Revisiting and Improving Generic Compositional Generalization of LLMs for Semantic Parsing in the Minimum Coverage Scenario

Anonymous ACL submission

Abstract

Compositional generalization is one of the important abilities that large language models (LLMs) need to have on semantic parsing tasks. Previous research typically relies on task-specific designs or a large number of samples in demonstrations to improve the compositional generalization of LLMs on semantic parsing. We revisit this issue and find that when the number of samples in a demonstration is limited to a theoretical lower bound for achieving compositional generalization (minimum coverage scenario), current advanced LLMs cannot arbitrarily achieve good compositional generalization generically on different semantic parsing tasks without task-specific designs. We propose Multi-level Component Composition (MC^2), a task-independent framework based on input primitives, which aims to generically help LLMs achieve compositional generalization in the minimum coverage scenario by selecting and organizing samples from multiple compositional levels that satisfy the primitive coverage. Experiments and analysis show that MC^2 can effectively improve compositional generalization of LLMs on different semantic parsing tasks in the minimum coverage scenario.

1 Introduction

Compositional generalization (Lake et al., 2016) refers to the ability to correctly handle unseen or rare combinations formed by known primitives. It is one of the inherent linguistic abilities of humans (Hupkes et al., 2020). Due to the potentially infinite number of combinations in languages (Chomsky, 1965), language models often encounter new combinations formed by known primitives in real-world tasks and require the ability of compositional generalization to cope with them. Therefore, compositional generalization of language models has been studied on many natural language processing tasks (Hupkes et al., 2022).

Semantic parsing (Kamath and Das, 2019) is one of the fields that have received the most attention in the research on compositional generalization. Tasks in this field aim to transform natural language into a more formal language to express its meaning more precisely, such as syntactic structure generation (Kim and Linzen, 2020), text-to-SQL (Keysers et al., 2020), etc. Since target languages typically have a strong compositionality, language models intuitively require the ability of compositional generalization for semantic parsing tasks.

The emergence of large language models (LLMs) drives the emergence of the in-context learning (Dong et al., 2024) paradigm for compositional generalization research on semantic parsing. In this paradigm, LLMs are asked to achieve the correct processing of test samples containing unseen combinations after seeing the demonstrations formed by samples drawn from the training corpus. Many research works have considered how compositional generalization of LLMs on semantic parsing tasks can be improved under the in-context-learning paradigm (Levy et al., 2023; An et al., 2023; Zhou et al., 2023; Drozdov et al., 2023). Limited by the performance of LLMs at the time, the methods used for improvement often rely on task-specific designs or a large number of samples in demonstrations, and thus have drawbacks in terms of generalizability or consumption. As the performance of LLMs improves, the question of *whether LLMs can generically achieve compositional generalization on different semantic parsing tasks with a rather limited number of samples in demonstrations and without task-specific designs* becomes a question worth revisiting.

To revisit this question, we introduce *minimum coverage scenario*. This scenario requires LLMs to achieve compositional generalization on semantic parsing tasks when the number of samples in a demonstration is limited to a theoretical lower bound for achieving compositional generalization. We find that existing advanced LLMs are unable

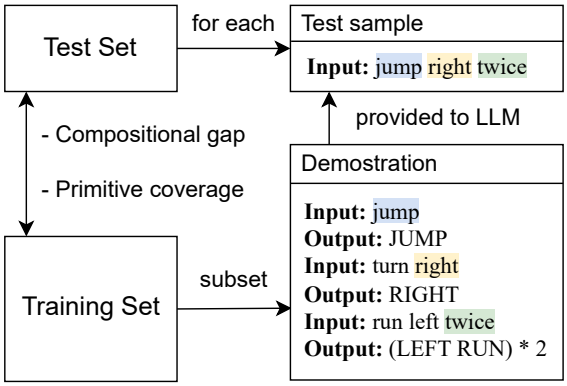

Figure 1: An illustration of compositional generalization of LLMs on the semantic parsing task in the in-context learning paradigm.

to arbitrarily achieve good compositional generalization on a variety of different semantic parsing tasks generically in the minimum coverage scenario *without task-specific designs*. This suggests that despite the continued development of LLMs, achieving generic compositional generalization on semantic parsing tasks in the minimum coverage scenario remains a challenge currently.

To generically improve the compositional generalization of LLMs on semantic parsing tasks in the minimum coverage scenario, we propose *Multi-level Component Composition* (MC^2), which is a task-independent framework based on input primitives. By three steps of chunking-matching-ordering, the framework selects samples from multiple compositional levels for LLMs and organizes the samples in a way that potentially exhibits compositional paths, while satisfying the basic requirement of the minimum coverage scenario. We show the effectiveness of the MC^2 framework through experiments on a variety of compositional generalization datasets of different types of semantic parsing tasks in the minimum coverage scenario. We also experimentally analyze (1) the importance of introducing multiple compositional levels, which is the core idea of the MC^2 framework, (2) the effect of steps in the framework, and (3) the impact of choices on method components and implementation rules in the framework. The experiments and analysis further confirm the design motivation of the MC^2 framework and demonstrate the room for improvement of the framework.

2 Minimum Coverage Scenario

In this section, we introduce the minimum coverage scenario, which is a challenging scenario for the compositional generalization of LLMs on semantic parsing tasks without task-specific designs.

2.1 Compositional Generalization of LLMs

Before the advent of LLMs, compositional generalization is typically evaluated using the train-test paradigm (Finegan-Dollak et al., 2018). Researchers construct the training set V and the test set W with significantly different distributions of combinations. The model is asked to be trained (or fine-tuned) on V and its performance on W is used to evaluate the ability of compositional generalization. A basic requirement to be met by sets V and W is that every input primitive appearing in W also appears in V, since the model needs to know at least the representation of the input primitive in the target to complete the composition (Lake and Baroni, 2018). This requirement is known as primitive coverage of W by V.

The emergence of LLMs introduces the in-context learning paradigm of compositional generalization (Levy et al., 2023; An et al., 2023). In this paradigm, the demonstration phase of in-context learning is regarded as the training phase. For each test sample s in W, a subset V' of samples in V that satisfy the primitive coverage of $\{s\}$ by V' is provided to the LLM as a demonstration, and then the LLM is asked to process the test sample. The testing of each test sample in W is independent. Figure 1 shows an illustration of compositional generalization of LLMs on the semantic parsing task in the in-context learning paradigm.

2.2 Definition

Based on the in-context learning paradigm, we introduce the concept of k-coverage scenario, which is a dynamic limit on the size of the subset V' (i.e., the number of samples in the demonstration). In the k-coverage scenario, for a test sample containing n input primitives, at most $k*n$ samples can be included in the demonstration. The limit becomes tighter as k decreases. Even an ideal model (i.e., one that can achieve compositional generalization as long as the demonstration satisfies primitive coverage) must, in the extreme case, be provided with n samples in the demonstration to satisfy primitive coverage to achieve compositional generalization. Thus, the 1-coverage scenario is a theoretical lower bound for achieving compositional generalization. We refer to the 1-coverage scenario as the minimum coverage scenario.

The minimum coverage scenario is valuable for

Model	k	SCAN	COGS	CFQ
DeepSeek	$k=1$	63.8	89.7	63.8
	$k=2$	86.3	90.8	74.9
	$k=4$	93.7	91.8	79.8
Qwen-plus	$k=1$	62.1	82.4	66.2
	$k=2$	80.7	84.9	75.3
	$k=4$	90.6	87.8	80.7

Table 1: Results of the pre-experiment in k-coverage scenario with $k \in \{1, 2, 4\}$. For SCAN and CFQ, we test on the MCD1 split.

research from both practical and cognitive perspectives. From the practical perspective, the minimum coverage scenario is the scenario with minimum input consumption for the compositional generalization of LLMs. From the cognitive perspective, the minimum coverage scenario is a challenge to the theoretical lower bound, demonstrating an upper bound on the ability of compositional generalization of LLMs on semantic parsing tasks.

2.3 Difficulty Estimation

To estimate the difficulty of the minimum coverage scenario, we conduct a pre-experiment to test the performance of advanced LLMs DeepSeek-2.5 (DeepSeek-AI et al., 2024) and Qwen-plus (Qwen et al., 2024) in k-coverage scenario with $k \in \{1, 2, 4\}$. The datasets and metrics of the pre-experiment are consistent with the main experiment (which we will describe in detail in Section 4). In the k-coverage scenario, for each of the n input primitives, k samples covering that primitive are randomly selected, for a total of $k * n$ mutually exclusive samples for the demonstration.

Table 1 shows the results of the pre-experiment. As k decreases, the performance of the LLMs on each dataset in the k-coverage scenario decreases significantly, reflecting the elevated difficulty. At $k = 1$, i.e., in the minimum coverage scenario, the performance of the LLMs is far from achieving perfect compositional generalization. The results provide an estimate of the challenge of the minimum coverage scenario to some extent, as the LLMs in this scenario are not yet able to achieve good compositional generalization on different semantic parsing tasks without task-specific designs under arbitrary choices for demonstration. This also motivates us to conduct research on improvements. Our subsequent framework proposal and experiments are all based on the minimum coverage scenario.

3 MC^2 Framework

In this section, we introduce the Multi-level Component Composition (MC^2) framework for generally helping LLMs achieve compositional generalization on different semantic parsing tasks in the minimum coverage scenario. The core motivation of the design is to help achieve compositional generalization by showing LLMs a compositional path with samples from multiple compositional levels. Moreover, the framework is designed to fulfill two requirements:

(1) The framework must be task-independent to generalize across different semantic parsing tasks. This requires the framework to be based on input primitives only, since the output formal language is usually task-relevant. It also requires that the framework does not need to have any prior knowledge of the task beyond the identification of the input primitive (see Appendix A for details), such as knowledge of the task-specific input syntax.

(2) The framework must be able to satisfy the requirements of primitive coverage and the number of samples in the demonstration in the minimum coverage scenario.

The framework consists of three steps: chunking, matching, and ordering. Figure 2 shows an illustration of the MC^2 framework.

3.1 Chunking

The chunking step aims at obtaining components from multiple compositional levels of the input and the relationships between the components as a basis for subsequent sample selection. In the chunking step, the input is decomposed compositionally to form a tree structure. Each node in the tree represents a component that is decomposed from the component represented by its parent node. For an input containing n input primitives, the final tree will contain n leaf nodes representing the primitives, and up to $n-1$ non-leaf nodes. The method of chunking is flexible, and any task-independent method capable of tree-like component decomposition can be chosen. We consider the following two methods of chunking:

PPL-based chunking. We use a small language model to compute the perplexity. For the component represented by each node, we enumerate the schemes that slice the component into two parts *L* / *R* and choose the scheme that minimizes the perplexity of the sentence "*Using / to split the sentence {component}. {L} / {R}*". According to the

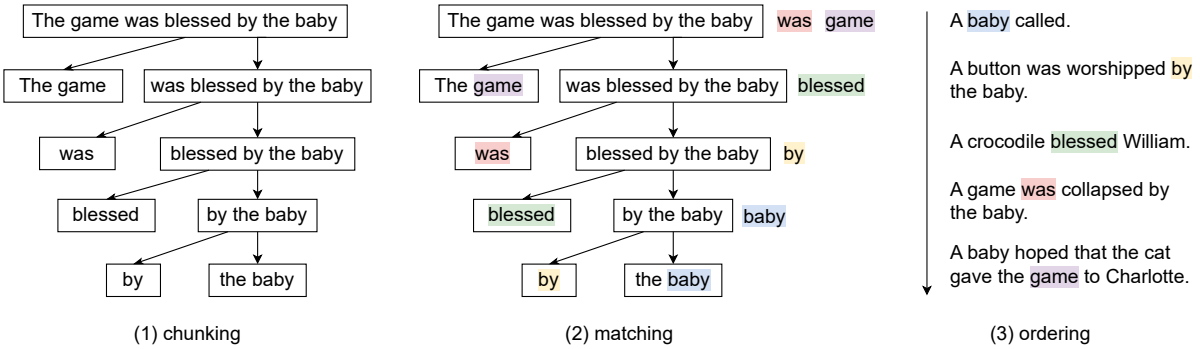

Figure 2: An illustration of the MC^2 framework. (1) Chunking: the input is component-wise chunked to form a tree structure. (2) Matching: each leaf node primitive (in color) corresponds to a non-leaf node, and the non-leaf node component matches the most similar sample in the training set that covers the corresponding primitive. (3) Ordering: samples matched by each non-leaf node are arranged in a specific order to form a demonstration.

scheme, we split the components and recursively process the two child nodes formed. The method always results in binary trees. See Appendix B for pseudo-code.

Off-the-shelf parser. The task of constituent parsing (Kitaev and Klein, 2018) aims at obtaining a tree-like constituent decomposition of the input sentence with constituent labels, which fits with the goal of the chunking step. Therefore, we can use an off-the-shelf constituent parser (based on a small language model) for the chunking step. The method may result in multinomial trees.

3.2 Matching

Based on the tree structure obtained from the chunking step, the matching step aims to select a total of n mutually exclusive samples from the training set that are similar to the components of the different compositional levels and satisfy the primitive coverage requirement.

For a non-leaf node x, we define the degree D_x as (the number of child nodes of x) $-$ 1. It always holds that $\sum D_x = n - 1$. After increasing D_{root} by 1, we select D_x samples for each non-leaf node x. We visit each leaf node y in reverse order of breadth-first search (BFS), find the deepest ancestor node x' of y for which the number of samples selected is less than $D_{x'}$, and then from the samples in the training set that have not been selected and cover the primitive represented by y, we will select a sample that is most similar to the component represented by x'. Finally, we will select n mutually exclusive samples that satisfy the primitive coverage requirement. The method of finding the most similar sample is flexible and can be chosen from any method that estimates the text similarity. We consider two methods: the BM25, and the cosine similarity of sentence vectors given by the small language model. See Algorithm 1 for pseudo-code.

Algorithm 1 Matching and Ordering

Input: tree structure T, training set V
Output: ordered list of samples L as the demonstration
  for all non-leaf node x in T **do**
    $D_x \leftarrow$ (the number of child nodes of x) $-$ ($x \neq$ root)
    $L_x \leftarrow$ empty list
  end for
  for all leaf node y in T in inverted BFS order **do**
    $p \leftarrow$ the primitive represented by y
    $x' \leftarrow$ parent node of y
    while $|L_{x'}| = D_{x'}$ **do**
      $x' \leftarrow$ parent node of x'
    end while
    $R \leftarrow \{s \mid s \in V \wedge (\forall x,\ s \notin L_x) \wedge \text{cover}\ (s, p)\}$
    $c \leftarrow$ the component represented by x'
    $L_{x'} \leftarrow L_{x'}$ + [the sample in R that is most similar to c]
  end for
  $L \leftarrow$ empty list
  for all non-leaf node x in T in inverted BFS order **do**
    $L \leftarrow L + L_x$
  end for
  return L

3.3 Ordering

The ordering step aims to order the selected samples in a way that potentially presents compositional paths. Based on the tree structure, we simply visit each non-leaf node in inverted BFS order and list the samples selected for each node in turn to form the demonstration. The BFS inverted order is the order of the compositional levels of the tree nodes from low to high, and to some extent characterizes the compositional paths that are progressively thought out from simple to complex components. See Algorithm 1 for pseudo-code.

4 Main Experiment

In this section, we conduct the main experiment to verify the effectiveness of the MC^2 framework on different semantic parsing tasks in the minimum coverage scenario and to verify the effect of the core idea of the framework and steps in it.

4.1 Datasets and Metrics

We conduct experiments on three representative datasets for compositional generalization research on semantic parsing: SCAN (Lake and Baroni, 2018), COGS (Kim and Linzen, 2020), and CFQ (Keysers et al., 2020). The three datasets correspond to three different semantic parsing tasks. Table 2 shows samples of the datasets.

SCAN. The SCAN dataset corresponds to the task of simplified natural language to action sequence translation. We use the maximum compound divergence (MCD) splits provided by Keysers et al. (2020), containing three training-test splits MCD1~MCD3. The MCD splits achieve high compositional divergence under similar atom distributions, which can effectively evaluate the ability of compositional generalization.

COGS. The COGS dataset corresponds to the task of syntactic structure generation for natural language. We use the original training-test split containing five categories of compositional generalization.

CFQ. The CFQ dataset corresponds to the task of text-to-SQL. This dataset contains realistic natural language questions and corresponding SPARQL queries. We use the original MCD splits containing the three training-test splits MCD1~MCD3.

For each split in each dataset, we sample 1,000 samples from the test set for testing. The metric is the accuracy under exact matching. Since sometimes the correct answer is not unique, we convert the model output and the standard answer to ensure correct matching. The details of the conversion are shown in Appendix C.

4.2 Evaluated LLMs

We conduct the main experiment on some advanced LLMs that provide API calls from official platforms, including the open-source LLM DeepSeek-2.5 and the closed-source LLM Qwen-plus and GPT-4o (OpenAI et al., 2024).

Dataset: SCAN (action sequence generation) **Input:** turn opposite left thrice and run around right twice **Output:** (LEFT LEFT) * 3 (RIGHT RUN RIGHT RUN RIGHT RUN RIGHT RUN) * 2
Dataset: COGS (syntactic structure generation) **Input:** Olivia believed that a donut was drawn by a girl. **Output:** believe (olivia, none, none) ccomp draw (girl, donut, none)
Dataset: CFQ (text-to-SQL) **Input:** What female person did M2 's employee and founder influence? **Output:** SELECT DISTINCT ?x0 WHERE { ?x0 a person . ?x0 influenced_by ?x1 . ?x0 has_gender female . ?x1 founded M2 . ?x1 employed_by M2 }

Table 2: Samples of the three semantic parsing datasets used in the experiments.

4.3 Settings

The small language models we use in our experiments include GPT-2-large (0.8B) (Radford et al., 2019) for PPL-based chunking, T5-large (0.8B) (Raffel et al., 2020) as a base model for the off-the-shelf parser BENEPAR (Kitaev and Klein, 2018), and all-mpnet-base-v2 (0.1B) (Song et al., 2020) for generating sentence vectors in the matching step. To verify the overall effect of the MC^2 framework and the effect of steps in it, our main experiment contains the following four settings:

Random. For each input primitive in turn, we randomly select a sample that is unselected and covers that primitive, and arrange the samples in the order of selection to form a demonstration.

Global Matching. Based on the previous setting, the random selection is replaced by the selection of the sample that is most similar to the whole input, using the same similarity estimation method as in the Matching step.

Chunking + Matching (C + M). The chunking and matching steps in the framework are used. The samples are arranged in the order that the covered primitives are in the input, consistent with the previous two settings.

Complete MC^2. The complete MC^2 framework is used, consisting of three steps.

In the main experiment, we fix the use of the off-the-shelf parser method in the chunking step and the vector method in the matching step. A

Model	Setting	$SCAN_1$	$SCAN_2$	$SCAN_3$	COGS	CFQ_1	CFQ_2	CFQ_3
DeepSeek	Random	63.8	65.2	55.9	89.7	63.8	68.0	68.7
	Global Matching	68.1	79.2	50.1	92.9	77.8	69.9	72.1
	C + M	74.1	81.1	51.1	**95.4**	78.7	75.0	77.3
	Complete MC^2	**76.6**	**88.4**	**57.5**	95.2	**79.8**	**76.4**	**77.6**
Qwen-plus	Random	62.1	66.7	56.1	82.4	66.2	68.4	69.3
	Global Matching	69.4	89.7	59.1	87.8	79.2	72.5	70.7
	C + M	73.6	92.0	58.5	88.5	79.8	77.4	74.6
	Complete MC^2	**78.2**	**94.9**	**59.4**	**89.1**	**79.9**	**77.5**	**74.7**
GPT-4o	Random	70.2	71.7	64.5	81.5	73.4	71.9	72.7
	Global Matching	72.4	84.5	58.9	86.8	80.6	78.3	77.9
	C + M	81.9	87.8	64.8	88.7	81.3	81.1	78.9
	Complete MC^2	**82.4**	**90.8**	**67.1**	**89.3**	**83.3**	**83.7**	**80.8**

Table 3: Results of the main experiment in the minimum coverage scenario. For SCAN and CFQ, numerical subscripts indicate the corresponding MCD split.

comparative analysis of the different methods will be shown in Section 5.

4.4 Results

Table 3 shows the results of the main experiment in the minimum coverage scenario. We analyze the effect of the core idea of the MC^2 framework and steps in it by comparing the results in different settings adjacent to each other.

Random vs. **Global Matching**. Matching using similarity metrics is intuitively a better selection strategy than random selection. In most comparisons, using Global Matching indeed leads to better performance than Random. However, we find that in some cases Global Matching brings a significant performance degradation (DeepSeek and GPT-4o on $SCAN_3$). This suggests that global-level selection based on certain similarity metrics may instead lead to weaker compositional generalization for some tasks or models.

Global Matching vs. **C + M**. Compared to Global Matching, C + M introduces the idea of multiple compositional levels. Unlike simply matching on the global input, C + M decomposes the components of different compositional levels with the help of the chunking step, and then matches the most similar samples for each component. As can be seen from the results, this hierarchical matching improves performance in most cases, and significantly in some comparisons. In the only exception where performance drops (Qwen-plus on $SCAN_3$), the drop is not significant. Overall, the chunking and matching steps that introduce the idea of multiple compositional levels effectively improve generic compositional generalization.

C + M vs. **Complete MC^2**. The complete MC^2 adds a a final ordering step compared to C + M, which changes the sample ordering from input order by matched primitive to order by matched tree nodes from lower to higher level. The results are similar to the previous set of comparisons. In most cases, the added ordering step improves performance, and the improvement is significant in some comparisons. In the only exception where performance drops (DeepSeek on COGS), the performance drop is also not significant. The results indicate that the addition of the ordering step overall improves generic compositional generalization.

From the results and analysis, we find that each step is important for performance improvement. The core idea of multiple compositional levels, which serves as the basis for component matching and sample ordering, plays an important role in generic improvement.

5 Analytical Experiments

In this section, we conduct analytical experiments on DeepSeek-2.5 and Qwen-plus on choices in the MC^2 framework, including the choice of method components and the choice of implementation rules. The purpose of the experiments is to analyze the impact of various choices on performance and to provide insight into the room for further improvement of the framework.

5.1 Method Components

In the MC^2 framework, the method of tree structure acquisition in the chunking step, and the method of finding the most similar samples in the matching

Model	Method	$SCAN_1$	$SCAN_2$	$SCAN_3$	COGS	CFQ_1	CFQ_2	CFQ_3
DeepSeek	PPL + BM25	67.9	84.2	56.4	92.2	79.3	75.7	74.4
	PPL + Vector	**78.4**	**93.4**	56.5	94.6	76.2	74.6	73.8
	Parser + BM25	69.3	83.1	**58.4**	91.7	78.2	75.7	**77.8**
	Parser + Vector	76.6	88.4	57.5	**95.2**	**79.8**	**76.4**	77.6
Qwen-plus	PPL + BM25	68.4	84.6	57.2	86.6	78.5	75.8	75.5
	PPL + Vector	72.3	93.3	56.4	**89.5**	76.9	75.2	73.7
	Parser + BM25	68.5	83.9	57.0	88.1	**80.0**	76.1	**76.2**
	Parser + Vector	**78.2**	**94.9**	**59.4**	89.1	79.9	**77.5**	74.7

Table 4: Results of experiments on choice combinations of method components in the minimum coverage scenario.

step, are components that can be flexibly changed. As described in Section 3, we consider two methods, PPL and Parser, in the chunking step, and two methods, BM25 and Vector, in the matching step. We conduct experiments on the combination of method choices for the two steps.

Table 4 shows the results of experiments. Of the four combinations, Parser + Vector demonstrates the best performance in most cases, while Parser + BM25 and PPL + Vector also perform best in some cases. For different LLMs, the combinations that show the best performance under the same split of the same task are different; for the same LLMs, the combinations that show the best performance under different tasks are not exactly the same. Overall, there is no combination that presents an absolute performance advantage.

For the performance of the framework, the impact of the chosen combination of method components is significant in some cases, reflected in the large performance difference between the best and worst performing combinations (e.g., $SCAN_1$ and $SCAN_2$). In addition, the performance of the framework may be limited by the performance of the method components in a particular case, e.g., the performance improvement of DeepSeek using Parser + Vector on $SCAN_3$ is relatively limited due to the performance of the Vector method. Therefore, finding task-independent method components and combinations that have better generic performance is one of the potential room for further performance improvement of the framework.

5.2 Implementation Rules

The MC^2 framework contains several implementation rules with multiple choices without altering the underlying design purpose. In each of the following implementation rules, we experiment with a choice different from the one in Section 3.

Visiting rule. In the matching and ordering steps, we use the BFS inverted order as the visiting order with the purpose of demonstrating the potential compositional paths. In the choice of visiting rule, any order that potentially demonstrates compositional paths can be taken into account. We experiment with the inverse order of depth-first search (DFS) as the visiting order, which characterizes another typical bottom-up compositional pattern.

Degree rule. In the matching step, the degree rule specifies the number of samples D_x matched by each non-leaf node x. The rule is to satisfy the requirement that the total number of samples in the minimum coverage scenario cannot exceed n. Any positive integer assignment to D_x that satisfies the sum of n can be used as a degree rule. We experiment with another rule that concentrates D_x on the root. Under this rule, $D_x = 1$ for non-root and non-leaf nodes x, and $D_{root} = (n - \text{number of non-root and non-leaf nodes})$.

Primitive coverage rule. In the matching step, each sample matched for a non-leaf node must cover the primitive corresponding to the corresponding leaf node. This rule is to ensure that each input primitive is covered after the sample matching is completed. We experiment with the rule of adding a branch to the original rule: when the primitive represented by the corresponding leaf node has already been covered by a sample that has been selected, the sample selected this time does not have to cover the primitive. This new rule still satisfies the primitive coverage requirement, potentially trading a reduction in the number of primitives in the demonstration for a larger sample selection space.

Table 5 shows the results of experiments. From the results, we find that there is no one implementation rule choice that exhibits an absolute perfor-

Model	Rule	$SCAN_1$	$SCAN_2$	$SCAN_3$	COGS	CFQ_1	CFQ_2	CFQ_3
DeepSeek	Original	76.6	88.4	57.5	95.2	79.8	76.4	77.6
	* Visiting	77.4	88.7	56.2	95.1	79.4	75.4	78.0
	* Degree	76.9	89.0	57.2	95.6	79.5	76.3	77.6
	* Primitive	76.9	87.3	57.7	95.4	80.2	75.9	76.4
Qwen-plus	Original	78.2	94.9	59.4	89.1	79.9	77.5	74.7
	* Visiting	79.2	94.0	60.1	90.5	82.0	77.1	77.8
	* Degree	77.9	94.8	59.9	89.6	80.9	76.2	75.1
	* Primitive	78.2	94.2	59.0	87.7	81.4	77.0	76.9

Table 5: Results of experiments on the choice of implementation rules in the minimum coverage scenario. In the Rule column, Original indicates that the implementation rules described in Section 3 are used, and other rows with * indicate that one of the corresponding rules in 5.2 is modified. Compared to Original, the performance improved by the modification is marked in red, and the performance degraded is marked in blue.

mance advantage, similar to the results for method components. Although LLMs may exhibit some preference for the choice of implementation rules (e.g., Qwen-plus prefers the DFS visiting rule), it does not appear that implementation rules before and after modification exhibit the same comparative results on all splits of an LLM.

While some of the modifications result in slightly larger performance changes (e.g., visiting rule on CFQ_3 for Qwen-plus), most of the modifications result in small performance changes. This implies that the framework is relatively insensitive to modifications to a single implementation rule without changing the underlying design purpose. However, better choices of implementation rules remain to be investigated, especially considering that the internal combinations of implementation rules and their combinations with other elements of the framework are still under-explored.

6 Related Work

For compositional generalization of LLMs on semantic parsing in the in-context learning paradigm, many works have proposed different methods for improvement. Levy et al. (2023) proposes Cover-LS. This method first defines the task-specific local structure that the output has. Then, pairs of (input, output local structures) in the training set are used to train a prediction model. Finally, the prediction model predicts the local structures corresponding to the test input and selects the demonstration samples that cover more of the predicted local structures. An et al. (2023) consider improvement based on the parsing tree on COGS. They consider structural similarity, diversity, and complexity to select the demonstration samples with the best matching parsing tree for the test sample. Drozdov et al. (2023) proposes least-to-most prompting to improve the performance of LLMs on SCAN. They divide the task into two subproblems, decomposition and mapping, provide LLMs with manually written samples of decomposition and mapping, and then ask LLMs to solve the two subproblems sequentially. On COGS and CFQ, which are more difficult to decompose subproblems, Drozdov et al. (2023) proposes dynamic last-to-most prompting. They manually write specific decomposition rules and examples for COGS and CFQ, and ask LLMs to decompose the problem. Then, they perform the selection of demonstration samples for the decomposed subproblems and ask LLMs to solve the subproblems step-by-step. As distinct from related work, the MC^2 framework does not rely on task-specific design such as guidance from manual annotations, and can always satisfy the requirement of the minimum coverage scenario for the number of demonstration samples.

7 Conclusion

In this work, we revisit the question of whether LLMs can generically achieve good compositional generalization on different semantic parsing tasks in the minimum coverage scenario without task-specific designs, and find that achieving this remains a challenge for current advanced LLMs. We propose the task-independent MC^2 framework to generically improve the compositional generalization of LLMs for different tasks on semantic parsing in the minimum coverage scenario. We illustrate the effectiveness and room for improvement of the framework and affirm the design motivation through experiments and analysis.

Limitations

This work investigates compositional generalization semantic parsing in the settings of Generic (no task-specific design) and Minimum Coverage Scenario (limiting the number of samples in the demonstration). The settings introduce potential limitations from a practical perspective, as the use of task-specific designs or larger sample sizes is fully allowed in real-world scenarios to improve performance. However, we believe that such a setting is still relevant since Generic characterizes the generalization ability applied to any task, and the Minimum Coverage Scenario is directly related to resource constraints on input consumption.

A potential limitation of the MC^2 framework is that performance gains are to some extent dependent on well-performing method components and their combinations, as demonstrated in the analytical experiments. This dependence indicates the direction of continued exploration to find better method components and combinations for performance improvement. We will continue our exploration in this direction.

Ethics Statement

We comply with the license to use language models for scientific research purposes only. The datasets we use do not contain any information that names or uniquely identifies individual people or offensive content.

The AI assistant we use in our work is Copilot (for simple code completion).

References

Shengnan An, Zeqi Lin, Qiang Fu, Bei Chen, Nanning Zheng, Jian-Guang Lou, and Dongmei Zhang. 2023. How do in-context examples affect compositional generalization? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 11027–11052. Association for Computational Linguistics.

Noam Chomsky. 1965. *Aspects of the Theory of Syntax*. 11. MIT press.

DeepSeek-AI, :, Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, Huazuo Gao, Kaige Gao, Wenjun Gao, Ruiqi Ge, Kang Guan, Daya Guo, Jianzhong Guo, Guangbo Hao, Zhewen Hao, Ying He, Wenjie Hu, Panpan Huang, Erhang Li, Guowei Li, Jiashi Li, Yao Li, Y. K. Li, Wenfeng Liang, Fangyun Lin, A. X. Liu, Bo Liu, Wen Liu, Xiaodong Liu, Xin Liu, Yiyuan Liu, Haoyu Lu, Shanghao Lu, Fuli Luo, Shirong Ma, Xiaotao Nie, Tian Pei, Yishi Piao, Junjie Qiu, Hui Qu, Tongzheng Ren, Zehui Ren, Chong Ruan, Zhangli Sha, Zhihong Shao, Junxiao Song, Xuecheng Su, Jingxiang Sun, Yaofeng Sun, Minghui Tang, Bingxuan Wang, Peiyi Wang, Shiyu Wang, Yaohui Wang, Yongji Wang, Tong Wu, Y. Wu, Xin Xie, Zhenda Xie, Ziwei Xie, Yiliang Xiong, Hanwei Xu, R. X. Xu, Yanhong Xu, Dejian Yang, Yuxiang You, Shuiping Yu, Xingkai Yu, B. Zhang, Haowei Zhang, Lecong Zhang, Liyue Zhang, Mingchuan Zhang, Minghua Zhang, Wentao Zhang, Yichao Zhang, Chenggang Zhao, Yao Zhao, Shangyan Zhou, Shunfeng Zhou, Qihao Zhu, and Yuheng Zou. 2024. Deepseek llm: Scaling open-source language models with longtermism. *Preprint*, arXiv:2401.02954.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, and Zhifang Sui. 2024. A survey on in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 1107–1128. Association for Computational Linguistics.

Andrew Drozdov, Nathanael Schärli, Ekin Akyürek, Nathan Scales, Xinying Song, Xinyun Chen, Olivier Bousquet, and Denny Zhou. 2023. Compositional semantic parsing with large language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-SQL evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360, Melbourne, Australia. Association for Computational Linguistics.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2020. Compositionality decomposed: How do neural networks generalise? *J. Artif. Intell. Res.*, 67:757–795.

Dieuwke Hupkes, Mario Giulianelli, Verna Dankers, Mikel Artetxe, Yanai Elazar, Tiago Pimentel, Christos Christodoulopoulos, Karim Lasri, Naomi Saphra, Arabella Sinclair, Dennis Ulmer, Florian Schottmann, Khuyagbaatar Batsuren, Kaiser Sun, Koustuv Sinha, Leila Khalatbari, Maria Ryskina, Rita Frieske, Ryan Cotterell, and Zhijing Jin. 2022. State-of-the-art generalisation research in NLP: a taxonomy and review. *CoRR*, abs/2210.03050.

Aishwarya Kamath and Rajarshi Das. 2019. A survey on semantic parsing. In *1st Conference on Automated Knowledge Base Construction, AKBC 2019, Amherst, MA, USA, May 20-22, 2019*.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin,

Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Najoung Kim and Tal Linzen. 2020. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 9087–9105. Association for Computational Linguistics.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Brenden M. Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2879–2888. PMLR.

Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. 2016. Building machines that learn and think like people. *CoRR*, abs/1604.00289.

Itay Levy, Ben Bogin, and Jonathan Berant. 2023. Diverse demonstrations improve in-context compositional generalization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 1401–1422. Association for Computational Linguistics.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoochian, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, Behrooz Ghorbani, Ben Leimberger, Ben Rossen, Ben Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Giertler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian Guarraci, Brian Hsu, Bright Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll Wainwright, Cary Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Christopher Hesse, Claudia Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, Dane Sherburn, Daniel Kappler, Daniel Levin, Daniel Levy, David Carr, David Farhi, David Mely, David Robinson, David Sasaki, Denny Jin, Dev Valladares, Dimitris Tsipras, Doug Li, Duc Phong Nguyen, Duncan Findlay, Edede Oiwoh, Edmund Wong, Ehsan Asdar, Elizabeth Proehl, Elizabeth Yang, Eric Antonow, Eric Kramer, Eric Peterson, Eric Sigler, Eric Wallace, Eugene Brevdo, Evan Mays, Farzad Khorasani, Felipe Petroski Such, Filippo Raso, Francis Zhang, Fred von Lohmann, Freddie Sulit, Gabriel Goh, Gene Oden, Geoff Salmon, Giulio Starace, Greg Brockman, Hadi Salman, Haiming Bao, Haitang Hu, Hannah Wong, Haoyu Wang, Heather Schmidt, Heather Whitney, Heewoo Jun, Hendrik Kirchner, Henrique Ponde de Oliveira Pinto, Hongyu Ren, Huiwen Chang, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian O'Connell, Ian Osband, Ian Silber, Ian Sohl, Ibrahim Okuyucu, Ikai Lan, Ilya Kostrikov, Ilya Sutskever, Ingmar Kanitscheider, Ishaan Gulrajani, Jacob Coxon, Jacob Menick, Jakub Pachocki, James Aung, James Betker, James Crooks, James Lennon, Jamie Kiros, Jan Leike, Jane Park, Jason Kwon, Jason Phang, Jason Teplitz, Jason Wei, Jason Wolfe, Jay Chen, Jeff Harris, Jenia Varavva, Jessica Gan Lee, Jessica Shieh, Ji Lin, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joanne Jang, Joaquin Quinonero Candela, Joe Beutler, Joe Landers, Joel Parish, Johannes Heidecke, John Schulman, Jonathan Lachman, Jonathan McKay, Jonathan Uesato, Jonathan Ward, Jong Wook Kim, Joost Huizinga, Jordan Sitkin, Jos Kraaijeveld, Josh Gross, Josh Kaplan, Josh Snyder, Joshua Achiam, Joy Jiao, Joyce Lee, Juntang Zhuang, Justyn Harriman, Kai Fricke, Kai Hayashi, Karan Singhal, Katy Shi, Kavin Karthik, Kayla Wood, Kendra Rimbach, Kenny Hsu, Kenny Nguyen, Keren Gu-Lemberg, Kevin Button, Kevin Liu, Kiel Howe, Krithika Muthukumar, Kyle Luther, Lama Ahmad, Larry Kai, Lauren Itow, Lauren Workman, Leher Pathak, Leo Chen, Li Jing, Lia Guy, Liam Fedus, Liang Zhou, Lien Mamitsuka, Lilian Weng, Lindsay McCallum, Lindsey Held, Long Ouyang, Louis Feuvrier, Lu Zhang, Lukas Kondraciuk, Lukasz Kaiser, Luke Hewitt, Luke Metz, Lyric Doshi, Mada Aflak, Maddie Simens, Madelaine Boyd, Madeleine Thompson, Marat Dukhan, Mark Chen, Mark Gray, Mark Hudnall, Marvin Zhang, Marwan Aljubeh, Mateusz Litwin, Matthew Zeng, Max Johnson, Maya Shetty, Mayank Gupta, Meghan Shah, Mehmet Yatbaz, Meng Jia Yang, Mengchao Zhong, Mia Glaese, Mianna Chen, Michael Janner, Michael Lampe, Michael Petrov, Michael Wu, Michele Wang, Michelle Fradin, Michelle Pokrass, Miguel Castro, Miguel Oom Temudo de Castro,

Mikhail Pavlov, Miles Brundage, Miles Wang, Minal Khan, Mira Murati, Mo Bavarian, Molly Lin, Murat Yesildal, Nacho Soto, Natalia Gimelshein, Natalie Cone, Natalie Staudacher, Natalie Summers, Natan LaFontaine, Neil Chowdhury, Nick Ryder, Nick Stathas, Nick Turley, Nik Tezak, Niko Felix, Nithanth Kudige, Nitish Keskar, Noah Deutsch, Noel Bundick, Nora Puckett, Ofir Nachum, Ola Okelola, Oleg Boiko, Oleg Murk, Oliver Jaffe, Olivia Watkins, Olivier Godement, Owen Campbell-Moore, Patrick Chao, Paul McMillan, Pavel Belov, Peng Su, Peter Bak, Peter Bakkum, Peter Deng, Peter Dolan, Peter Hoeschele, Peter Welinder, Phil Tillet, Philip Pronin, Philippe Tillet, Prafulla Dhariwal, Qiming Yuan, Rachel Dias, Rachel Lim, Rahul Arora, Rajan Troll, Randall Lin, Rapha Gontijo Lopes, Raul Puri, Reah Miyara, Reimar Leike, Renaud Gaubert, Reza Zamani, Ricky Wang, Rob Donnelly, Rob Honsby, Rocky Smith, Rohan Sahai, Rohit Ramchandani, Romain Huet, Rory Carmichael, Rowan Zellers, Roy Chen, Ruby Chen, Ruslan Nigmatullin, Ryan Cheu, Saachi Jain, Sam Altman, Sam Schoenholz, Sam Toizer, Samuel Miserendino, Sandhini Agarwal, Sara Culver, Scott Ethersmith, Scott Gray, Sean Grove, Sean Metzger, Shamez Hermani, Shantanu Jain, Shengjia Zhao, Sherwin Wu, Shino Jomoto, Shirong Wu, Shuaiqi, Xia, Sonia Phene, Spencer Papay, Srinivas Narayanan, Steve Coffey, Steve Lee, Stewart Hall, Suchir Balaji, Tal Broda, Tal Stramer, Tao Xu, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Cunninghman, Thomas Degry, Thomas Dimson, Thomas Raoux, Thomas Shadwell, Tianhao Zheng, Todd Underwood, Todor Markov, Toki Sherbakov, Tom Rubin, Tom Stasi, Tomer Kaftan, Tristan Heywood, Troy Peterson, Tyce Walters, Tyna Eloundou, Valerie Qi, Veit Moeller, Vinnie Monaco, Vishal Kuo, Vlad Fomenko, Wayne Chang, Weiyi Zheng, Wenda Zhou, Wesam Manassra, Will Sheu, Wojciech Zaremba, Yash Patil, Yilei Qian, Yongjik Kim, Youlong Cheng, Yu Zhang, Yuchen He, Yuchen Zhang, Yujia Jin, Yunxing Dai, and Yury Malkov. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

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. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. In *Advances in Neural Information Processing Systems*, volume 33, pages 16857–16867. Curran Associates, Inc.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

A Primitive Identification

For input, we mostly simply use spaces as separators to identify primitives and ignore cases when matching primitives. In the following cases, we perform special identification to avoid redundant or unusual primitives: (1) articles (e.g., the, a, an) are not considered as primitives, (2) punctuation marks are separated from the word and are not considered as primitives, and (3) for the SCAN dataset, opposite and around, which act as special ideograms, are combined with the directional adverbs that follow them to form a single primitive.

B PPL-based Chunking

Algorithm 2 shows pseudo-code for PPL-based chunking.

Algorithm 2 PPL-based Chunking

function: chunking
Input: input primitives $p_1, p_2, ..., p_n$
Output: root node x of the tree structure
  $x \leftarrow$ new node $(p_1, p_2, ..., p_n)$
  if $n = 1$ **then**
    return x
  end if
  $m, v \leftarrow 0, \infty$
  for i from 1 to $n - 1$ **do**
    sent $\leftarrow$ "Using / to split the sentence $\{p_1, p_2, ..., p_n\}$. $\{p_1, p_2, ..., p_i\}$ / $\{p_{i+1}, p_{i+2}, ..., p_n\}$"
    if PPL (sent) $< v$ **then**
      $m, v \leftarrow i$, PPL (sent)
    end if
  end for
  $l \leftarrow$ chunking $(p_1, p_2, ..., p_m)$
  $r \leftarrow$ chunking $(p_m, p_{m+1}, ..., p_n)$
  set l, r to be the left and right son nodes of x
  return x

C Conversion Details

We match the model outputs and standard answers after a uniform conversion. The purpose of the conversion is to unify all possible correct answers to a unique one, which is consistent with the result of the conversion of the standard answer. For all datasets, we ignore redundant spaces and information other than answers. For the SCAN dataset, we expand the parentheses with multipliers by the number of repetitions shown by the multipliers to get the only correct answer without parentheses. For the COGS dataset, we ignore all case issues. For the CFQ dataset, following Drozdov et al. (2023), we uniquely orient the bidirectional relation and perform iterative sorting and normalization to ensure the uniqueness of the answer.