SPOR: A Comprehensive and Practical Evaluation Method for Compositional Generalization in Data-to-Text Generation

Anonymous ACL submission

Abstract

Compositional generalization is an important ability of language models and has many different manifestations. For data-to-text generation, previous research on this ability is limited to a single manifestation called *Systematicity* and lacks consideration of large language models (LLMs), which cannot fully cover practical application scenarios. In this work, we propose SPOR, a comprehensive and practical evaluation method for compositional generalization in data-to-text generation. SPOR includes four aspects of manifestations (*Systematicity*, *Productivity*, *Order invariance*, and *Rule learnability*) and allows high-quality evaluation without additional manual annotations based on existing datasets. We demonstrate SPOR on two different datasets and evaluate some existing language models including LLMs. We find that the models are deficient in various aspects of the evaluation and need further improvement. Our work shows the necessity for comprehensive research on different manifestations of compositional generalization in data-to-text generation and provides a framework for evaluation.

1 Introduction

Data-to-text generation (Gatt and Krahmer, 2018) is an important task in natural language generation (NLG). It aims to generate fluent and faithful text based on structured data input and is critical in many NLG systems, such as report generation (Wiseman et al., 2017), oriented dialogues (Mehta et al., 2022), etc. In data-to-text generation, structured data input is compositional, i.e., it can be considered as a combination of elements formed according to certain rules. Therefore, in order to handle the practical data-to-text generation, the language models should have the ability to recombine previously learned elements with certain rules to map new inputs made up from these elements to their correct output (Hupkes et al., 2022), which is the so-called compositional generalization.

Compositional generalization is an important ability of language models for many tasks. In semantic parsing and mathematical reasoning tasks, many different manifestations of this ability have been studied (Hupkes et al., 2020; Ontañón et al., 2022), such as *systematicity* (handle combinations unseen during training), *productivity* (extrapolate to longer sequences than those seen during training), etc. For compositional generalization in data-to-text generation, only *systematicity* receives attention (Mehta et al., 2022), and research on other manifestations is lacking. The single *systematic* manifestation cannot fully cover practical application scenarios of compositional generalization and cannot comprehensively reflect this ability of language models in data-to-text generation. Although research on different manifestations of compositional generalization in data-to-text generation is necessary, there is currently no comprehensive evaluation method to support such research.

To solve this problem, we propose SPOR, a comprehensive and practical evaluation method for compositional generalization in a data-to-text generation. Based on the manifestations of compositional generalization mentioned in Hupkes et al. (2020), SPOR includes four aspects of compositional generalization in data-to-text generation:

- *Systematicity.* The ability to handle data combinations unseen during training.
- *Productivity.* The ability to handle a larger amount of data within a sample than seen during training.
- *Order invariance.* The ability to maintain the fidelity and proper data ordering of the output text when the input order of data in an unordered set is changed.
- *Rule learnability.* The ability to actually learn and apply *copy rule* for generation, rather than memorize specific mappings.

For each aspect, we propose the corresponding methods for dataset construction and evaluation. Based on existing datasets, we mainly perform repartition (Keysers et al., 2020) and element modification to construct datasets for our evaluation. Overall, the evaluation method SPOR has the following properties:

- *Necessity.* The ability or property in each aspect manifests compositional generalization and is required by the model for practical data-to-text generation.
- *High evaluation quality.* For each aspect, the evaluation method can effectively evaluate the corresponding ability or property.
- *Low construction cost.* Based on existing datasets, the dataset used for evaluation does not require additional manual annotation and can be constructed automatically.

We demonstrate SPOR on two existing datasets for data-to-text generation and evaluate some existing language models. Previous research on compositional generalization in data-to-text generation lacks consideration of large language models (LLMs) due to the lack of methods to directly fine-tune and apply LLMs to data-to-text generation in the past. Nowadays, advanced Parameter-Efficient Fine-Tuning such as LoRA (Hu et al., 2022) provides the methods, and the consideration of LLMs becomes necessary. Therefore, we include some advanced LLMs in our evaluation to partially fill the gap in previous research.

2 Preliminaries

In this section, we provide a brief description of the datasets that SPOR is demonstrated on, the evaluated models, and the evaluation metrics.

2.1 Datasets

We demonstrate SPOR on two data-to-text generation datasets, WebNLG (Gardent et al., 2017) and E2E (Novikova et al., 2017). Both contain (D, T) pairs, where D is the input data and T is the text that verbalizes the data.

WebNLG is a realistic multi-domain dataset. In WebNLG, D is an unordered set of 1~7 triples $\langle s, p, o \rangle$, where s, p, o represents subject, predicate, and object, respectively. We select the latest version, WebNLG+ (Ferreira et al., 2020), which covers more domains and contains more samples than the original version.

< Bananaman, starring, Bill Oddie > < Bill Oddie, birth place, Lancashire > Bill Oddie, who was born in Lancashire, starred in Bananaman.
name[The Phoenix], eatType[pub], food[French], priceRange[more than £30], customer rating[5 out of 5] The Phoenix is a pub with French food. It has a customer rating of 5 out of 5 and a price range of more than £30.

Figure 1: Examples of data-text pairs in WebNLG (above) and E2E (below).

E2E is a dataset in the restaurant domain. In E2E, D is a name with an unordered set of 1~7 pairs (a, v), where a, v represents attribute and value, respectively. We select the cleaned version (Dusek et al., 2019), which fixes the data to eliminate inconsistencies between the data and the text.

Figure 1 shows examples of data-text pairs in WebNLG and E2E. See Appendix A for more details on the two datasets.

2.2 Models

We evaluate some smaller-sized, previously state-of-the-art language models in data-to-text generation, including two encoder-decoder language models T5-large (Raffel et al., 2020) and BART-large (Lewis et al., 2020), and one causal language model GPT-2-large (Radford et al., 2019). We also evaluate some advanced LLMs, including one encoder-decoder language model T5-11b (Chung et al., 2022), and two causal language models Mistral-7b (Jiang et al., 2023) and Llama-2-13b (Touvron et al., 2023). For data input, we use the linearization method (Kale and Rastogi, 2020). Following previous work in data-to-text generation (Mehta et al., 2022), we use fine-tuning method and treat the fine-tuning phase as the training phase. We use LoRA fine-tuning, which has better performance than full fine-tuning in data-to-text generation (Hu et al., 2022). See Appendix B for details about model size, input, training, and inference.

2.3 Metrics

We use PARENT (Dhingra et al., 2019) as the performance metric to measure the quality of the model's output. PARENT is a metric designed for data-to-text generation tasks, which considers the alignment of the output to both input data and reference texts. PARENT better reflects the semantic fidelity of the output and has a stronger correlation with human judgments than reference-only-based

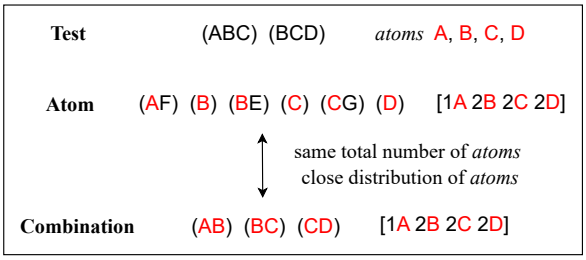

Figure 2: An example of datasets for the *systematicity* evaluation. Each pair of brackets denotes a sample and each letter (A~G) denotes a data unit.

metrics. Metrics other than the performance metric are described in the corresponding aspects.

3 Evaluation Method

In this section, we describe each aspect of SPOR. Each subsection corresponds to an aspect that includes the overview, how to construct the dataset, how to perform the evaluation, and the results and analysis. We regard triples and attribute-value pairs as data units for WebNLG and E2E, respectively. For all results reported, we run experiments three times with different random seeds and average the results to avoid contingency. Appendix C provides statistics of constructed datasets. Appendix D provides the qualitative analysis of evaluations, showing specific samples with model outputs.

3.1 Systematicity

The first aspect we evaluate is *systematicity* (Hupkes et al., 2020). *Systematicity* is a notion frequently used in tests of compositional generalization (Lake and Baroni, 2018; Kim and Linzen, 2020; Hupkes et al., 2020; Keysers et al., 2020), which refers to the ability to handle combinations of known elements that are not seen during training. In the data-to-text generation task, the elements refer to the data. Although a large corpus allows the model to see a large amount of data, the possible combinations of data are too numerous to be fully covered. In practical applications, the model will often see combinations of known data in the input that are not seen during training, so the ability to handle unseen combinations of data is important.

In the *systematicity* evaluation, by reconstructing the dataset, we allow the model to see all data units in the test set during training, but not any combination of them. In this case, the model needs *systematicity* to handle unseen combinations at test time. We use the model performance in this case as the *systematicity* metric. Based on the same test set, we also construct the case where the model can see combinations of data units to test whether the model's performance when it cannot see combinations is comparable to that when it can.

Algorithm 1 Construction of **Atom** and the test set

Input: original dataset S
Output: **Atom** (A), test set (T), **Blocked** (B)

```
T, A, B ← ∅
while S ≠ ∅ do
    x ← randomly selected sample in S
    S ← S − {x}
    R ← {y | y ∈ A ∪ S ∧ y ∉ B ∧ |y ∩ x| = 1}
    if x ⊆ ⋃R and max_{y∈A} |y ∩ x| ≤ 1 then
        T ← T ∪ {x}
        S ← S − R
        A ← A ∪ R
        B ← B ∪ {y | y ∈ S ∧ |y ∩ x| > 1}
    end if
end while
```

3.1.1 Dataset Construction

We construct one test set and two training sets **Atom (A)** and **Combination (C)**. Figure 2 illustrates the goal of our construction. We call the data units that appear in the test set *atoms*. Both **Atom** and **Combination** cover all *atoms*, and they have the same total number of *atoms* and close distribution of *atoms*. However, **Atom** does not contain any combination of *atoms*, but **Combination** does.

We use Algorithm 1 to construct **Atom** and the test set. We assume that the original dataset is the set S and each sample x in S is a set of data units. For a set x, we use $|x|$ to denote the number of data units it contains. For a set S containing sets, we use $\bigcup S$ to denote the union of the sets it contains, i.e., $\bigcup S$ is the set of all data units occurring in S.

Initially, both **Atom** and the test set are empty sets, and we set an initially empty auxiliary set **Blocked** to store samples containing combinations of *atoms*. Each time, we remove a sample x from S and check all samples in the current **Atom** and samples in S that are not in **Blocked** and include only one data unit in x. If these samples cover all data units in x, and **Atom** does not contain combinations of data units in x, then we:

- Add x to the test set.
- Remove samples in S that are not in **Blocked** and include only one data unit in x, and add them to **Atom**.
- Add samples in S that include more than one data unit in x to **Blocked**.

Algorithm 2 Construction of **Combination**

Input: **Atom** (A), test set (T), **Blocked** (B), divergence measure function $\mathcal{D}$, threshold r
Output: **Combination** (C)

```
C, A' ← A
B ← B − T
T ← ⋃T
define function F(x, G) as Σ_{y∈G} |x ∩ y|
define function V(x) as F(x ∩ T, A) − F(x ∩ T, C)
while B ≠ ∅ do
    x ← sample in S with maximum V(x)
    B ← B − {x}
    R ← ∅
    for all y ∈ A' in ascending order of V(y) do
        R' ← R ∪ {y}
        if |⋃R'| ≤ |x| and T ⊆ (C − R') ∪ {x} then
            R ← R'
        end if
    end for
    if |⋃R| = |x| and D(A, (C − R) ∪ {x}) ≤ r then
        C ← (C − R) ∪ {x}
        A' ← A' − R
    end if
end while
```

This process is repeated until S is empty. Under this construction method, **Atom** covers all *atoms* but does not contain any combination of *atoms*. The samples containing combinations of *atoms* are all in **Blocked**.

We then use Algorithm 2 to construct **Combination**. The core idea of Algorithm 2 is to replace samples in **Atom** with samples that have combinations of *atoms* to obtain **Combination**. We initialize **Combination** with **Atom**. For each sample x in **Blocked** but not in the test set, we try to replace a cluster of samples belonging to **Atom** with x in **Combination**, ensuring that **Combination** still covers all *atoms* and the total number of *atoms* remains the same after the replacement. Each replacement makes **Combination** have one more sample with combinations of *atoms*.

To ensure that the distributions of *atoms* in **Atom** and **Combination** are close, we perform the replacement only if the divergence of the two distributions after the replacement does not exceed a threshold r. Following Keysers et al. (2020), we measure the divergence using the Chernoff coefficient $\mathcal{D}(P, Q) = 1 - \sum_k p_k^{0.5} q_k^{0.5} \in [0, 1]$ (Chung et al., 1989) and set the threshold $r = 0.02$, where p_k and q_k denote the proportion of the atom k in datasets P and Q, respectively. Random replacements will cause the divergence to reach the threshold too early. To avoid this, we define $\mathcal{V}(x)$ as the subtraction of the total occurrences of *atoms* from x in **Atom** and **Combination**, and try to use samples with high $V(x)$ to replace samples with low $V(x)$. This replacement method controls the growth of divergence, allowing more replacements to occur and thus allowing **Combination** to contain more combinations of *atoms*.

	WebNLG		E2E	
	A	**C**	**A**	**C**
T5-large	66.14	66.54	49.19	52.76
BART-large	64.44	64.80	50.49	52.63
GPT-2-large	63.98	64.93	51.82	52.95
T5-11b	**68.93**	**69.07**	**53.78**	**54.72**
Mistral-7b	66.87	67.09	53.06	54.22
Llama-2-13b	65.87	66.18	51.28	53.35

Table 1: Performance of models on the two training sets for the *systematicity* evaluation.

3.1.2 Evaluation

We train the model on **Atom** and **Combination** respectively and test the performance of the two trained models on the test set. We evaluate *systematicity* of the model by the performance on **Atom**. We use the performance on **Combination** as a bound to analyze the *systematicity* level of the model.

3.1.3 Results and Analysis

Table 1 shows the results of the *systematicity* evaluation. On WebNLG, T5-11b performs best on **Atom**, showing the strongest *systematicity*. Among the LLMs, both t5-11b and Mistral-7b outperform all the smaller LMs on **Atom**, reflecting an improvement in *systematicity*. However, all models, including LLMs, show performance gaps on **Atom** and **Combination**, which indicates that when the model cannot see combinations during training, it is unable to handle combinations as well as when it can see. This reflects a deficiency in *systematicity* of the model. The results on E2E are similar, and the performance gaps on **Atom** and **Combination** on E2E are more significant than on WebNLG, which further confirms the deficiency in *systematicity* of the model. In conclusion, the LLMs overall show an improvement in *systematicity* compared to the smaller LMs but do not eliminate the deficiency in *systematicity* of the model.

3.2 Productivity

The second aspect we evaluate is *productivity* (Hupkes et al., 2020). *Productivity*, in the context of

	WebNLG						E2E					
	$N=3$		$N=4$		$N=5$		$N=3$		$N=4$		$N=5$	
	I	**V**	**I**	**V**	**I**	**V**	**I**	**V**	**I**	**V**	**I**	**V**
T5-large	68.24	69.82	68.32	70.11	68.36	68.71	61.27	62.91	64.31	64.91	63.81	64.11
BART-large	67.58	69.17	67.54	69.89	68.84	69.17	62.59	62.98	64.31	64.68	63.37	63.71
GPT-2-large	63.95	66.43	64.96	68.61	65.25	66.90	57.81	62.89	64.22	65.17	63.99	64.15
T5-11b	**70.86**	**71.10**	**70.03**	70.15	**69.57**	**69.83**	**62.79**	63.33	63.97	64.48	63.89	64.25
Mistral-7b	68.92	70.55	69.43	**71.09**	69.41	69.63	62.71	**64.53**	**65.13**	**66.06**	64.18	**64.82**
Llama-2-13b	68.77	69.78	69.55	70.30	69.08	69.23	61.18	62.76	64.46	64.86	**64.22**	64.40

Table 2: Performance of models trained on the two training sets with the number threshold $N \in \{3, 4, 5\}$ for the *productivity* evaluation.

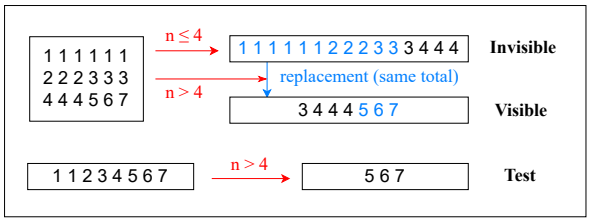

Figure 3: An example of datasets with threshold $N = 4$ for the *productivity* evaluation. Each number represents a sample with a corresponding number of data units.

compositionality, refers to the ability to extrapolate to longer sequences than those seen during training (Ontañón et al., 2022). Similar to *systematicity*, *productivity* is also a notion frequently used in tests of compositional generalization (Lake and Baroni, 2018; Hupkes et al., 2020; Ontañón et al., 2022). In the data-to-text generation task, *productivity* corresponds to the ability to handle a larger amount of data in the input than those seen during training. In practical applications, the amount of data contained in an input can be arbitrarily large, and it is impossible for a finite corpus to cover inputs with arbitrarily large amounts of data. The model will often encounter inputs with a larger amount of data than those seen during training and should have the ability to handle this situation.

In the *productivity* evaluation, we limit the number of data units of each sample during training, and test how the model performs when handling a larger amount of input data units than those seen during training. On the same test set, we also test the model trained with samples without the limit on the number of input data to see whether the model's performance with the limit is comparable to that without the limit.

3.2.1 Dataset Construction

We construct one test set and two training sets **Invisible (I)** and **Visible (V)**. We start by setting a number threshold N. We construct **Invisible** using all samples with no more than N data units. Similar to Algorithm 2, we replace the samples in **Invisible** with samples with more than N data units to obtain **Visible**, ensuring that the total numbers of data units in **Invisible** and **Visible** are the same and that the divergence of the distribution is less than the threshold $r = 0.02$ (using the same metric as in *systematicity*). We construct the test set using all samples with more than N data units in the original test. We ensure that any data unit in the test set is present in both **Invisible** and **Visible**. Our experiments try the number threshold $N \in \{3, 4, 5\}$. Figure 3 shows an example of dataset construction.

3.2.2 Evaluation

We train the model on **Invisible** and **Visible** respectively and test the performance of the two trained models on the test set. We evaluate *productivity* of the model by the performance on **Invisible**. We use the performance on **Visible** as a bound to analyze the *productivity* level of the model.

3.2.3 Results and Analysis

Table 2 shows the results of the *productivity* evaluation. On WebNLG, T5-11b performs best on **Invisible** with different thresholds. On E2E, the best performing model on **Invisible** with each threshold is one of the LLMs. The LLMs overall show stronger *productivity* than the smaller LMs. However, all models, including LLMs, show performance gaps on **Invisible** and **Visible** on both WebNLG and E2E, which indicates that when the model cannot see samples with the number of input data units exceeding the threshold during training, it is unable to handle such samples as well as when it can see. This reflects a deficiency in *productivity* of the model. The performance gaps of most models on **Invisible** and **Visible** are more significant for smaller thresholds, indicating that the deficiency

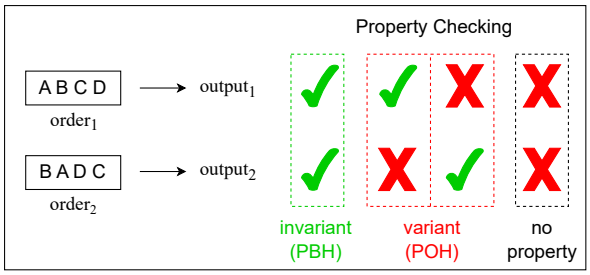

Figure 4: An illustration of the *order invariance* evaluation. Each letter (A~D) denotes a data unit.

in *productivity* is more pronounced when the maximum number of input data units within a sample seen during training decreases. In conclusion, the LLMs overall show an improvement in *productivity* compared to the smaller LMs but do not eliminate the deficiency in *productivity* of the model.

3.3 Order Invariance

The third aspect we evaluate is *order invariance*. This notion is previously studied by Wang et al. (2023), who finds that LLMs are sensitive to the order of options in multiple choice task. In the data-to-text generation task, *order invariance* refers to the ability that a model's output text maintains the fidelity and proper ordering of data when the same unordered set of data is input in different orders. Having *order invariance* means that the model can decompose the input into the set of data units and recombine them properly, regardless of the order of data units in the input, which reflects compositional generalization. In practical application scenarios, there are often cases where the data does not have a known linear order, and thus the model is required to have *order invariance* to ensure the fidelity and proper data ordering of the output texts under any data input order.

In the *order invariance* evaluation, for the same set of data units, we use two different input orders and then evaluate whether outputs maintain the fidelity and proper data ordering under both input orders. Further, we investigate the effect of the training process on *order invariance*. We construct a training set in which data units are arranged in the input in the same order as they appear in the text. We evaluate whether using such a training set makes the model more inclined to arrange data units in the text according to input order and whether it affects the *order invariance* of the model.

3.3.1 Dataset Construction

We design a search algorithm to find the occurrence position of data units in the text (see Appendix E for details). For each data-text pair in the original training set, we arrange the data units in the input according to their occurrence in the text, forming the training set **Match (M)**. Correspondingly, **Original (O)** refers to the original training set.

3.3.2 Evaluation

We train the model on **Original**. For each sample of the original test set, we randomize the order of the input data units to form two different inputs. We determine the set of data units contained in the output and the order of the data units, and then consider two properties: (1) The output is considered to have **fidelity** if the set of data units exactly matches the input. (2) The output is considered to have **proper data ordering** if the order of the data units satisfies $k > 0$ with the order of at least one reference text, where $k \in [-1, 1]$ is the Kendall coefficient (Abdi, 2007), which measures the correlation of two orders. For each of the two properties, we evaluate the proportion of both outputs having the property (PBH) and the proportion of only one output having the property (POH). A model with high *order invariance* on the property should have a higher PBH. Relatively, POH reflects the order variance of the model. Figure 4 shows an illustration of the evaluation.

3.3.3 Additional Tests

To investigate the effect of the order consistency of data units in input and output in the training set, we train the model on **Match** and perform additional tests. Besides fidelity and proper data ordering in the evaluation, we also perform the following tests on the models trained on **Original** and **Match**. First, for the input and model output of the original test set, we determine the order of data units in the output, and then calculate its correlation with the input order of the data units (CWIO). We use the Kendall coefficient to measure the correlation. A higher correlation means that the model is more inclined to arrange data units in the text according to input order. Second, we test the performance of the model on the original test set to see the effect of different training sets on the performance.

3.3.4 Results and Analysis

Table 3 shows the results of the *order invariance* evaluation. When trained on **Original**, on fidelity,

	WebNLG						E2E					
	Fidelity		Ordering		CWIO	PERF	Fidelity		Ordering		CWIO	PERF
	PBH	POH	PBH	POH			PBH	POH	PBH	POH		
T5-large	97.56	1.67	87.15	6.84	+0.13	67.95	91.39	6.80	77.22	10.15	+0.51	63.07
BART-large	97.65	0.94	88.69	3.98	+0.10	66.96	98.05	0.90	82.26	3.59	+0.52	62.58
GPT-2-large	90.55	6.86	82.64	9.90	+0.11	67.64	74.37	19.22	68.08	10.99	+0.50	62.60
T5-11b	**99.10**	0.64	**89.05**	4.53	+0.10	68.47	**99.12**	0.60	**82.75**	3.68	+0.57	62.56
Mistral-7b	96.49	2.67	86.29	7.80	+0.11	**68.69**	96.49	3.08	82.28	4.46	+0.42	**63.91**
Llama-2-13b	96.69	2.33	87.28	6.86	+0.09	68.07	96.88	2.75	78.50	7.54	+0.46	62.81
T5-large	94.63	4.55	53.56	39.98	+0.81	65.53	98.36	1.62	37.28	43.95	+0.95	55.74
BART-large	92.45	5.94	54.78	38.14	+0.76	64.12	97.25	2.67	37.58	43.91	+0.95	56.98
GPT-2-large	81.06	15.80	54.84	37.74	+0.76	65.07	85.34	11.77	**38.96**	42.31	+0.95	56.52
T5-11b	**96.58**	3.01	54.93	38.81	+0.76	66.04	**99.28**	0.72	37.05	43.56	+0.94	56.21
Mistral-7b	94.65	4.64	**54.95**	38.46	+0.79	**66.29**	97.95	1.93	37.24	43.40	+0.94	**57.37**
Llama-2-13b	91.44	7.38	54.59	39.00	+0.78	65.66	97.97	1.99	37.38	43.68	+0.95	56.52

Table 3: Results of models trained on **Original** (above) and **Match** (below) for the *order invariance* evaluation. CWIO refers to the correlation with the input order. PERF refers to the performance on the original test set.

T5-11b has the highest PBH on both WebNLG and E2E, showing the strongest *order invariance*. As a smaller LM, BART-large has the second highest PBH, which is higher than LLMs Mistral-7b and Llama-2-13b. From the POH we can see that all models show *order variant* cases on fidelity, i.e., for two input orders of the same set of data units, a model may show fidelity in one order but not in the other. On proper data ordering, the results are similar to fidelity and show a larger proportion of *order variant* cases. This means that for two input orders of data units, the two outputs of the model may differ in their data ordering, where one is proper and the other is not. Overall, the models are deficient in *order invariance* on both fidelity and proper data ordering.

Compared to **Original**, when trained on **Match**, the CWIO of the model is significantly higher, indicating that the model is more inclined to arrange the data units in the text according to input order. This inclination about ordering leads to a decrease in *order invariance* on proper data ordering. An unexpected finding is that the inclination also affects *order invariance* on fidelity, overall leading to a decrease on WebNLG and an increase on E2E (see Appendix D.3 for the discussion). The performance of the model trained on **Match** is significantly lower than on **Original**, indicating that high order consistency of data units in input and output during training negatively affects the performance when the order of input data units is arbitrary.

3.4 Rule Learnability

Models with high compositionality have the "willingness to prefer rules over memorization" (Hupkes et al., 2020), i.e., they tend to apply observed rules to recombine elements rather than simply memorizing combinations of elements. Based on this understanding, we propose the last aspect of the evaluation, *rule learnability*, which refers to the ability to learn rules from training and apply them during testing. Our evaluation focuses on the *copy rule* (Gehrmann et al., 2018) in data-to-text generation, which refers to the rule that certain information involved in the text (e.g., entities, numeric values) should be copied directly from the data to ensure the fidelity of the text.

In the *rule learnability* evaluation, we replace some entities or numeric values that should be copied with phrases that hide information, and then check whether the model correctly applies the *copy rule*. A correct copy should not have *omissions* of phrases that hide information or *hallucinations* of outputting entities and numeric values that have been hidden. If the model only memorizes specific mappings that conform to the *copy rule* during training, rather than actually learning the *copy rule*, then it will not be able to correctly apply the *copy rule* to the phrases that hide information.

3.4.1 Dataset Construction

On WebNLG, the *copy rule* is mainly applied to entities. For each sample in the original WebNLG test set, we find the entities that act as subjects and are copied in every reference text, and replace these entities in the input with "Entity i" (i denotes the entity's label, which is used to distinguish between different entities). On E2E, the *copy rule* is mainly applied to values, and we focus on numeric values. Similar to WebNLG, we replace the numeric value

< Entity 1, starring, Entity 2 >
< Entity 2, birth place, Lancashire >

Entity 1: Bananaman / Entity 2: Bill Oddie

---

name [The Phoenix], eatType [pub], food[French], priceRange [more than Value A], customerRating [Value B out of 5]

Value A: £30 / Value B: 5

Figure 5: An example of dataset construction for the *rule learnability* evaluation.

with "Value i". If a value contains more than one numeric value, only the first one will be replaced. Figure 5 shows an example of dataset construction.

3.4.2 Evaluation

We train the model on the original training set and then check the output of the model on the replaced inputs. The result of checking each sample can be represented as **(a, b)**, where **a** $\in$ **{0, 1}** indicates whether all phrases that hide information are copied correctly (using fuzzy matching, see Appendix F for details), and **b** $\in$ **{0, 1}** indicates whether the hidden entities or numeric values appear. In the representation of the result, **a = 0** implies *omissions* and **b = 1** implies *hallucinations*. Of the four possible results, only **(1, 0)** indicates that the *copy rule* is correctly applied. We count the proportions of the four cases and evaluate the *rule learnability* by the proportion of **(1, 0)**.

3.4.3 Results and Analysis

Table 4 shows the results of *rule learnability* evaluation. On WebNLG, all models apply the *copy rule* less than 90% correctly. The errors are mainly concentrated on the **(0, 0)** case. This case indicates that the model does not have the *hallucinations* of outputting entities that have been hidden, but it has *omissions* of phrases that hide information. Among all the models, T5-large and BART-large have relatively high correct rates. The LLMs do not show higher correct rates compared to the smaller LMs. All LLMs have a correct rate of less than 80%.

The results shown on E2E are different. On E2E, the LLMs have high correct rates and outperform the smaller LMs. Among the LLMs, both Mistral-7b and Llama-2-13b are almost completely correct. Among the smaller LMs, BART-large and GPT-2-large show very low correct rates. Their proportions of **(0, 1)** are both high, indicating that there

	(0, 0)	**(0, 1)**	**(1, 0)**	**(1, 1)**
T5-large	10.16	0.31	**89.32**	0.21
BART-large	10.64	1.30	87.59	0.48
GPT-2-large	19.43	1.69	78.44	0.43
T5-11b	17.35	3.02	79.62	0.02
Mistral-7b	19.08	1.40	79.04	0.48
Llama-2-13b	21.15	0.45	78.11	0.29
T5-large	2.64	1.44	95.93	0.00
BART-large	13.17	57.57	29.26	0.00
GPT-2-large	15.28	48.19	36.06	0.46
T5-11b	0.05	2.38	97.57	0.00
Mistral-7b	0.65	0.00	**99.35**	0.00
Llama-2-13b	0.86	0.00	99.14	0.00

Table 4: Results of the *rule learnability* evaluation on WebNLG (above) and E2E (below). Each column represents the proportion of the corresponding case.

are serious *hallucinations* of outputting numeric values that have been hidden. When outputting these numeric values, the model tends not to output the corresponding phrases that information, resulting in *omissions*. Their proportions of **(0, 0)** also indicate the presence of simple *omissions* unrelated to the *hallucinations*.

In summary, the results show that all models, including LLMs, are unable to achieve high correct copy rates on both WebNLG and E2E, and that *omissions* and *hallucinations* are prevalent in the models. This indicates that for *copy rules* in data-to-text generation, the models are deficient in *rule learnability* and need further improvement.

4 Conclusions

In this work, we propose SPOR, a comprehensive and practical evaluation method for compositional generalization in data-to-text generation, which includes four aspects of manifestations: *systematicity*, *productivity*, *order invariance*, and *rule learnability*. We demonstrate on WebNLG and E2E how SPOR enables evaluations without additional manual annotations based on existing datasets. We evaluate some existing language models, including LLMs. We find that the models are deficient in various aspects of compositional generalization in data-to-text generation and need further improvement. Our work supports comprehensive research on different manifestations of compositional generalization in data-to-text generation and provides a framework for identifying and evaluating improvements in this ability of language models.

Limitations

A limitation of our work is the limited size of the models evaluated. Although we include some LLMs in our evaluation, due to the need for fine-tuning with limited resources, the size of the LLMs does not exceed 13b. Resource constraints make it difficult to apply fine-tuning methods on larger LMs, and there is currently no effective method for directly applying larger LMs to data-to-text generation. One possible method is in-context learning, which performs inference directly but adds a prefix to the input that demonstrates a small number of samples for the model to learn. In the in-context learning style, the training phase of compositional generalization corresponds to the sample demonstration in the prefix, and the evaluation needs to consider the method of sample demonstration selection. We will continue to follow the progress of applying larger LMs to data-to-text generation and explore evaluation methods for compositional generalization in data-to-text generation of larger LMs.

Ethics Statement

The datasets and models we use are open-source and we use them for scientific research purposes only. The datasets we construct will also be open source for scientific research purposes. The datasets we use and construct do not contain any information that names or uniquely identifies individual people or offensive content.

Since we use the realistic dataset WebNLG, we are particularly concerned with data faithfulness, i.e., all data in the reconstructed evaluation dataset must not show information that contradicts the original realistic dataset, which may be inconsistent with the real world and may be harmful. In the *systematicity*, *productivity*, and *order invariance* evaluations, we do not modify the information in any triple. In the *rule learnability* evaluation, we only hide the information, and no new information is generated. Therefore, the data used in the evaluation do not contain information that contradicts the original realistic dataset.

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

References

Hervé Abdi. 2007. The kendall rank correlation coefficient. *Encyclopedia of Measurement and Statistics. Sage, Thousand Oaks, CA*, pages 508–510.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

J.K Chung, P.L Kannappan, C.T Ng, and P.K Sahoo. 1989. Measures of distance between probability distributions. *Journal of Mathematical Analysis and Applications*, 138(1):280–292.

Bhuwan Dhingra, Manaal Faruqui, Ankur P. Parikh, Ming-Wei Chang, Dipanjan Das, and William W. Cohen. 2019. Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4884–4895. Association for Computational Linguistics.

Ondrej Dusek, David M. Howcroft, and Verena Rieser. 2019. Semantic noise matters for neural natural language generation. In *Proceedings of the 12th International Conference on Natural Language Generation, INLG 2019, Tokyo, Japan, October 29 - November 1, 2019*, pages 421–426. Association for Computational Linguistics.

Thiago Castro Ferreira, Claire Gardent, Nikolai Ilinykh, Chris van Der Lee, Simon Mille, Diego Moussallem, and Anastasia Shimorina. 2020. The 2020 Bilingual, Bi-Directional WebNLG+ Shared Task Overview and Evaluation Results (WebNLG+ 2020). In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, Dublin/Virtual, Ireland.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The webnlg challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation, INLG 2017, Santiago de Compostela, Spain, September 4-7, 2017*, pages 124–133. Association for Computational Linguistics.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *J. Artif. Intell. Res.*, 61:65–170.

Sebastian Gehrmann, Falcon Z. Dai, Henry Elder, and Alexander M. Rush. 2018. End-to-end content and plan selection for data-to-text generation. In *Proceedings of the 11th International Conference on Natural Language Generation, Tilburg University, The Netherlands, November 5-8, 2018*, pages 46–56. Association for Computational Linguistics.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2020. Compositionality decomposed: How do neural networks generalise? (extended abstract). In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 5065–5069. ijcai.org.

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.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *CoRR*, abs/2310.06825.

Mihir Kale and Abhinav Rastogi. 2020. Text-to-text pre-training for data-to-text tasks. In *Proceedings of the 13th International Conference on Natural Language Generation, INLG 2020, Dublin, Ireland, December 15-18, 2020*, pages 97–102. Association for Computational Linguistics.

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.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

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.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics.

Sanket Vaibhav Mehta, Jinfeng Rao, Yi Tay, Mihir Kale, Ankur Parikh, and Emma Strubell. 2022. Improving compositional generalization with self-training for data-to-text generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 4205–4219. Association for Computational Linguistics.

Jekaterina Novikova, Ondrej Dusek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue, Saarbrücken, Germany, August 15-17, 2017*, pages 201–206. Association for Computational Linguistics.

Santiago Ontañón, Joshua Ainslie, Zachary Fisher, and Vaclav Cvicek. 2022. Making transformers solve compositional tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 3591–3607. Association for Computational Linguistics.

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.

Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2020. Investigating pretrained language models for graph-to-text generation. *CoRR*, abs/2007.08426.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa,

Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Peiyi Wang, Lei Li, Liang Chen, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023. Large language models are not fair evaluators. *CoRR*, abs/2305.17926.

Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2253–2263. Association for Computational Linguistics.

A Original Dataset Details

In the original WebNLG dataset, 10 domains are present in the training set and can be used in the evaluation. We select the latest version, WebNLG+, which increases the number of available domains to 16 and contains more samples. For the samples used for testing, we retain only samples in which all data units appear in the training set. After processing, WebNLG+ contains 3,873 distinct triples, 13,211 samples for training, and 2,179 samples for testing.

The E2E dataset contains 7 attributes and 45 distinct attribute-value pairs. The original E2E dataset has some inconsistencies between the data and the text, which is partially fixed in the clean version. We perform further filtering based on the clean version, retaining only samples in which all input values have matches in the text. After processing, E2E contains 6,735 samples for training and 1,635 samples for testing.

B Model Details

The models we evaluate include T5-large (738M), BART-large (406M), GPT2-large (774M), T5-11b, Mistral-7b, and Llama-2-13b. All models are downloaded from HuggingFace, and training and inference are based on the transformers library. Each item in our experiment is done on a single NVIDIA A800 80G GPU.

For model input, we use the linearization method (Ribeiro et al., 2020; Kale and Rastogi, 2020). For WebNLG, we add the special identifiers <head>, <relation>, and <tail> before the subject, predicate, and object of each triple, and then linearly concatenate all triples to form the input. For E2E, We form the input by linearly concatenating each attribute-value pair in the form of "attribute[value]". Following Ribeiro et al. (2020), for WebNLG, we add a prefix “translate from Triple to Text:” before the input. Similarly, we use the prefix "translate from MR to Text:" for E2E.

For model training, the optimizer is Adam (Kingma and Ba, 2015). The learning rate is 1e-4, and the batch size is 6. We train the model for 10 epochs. For model inference, the beam width is 5.

For *systematicity* and *productivity* evaluations, we report the best results on the test set among all checkpoints. For *order invariance* and *rule learnability* evaluations, we report the results of the checkpoint that has the best performance on the original test set.

	WebNLG		E2E	
	A	**C**	**A**	**C**
# samples	4,717	3,256	3,351	1,390
# data units	9,636	8,267	13,311	7,043
# *atoms*	5,281	5,281	3,298	3,298
# combination pairs	0	1,969	0	2,670

Table 5: Some statistics about the training sets for the *systematicity* evaluation.

		1	**2**	**3**	**4**	**5**	**6**	**7**
$N=3$	**I**	249	193	239	0	0	0	0
	V	19	18	9	56	57	44	71
$N=4$	**I**	249	193	239	260	0	0	0
	V	0	17	35	25	148	99	117
$N=5$	**I**	249	193	239	260	227	0	0
	V	9	52	128	99	34	203	178
$N=3$	**I**	86	592	1,480	0	0	0	0
	V	0	66	633	0	414	148	103
$N=4$	**I**	86	592	1,480	2,151	0	0	0
	V	0	80	1,227	1,601	4	543	113
$N=5$	**I**	86	592	1,480	2,151	1,612	0	0
	V	0	389	1,400	2,029	1,435	219	113

Table 6: Number of samples in training sets for the *productivity* evaluation with each number (from 1 to 7) of data units in WebNLG (above) and E2E (below).

C Constructed Dataset Statistics

C.1 Systematicity

Table 5 shows the statistics about the training sets for the *systematicity* evaluation. The size of the test set for the *systematicity* evaluation is related to the number of distinct data units contained in the original dataset. For a dataset like E2E with a small number of distinct data units, it is more difficult to construct a large test set. To maximize the size of the test set, we randomly pick x in Algorithm 1 among those x that have the largest $|x|$ and perform multiple random constructions, taking the one with the largest test set size. The test set contains 2,360 samples on WebNLG and 156 samples on E2E.

C.2 Productivity

Each sample in WebNLG contains 1 to 7 data units. Samples with 6 and 7 data units only cover four domains: *Astronaut*, *Monument*, *University*, and *Company*. To avoid inconsistent domain distributions of training sets, we only use samples from these four domains to construct the datasets for the *productivity* evaluation on WebNLG. Table 6 shows the number of samples in training sets with each number of input triples. For $N \in \{3, 4, 5\}$, the test set of WebNLG contains 219 / 153 / 99 samples, and the test set of E2E contains 1,314 / 1,002 / 477 samples.

C.3 Order Invariance

For the *order invariance* evaluation on fidelity and proper data ordering, we remove samples with only one data unit and samples where the order of the data units in the text cannot be determined. The test set of WebNLG contains 1,559 samples, and the test set of E2E contains 1,623 samples.

C.4 Rule Learnability

For the *rule learnability* evaluation, on WebNLG, we retain only samples in which there is at least one entity that satisfies the replacement condition. The final test set contains 1,614 samples. On E2E, since the training data guarantees copies of values, we can construct samples without reference texts to cover more combinations. We enumerate the values of 6 attributes (except the attribute *near*, which is similar to *name*) and ensure that at least one value contains the numeric value, resulting in 1,440 samples. When a phrase that hides information corresponds to more than one value, the occurrence of any one of these values is considered a hallucination.

D Qualitative Analysis of Evaluations

Table 7 ~ 10 show some specific samples with model outputs in each aspect of the evaluation.

D.1 Systematicity

Table 7 shows samples from Llama-2-13b in the *systematicity* evaluation. On WebNLG, the issue on fidelity is the omission of data units, and the issue on fluency is the stiff expression (the model repeatedly enumerates data units by applying the same pattern, and lacks fluency in articulation). On E2E, the issues center on fluency similar to those shown on WebNLG. The stiff expression can be attributed to the difficulty of models trained on the **Atom** in handling unseen combinations.

D.2 Productivity

Table 8 shows samples from Llama-2-13b in the *productivity* evaluation. The issues center on fidelity. In addition to the omissions present on WebNLG and E2E, hallucinations are found on E2E. The fidelity issue can be attributed to the difficulty of models trained on **Invisible** in handling a larger number of input data units.

D.3 Order Invariance

Table 9 shows samples from Llama-2-13b in the *order invariance* evaluation. On WebNLG, for the model trained on **Original**, both outputs have fidelity. However, the data ordering of Output 1 is improper, while that of Output 2 is proper (for < Trance music, stylistic origin, Pop music >, it should be next to < Andrew Rayel, genre, Trance music >, not isolated at the end). For the model trained on **Match**, the order of the output data units is consistent with the input order. When the input order is not the proper data ordering, the model may try to apply complex grammar on an unnatural order of data units, which results in some data units not being generated as demonstrated in the sample. On E2E, the two outputs on **Original** are consistent in ordering but vary in fidelity. The two outputs on **Match** have exactly the same data ordering as the inputs, resulting in a stiff expression. However, from the experimental results, such a form of output on improves *order invariance* on fidelity on E2E. We hypothesize that due to the relatively simple grammar of E2E, this form does not lead to omissions as on WebNLG, and the model may be easier to maintain fidelity because there is no need to rearrange the data units.

D.4 Rule Learnability

Table 10 shows samples of error cases in the *rule learnability* evaluation. The most frequent error case on WebNLG is **(0, 0)**. In the sample of **(0, 0)** on WebNLG, there is no hallucination in the output but "Entity 1" is omitted, resulting in a factual error. The other two samples demonstrate cases with hallucinations. On a realistic dataset like WebNLG, the hallucination may be a correct inference based on known information but does not satisfy the requirement for fidelity in data-to-text generation. The poorer performing models on E2E, such as BART-large / GPT-2-large, have a large proportion of **(0, 1)** cases. In the sample of **(0, 1)** on E2E, the model outputs "5 out of 5" instead of "Value B of 5", which is a hallucination with the omission. On E2E, known information is irrelevant to the hidden numeric value, so the hallucination is unfounded. The sample of **(0, 0)** demonstrates an omission unrelated to the hallucination, which is the only case of errors for the better performing models on E2E such as Mistral-7b / Llama-2-13b.

E Search Algorithm for Order-Invariance Evaluation

For each data-text pair in WebNLG, we first locate where the entities in the data appear in the text.

Although most of the entities appear unchanged in the text, variations still exist, such as token discontinuities or token distortions. However, discontinuous tokens are not too far away from each other, and the degree of token distortion is not too large. Therefore, we use the following algorithm for localization:

1. We first slice the entity into tokens, and for each token t, find the set of candidate-matching tokens in the text with the smallest edit distance from t and no more than min (2, length of t).

2. Keep all non-empty candidate sets, and then use depth-first search to select a position in each candidate set such that the final variance of all positions is minimized as the token position representation of the entity. If there are multiple minimum variance representations, then all are retained.

3. The entities are sorted by the number of position representations retained from smallest to largest, and then one representation is selected for each entity and the smallest position number in the representation is used to represent that entity. We require that the position number representing an entity cannot appear in the representations of other entities, and if it cannot be satisfied, then the position number of this entity is set to a large boundary value (the percentage of such cases is about 1.6%).

After determining the position number of each entity, we determine the order of triples. We consider the set of triples as an undirected graph, and each triple represents a connected edge between the subject and the object. For each triple, if the degree of the subject and object are different, we take the position of the entity with the smaller degree to represent the position of the triple, otherwise, we take the larger of the two entity positions to represent the position of the triple. According to the position number of triple, we get the order of triple. The order relationship between triples with the same position number follows the input.

On E2E, since the training data guarantees copies of values, we use strict matching to localize the values.

F Fuzzy Matching for Rule-Learnability Evaluation

In the *rule learnability* evaluation, for the checking of copying phrases that hide information, we find that there are cases where the model does not perform strict copying, but semantically completes the copying, which should also be considered correct. Therefore, in addition to strictly correct copying, the following cases are also considered as correct copying:

- Case is ignored. For example, "entity 1" and "value b" are considered correct.
- Numeric symbols can be changed to ordinal numbers. For example, "1st Entity" is considered correct.
- If the symbol is copied, it is allowed not to copy "Entity" or "Value". For example, "Its customer rating is B out of 5." is considered correct.

The fuzzy matching covers most cases of semantically completed copies, which makes the checking of copying more accurate.

Dataset	Sample
WebNLG	**Input:** < Ayam penyet, region, Malaysia >, < Ayam penyet, country, Java >, < Ayam penyet, ingredient, Fried chicken >, < Ayam penyet, main ingredient, Squeezed or smashed fried chicken served with sambal >, < Ayam penyet, serving temperature, Hot > **Combination:** Ayam penyet is a dish from Malaysia and Java. It includes fried chicken which is squeezed or smashed and served with sambal. It should be served hot. **Atom:** Ayam penyet is a dish from the region of Malaysia and Java. It contains fried chicken and is served hot. **Performance:** (73.90, 56.98) **Issue:** Omission of data units
	Input: < Spain, leader, Felipe VI of Spain >, < Spain, language, Spanish language >, < Spain, currency, Euro >, < Ajoblanco, country, Spain >, < Spain, demonym, Spaniards > **Combination:** Ajoblanco is a dish from Spain, where the currency is the euro and the language is Spanish. The country is led by Felipe VI and the people who live there are called Spaniards. **Atom:** Ajoblanco is a food found in Spain, where Felipe VI of Spain is the leader, Spanish is spoken, the Euro is the currency and Spaniards live. **Performance:** (62.93, 8.43) **Issue:** Stiff expression
E2E	**Input:** name[Wildwood], eat type[restaurant], food[French], area[riverside], near[Raja Indian Cuisine] **Combination:** Wildwood is a French restaurant near Raja Indian Cuisine in the riverside area. **Atom:** Wildwood is a restaurant providing French food It is located in the riverside. It is near Raja Indian Cuisine. **Performance:** (59.89, 24.93) **Issue:** Stiff expression / Missing punctuation

Table 7: Samples from Llama-2-13b in the *systematicity* evaluation. **Combination** and **Atom** refer to the output of the same model trained on the corresponding dataset. **Performance** refers to the performance scores of the two outputs.

Dataset	Sample
WebNLG	**Input:** < Elliot See, alma mater, University of Texas at Austin >, < University of Texas at Austin, affiliation, University of Texas System >, < Elliot See, birth place, Dallas >, < Elliot See, death place, St. Louis >, < Elliot See, status, Deceased > **Visible:** Elliot See was born in Dallas and died in St. Louis. He attended the University of Texas at Austin, which is affiliated to the University of Texas system. **Invisible:** Elliot See was born in Dallas, Texas, graduated from the University of Texas at Austin and died in St Louis. **Performance:** (82.85, 70.91) **Issue:** Omission of data units
E2E	**Input:** name[Browns Cambridge], eat type[coffee shop], food[Chinese], customer rating[average], area[city centre], area[riverside], family friendly[no], near[Crowne Plaza Hotel] **Visible:** Browns Cambridge is a Chinese coffee shop located in the city centre near the Crowne Plaza Hotel. It is not family friendly and has an average customer rating. It is located in the riverside area. **Invisible:** Browns Cambridge is a coffee shop providing Chinese food It is located in the city centre. Its customer rating is average. It is not family friendly. It is near Crowne Plaza Hotel. **Performance:** (62.84, 53.84) **Issue:** Omission of data units
	Input: name[Clowns], eat type[coffee shop], food[English], customer rating[5 out of 5], near[Clare Hall] **Visible:** Clowns is a coffee shop near Clare Hall. It serves English food and has a customer rating of 5 out of 5. **Invisible:** Clowns is a coffee shop providing English food in the higher price range. Its customer rating is 5 out of 5. It is near Clare Hall. **Performance:** (64.59, 23.47) **Issue:** Hallucination

Table 8: Samples from Llama-2-13b in the *productivity* evaluation with the threshold $N = 3$. **Visible** and **Invisible** refer to the output of the same model trained on the corresponding dataset. **Performance** refers to the performance scores of the two outputs.

Dataset	Sample
WebNLG	**Input:** < Trance music, stylistic origin, Pop music >, < Andrew Rayel, genre, Trance music >, < Jwaydan Moyine, associated band/associated musical artist, John Digweed >, < Andrew Rayel, associated band/associated musical artist, Jwaydan Moyine >
	Input Order 1: < Trance music, stylistic origin, Pop music > → < Andrew Rayel, genre, Trance music > → < Jwaydan Moyine, associated band/associated musical artist, John Digweed > → < Andrew Rayel, associated band/associated musical artist, Jwaydan Moyine > **Original 1:** Andrew Rayel is a Trance musician who is associated with the musical artist Jwaydan Moyine. Moyine is associated with the musical artist John Digweed. Trance music originated from pop music. **Match 1:** Trance music originated from pop music and is performed by artists such as Andrew Rayel and John Digweed.
	Input Order 2: < Jwaydan Moyine, associated band/associated musical artist, John Digweed > → < Trance music, stylistic origin, Pop music > → < Andrew Rayel, associated band/associated musical artist, Jwaydan Moyine > → < Andrew Rayel, genre, Trance music > **Original 2:** Andrew Rayel's musical genre is Trance which has its origins in Pop music. He is associated with the musical artist Jwaydan Moyine who is associated with the musical artist John Digweed. **Match 2:** Jwaydan Moyine is associated with the musical artist John Digweed and with Trance musician Andrew Rayel.
E2E	**Input:** name[The Eagle], family friendly[no], price range[cheap], area[city centre], near[Burger King], customer rating[average], food[Chinese], eat type[coffee shop]
	Input Order 1: family friendly[no] → price range[cheap] → area[city centre] → near[Burger King] → customer rating[average] → food[Chinese] → eat type[coffee shop] **Original 1:** The Eagle is a cheap Chinese coffee shop in the city centre near Burger King. It has an average customer rating. **Match 1:** The Eagle is not family friendly, but is cheap. It is located in the city centre near Burger King. It has an average customer rating and serves Chinese food. It is a coffee shop.
	Order 2: food[Chinese] → customer rating[average] → near[Burger King] → price range[cheap] → area[city centre] → family friendly[no] → eat type[coffee shop] **Original 2:** The Eagle is a cheap Chinese coffee shop in the city centre near Burger King. It has an average customer rating and is not family friendly. **Match 2:** The Eagle serves Chinese food. It has an average customer rating, is near Burger King, is cheap, is in the city centre, is not family friendly, and is a coffee shop.

Table 9: Samples from Llama-2-13b in the *order invariance* evaluation. **Original** and **Match** refer to the output of the same model trained on the corresponding dataset. We mark the position of data units in the output in red.

Dataset	Case	Sample
WebNLG	**(0, 0)**	**Input:** < Antares (rocket), manufacturer, Yuzhnoye Design Office >, < Entity 1, launch site, Vandenberg Air Force Base >, < Antares (rocket), comparable, Entity 1 >, <Antares (rocket), launch site, Mid-Atlantic Regional Spaceport >, < Entity 1, country origin, United States > **Hidden:** (Entity 1, Delta II) **Output:** The Antares rocket, manufactured by the Yuzhnoye Design Office, was launched from the Mid-Atlantic Regional Spaceport and the Vandenberg Air Force Base in the United States.
	(0, 1)	**Input:** < Entity 1, academic discipline, Mathematics >, < Entity 1, first publication year, 1878 >, < Entity 1, abbreviation, Am. J. Math >, < Entity 1, issn number, 1080-6377 > **Hidden:** (Entity 1, American Journal of Mathematics) **Output:** The American Journal of Mathematics (abbreviated to Am. J. Math.) has the ISSN number 1080-6377.
	(1, 1)	**Input:** < Entity 2, leader, Mulatu Teshome >, < Entity 2, leader, Hailemariam Desalegn >, < Addis Ababa, is part of, Addis Ababa Stadium >, < Entity 1, location, Addis Ababa >, < Addis Ababa, country, Entity 2 > **Hidden:** (Entity 1, Addis Ababa City Hall), (Entity 2, Ethiopia) **Output:** Addis Ababa Stadium is located in Addis Ababa, Ethiopia. Entity 1 is located in Addis Ababa. Mulatu Teshome and Hailemariam Desalegn are leaders of Entity 2.
E2E	**(0, 0)**	**Input:** name[The Twenty Two], price range[less than Value A], area[city centre], eat type[pub], food[Italian], near[Café Rouge], family friendly[no], customer rating[Value B out of 5] **Hidden:** (Value A, £20), (Value B, 1/3/5) **Output:** The Twenty Two is a pub located in the city centre near Café Rouge. It serves Italian food and has a customer rating of Value B out of 5. It is not family friendly.
	(0, 1)	**Input:** name[Loch Fyne], eat type[coffee shop], customer rating[Value B out of 5], area[city centre], price range[more than Value A], food[English], near[The Sorrento] **Replacement:** (Value A, £30), (Value B, 1/3/5) **Output:** Loch Fyne is a coffee shop near The Sorrento in the city centre. It has a customer rating of 5 out of 5 and serves English food at a price range of more than Value A.
	(1, 1)	**Input:** name[Alimentum], area[city centre], near[The Six Bells], eat type[pub], food[Chinese], price range[more than Value A] **Replacement:** (Value A, £30) **Output:** more than Value A, Alimentum is a pub that provides Chinese food in the more than £30 price range. It is located in the city centre.

Table 10: Samples from Llama-2-13b / T5-11b / GPT-2-large that performs incorrect copies in the *rule learnability* evaluation. We mark copies of phrases that hide information in blue and occurrences of hidden entities or numerical values in red.