The Validity of Evaluation Results: Assessing Concurrence Across Compositionality Benchmarks

Kaiser Sun **Adina Williams** **Dieuwke Hupkes**
Meta AI
hsun74@cs.jhu.edu
{adinawilliams, dieuwkehupkes}@meta.com

Abstract

NLP models have progressed drastically in recent years, according to numerous datasets proposed to evaluate performance. Questions remain, however, about how particular dataset design choices may impact the conclusions we draw about model capabilities. In this work, we investigate this question in the domain of compositional generalization. We examine the performance of six modeling approaches across 4 datasets, split according to 8 compositional splitting strategies, ranking models by 18 compositional generalization splits in total. Our results show that: i) the datasets, although all designed to evaluate compositional generalization, rank modeling approaches differently; ii) datasets generated by humans align better with each other than they with synthetic datasets, or than synthetic datasets among themselves; iii) generally, whether datasets are sampled from the same source is more predictive of the resulting model ranking than whether they maintain the same interpretation of compositionality; and iv) which lexical items are used in the data can strongly impact conclusions. Overall, our results demonstrate that much work remains to be done when it comes to assessing whether popular evaluation datasets measure what they intend to measure, and suggests that elucidating more rigorous standards for establishing the validity of evaluation sets could benefit the field.[1]

1 Introduction

Over the past few years, NLP has made astonishing progress on almost all language-related tasks proposed by the community. Concurrently, a plethora of benchmark datasets has emerged for evaluating the skills of NLP models and exposing their strengths and weaknesses (Chowdhery et al. 2022, *inter alia*). These datasets focus on a variety of

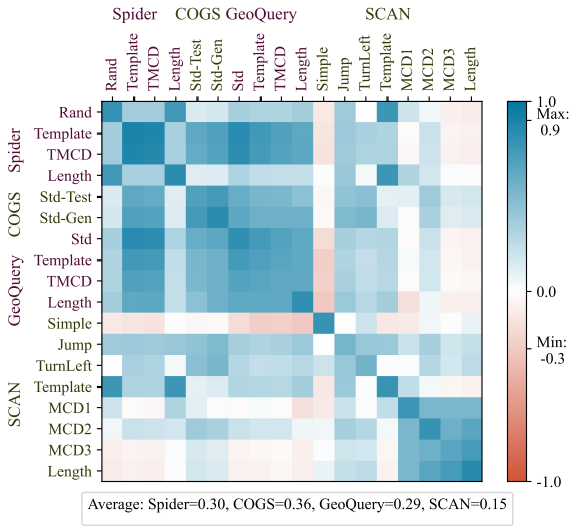

Figure 1: Pairwise concurrence values averaged across models for each dataset–split pair. Values closer to 1.0 (blue) denote a more similar ranking of models according to their performance on the dataset and split. The dataset and split font color indicate whether the data was generated by humans (purple) or synthetically using rules (green).

different aspects of model capabilities, that are increasingly not mutually exclusive: oftentimes, multiple benchmarks are available that target the same capability or skill, using (slightly) different metrics, design choices, and/or conceptual approaches. For instance, Hupkes et al. (2023) report that many recent studies on generalization used different *shift sources* to study the same types of *generalization* (see Figure 2).[2]

However, somewhat surprisingly, despite a wealth of work in the domain of evaluation and generalization, there is very little research that assesses whether multiple datasets designed to measure the same ability also yield the same conclusions. This makes it difficult for practitioners to conduct informed evaluation dataset selection and,

[1]Code to reproduce the experiments can be found at https://github.com/facebookresearch/CompositionalityValidity.

[2]Plot generated using the visualisation tool on https://genbench.org/visualisations.

Proceedings of the 27th Conference on Computational Natural Language Learning (CoNLL), pages 274–293
December 6–7, 2023.

perhaps even more concerning, impedes our understanding of how well different datasets measure what they intend to measure. While establishing *construct validity* and *construct reliability* – for instance through comparing the results of tests with other tests that intend to measure the same thing – is common practice in the social sciences (Westen and Rosenthal, 2003; Jacobs and Wallach, 2021), it is not the standard in the field of NLP.

In this work, we argue that establishing such standards is much needed in our field, and we present a detailed set of experiments that assesses construct validity in the domain of *compositional generalization*. Following Liu et al. (2021), we use *concurrence* to measure the extent to which 8 different *compositional splitting strategies* for 4 different datasets – SCAN, GeoQuery, COGS, and Spider – provide similar rankings for 6 different modeling approaches – BART, T5, Transformer, uni- and biLSTMS, and Neural-BTG. We find that, in general, the conclusions drawn from one dataset split typically do not align with the results from another dataset split. In a range of experiments, we explore if that could be attributed to whether the underlying data are synthetic or human-generated, to the compositional splitting strategy is used to create the data (a.k.a. what interpretation of compositionality), or to uncontrolled exposure to lexical items that also occurred during pretraining.

We find that concurrence values are generally low: only 10 out of 153 pairs of dataset splits have a concurrence value that surpasses the threshold for high concurrence. Furthermore, results from human-authored datasets concur much more than results from synthetic datasets. On the contrary, dataset splits that share the same interpretation of compositionality – as defined by their splitting strategy – hardly concur with each other: the underlying data plays a more important role in model rankings. Lastly, aligned with the findings of Kim et al. (2022), we find that carefully controlling the lexical items in a compositional split has a positive impact on concurrence. Overall, our results suggest that much work remains to be done to evaluate compositional generalization, and more generally that having more rigorous standards for establishing the validity of evaluation sets should be prioritized in the future.

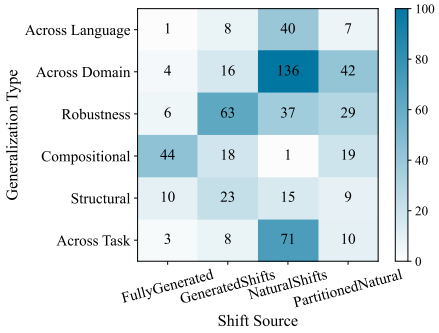

Figure 2: Generalization studies published in the ACL anthology (2015-2022), across different *shift sources*.

2 Related Work

In this section, we provide an overview of datasets commonly used for assessing compositional generalization, and we discuss previous attempts to compare performance across benchmarks.

Datasets for Compositional Generalization Since the introduction of *SCAN* in 2018 (Lake and Baroni, 2018), many datasets have been proposed to assess compositional generalization in neural networks. Several of them were direct follow-ups to SCAN that aimed to extend the original dataset or mitigate various issues perceived with it. For instance, Bastings et al. (2018) introduced NACS, a 'reversed' version of SCAN; Loula et al. (2018) introduced new splits using the original dataset; Ruis et al. (2020) introduced a multimodal, grounded version of the benchmark; and Patel et al. (2022) increased the number of primitives. Recently, Valvoda et al. (2022) proposed a transducer-based procedure for generating myriad synthetic datasets similar to SCAN to investigate which formal properties impact the results. Other artificially generated datasets available to evaluate compositionality are PCFG SET (Hupkes et al., 2020), COGS (Kim and Linzen, 2020), and the dataset proposed by Oren et al. (2021).

Datasets that use more natural (but often still templated) data are typically situated in the domain of machine translation – such as Li et al. (2021), Dankers et al. (2022) and Raunak et al. (2019) – or semantic parsing – e.g. Finegan-Dollak et al. (2018); Keysers et al. (2019); Shaw et al. (2021); Cui et al. (2022). Finally, Thrush et al. (2022) introduce Winoground, aimed to assess compositionality in text-to-image models. In our work, we focus on datasets that target compositionality in the domain of semantic parsing, with the addition of

SCAN for its sheer popularity.

Performance across benchmarks Several recent works across NLP have been interested in the extent to which strong performance on one task, setting, or dataset transfers to strong performance on another. Typically, such experiments are motivated by transfer learning, rather than establishing the validity of evaluation results. For instance, Vu et al. (2020), Ye et al. (2021), Luo et al. (2022), Padmakumar et al. (2022), and Weber et al. (2021) all investigate to what extent performance transfers across tasks. More closely related to our study, is the work presented by Liu et al. (2021), who quantify the measurement of benchmark agreement on model rankings and compare it in question answering. In our work, we adopt their definition of comparability across datasets.

In the context of compositional generalization, the work most closely related to ours is the study presented by Chaabouni et al. (2021), in which they investigate whether the performance improvements on the synthetic dataset SCAN transfer to the naturalistic setting. We largely confirm their results, but consider compositionality benchmarks more broadly, not only considering the synthetic v.s natural dimension, but also interpretations of compositionality and lexical items exposed during pretraining.

3 Methodology

We compare how the conclusions drawn from 18 different compositional generalization splits – defined over 4 different datasets with 8 compositional splitting strategies – compare across 6 modeling approaches. In this section, we describe the datasets and modeling approaches we consider and provide details on training and hyperparameter selection.

3.1 Models

For our experiments, we consider both pretrained and train-from-scratch approaches that have previously been considered in the context of compositional generalization.

BART & T5 We use the pretrained seq2seq models BART (Lewis et al., 2020) and T5 (Raffel et al., 2020) to enable easy comparison with prior work. In the case of BART, order-based noising strategies are used, which may encourage the model to learn to better represent linguistic structure.

COGS	Input:	Mila liked that the cake was offered to Emma .
	Output:	`* cake ( x _ 4 ) ; like . agent ( x _ 1 , Mila ) AND like . ccomp ( x _ 1 , x _ 6 ) AND offer . theme ( x _ 6 , x _ 4 ) AND offer . recipient ( x _ 6 , Emma )`
SCAN	Input:	turn left after jump twice
	Output:	`I_JUMP I_JUMP I_TURN_LEFT`
GeoQuery	Input:	how much population does m0 have
	Output:	`answer ( intersection ( river , loc_2 ( m0 ) ) )`
Spider	Input:	flight_1: what is the average distance and price for all flights from la?
	Output:	`select avg(distance) , avg(price) from flight where origin = "los angeles"`

Table 1: Examples of instances in each dataset used in our experiments.

LSTM & Transformer To ensure coverage of models without pre-trained knowledge, we use a uni-directional LSTM (Hochreiter and Schmidhuber, 1997), a bi-directional LSTM, and a vanilla transformer (Vaswani et al., 2017).

Neural-BTG We include one modeling approach specifically designed to address compositionality: Neural-BTG (Wang et al., 2022), composed of a discriminative parser based on a bracketing transduction grammar (BTG; Wu, 1997) and a neural seq2seq model.

3.2 Data

We consider four different datasets designed to test compositional generalization. We focus on datasets for semantic parsing and include SCAN as the most commonly used dataset for compositionality overall. Three of these datasets contain different curated *splits* that target different interpretations of compositionality. Two of the datasets (SCAN and COGS) are synthetic datasets that are generated with rules, while the other two (Spider and GeoQuery) are natural datasets, authored by humans. Examples for all datasets and descriptions of all curated splits can be found in Appendix A.

SCAN Consisting of a set of commands and the corresponding action sequences, SCAN (Lake and Baroni, 2018) is one of the most popular synthetic datasets to study compositional generalization. We include the *simple*, *length*, *add primitive*, *template* splits from Lake and Baroni (2018). In addition to original SCAN splits, we also use the maximum compound divergence (MCD) splits of SCAN proposed by Keysers et al. (2020).

COGS Kim and Linzen (2020) introduced COGS, a synthetic semantic parsing dataset generated by a rule-based approach, which covers a larger variety of grammar rules than SCAN does. The inputs in COGS are English sentences, generated by a probabilistic context-free grammar. The corresponding output, which is the semantic interpretation of the input, is annotated with the logical formalism of Reddy et al. (2017). COGS includes a randomly sampled test set and an out-of-distribution compositional generalization set.

GeoQuery GeoQuery (Tang and Mooney, 2001; Zelle and Mooney, 1996) is a text-to-QL dataset containing naturalistic examples. We use the four compositional generalization splits defined on this dataset by Shaw et al. (2021): *random/standard*, *length*, *template*, and *Target Maximum Compound Divergence (TMCD)*.

Spider Spider (Yu et al., 2018) is originally designed for cross-domain semantic parsing. We use the compositional generalization splits for Spider defined by Shaw et al. (2021), which match their splits for GeoQuery: *random/standard*, *length*, *template*, and *TMCD*.

3.3 Training Setup

We train/fine-tune the models on the train partition of each dataset described above and evaluate them on the corresponding test set. For T5 on GeoQuery and Spider as well as LSTM and Transformers on COGS, we use the hyperparameters provided in Shaw et al. (2021) and Kim and Linzen (2020), respectively. We followed Orhan (2021) to train T5 and Yao and Koller (2022) to train BART on COGS. For the remaining model-dataset combinations, we perform a hyperparameter search for each dataset, with 10% of instances randomly chosen to be used for tuning. Details can be found in Appendix C. We use three different random seeds for each training run and use five random seeds for each training run of LSTM, to compensate for LSTM's higher variation in performance across seeds. For models with existing evaluations on a dataset, we compare to these previous measures of performance to ensure that our replication results align with previously reported numbers (Keysers et al., 2020; Kim and Linzen, 2020; Orhan, 2021; Shaw et al., 2021; Yao and Koller, 2022; Sun et al., 2023b).

3.4 Evaluation

For most datasets, we use exact match (EM) accuracy. EM is a binary metric that only counts an output as correct if it matches the target output exactly, and is most frequently used for the datasets we consider. During initial experiments, we found that, in many cases, EM accuracy may be too strict for our purposes. In some cases, models' tokenizers may prefer slightly different spacing – a phenomenon also reported by Sun et al. (2023a) – in others, models lack specific tokens in their vocabulary. Neither of these things is indicative of a model's compositional generalization capability, and we therefore choose to normalize model outputs before applying EM accuracy. In Appendix D, we include examples of such cases, and we report the differences between EM scores with and without our normalization step. For Spider, the original dataset also uses a more lenient EM implementation. For consistency reasons, we use the same implementation across all datasets, but we report Spider EM scores in Appendix E to compare with previous work.

3.5 Measuring Concurrence

To measure how similarly different dataset splits rank different modeling approaches, we use the concept of *concurrence* introduced by Liu et al. (2021). The concurrence between two dataset splits is defined as the correlation between the performances of different modeling approaches for those splits. More specifically, the concurrence $\text{CONCUR}(D_1, D_2; \mathcal{A}, \text{Eval})$ between two dataset splits D_1 and D_2, given a set of modeling approaches $\mathcal{A}$ and evaluation function Eval, is defined as:

$$\text{CONCUR}(D_1, D_2; \mathcal{A}, \text{Eval}) = \text{CORR}(P_1, P_2),$$

where CORR is some correlation function and P_i is the variable that holds the scores of $\text{Eval}(a, D_i)$ for all $a \in \mathcal{A}$. For CORR, Liu et al. (2021) considered both Pearson (r) and Kendall rank (τ). Because we are interested in how benchmarks rank model performance, we report the concurrence values under Kendall's τ unless specified otherwise. We refer to the concurrence between the dataset split and itself as *self-concurrence*, the value of which is purely affected by seed variation across training runs. We see self-concurrence, which would be 1.0 if there is no variation across seeds, as an upper bound for the concurrence values across dataset splits.

4 Results

We now present our results, starting with a discussion of the performance of models on the datasets (§4.1) and the concurrence scores between the performances (§4.2), we then proceed to look at the relationship between synthetic and natural compositionality datasets (§4.3), and how this interacts with the choice of definition of compositionality and underlying dataset (§4.4). We finish our results section with a short investigation into the impact of the choice of lexical items in data (§4.5).

4.1 Overall Performance

In Table 2, we show the performance of all models on all dataset splits under consideration, as well as the average performance per dataset split (last column). Our scores are generally close to the scores reported in previous work, for the (dataset split, architecture) combinations for which previous results exist (Sun et al., 2023b), with the exception of the results for Spider, for which we use a different metric. All models perform reasonably well on the random splits of each datasets (first row for each dataset in Table 2), but most struggle with various generalization splits. While some splits are difficult across the board, other difficulties appear more model-dependent. For instance, while all models are weak on the *length* and *MCD* splits of SCAN and *length* split of Spider, COGS is difficult for some models (e.g., BTG) but much less for others (e.g., T5). Similarly, some models perform well on one of the datasets or one of the splits, but perform poorly on the others. BART, for instance, maintains high performance on GeoQuery and COGS, but performs even worse than non-pretrained models on some splits of SCAN, while BTG performs well on GeoQuery but fails on many splits of SCAN. T5 has high performance on most datasets, but is outperformed by the unidirectional LSTM on the *length* split of SCAN. SCAN, in particular, appears to be challenging for all models, with the *TurnLeft* split being the only exception.[3]

4.2 Overall Concurrence

It is not difficult to tell from Table 2 that the performance of a model on one dataset is not predictive of its performance on the others. To quantitatively substantiate this observation, we compute the concurrences between the different dataset splits, which we visualize in Figure 1. On average, the concurrence between dataset splits is low: a mere 0.22, far below the average self-concurrence of 0.76 that (model, split) combinations have across different seeds. Interestingly, even these average self-concurrence values are lower than the 0.8 that Liu et al. (2021) used as a threshold for "high" concurrence, indicating that performance on the same compositional dataset is not very stable across runs.[4] Consequently, we lower the threshold to 0.7 here, which is approximately 90% of the average self-concurrence. Of the 153 pairs of dataset split we compare in this experiment, only 10 pairs surpass this threshold. Somewhat surprisingly, perhaps, many of the highest values (reported in Table 3), are concurrences between i.i.d. splits and compositional splits.

Considering the concurrence of each dataset with all other datasets (excluding self-concurrence, values are reported below Figure 1), we can see that performance COGS, with an average τ of 0.36 is most predictive of performance on other datasets. Furthermore, the three semantic parsing datasets have much higher average concurrence than SCAN, suggesting that compositionality on one task may not be predictive of compositionality on another.

4.3 Synthetic vs natural data

Why are these concurrence values so low? The first hypothesis that we explore is that performance on strongly structured templated data may not correlate with performance on datasets that are authored by humans. To this end, we compute the average concurrence values of three combinations of dataset split pairs, natural-natural, natural-synthetic and synthetic-synthetic, and include an example of each pair type in Figure 3. We find that splits of natural datasets concur much better than splits of synthetic datasets (0.54 v.s. 0.22); the worst is concurrence between synthetic and natural dataset splits (0.19). The same finding can be observed in Figure 6, which we will use later to explore the relationship between concurrence values and performance in §4.6.

These results are in line with earlier studies that suggested that performance on synthetic compositionality datasets may not transfer to more re-

[3]While architectures exist that obtain high scores on SCAN, such as the ones introduced by Shaw et al. (2021) and Kim (2021), they are too narrowly scoped for our current study and we thus do not consider them.

[4]This finding is in line with results reported by Liska et al. (2018), who find a range of different generalization performances on a simple but highly compositional look-up table task.

Dataset	Split	LSTM Uni		LSTM Bi		Transformer		T5		BART		BTG		Avg
COGS	*Std-Test*	99.3	±.0	99.1	±.01	99.5	±.0	99.7	±.0	99.7	±.0	68.8	±.01	94.3
	Std-Gen	21.3	±.05	14.8	±.08	56.1	±.06	82.9	±.0	78.6	±.0	2.8	±.01	42.8
SCAN	*Simple*	99.9	±.0	99.9	±.0	100.0	±.0	94.9	±.01	99.1	±.01	12.3	±.01	84.4
	Jump	0.4	±.01	0.0	±.0	0.1	±.0	95.0	±.01	0.4	±.01	0.0	±.0	16.0
	TurnLeft	61.1	±.13	34.1	±.06	64.8	±.11	70.3	±.12	63.1	±.19	8.9	±.01	50.4
	Template	0.2	±.0	0.3	±.01	1.1	±.0	34.3	±.03	0.0	±.0	0.9	±.01	6.1
	MCD1	5.9	±.06	12.2	±.07	1.1	±.0	24.6	±.01	0.4	±.01	1.8	±.01	7.7
	MCD2	6.7	±.03	5.8	±.03	1.2	±.0	34.1	±.01	1.6	±.0	0.5	±.0	8.3
	MCD3	8.7	±.04	7.8	±.02	0.7	±.0	11.1	±.01	1.2	±.01	0.8	±.01	5.0
	Length	15.3	±.04	11.8	±.01	0.0	±.0	14.1	±.01	0.7	±.01	0.0	±.0	7.0
GeoQuery	*Std*	74.0	±.06	78.9	±.04	82.3	±.02	92.5	±.01	89.2	±.01	79.0	±.01	82.6
	Template	46.5	±.06	55.9	±.07	56.7	±.04	91.0	±.0	77.1	±.06	53.5	±.06	63.5
	TMCD	35.8	±.02	37.1	±.02	37.9	±.01	54.1	±.0	48.2	±.0	36.9	±.0	41.7
	Length	18.5	±.03	16.2	±.02	22.0	±.01	41.1	±.01	36.1	±.01	20.7	±.02	25.8
Spider	*Rand*	33.4	±.02	36.9	±.01	42.5	±.01	68.0	±.0	32.7	±.01	40.1	±.01	42.3
	Template	1.0	±.0	2.2	±.01	4.6	±.0	39.6	±.01	21.6	±.01	1.9	±.0	11.8
	TMCD	4.6	±.01	6.0	±.01	7.5	±.01	47.2	±.01	31.2	±.03	5.5	±.0	17.0
	Length	12.7	±.01	14.0	±.01	17.5	±.01	35.4	±.01	7.4	±.0	14.0	±.01	16.8

Table 2: Model exact-match accuracy on datasets averaged across random seeds, with standard deviation.

Dataset A	Dataset B	Split A	Split B	Concur
Spider	Spider	*Template*	*TMCD*	0.88
GeoQuery	Spider	*Std*	*Template*	0.84
GeoQuery	Spider	*Std*	*TMCD*	0.83
SCAN	Spider	*Template*	*Rand*	0.76
SCAN	Spider	*Template*	*Length*	0.76
Spider	Spider	*Rand*	*Length*	0.75
GeoQuery	Spider	*Template*	*Template*	0.74
GeoQuery	Spider	*Template*	*TMCD*	0.73
GeoQuery	GeoQuery	*Std*	*Template*	0.73
SCAN	SCAN	*Length*	*MCD3*	0.72

Table 3: High concurrence values ($\geq$ 0.7) among all pairs of dataset splits, excluding self-concurrence.

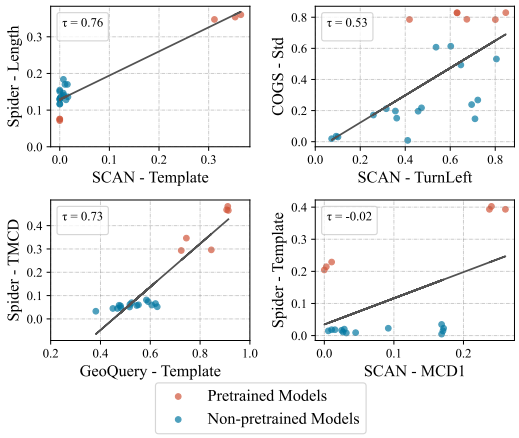

Figure 3: Performance of one dataset split versus another. Upper left is an example of high concurrence pair between a synthetic and a natural dataset; upper right is an example of low concurrence within synthetic datasets; lower left is an example of high concurrence within natural datasets; lower right is an example of low concurrence between natural and synthetic datasets.

alistic scenarios (Chaabouni et al., 2021; Shaw et al., 2021), and underline the point made by Dankers et al. (2022), who argue that compositionality should be studied in its natural habitat. Also the concurrence between dataset splits with naturalistic data is well below the threshold for high concurrence, suggesting that there exist factors beyond dataset creation strategy that can affect how compositionality benchmarks rank modeling approaches.

4.4 Interpretations of compositionality

The next hypothesis that we consider is that concurrence values are low because different dataset splits investigate different types of compositionality (cf. Hupkes et al., 2020). In compositional evaluation datasets, the interpretation of compositionality is operationalized through its *splitting strategy*. One splitting strategy may, for instance, define compositional generalization as generalization to longer lengths, whereas another instead focuses on generalization to novel vocabulary items. These different interpretations of compositionality could potentially require different model capabilities. Could it be that our concurrence values are low because different splits in fact focus on different types of compositional generalization?

To investigate this, we group the concurrence values by four dataset pair types – different datasets with the same splitting strategy, the same dataset with different splitting strategies, different datasets with different splitting strategies, and the same dataset with the same splitting strategy – and plot them in Figure 4. Predictably, datasets concur most with themselves (red line). We also see that which data a splitting approach is applied to is more important than the interpretation of compositionality (cyan and dark blue lines, respectively): concur-

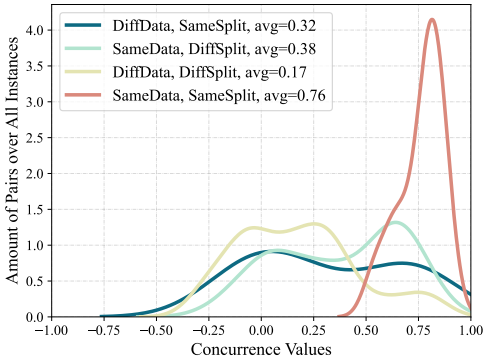

Figure 4: Distribution of concurrence values among all dataset splits. The color of the bar indicates whether the splits in the pair share the same dataset origin and/or the same splitting strategy.

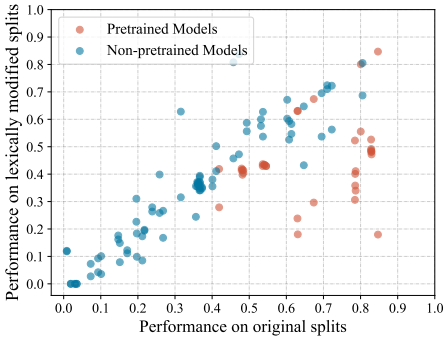

Figure 5: Performance of the original split versus the splits with lexical items replaced. Performance of pretrained models decreases when train on the splits with lexical items that are not previously seen in pretraining.

Dataset A	Dataset B	Concur	Dataset A	Dataset B	Concur
COGS	GeoQuery	0.54	COGS	SCAN	0.01
COGS	Spider	0.26	SCAN	Spider	0.01
GeoQuery	Spider	0.23	GeoQuery	SCAN	- 0.09

Table 4: Concurrence between length splits of datasets.

rence between experiments that share the same source of data averages at 0.38, whereas different data but the same splitting strategy results in an average concurrence of 0.32. However, when both the source of data and splitting strategy are different (yellow line), the concurrence values shift leftward, suggesting that the data type and splitting strategy pose different kinds of difficulties for the modelling approaches considered.

Length Generalization Because not every dataset in previous work applied all the splitting strategies, we follow-up with a small experiment in a split shared across all datasets: *length generalization* splits.[5] The concurrence values between the different length splits, shown in Table 4, are generally low, ranging from -0.09 to 0.54 and averaging at 0.16. This additional experiment confirms that even when benchmarks maintain the same interpretation of compositionality, there may still be substantial differences in model rankings, depending on the underlying data.

4.5 The influence of lexical items

In Table 2, we can see that pretrained models achieve the highest accuracies and in Table 3 that the highest concurrence values are between two natural datasets. In this section, we dive into the differences between pretrained and trained-from-scratch models, and investigate the extent to which those differences affect the concurrence results. In particular, we investigate whether the presence of uncontrolled lexical exposure during pretraining may impact the performance of pretrained models, implying their accuracy numbers may not solely reflect their compositional abilities, as suggested by Kim et al. (2022). Were this to happen, a misalignment in the evaluation between pretrained and non-pretrained models would contribute to variation in the concurrence values, where the performance of pretrained models is overestimated due to lexical exposure in pretraining.

To test for possible effects of lexical exposure, we extend the experiment from Kim et al. (2022) – who conducted it for COGS – to the TMCD and Std split of GeoQueory, and the TurnLeft split of SCAN[6] In both cases, we swap out lexical items with strings of similar length that act as "wug words" (Berko, 1958), or, in other words, previously unattested and therefore meaningless lexical items. Following Kim et al. (2022), we generate the strings in two ways:

- *Rstr:* We randomly sample lowercase characters from the Latin script with replacements.
- *Rcvcv:* We alternately sample a vowel after a consonant from the Latin script.

We train the models on all modified splits and compute the performance (Figure 5). We also compute the concurrence between the original split and the modified split (Table 5a and Table 5b).

[5] As the original COGS dataset did not come with a length generalization split, we generate one ourselves.

[6] In both these cases, particular lexical items are purposefully left out of the training set, to be evaluated at test time. If those lexical items were also present in the uncontrolled pretraining corpus, this would thus break the test.

Dataset	Split A	Split B	Concur
GeoQuery	Std	Std-Rcvcv	0.69
		Std-Rstr	0.54
	TMCD	TMCD-Rstr	0.65
		TMCD-Rcvcv	0.63
COGS	Std	RandStr	0.60
		Randcvcv	0.59
SCAN	TurnLeft	TurnLeftRcvcv	0.29
		TurnLeftRStr	0.23

(a) Concurrence between the original split and lexically-processed splits.

Dataset A	Split A	Dataset B	Split B	Concur
COGS	*Length*	GeoQuery	*TMCD-Rcvcv*	0.84
GeoQuery	*Std-Rcvcv*	GeoQuery	*TMCD-Rcvcv*	0.83
COGS	*Std*	GeoQuery	*TMCD-Rcvcv*	0.82
GeoQuery	*TMCD-Rstr*	Spider	*Template*	0.82
GeoQuery	*TMCD-Rcvcv*	Spider	*Template*	0.81
COGS	*Length*	GeoQuery	*TMCD-Rstr*	0.81
COGS	*Length*	GeoQuery	*Std-Rcvcv*	0.8
GeoQuery	*Std-Rcvcv*	GeoQuery	*TMCD-Rstr*	0.8
GeoQuery	***TMCD-Rstr***	**Spider**	***TMCD***	**0.79**
GeoQuery	***TMCD-Rcvcv***	**Spider**	***TMCD***	**0.79**
COGS	*Std*	GeoQuery	*Std-Rcvcv*	0.78
GeoQuery	*Std*	GeoQuery	*TMCD-Rstr*	0.77
GeoQuery	*Std*	GeoQuery	*TMCD-Rcvcv*	0.75
COGS	*Std*	GeoQuery	*TMCD-Rstr*	0.74
GeoQuery	*Template*	Spider	*TMCD*	0.73
GeoQuery	*Std-Rcvcv*	Spider	*Template*	0.73
COGS	*RandStr*	GeoQuery	*Std-Rstr*	0.73
COGS	*Std*	GeoQuery	*Std-Rstr*	0.72
GeoQuery	*Std-Rstr*	GeoQuery	*TMCD-Rcvcv*	0.71
GeoQuery	*Std-Rcvcv*	Spider	*TMCD*	0.71
COGS	*Randcvcv*	GeoQuery	*Std-Rstr*	0.7

(b) High concurrence values after introducing lexically-processed splits, excluding self-concurrence or concurrence between lexically-processed splits that share the same origin.

Table 5: Performance and Concurrence between the lexically-processed splits of datasets.

In Figure 5, we see that the performance of the pretrained models drops drastically when the lexical items are replaced, while the non-pretrained models' performance does not, confirming the results of Kim et al. (2022). In addition, the concurrence between the original splits and the modified splits for all datasets is below our set threshold – albeit higher than other comparisons we have seen before (Table 5a) – implying that replacing lexical items results in yet another new ranking of modeling approaches for compositionality.

We then compute the concurrence between the same set of splits before and after the lexical exposure edits: *within* the group of splits that are selected for the lexical changes, the concurrence values decrease from 0.49 to 0.41, while the average concurrence values of these splits with *other* splits that haven't undergone lexical edits slightly increase from 0.25 to 0.26 (e.g. concurrence between GeoQuery and Spider TMCD splits increases when GeoQuery TMCD split applies the lexical changes), with many more dataset split pairs surpassing the $\tau = 0.7$ bar for high concurrence (Table 5b).

A closer look explains this apparent contrast: the overall low-concurring dataset SCAN – which makes up 12.5% of the lexically edited splits, drags down the concurrence values within that group. Excluding SCAN, the within-group concurrence values also increase, from 0.63 to 0.66. These results do thus not only confirm that controlling lexical exposure is important when evaluating compositionality in pretrained models, but also further exemplify our earlier finding that compositionality scores – for neural models – strongly depend task and dataset. We further analyze the influence of tasks to compositionality results in Appendix F.

4.6 Other confounding factors

We have explored a range of factors that may impact the evaluation of compositionality, such as the nature of the underlying data and task, the interpretation of compositionality, and the choice of lexical items. We wrap up our analysis by verifying that our results are not driven by specific performance scores: we verify that concurrence values are not skewed by datasets for which performances are saturated or close to random. To assess this, we compute the correlation between the average performance between two datasets and their concurrence, as plotted in Figure 6. As can be seen, there is no apparent relation between average performance and concurrence: difficult datasets do not concur less or more than easier ones, and dataset saturation (or the opposite: random performance) appears not to impact the results. A correlation test confirms this visually observed pattern: the Pearson correlation coefficient between performance and concurrence is near zero ($r = 0.026$).

5 Conclusion

In this paper, we explored how different evaluation choices impact the conclusions drawn from the experiments evaluating compositionality. Using compositional generalization datasets and models ranging from trained-from-scratch to pretrained, we conduct a series of experiments to understand whether datasets consistently rank models in terms of their generalizability, and we find little consistency. When we perform further analysis to try to better understand this inconsistency, we find that comparing within the training setting (pretrained v.s. trained-from-scratch) or data creation type (synthetically generated v. naturally generated) does

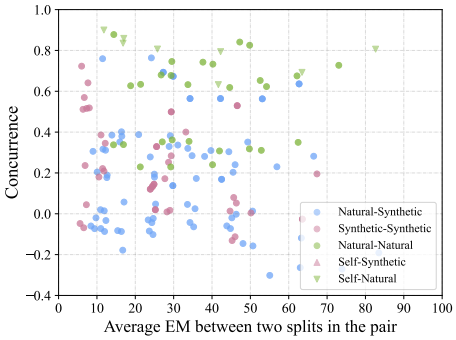

Figure 6: Values of concurrences with respect to pairwise averaged performance among the splits shown in Table 2. The color of dots indicates the type of split pairs. The triangle-shape dots indicates the values of self-concurrence.

not increase consistency. However, better controlling the lexical items can help us draw more consistent conclusions, at least for datasets that share the same notion of compositionality. We leave the investigation into how task selection might affect evaluation results for compositional generalization to further research. Overall, our results suggest that to evaluate compositional generalization consistently, clearer definitions of compositionality are needed, as well as more careful consideration of evaluation design and more thorough dataset evaluations.

References

Jasmijn Bastings, Marco Baroni, Jason Weston, Kyunghyun Cho, and Douwe Kiela. 2018. Jump to better conclusions: SCAN both left and right. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 47–55, Brussels, Belgium. Association for Computational Linguistics.

Jean Berko. 1958. The child's learning of english morphology. *Word*, 14(2-3):150–177.

Rahma Chaabouni, Roberto Dessì, and Eugene Kharitonov. 2021. Can transformers jump around right in natural language? assessing performance transfer from SCAN. In *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 136–148, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, et al. 2022. PaLM: Scaling language modeling with pathways. *CoRR*, abs/2204.02311.

Ruixiang Cui, Rahul Aralikatte, Heather Lent, and Daniel Hershcovich. 2022. Compositional generalization in multilingual semantic parsing over Wikidata. *Transactions of the Association for Computational Linguistics*, 10:937–955.

Verna Dankers, Elia Bruni, and Dieuwke Hupkes. 2022. The paradox of the compositionality of natural language: A neural machine translation case study. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4154–4175, Dublin, Ireland. Association for Computational Linguistics.

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.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Arian Hosseini, Ankit Vani, Dzmitry Bahdanau, Alessandro Sordoni, and Aaron Courville. 2022. On the compositional generalization gap of in-context learning. In *Proceedings of the Fifth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 272–280, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2020. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 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. 2023. A taxonomy and review of generalization research in nlp. *Nature Machine Intelligence*, 5(10):1161–1174.

Abigail Z Jacobs and Hanna Wallach. 2021. Measurement and fairness. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 375–385.

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 *International Conference on Learning Representations*.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, et al. 2019. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*.

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)*, pages 9087–9105, Online. Association for Computational Linguistics.

Najoung Kim, Tal Linzen, and Paul Smolensky. 2022. Uncontrolled lexical exposure leads to overestimation of compositional generalization in pretrained models. *arXiv preprint arXiv:2212.10769*.

Yoon Kim. 2021. Sequence-to-sequence learning with latent neural grammars. In *Advances in Neural Information Processing Systems*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pages 2873–2882. 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*, pages 7871–7880, Online. Association for Computational Linguistics.

Yafu Li, Yongjing Yin, Yulong Chen, and Yue Zhang. 2021. On compositional generalization of neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4767–4780, Online. Association for Computational Linguistics.

Adam Liska, Germán Kruszewski, and Marco Baroni. 2018. Memorize or generalize? searching for a compositional RNN in a haystack. *CoRR*, abs/1802.06467.

Nelson F Liu, Tony Lee, Robin Jia, and Percy Liang. 2021. Do question answering modeling improvements hold across benchmarks? *arXiv preprint arXiv:2102.01065*.

João Loula, Marco Baroni, and Brenden Lake. 2018. Rearranging the familiar: Testing compositional generalization in recurrent networks. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 108–114, Brussels, Belgium. Association for Computational Linguistics.

Yifei Luo, Minghui Xu, and Deyi Xiong. 2022. CogTaskonomy: Cognitively inspired task taxonomy is beneficial to transfer learning in NLP. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 904–920, Dublin, Ireland. Association for Computational Linguistics.

Inbar Oren, Jonathan Herzig, and Jonathan Berant. 2021. Finding needles in a haystack: Sampling structurally-diverse training sets from synthetic data for compositional generalization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10793–10809, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

A Emin Orhan. 2021. Compositional generalization in semantic parsing with pretrained transformers. *arXiv preprint arXiv:2109.15101*.

Vishakh Padmakumar, Leonard Lausen, Miguel Ballesteros, Sheng Zha, He He, and George Karypis. 2022. Exploring the role of task transferability in large-scale multi-task learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2542–2550, Seattle, United States. Association for Computational Linguistics.

Arkil Patel, Satwik Bhattamishra, Phil Blunsom, and Navin Goyal. 2022. Revisiting the compositional generalization abilities of neural sequence models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 424–434, Dublin, Ireland. Association for Computational Linguistics.

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

Vikas Raunak, Vaibhav Kumar, Florian Metze, and Jaimie Callan. 2019. On compositionality in neural machine translation. In *NeurIPS 2019 Context and Compositionality in Biological and Artificial Neural Systems Workshop*.

Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata. 2017. Universal semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 89–101, Copenhagen, Denmark. Association for Computational Linguistics.

Laura Ruis, Jacob Andreas, Marco Baroni, Diane Bouchacourt, and Brenden M. Lake. 2020. A benchmark for systematic generalization in grounded language understanding. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. 2021. Compositional generalization and natural language variation: Can a semantic parsing approach handle both? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 922–938, Online. Association for Computational Linguistics.

Kaiser Sun, Peng Qi, Yuhao Zhang, Lan Liu, William Yang Wang, and Zhiheng Huang. 2023a. Tokenization consistency matters for generative models on extractive NLP tasks. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. Association for Computational Linguistics.

Kaiser Sun, Adina Williams, and Dieuwke Hupkes. 2023b. A replication study of compositional generalization works on semantic parsing. *ReScience C*, 9(2):44.

Lappoon R Tang and Raymond J Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *European Conference on Machine Learning*, pages 466–477. Springer.

Tristan Thrush, Ryan Jiang, Max Bartolo, Amanpreet Singh, Adina Williams, Douwe Kiela, and Candace Ross. 2022. Winoground: Probing vision and language models for visio-linguistic compositionality. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5238–5248.

Josef Valvoda, Naomi Saphra, Jonathan Rawski, Adina Williams, and Ryan Cotterell. 2022. Benchmarking compositionality with formal languages. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 6007–6018, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

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

Tu Vu, Tong Wang, Tsendsuren Munkhdalai, Alessandro Sordoni, Adam Trischler, Andrew Mattarella-Micke, Subhransu Maji, and Mohit Iyyer. 2020. Exploring and predicting transferability across NLP tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7882–7926, Online. Association for Computational Linguistics.

Bailin Wang, Ivan Titov, Jacob Andreas, and Yoon Kim. 2022. Hierarchical phrase-based sequence-to-sequence learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8211–8229, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Lucas Weber, Jaap Jumelet, Elia Bruni, and Dieuwke Hupkes. 2021. Language modelling as a multi-task problem. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2049–2060, Online. Association for Computational Linguistics.

Drew Westen and Robert Rosenthal. 2003. Quantifying construct validity: two simple measures. *Journal of personality and social psychology*, 84(3):608.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Yuekun Yao and Alexander Koller. 2022. Structural generalization is hard for sequence-to-sequence models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5048–5062, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. 2021. CrossFit: A few-shot learning challenge for cross-task generalization in NLP. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7163–7189, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055.

A Dataset examples

For convenience, we include a brief description with examples of all datasets we consider in our experiments in Table 6. The description of each split and the number of instances in each dataset split is shown in Table 7 and Table 8.

SCAN Consisting of a set of commands and the corresponding action sequences, SCAN (Lake and Baroni, 2018) is one of the most popular synthetic datasets to study compositional generalization. The model is given commands like `jump left` and is expected to predict action sequences like `LTURN JUMP`. We include the *simple*, *length*, *add primitive*, *template* splits from Lake and Baroni (2018). In addition to original SCAN splits, we also use maximum compound divergence (MCD) splits of SCAN proposed by Keysers et al. (2020).

COGS Kim and Linzen (2020) introduce COGS, a synthetic semantic parsing dataset generated by a rule-based approach, which covers a larger variety of grammar rules than SCAN does. The inputs in COGS are English sentences, generated by a probabilistic context-free grammar. The corresponding output, which is the semantic interpretation of the input, is annotated with the logical formalism in Reddy et al. (2017). COGS includes a randomly sampled test set and an out-of-distribution compositional generalization set.

GeoQuery GeoQuery (Tang and Mooney, 2001; Zelle and Mooney, 1996) is a text-to-QL dataset containing naturalistic examples. We use the four compositional generalization splits defined on this dataset by Shaw et al. (2021): We use the splits in Shaw et al. (2021), in which all entity mentions are converted with placeholders and use Functional Query Language (FunQL) as the target representation. *random/standard*, *length*, *template*, and *Target Maximum Compound Divergence (TMCD)*. The TMCD split is an extension of MCD splits in SCAN, with the capability to be applied to non-synthetic datasets.

Spider Spider (Yu et al., 2018) is originally designed for cross-domain semantic parsing, and targets a challenging kind of generalization, generalization to new database schemata, using different databases for the training and test set. It also uses SQL for a more complex syntax. We use the compositional generalization splits for Spider defined by Shaw et al. (2021), which match their splits for GeoQuery: *random/standard*, *length*, *template*, and *TMCD*. In the same paper, Shaw et al. (2021) split Spider into the same four splits as GeoQuery and adopt a setting where databases are shared between train and test examples so that the dataset splits can be dedicated to evaluating compositional generalization.

B License of Artifacts

We include the licenses and intended usage of artifacts used in this work in Table 9.

C Hyperparameters

For the models and dataset combinations that have already been trained by prior works, we adopt the same set of hyperparameters. For the remaining combinations, we tune the hyperparameters on a random split of the original dataset, with 90% data in the training set and 10% data in the test set. We describe the final hyperparamters below.

For T5 with GEOQUERY and SPIDER, we follow the same hyperparameter setup as Shaw et al., 2021. For LSTM and Transformer with COGS, we follow the same hyperparameter setup as in Kim and Linzen, 2020. For T5 with COGS, we follow the training strategy from (Orhan, 2021).

For other datasets, we tune the learning rate of T5 and BART in $[10^{-5}, 10^{-4}, 10^{-3}]$. We tune the dropout rate in $[0.0, 0.1, 0.5]$ and layers in $[1, 2]$ for LSTMs; dropout rate in $[0.0, 0.1, 0.5]$ and layers in $[2, 4, 8]$ for Transformer. For BTG, we tune the vocabulary size between 200 and 800, as well as the learning rate in $[1.0 \times 10^{-4}, 3.0 \times 10^{-4}]$.

COGS	**Input:**	Mila liked that the cake was offered to Emma .
	Output:	* cake (x _ 4) ; like . agent (x _ 1 , Mila) AND like . ccomp (x _ 1 , x _ 6) AND offer . theme (x _ 6 , x _ 4) AND offer . recipient (x _ 6 , Emma)
SCAN	**Input:**	turn left after jump twice
	Output:	I_JUMP I_JUMP I_TURN_LEFT
NACS	**Input:**	run thrice after jump around left
	Output:	I_TURN_LEFT I_JUMP I_TURN_LEFT I_JUMP I_TURN_LEFT I_JUMP I_TURN_LEFT I_JUMP I_RUN I_RUN I_RUN
GeoQuery	**Input:**	how much population does m0 have
	Output:	answer (intersection (river , loc_2 (m0)))
Spider	**Input:**	flight_1: what is the average distance and price for all flights from la?
	Output:	select avg(distance) , avg(price) from flight where origin = "los angeles"

Table 6: Examples of instances in each dataset used in our experiments.

Split	Dataset	Description
random/standard/simple	COGS, SCAN, GeoQuery, Spider	Split the dataset randomly.
length	COGS, SCAN, GeoQuery, Spider	Split the dataset according to the input length.
template	SCAN, GeoQuery, Spider	Split the dataset based on a given string template.
TurnLeft	SCAN	Compositional commands of `TurnLeft` are isolated in training set.
Jump	SCAN	Compositional commands of `Jump` are isolated in training set.
MCD	SCAN	Split according to maximum compound divergence.
TMCD	GeoQuery, Spider	Natural counterpart of MCD, split the data based on target MCD.
Gen	COGS	Not a splitting strategy, but a collection of specially generated samples designed to test 21 cases of generalization in COGS.

Table 7: Summary of each split and their designated dataset we use.

D Evaluation: Variants of Exact Match Accuracy

Dataset	Split	T5	BART	BTG
COGS	*Std-Test*	99.7	0.0	0.0
	Std-Gen	82.9	0.0	0.0
	Rcvcv-Test	99.7	0.0	0.0
	Rstr-Test	99.8	0.0	0.0
	Rcvcv-Gen	50.0	0.0	0.0
	Rstr-Gen	48.0	0.0	0.0
	Length	37.9	0.0	0.0
Spider	Rand	60.1	26.2	32.4
	Template	34.9	18.1	1.8
	TMCD	38.3	23.5	4.9
	Length	33.9	6.1	11.9
GeoQuery	Std	77.1	0.0	0.0
	Std-Rcvcv	74.3	0.0	0.0
	Std-Rstr	73.5	0.0	0.0
	Template	76.5	0.0	0.0
	Length	39.5	0.0	0.0
	TMCD	40.7	0.0	0.0
	TMCD-Rcvcv	31.6	0.0	0.0
	TMCD-Rstr	31.4	0.0	0.0

Table 10: Percentage difference between raw EM implementation and EM implementation that ignore harmless space (space-lenient EM - raw EM). SCAN and NACS are omitted because models do not have this issue on them. LSTMs do not display this issue; the difference for Transformer is under 0.1% for each datset.

The most intuitive implementation of exact match accuracy is directly comparing the output text string with the gold sequence, without any post-processing. However, we found this to be unnecessarily strict for some models, such as T5, which does not have the "<" symbol, which appears in a large number of instances, in the vocabulary and required post-processing to replace the UNK tokens with "<". In addition, although the location of space should not change the correctness of a prediction for our evaluated datasets, often incorrect spaces led to wrong evaluation when direct text comparison is used. Table 11 shows an example of such an instance. With the leniency on spaces, T5's exact match value changed from zero accuracy on a whole dataset (COGS) to performing among the best on all datasets (Table 10); this is likely due to the tokenization of special tokens with space, as noted in Sun et al. (2023a).

E Spider performance

Split	LSTM Uni	LSTM Bi	Trans-former	T5	BART	BTG
Rand	0.0	0.0	0.0	77.8	34.8	46.2
Template	1.4	2.7	3.2	52.5	25.5	3.5
TMCD	0.1	0.1	0.1	57.6	37.9	6.9
Length	0.9	0.6	0.3	44.4	9.0	16.5

Table 12: Model exact-match accuracy with Spider EM. A large amount of output of LSTM and Transformer are deemed as invalid SQL due to special tokens.

The official release of Spider (Yu et al., 2018) uses a different variant of exact match accuracy, which is more lenient than the version we used. We include a table of model performance on splits of Spider, evaluated with the official Spider metric in

Dataset	Split	Train	Validation	Test	Overall	
COGS	no_mod	24155	3000	3000	21000	51155
	random_cvcv	24155	3000	3000	21000	51155
	random_str	24155	3000	3000	21000	51155
	length	24156	-	23999	-	48155
GeoQuery	standard	600	-	280	-	880
	length	440	-	440	-	880
	template	441	-	439	-	880
	tmcd	440	-	440	-	880
SCAN	simple	16728	-	4182	-	20910
	length	16990	-	3920	-	20910
	mcd1	8365	1045	1045	-	10455
	mcd2	8365	1045	1045	-	10455
	mcd3	8365	1045	1045	-	10455
	addprim_jump	14670	-	7706	-	22376
	addprim_turn_left	21890	-	1208	-	23098
	jump_random_cvcv	14670	-	7706	-	22376
	jump_random_str	14670	-	7706	-	22376
	turn_left_random_cvcv	21890	-	1208	-	23098
	turn_left_random_str	21890	-	1208	-	23098
Spider	random	3282	-	1094	-	4376
	length	3282	-	1094	-	4376
	template	3280	-	1096	-	4376
	tmcd	3282	-	1094	-	4376

Table 8: Number of instances for each dataset in each optimization split.

Artifact	License	Intended Usage
COGS	MIT	A dataset focuses on compositional generalization
SCAN	BSD	A dataset focuses on compositional generalization.
GeoQuery	ODC-BY 1.0 license	A database query datasets for U.S. geography.
Spider	CC BY-SA 4.0	A cross-domain semantic parsing and text-to-SQL dataset.
NACS	CC-NC	A dataset focuses on compositional generalization.
Neural-BTG	MIT	A neural transducer for sequence-to-sequence tasks.
LSTM, Transformer (OpenNMT-py (Klein et al., 2017))	MIT	Models for sequence-to-sequence tasks.
T5	Apache-2.0	A pre-trained model for sequence-to-sequence tasks.
BART	Apache-2.0	A pre-trained model for sequence-to-sequence tasks.

Table 9: License and intended usage for the artifacts we used.

Table 12.

F The influence of task similarity

As briefly mentioned in §4.5, task formulation can be another factor that affects the agreement between datasets. To understand the effect of task similarity on the conclusion obtained from compositionality benchmarks, we add in the NACS dataset (Bastings et al., 2018) for existing experiments, as all three datasets except for SCAN are semantic parsing tasks, while SCAN falls under a navigation task. NACS is introduced as a dataset that is similar to SCAN but requires mapping actions back to the original commands, and it is thus more complex for models compared to SCAN and will not allow simple models to gain unintended high performance. We train models on NACS with the same hyperparameter tuning and training strategy as in §3, compute the concurrence between NACS and other datasets, and look at the effect of different splitting strategy between SCAN and NACS. The results are discussed below.

F.1 Overall Performance and Concurrence

The overall performance and concurrence including NACS are shown in Table 15 and Figure 7. The concurrence values between NACS and SCAN is surprisingly low compared to the concurrence values between NACS and other datasets, with the *length* split being the only exception, suggesting that even when the underlying tasks are the same, the datasets may provide very different model rankings. In terms of the distribution of concurrence values by type of data split pairs (Figure 8), the conclusion in §4.4 persists: the source of the dataset matters more than the interpretation of compositonality (splitting strategy).

F.2 Length Split of NACS

Out of the four splits of NACS, the *length* split is the only split that results in a high concurrence with tsplits of SCAN (Figure 7). The *length* split of SCAN and NACS is also the only length splits pair

Input:	Zoe thought that a hippo cleaned .
Output:	think. agent (x _ 1, Zoe) AND think. ccomp (x _ 1, x _ 5) AND hippo (x _ 4) AND clean. agent (x _ 5, x _ 4)
Prediction:	think. agent (x _ 1, Zoe) AND think. ccomp (x _ 1, x _ 5) AND hippo (x _ 4) AND clean. agent (x _ 5, x _ 4)

Table 11: Examples of instance where the model is only mistaken on the space.

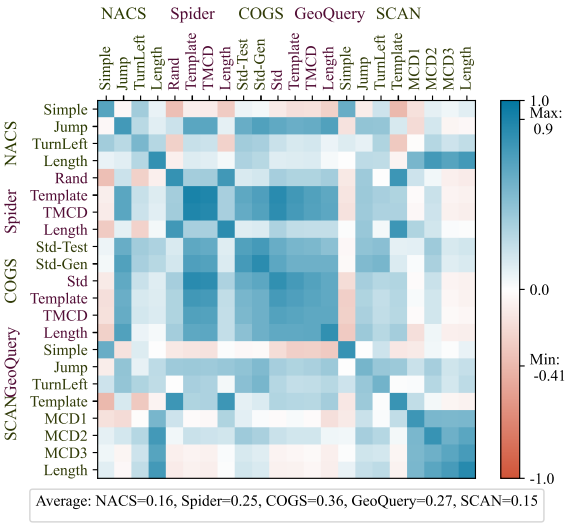

Figure 7: Distribution of concurrence values between each dataset and split pairs.

Dataset A	Dataset B	Split A	Split B	Concur
Spider	Spider	*Template*	*TMCD*	0.88
GeoQuery	Spider	*Std*	*Template*	0.84
GeoQuery	Spider	*Std*	*TMCD*	0.83
SCAN	Spider	*Template*	*Rand*	0.76
SCAN	Spider	*Template*	*Length*	0.76
Spider	Spider	*Rand*	*Length*	0.75
GeoQuery	Spider	*Template*	*Template*	0.74
SCAN	NACS	*MCD2*	*Length*	0.74
GeoQuery	Spider	*Template*	*TMCD*	0.73
SCAN	NACS	*Length*	*Length*	0.73
GeoQuery	GeoQuery	*Std*	*Template*	0.73
SCAN	SCAN	*Length*	*MCD3*	0.72

Table 13: High concurrence values (≥ 0.7) among all pairs of dataset splits, excluding self-concurrence.

that exceed the boundary set for high concurrence (Table 14). It is likely because that both *length* split of NACS and the splits that it has high concurrence with are extremely difficult split that many models fail on.

G Performance and concurrence across all setups

The performance of all models on all the curated splits for each dataset is shown in Table 15. The concurrence between all datasets and split pairs in this work is shown in Figure 9 and the exact values are included in Table 17.

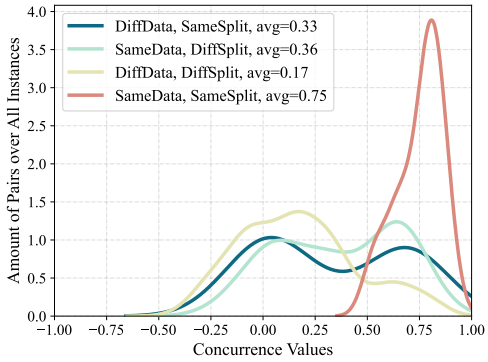

Figure 8: Distribution of concurrence values among all dataset splits. The color of the bar indicates whether the splits in the pair share the same dataset origin and/or the same splitting strategy.

Dataset A	Dataset B	Concur	Dataset A	Dataset B	Concur
SCAN	NACS	0.73	GeoQuery	NACS	0.08
COGS	GeoQuery	0.54	Spider	NACS	0.04
COGS	Spider	0.26	SCAN	Spider	0.01
COGS	NACS	0.24	COGS	SCAN	0.01
GeoQuery	Spider	0.23	GeoQuery	SCAN	-0.09

Table 14: Concurrence between length splits of datasets.

H Mistakes that model make in both random splits and generalization splits

The in-distribution performance may also be a confounder when at least one of the models does not perform as well on an in-distribution test set, or in a random split of the data. Qualitatively, we observe that models sometimes make the same trivial mistakes in both a random split and a generalization split, making the resulting raw metric unrepresentative of compositionality. For example, BART makes mistakes on parentheses, adding or dropping them on both standard split and generalization splits of GeoQuery (Table 18); BTG cannot tell left from right in the *simple* split of SCAN, and the same type of mistake continues to appear in the *template* split. While simple mistakes like these and the space tokenization issue mentioned in Section 3.4 can be easily resolved by adopting a post-processing protocol or rules to ignore when computing EM, other types of less identifiable errors may also be present and harder to patch. Since many of the models do not achieve near-perfect performance on the random splits, to what extent they

Dataset	Split	LSTM Uni		LSTM Bi		Transformer		T5		BART		BTG		Avg
COGS	*Std-Test*	99.3	±.0	99.1	±.01	99.5	±.0	99.7	±.0	99.7	±.0	68.8	±.01	94.3
	Rcvcv-Test	99.4	±.0	99.1	±.0	99.5	±.0	99.7	±.0	99.7	±.0	68.1	±.0	94.2
	Rstr-Test	99.4	±.0	99.0	±.01	99.6	±.0	99.8	±.0	99.7	±.0	68.4	±.0	94.3
	Std-Gen	21.3	±.05	14.8	±.08	56.1	±.06	82.9	±.0	78.6	±.0	2.8	±.01	42.8
	Rcvcv-Gen	22.6	±.04	10.1	±.02	57.6	±.02	50.0	±.02	44.5	±.07	0.0	±.0	30.8
	Rstr-Gen	22.3	±.07	14.7	±.03	56.6	±.03	48.0	±.01	33.5	±.03	0.0	±.0	29.2
	Length	20.7	±.01	24.9	±.01	28.7	±.02	37.9	±.0	34.1	±.01	20.5	±.0	27.8
SCAN	*Simple*	99.9	±.0	99.9	±.0	100.0	±.0	94.9	±.01	99.1	±.01	12.3	±.01	84.4
	Jump	0.4	±.01	0.0	±.0	0.1	±.0	95.0	±.01	0.4	±.01	0.0	±.0	16.0
	Template	0.2	±.0	0.3	±.01	1.1	±.0	34.3	±.03	0.0	±.0	0.9	±.01	6.1
	MCD1	5.9	±.06	12.2	±.07	1.1	±.0	24.6	±.01	0.4	±.01	1.8	±.01	7.7
	MCD2	6.7	±.03	5.8	±.03	1.2	±.0	34.1	±.01	1.6	±.0	0.5	±.0	8.3
	MCD3	8.7	±.04	7.8	±.02	0.7	±.0	11.1	±.01	1.2	±.01	0.8	±.01	5.0
	Length	15.3	±.04	11.8	±.01	0.0	±.0	14.1	±.01	0.7	±.01	0.0	±.0	7.0
	TurnLeft	61.1	±.13	34.1	±.06	64.8	±.11	70.3	±.12	63.1	±.19	8.9	±.01	50.4
	TurnLeftRcvcv	69.4	±.14	42.8	±.14	60.4	±.12	20.0	±.03	37.7	±.15	3.5	±.01	39.0
	TurnLeftRStr	59.0	±.18	43.5	±.1	61.9	±.1	17.7	±.02	23.9	±.17	2.4	±.0	34.7
NACS	*Simple*	100.0	±.0	100.0	±.0	100.0	±.0	94.6	±.0	100.0	±.0	6.1	±.01	83.5
	Jump	0.1	±.0	0.2	±.0	0.2	±.0	95.8	±.01	67.6	±.04	0.0	±.0	27.3
	TurnLeft	63.3	±.12	62.0	±.13	54.4	±.11	64.9	±.04	82.4	±.13	9.2	±.01	56.0
	Length	12.7	±.02	13.2	±.01	0.0	±.0	14.3	±.0	9.3	±.02	0.0	±.0	8.2
Spider	*Rand*	33.4	±.02	36.9	±.01	42.5	±.01	68.0	±.0	32.7	±.01	40.1	±.01	42.3
	Template	1.0	±.0	2.2	±.01	4.6	±.0	39.6	±.01	21.6	±.01	1.9	±.0	11.8
	TMCD	4.6	±.01	6.0	±.01	7.5	±.01	47.2	±.01	31.2	±.03	5.5	±.0	17.0
	Length	12.7	±.01	14.0	±.01	17.5	±.01	35.4	±.01	7.4	±.0	14.0	±.01	16.8
GeoQuery	*Std*	74.0	±.06	78.9	±.04	82.3	±.02	92.5	±.01	89.2	±.01	79.0	±.01	82.6
	Std-Rcvcv	76.7	±.03	78.9	±.02	80.5	±.01	89.4	±.0	84.2	±.0	69.0	±.03	79.8
	Std-Rstr	77.1	±.01	78.6	±.02	82.7	±.01	88.8	±.01	79.9	±.0	65.8	±.01	78.8
	Template	46.5	±.06	55.9	±.07	56.7	±.04	91.0	±.0	77.1	±.06	53.5	±.06	63.5
	Length	18.5	±.03	16.2	±.02	22.0	±.01	41.1	±.01	36.1	±.01	20.7	±.02	25.8
	TMCD	35.8	±.02	37.1	±.02	37.9	±.01	54.1	±.0	48.2	±.0	36.9	±.0	41.7
	TMCD-Rcvcv	35.9	±.01	36.7	±.01	37.5	±.0	43.3	±.0	40.8	±.01	34.3	±.0	38.1
	TMCD-Rstr	35.5	±.01	37.7	±.01	37.6	±.0	43.1	±.0	41.4	±.0	35.3	±.01	38.4

Table 15: Model exact-match accuracy on datasets averaged across random seeds, with standard deviation.

make the mistakes in the standard split again in the generalization splits requires further research.

We also include a Genbench evaluation card (Hupkes et al., 2023) in Table 19.

I Limitations

While we explore the consequences of the modeling approach on concurrence, we have focused mainly on models trained from scratch to perform compositional generalization or pretrained models which have been finetuned. Another possible area of investigation would be to explore the extent to which a model's compositional generalization abilities also transfer to in-context evaluations (Hosseini et al., 2022). We leave this question for future work.

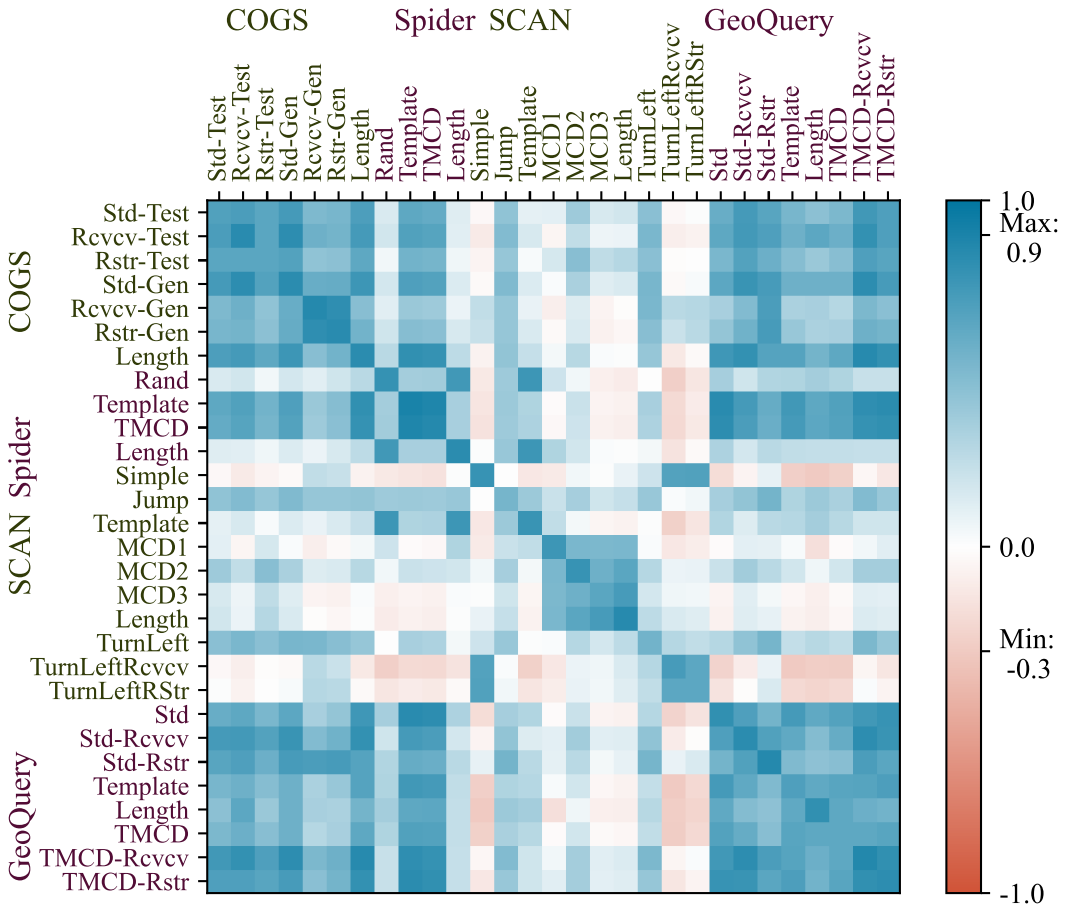

Figure 9: Distribution of concurrence values between each dataset and split pairs.

Dataset A	Dataset B	Split B	Split A	Concur	Dataset A	Dataset B	Split A	Split B	Concur
Spider	Spider	*TMCD*	*Template*	0.88	COGS	GeoQuery	*RandStr*	*TMCD-Rstr*	0.54
COGS	GeoQuery	*TMCD-Rcvcv*	*Length*	0.84	GeoQuery	SCAN	*Std-Rstr*	*TurnLeft*	0.54
GeoQuery	Spider	*Template*	*Std*	0.84	COGS	SCAN	*Std*	*TurnLeft*	0.53
GeoQuery	Spider	*Template*	*TMCD-Rstr*	0.84	COGS	SCAN	*Randcvcv*	*TurnLeft*	0.52
GeoQuery	GeoQuery	*TMCD-Rcvcv*	*Std-Rcvcv*	0.83	SCAN	SCAN	*MCD1*	*MCD2*	0.52
GeoQuery	Spider	*TMCD*	*Std*	0.83	SCAN	SCAN	*Length*	*MCD1*	0.52
COGS	GeoQuery	*TMCD-Rcvcv*	*Std*	0.82	COGS	GeoQuery	*Randcvcv*	*TMCD-Rcvcv*	0.51
COGS	COGS	*RandStr*	*Randcvcv*	0.82	GeoQuery	SCAN	*TMCD-Rcvcv*	*TurnLeft*	0.51
GeoQuery	Spider	*Template*	*TMCD-Rcvcv*	0.81	SCAN	SCAN	*MCD1*	*MCD3*	0.51
COGS	Spider	*Template*	*Length*	0.81	COGS	SCAN	*Std*	*Jump*	0.5
GeoQuery	Spider	*TMCD*	*TMCD-Rstr*	0.81	GeoQuery	GeoQuery	*Std-Rstr*	*Template*	0.5
GeoQuery	GeoQuery	*TMCD-Rstr*	*TMCD-Rcvcv*	0.81	GeoQuery	SCAN	*TMCD-Rcvcv*	*Jump*	0.49
COGS	GeoQuery	*TMCD-Rstr*	*Length*	0.8	COGS	GeoQuery	*Randcvcv*	*Std-Rcvcv*	0.49
COGS	GeoQuery	*Std-Rcvcv*	*Length*	0.8	GeoQuery	GeoQuery	*Std-Rcvcv*	*Length*	0.48
COGS	Spider	*TMCD*	*Length*	0.79	COGS	Spider	*RandStr*	*Template*	0.47
GeoQuery	Spider	*TMCD*	*TMCD-Rcvcv*	0.79	COGS	SCAN	*RandStr*	*TurnLeft*	0.47
GeoQuery	GeoQuery	*TMCD-Rstr*	*Std*	0.79	COGS	COGS	*Randcvcv*	*Length*	0.47
GeoQuery	GeoQuery	*TMCD-Rstr*	*Std-Rcvcv*	0.78	GeoQuery	GeoQuery	*Std-Rstr*	*TMCD*	0.46
COGS	GeoQuery	*Std-Rcvcv*	*Std*	0.78	COGS	GeoQuery	*Randcvcv*	*TMCD-Rstr*	0.46
COGS	COGS	*Length*	*Std*	0.76	COGS	Spider	*RandStr*	*TMCD*	0.46
SCAN	Spider	*Rand*	*Template*	0.76	GeoQuery	GeoQuery	*Std-Rstr*	*Length*	0.44
SCAN	Spider	*Length*	*Template*	0.76	GeoQuery	SCAN	*Std-Rcvcv*	*TurnLeft*	0.43
COGS	GeoQuery	*Std*	*Length*	0.75	COGS	SCAN	*Length*	*Jump*	0.43
GeoQuery	GeoQuery	*TMCD-Rcvcv*	*Std*	0.75	GeoQuery	SCAN	*Std-Rcvcv*	*Jump*	0.42
Spider	Spider	*Length*	*Rand*	0.75	COGS	GeoQuery	*RandStr*	*Std*	0.42
GeoQuery	Spider	*Template*	*Template*	0.74	COGS	SCAN	*Randcvcv*	*Jump*	0.41
GeoQuery	Spider	*TMCD*	*Template*	0.73	GeoQuery	SCAN	*TMCD-Rstr*	*TurnLeft*	0.41
GeoQuery	Spider	*Template*	*Std-Rcvcv*	0.73	COGS	SCAN	*Length*	*TurnLeft*	0.41
GeoQuery	GeoQuery	*Template*	*Std*	0.73	COGS	SCAN	*RandStr*	*Jump*	0.41
COGS	GeoQuery	*Std-Rstr*	*RandStr*	0.73	COGS	GeoQuery	*RandStr*	*Template*	0.4
COGS	GeoQuery	*TMCD-Rstr*	*Std*	0.72	GeoQuery	SCAN	*TMCD-Rstr*	*Jump*	0.4
SCAN	SCAN	*MCD3*	*Length*	0.72	SCAN	Spider	*Jump*	*Length*	0.4
COGS	GeoQuery	*Std-Rstr*	*Std*	0.72	SCAN	SCAN	*Jump*	*TurnLeft*	0.4
GeoQuery	GeoQuery	*TMCD-Rcvcv*	*Std-Rstr*	0.71	COGS	Spider	*Randcvcv*	*Template*	0.39
GeoQuery	Spider	*TMCD*	*Std-Rcvcv*	0.71	GeoQuery	SCAN	*Length*	*Jump*	0.39
COGS	GeoQuery	*Std-Rstr*	*Randcvcv*	0.7	SCAN	SCAN	*Jump*	*Template*	0.39
GeoQuery	GeoQuery	*TMCD-Rstr*	*Template*	0.7	SCAN	Spider	*Jump*	*Template*	0.39
COGS	Spider	*Template*	*Std*	0.69	SCAN	Spider	*Jump*	*Rand*	0.38
GeoQuery	GeoQuery	*Std-Rcvcv*	*Std*	0.69	SCAN	Spider	*Jump*	*TMCD*	0.38
SCAN	SCAN	*TurnLeftRStr*	*Simple*	0.68	COGS	Spider	*Randcvcv*	*TMCD*	0.38
GeoQuery	GeoQuery	*Std-Rstr*	*Std-Rcvcv*	0.68	GeoQuery	SCAN	*Std-Rcvcv*	*MCD2*	0.37
GeoQuery	Spider	*Template*	*TMCD*	0.68	Spider	Spider	*Rand*	*TMCD*	0.36
GeoQuery	Spider	*TMCD*	*TMCD*	0.68	GeoQuery	SCAN	*TMCD-Rstr*	*MCD2*	0.36
SCAN	SCAN	*TurnLeftRcvcv*	*Simple*	0.68	GeoQuery	Spider	*Length*	*Rand*	0.35
GeoQuery	GeoQuery	*TMCD*	*Std*	0.68	GeoQuery	SCAN	*TMCD-Rcvcv*	*MCD2*	0.35
COGS	Spider	*TMCD*	*Std*	0.67	Spider	Spider	*Rand*	*Template*	0.35
COGS	GeoQuery	*Std-Rstr*	*Length*	0.67	GeoQuery	SCAN	*Length*	*Template*	0.35
COGS	GeoQuery	*Template*	*Length*	0.67	GeoQuery	SCAN	*Std*	*Jump*	0.35
GeoQuery	GeoQuery	*TMCD-Rcvcv*	*Template*	0.66	GeoQuery	Spider	*Std*	*Rand*	0.35
GeoQuery	GeoQuery	*TMCD*	*Template*	0.65	SCAN	SCAN	*MCD2*	*Jump*	0.35
GeoQuery	GeoQuery	*TMCD-Rstr*	*TMCD*	0.65	COGS	GeoQuery	*RandStr*	*TMCD*	0.34
GeoQuery	GeoQuery	*TMCD-Rstr*	*Std-Rstr*	0.65	Spider	Spider	*Length*	*TMCD*	0.34
SCAN	SCAN	*MCD2*	*Length*	0.64	Spider	Spider	*Length*	*Template*	0.34
SCAN	SCAN	*TurnLeftRStr*	*TurnLeftRcvcv*	0.64	COGS	GeoQuery	*Randcvcv*	*Std*	0.34
COGS	GeoQuery	*Std*	*Std*	0.64	SCAN	Spider	*TurnLeft*	*Template*	0.34
GeoQuery	Spider	*TMCD*	*Length*	0.63	COGS	GeoQuery	*Randcvcv*	*Length*	0.34
GeoQuery	GeoQuery	*TMCD*	*Length*	0.63	GeoQuery	SCAN	*TMCD*	*Jump*	0.33
GeoQuery	GeoQuery	*TMCD-Rcvcv*	*TMCD*	0.63	COGS	SCAN	*Std*	*MCD2*	0.33
COGS	GeoQuery	*TMCD*	*Length*	0.63	COGS	GeoQuery	*Randcvcv*	*Template*	0.33
GeoQuery	Spider	*Template*	*Length*	0.63	COGS	GeoQuery	*RandStr*	*Length*	0.32
GeoQuery	GeoQuery	*Length*	*Std*	0.62	SCAN	Spider	*TurnLeft*	*TMCD*	0.32
GeoQuery	GeoQuery	*Template*	*Length*	0.62	GeoQuery	Spider	*Std*	*Length*	0.32
GeoQuery	GeoQuery	*Template*	*Std-Rcvcv*	0.62	SCAN	Spider	*Template*	*TMCD*	0.32
GeoQuery	Spider	*Template*	*Std-Rstr*	0.6	SCAN	Spider	*MCD1*	*Length*	0.31
COGS	COGS	*RandStr*	*Std*	0.6	GeoQuery	Spider	*Template*	*Rand*	0.31
GeoQuery	GeoQuery	*TMCD*	*Std-Rcvcv*	0.6	GeoQuery	SCAN	*Template*	*Jump*	0.31
COGS	COGS	*Randcvcv*	*Std*	0.59	GeoQuery	Spider	*TMCD*	*Rand*	0.31
GeoQuery	Spider	*TMCD*	*Std-Rstr*	0.58	SCAN	Spider	*Template*	*Template*	0.31
GeoQuery	GeoQuery	*TMCD-Rcvcv*	*Length*	0.57	GeoQuery	SCAN	*Std*	*Template*	0.3
COGS	GeoQuery	*TMCD-Rcvcv*	*RandStr*	0.57	GeoQuery	Spider	*Std-Rstr*	*Rand*	0.3
SCAN	SCAN	*MCD3*	*MCD2*	0.57	COGS	SCAN	*Randcvcv*	*TurnLeftRStr*	0.29
COGS	GeoQuery	*Length*	*Std*	0.56	SCAN	SCAN	*TurnLeft*	*TurnLeftRcvcv*	0.29
COGS	GeoQuery	*TMCD*	*Std*	0.56	GeoQuery	SCAN	*Template*	*Template*	0.28
COGS	GeoQuery	*Template*	*Std*	0.56	SCAN	SCAN	*MCD2*	*TurnLeft*	0.28
COGS	GeoQuery	*Std-Rcvcv*	*RandStr*	0.56	COGS	GeoQuery	*Randcvcv*	*TMCD*	0.28
GeoQuery	GeoQuery	*TMCD-Rstr*	*Length*	0.55	GeoQuery	SCAN	*Std*	*TurnLeft*	0.28
COGS	COGS	*Length*	*RandStr*	0.55	COGS	SCAN	*Length*	*MCD2*	0.28
GeoQuery	SCAN	*Jump*	*Std-Rstr*	0.54	GeoQuery	SCAN	*Length*	*TurnLeft*	0.28
COGS	GeoQuery	*Length*	*Length*	0.54	GeoQuery	SCAN	*TMCD*	*Template*	0.28
GeoQuery	GeoQuery	*Std-Rstr*	*Std*	0.54	GeoQuery	Spider	*Std-Rstr*	*Length*	0.27

Table 16: Concurrence Values.

Dataset A	Dataset B	Split A	Split B	Concur	Dataset A	Dataset B	Split A	Split B	Concur
COGS	Spider	*Length*	*Rand*	0.27	SCAN	SCAN	*Jump*	*TurnLeftRcvcv*	0.02
GeoQuery	SCAN	*Std-Rstr*	*Template*	0.27	COGS	SCAN	*Std*	*MCD1*	0.02
GeoQuery	SCAN	*Std-Rstr*	*MCD2*	0.27	SCAN	Spider	*MCD3*	*Length*	0.02
COGS	SCAN	*RandStr*	*TurnLeftRStr*	0.27	COGS	SCAN	*Length*	*MCD3*	0.02
COGS	Spider	*Length*	*Length*	0.26	GeoQuery	SCAN	*TMCD-Rcvcv*	*TurnLeftRStr*	0.02
SCAN	SCAN	*Length*	*TurnLeft*	0.25	SCAN	SCAN	*MCD1*	*TurnLeft*	0.02
GeoQuery	SCAN	*TMCD*	*TurnLeft*	0.24	SCAN	Spider	*Length*	*Length*	0.01
GeoQuery	Spider	*Template*	*Length*	0.24	COGS	SCAN	*Length*	*Length*	0.01
COGS	SCAN	*Randcvcv*	*Simple*	0.24	SCAN	SCAN	*Simple*	*MCD3*	0.01
SCAN	SCAN	*MCD1*	*Template*	0.24	SCAN	Spider	*Simple*	*Length*	0.01
SCAN	SCAN	*TurnLeft*	*TurnLeftRStr*	0.23	SCAN	SCAN	*TurnLeft*	*Template*	0.01
GeoQuery	SCAN	*Template*	*TurnLeft*	0.23	SCAN	SCAN	*Simple*	*Jump*	0.0
GeoQuery	Spider	*TMCD*	*Length*	0.23	SCAN	Spider	*TurnLeft*	*Rand*	-0.0
GeoQuery	Spider	*Length*	*Length*	0.23	COGS	SCAN	*Randcvcv*	*Length*	-0.01
GeoQuery	Spider	*TMCD-Rcvcv*	*Length*	0.22	GeoQuery	SCAN	*Std-Rcvcv*	*TurnLeftRStr*	-0.01
SCAN	SCAN	*Length*	*Jump*	0.22	GeoQuery	SCAN	*Std*	*MCD1*	-0.02
COGS	SCAN	*Length*	*Template*	0.22	SCAN	Spider	*MCD1*	*Template*	-0.02
GeoQuery	Spider	*TMCD-Rstr*	*Length*	0.22	GeoQuery	SCAN	*TMCD*	*MCD1*	-0.02
GeoQuery	Spider	*TMCD-Rcvcv*	*Rand*	0.22	COGS	SCAN	*Std*	*TurnLeftRcvcv*	-0.02
GeoQuery	Spider	*TMCD-Rstr*	*Rand*	0.21	COGS	SCAN	*RandStr*	*MCD1*	-0.02
COGS	SCAN	*RandStr*	*Simple*	0.21	COGS	SCAN	*Std*	*Simple*	-0.03
SCAN	SCAN	*MCD1*	*Jump*	0.21	COGS	SCAN	*Length*	*TurnLeftRStr*	-0.03
SCAN	Spider	*MCD2*	*Template*	0.2	SCAN	Spider	*TurnLeftRStr*	*Length*	-0.03
GeoQuery	SCAN	*Std*	*MCD2*	0.2	GeoQuery	SCAN	*TMCD*	*MCD3*	-0.03
COGS	SCAN	*RandStr*	*TurnLeftRcvcv*	0.2	SCAN	Spider	*MCD1*	*TMCD*	-0.03
SCAN	SCAN	*Simple*	*TurnLeft*	0.2	GeoQuery	SCAN	*TMCD-Rcvcv*	*Simple*	-0.04
SCAN	Spider	*MCD1*	*Rand*	0.19	GeoQuery	SCAN	*Template*	*MCD3*	-0.04
SCAN	Spider	*MCD2*	*TMCD*	0.19	GeoQuery	SCAN	*TMCD*	*Length*	-0.04
GeoQuery	Spider	*Std-Rcvcv*	*Rand*	0.18	COGS	SCAN	*RandStr*	*Length*	-0.04
GeoQuery	SCAN	*TMCD-Rcvcv*	*Template*	0.18	SCAN	SCAN	*MCD3*	*Template*	-0.05
COGS	Spider	*RandStr*	*Rand*	0.18	GeoQuery	SCAN	*TMCD-Rcvcv*	*TurnLeftRcvcv*	-0.05
GeoQuery	SCAN	*TMCD-Rstr*	*Template*	0.18	GeoQuery	SCAN	*Std-Rcvcv*	*Simple*	-0.06
SCAN	SCAN	*MCD3*	*Jump*	0.18	GeoQuery	SCAN	*Std*	*MCD3*	-0.06
GeoQuery	SCAN	*TMCD*	*MCD2*	0.18	SCAN	Spider	*MCD3*	*Template*	-0.06
SCAN	Spider	*MCD2*	*Length*	0.18	COGS	SCAN	*Randcvcv*	*MCD3*	-0.06
GeoQuery	SCAN	*Template*	*MCD2*	0.17	GeoQuery	SCAN	*TMCD-Rstr*	*TurnLeftRStr*	-0.06
SCAN	SCAN	*MCD3*	*TurnLeft*	0.17	GeoQuery	SCAN	*Template*	*Length*	-0.06
COGS	Spider	*Std*	*Rand*	0.17	COGS	SCAN	*Length*	*Simple*	-0.07
GeoQuery	Spider	*Std-Rcvcv*	*Length*	0.17	SCAN	SCAN	*Length*	*Template*	-0.07
COGS	Spider	*RandStr*	*Length*	0.15	SCAN	Spider	*MCD3*	*TMCD*	-0.07
GeoQuery	SCAN	*Std-Rstr*	*TurnLeftRStr*	0.15	GeoQuery	SCAN	*Std*	*Length*	-0.07
COGS	SCAN	*RandStr*	*MCD2*	0.15	SCAN	Spider	*Length*	*Template*	-0.07
SCAN	SCAN	*Length*	*TurnLeftRcvcv*	0.14	COGS	SCAN	*RandStr*	*MCD3*	-0.07
COGS	SCAN	*Std*	*Length*	0.14	GeoQuery	SCAN	*Length*	*MCD3*	-0.08
COGS	SCAN	*RandStr*	*Template*	0.14	SCAN	Spider	*MCD3*	*Rand*	-0.08
COGS	Spider	*Std*	*Length*	0.14	COGS	SCAN	*Randcvcv*	*MCD1*	-0.09
GeoQuery	SCAN	*TMCD-Rcvcv*	*Length*	0.14	GeoQuery	SCAN	*Length*	*Length*	-0.09
COGS	SCAN	*Std*	*Template*	0.13	SCAN	Spider	*Length*	*TMCD*	-0.09
GeoQuery	SCAN	*Std-Rcvcv*	*Template*	0.13	SCAN	SCAN	*MCD1*	*TurnLeftRStr*	-0.09
COGS	SCAN	*Randcvcv*	*MCD2*	0.13	SCAN	Spider	*Length*	*Rand*	-0.1
GeoQuery	SCAN	*TMCD-Rstr*	*Length*	0.12	SCAN	Spider	*TurnLeftRStr*	*Template*	-0.11
SCAN	SCAN	*Length*	*TurnLeftRStr*	0.12	GeoQuery	SCAN	*Std-Rcvcv*	*TurnLeftRcvcv*	-0.11
COGS	SCAN	*Std*	*MCD3*	0.12	SCAN	SCAN	*Simple*	*MCD1*	-0.11
GeoQuery	SCAN	*Std-Rcvcv*	*Length*	0.12	SCAN	Spider	*TurnLeftRStr*	*TMCD*	-0.12
GeoQuery	SCAN	*TMCD-Rstr*	*MCD1*	0.11	SCAN	Spider	*Simple*	*Rand*	-0.12
COGS	Spider	*Randcvcv*	*Rand*	0.11	COGS	SCAN	*Length*	*TurnLeftRcvcv*	-0.12
GeoQuery	SCAN	*TMCD-Rcvcv*	*MCD3*	0.11	GeoQuery	SCAN	*TMCD-Rstr*	*Simple*	-0.13
GeoQuery	SCAN	*Std-Rcvcv*	*MCD3*	0.11	SCAN	SCAN	*MCD1*	*TurnLeftRcvcv*	-0.13
GeoQuery	SCAN	*TMCD-Rstr*	*MCD3*	0.11	SCAN	SCAN	*Simple*	*Template*	-0.13
GeoQuery	SCAN	*Std-Rcvcv*	*MCD1*	0.1	SCAN	Spider	*TurnLeftRStr*	*Rand*	-0.14
GeoQuery	SCAN	*Std-Rstr*	*MCD1*	0.1	GeoQuery	SCAN	*TMCD-Rstr*	*TurnLeftRcvcv*	-0.14
GeoQuery	SCAN	*Std-Rstr*	*Simple*	0.09	SCAN	Spider	*Simple*	*Template*	-0.15
GeoQuery	SCAN	*Std-Rstr*	*Length*	0.09	SCAN	SCAN	*TurnLeftRStr*	*Template*	-0.15
GeoQuery	SCAN	*Std-Rstr*	*TurnLeftRcvcv*	0.08	SCAN	Spider	*TurnLeftRcvcv*	*Length*	-0.15
COGS	SCAN	*Randcvcv*	*Template*	0.08	GeoQuery	SCAN	*Std*	*TurnLeftRStr*	-0.15
SCAN	SCAN	*MCD2*	*TurnLeftRStr*	0.08	SCAN	Spider	*Simple*	*TMCD*	-0.16
SCAN	SCAN	*Simple*	*Length*	0.08	GeoQuery	SCAN	*Length*	*MCD1*	-0.18
COGS	Spider	*Randcvcv*	*Length*	0.07	GeoQuery	SCAN	*Std*	*Simple*	-0.19
SCAN	SCAN	*MCD2*	*TurnLeftRcvcv*	0.07	SCAN	Spider	*TurnLeftRcvcv*	*Template*	-0.2
SCAN	SCAN	*MCD3*	*TurnLeftRcvcv*	0.06	GeoQuery	SCAN	*TMCD*	*TurnLeftRStr*	-0.21
GeoQuery	SCAN	*Length*	*MCD2*	0.06	GeoQuery	SCAN	*Template*	*TurnLeftRStr*	-0.21
SCAN	SCAN	*Simple*	*MCD2*	0.05	SCAN	Spider	*TurnLeftRcvcv*	*TMCD*	-0.22
GeoQuery	SCAN	*TMCD-Rcvcv*	*MCD1*	0.05	GeoQuery	SCAN	*Length*	*TurnLeftRStr*	-0.24
SCAN	SCAN	*MCD3*	*TurnLeftRStr*	0.05	GeoQuery	SCAN	*Std*	*TurnLeftRcvcv*	-0.25
SCAN	SCAN	*Jump*	*TurnLeftRStr*	0.05	SCAN	SCAN	*TurnLeftRcvcv*	*Template*	-0.26
SCAN	Spider	*MCD2*	*Rand*	0.05	GeoQuery	SCAN	*TMCD*	*Simple*	-0.26
SCAN	Spider	*TurnLeft*	*Length*	0.05	GeoQuery	SCAN	*Template*	*Simple*	-0.27
GeoQuery	SCAN	*Std-Rstr*	*MCD3*	0.05	SCAN	Spider	*TurnLeftRcvcv*	*Rand*	-0.27
SCAN	SCAN	*MCD2*	*Template*	0.04	GeoQuery	SCAN	*Length*	*TurnLeftRcvcv*	-0.28
COGS	SCAN	*Length*	*MCD1*	0.04	GeoQuery	SCAN	*TMCD*	*TurnLeftRcvcv*	-0.29
COGS	SCAN	*Std*	*TurnLeftRStr*	0.03	GeoQuery	SCAN	*Template*	*TurnLeftRcvcv*	-0.3
GeoQuery	SCAN	*Template*	*MCD1*	0.02	GeoQuery	SCAN	*Length*	*Simple*	-0.3

Table 17: Concurrence Values (Cont).

Example 1.	BART on GeoQuery *standard* and *template*
Input:	what are the highest points of all the states
Output:	answer (highest (intersection (place , loc_2 (state))))
Prediction:	answer (highest (intersection (place , loc_2 (state)))))
Input:	what is the adjacent state of m0
Output:	answer (intersection (state , next_to_2 (m0)))
Prediction:	answer (intersection (state , next_to_2 (m0))))
Example 2.	BTG on GeoQuery *simple* and *TurnLeft*
Input:	run left thrice and look opposite right thrice
Output:	TURN_LEFT RUN TURN_LEFT RUN TURN_LEFT RUN TURN_RIGHT TURN_RIGHT LOOK TURN_RIGHT TURN_RIGHT LOOK TURN_RIGHT TURN_RIGHT I_LOOK
Prediction:	TURN_LEFT RUN TURN_LEFT RUN TURN_LEFT RUN TURN_LEFT TURN_LEFT LOOK TURN_LEFT TURN_LEFT LOOK TURN_LEFT TURN_LEFT LOOK
Input:	look right after turn left
Output:	TURN_LEFT TURN_RIGHT LOOK
Prediction:	TURN_LEFT TURN_LEFT LOOK

Table 18: Examples of instance where the model makes both mistakes in random split and generalization split. The first instance is the output of BART on *standard* split of GeoQuery, and the second entry is BART making a similar mistake on *template* split of GeoQuery; the second instance is output of BTG on *simple* split of SCAN, and a similar instance making the same directional mistake on the *TurnLeft* split.

Motivation			
Practical	*Cognitive* □ △ ◯ ⊙	*Intrinsic*	*Fairness*
Generalisation type			
Compositional □ △ ◯ ⊙	*Structural*	*Cross Task* *Cross Language* *Cross Domain*	*Robustness*
Shift type			
Covariate □ △ ◯ ⊙	*Label*	*Full*	*Assumed*
Shift source			
Naturally occuring	*Partitioned natural* □ △	*Generated shift*	*Fully generated* ◯ ⊙
Shift locus			
Train–test □ ◯	*Finetune train–test* △ ⊙	*Pretrain–train*	*Pretrain–test*

Table 19: A GenBench evaluation card (Hupkes et al., 2023) that summarizes our experiments. □= Experiments of LSTM and Transformer on GeoQuery and Spider; △= Experiments of T5 and BART on GeoQuery and Spider; ◯= Experiments of LSTM and Transformer on COGS and SCAN; ⊙= Experiments of T5 and BART on COGS and SCAN.