Structurally Diverse Sampling Reduces Spurious Correlations in Semantic Parsing Datasets

Shivanshu Gupta1 Sameer Singh1,2 Matt Gardner3
1University of California Irvine 2Allen Institute for AI 3Microsoft Semantic Machines
{shivag5,sameer}@uci.edu, mattgardner@microsoft.com

Abstract
A rapidly growing body of research has demonstrated the inability of NLP models to generalize compositionally and has tried to alleviate it through specialized architectures, training schemes, and data augmentation, among other approaches. In this work, we study a different relatively under-explored approach: sampling diverse train sets that encourage compositional generalization. We propose a novel algorithm for sampling a structurally diverse set of instances from a labeled instance pool with structured outputs. Evaluating on 5 semantic parsing datasets of varying complexity, we show that our algorithm performs competitively with or better than prior algorithms in not only compositional template splits but also traditional IID splits of all but the least structurally diverse datasets. In general, we find that diverse train sets lead to better generalization than random training sets of same size in 9 out of 10 dataset-split pairs, with over 10% absolute improvement in 5, providing further evidence to their sample efficiency. Moreover, we show that structural diversity also makes for more comprehensive test sets that require diverse training to succeed on. Finally, we use information theory to show that reduction in spurious correlations between substructures may be one reason why diverse training sets improve generalization.

1 Introduction
Systematic compositionality—expressing novel complex concepts as a systematic composition of expressions for simpler concepts—is the property underlying human languages’ immense expressive power and productivity (Lake et al., 2017; Fodor and Pylyshyn, 1988). However, NLP models, in spite of large-scale pretraining, have been shown to struggle to generalize compositionally to novel composite expressions in the context of semantic parsing (Lake and Baroni, 2018; Loula et al., 2018; Kim and Linzen, 2020).

Data augmentation has been extensively explored for compositional generalization (Akyürek et al., 2021; Guo et al., 2021; Wang et al., 2021; Guo et al., 2020; Qiu et al., 2021). However, training instances that overlap in their structure—such as abstract syntax trees (ASTs) of target programs—are sub-optimal and any augmentation mechanism that ignore this will require unnecessarily large training data to achieve compositional generalization. One way to improve sample efficiency is to train on a diverse training set—one that contains a variety of structures. The prior work exploring the efficiency of diverse training sets has been very limited in scope. Oren et al. (2021) demonstrate that template diversity improved sample efficiency in subsamples from a pool of instances as compared to random subsamples. However, they only experiment with a single dataset and given that templates themselves overlap in structure, diversifying over them can itself be inefficient. Bogin et al. (2022) demonstrate benefits of diversifying over bigrams—adjacent nodes in the AST of a logical form—which are too fine-grained. It is reasonable to expect that the optimal granularity of substructures over which to diversify would be somewhere in the middle of the extremes of templates and bigrams. Moreover, while they show benefits from diversity in compositional splits, there is limited evidence for whether they also help in the traditional IID setting.

In this study, we take a broader look at the efficacy of diversity in datasets. We propose a general, substructure-agnostic recipe for structurally diverse subsampling algorithms and use that to formulate a novel algorithm that diversifies over program subtrees. Evaluating on template and IID splits of 5 different semantic parsing datasets of varying complexities, we find that structurally diverse train sets perform best in 9 out of 10 splits with over 10 points improvements in exact match accuracy in 5. In particular, our proposed SUBLTRE
sity algorithm is the most consistent, performing better than Template and Bigram diversity in 6 out of 10 splits. Interestingly, we find that the original versions of both bigram and template diversity algorithms often perform worse than random subsampling in IID splits but improve with some simple modifications.

We further show that diverse subsampling can also be used to sample test sets that more comprehensively cover the space of instance structures and hence are far more challenging than IID splits for models trained on random train sets than for those trained on diverse sets. Finally, we use information theory to explore the improved generalization from diverse train sets and find that the weakening of spurious correlations between substructures might be one way diversity helps.

2 Setup

While the idea of diverse subsampling is applicable to any task with structured inputs or outputs, the sampling algorithms we use in this work operate on trees, and we focus in particular on abstract syntax trees from output programs found in semantic parsing tasks. In semantic parsing, an utterance $x$ has to be parsed into an executable program or logical form $z$. Further, following Oren et al. (2021) and Bogin et al. (2022), we assume access to a pool of instances $D_{pool} = \{e_1, \ldots, e_n\}$ where $e_i = (x_i, z_i)$, from which we want to subsample instances. We view each instance as a bag of substructures: $C(e_i) = C(x_i, z_i) = \{c_1^{(i)}, \ldots, c_m^{(i)}\} \subseteq C$, where $C$ is the mapping from instances to their substructures and $C$ is the set of all substructures. The substructures $c_j$ may be shared between instances.

The goal of a structurally-diverse subsampling algorithm is to select instances that contain among them as many substructures in as many different contexts as possible. When used for training, the hypothesis is that such a subsample will give the model more information about the system underlying the instances than a random subsample and hence improve generalization and sample efficiency. Intuitively, randomly sampling instances from a grammar, or from human annotators, is likely to produce skewed distributions with spurious correlations between substructures; a diverse sampling algorithm minimizes these correlations. Similarly, when subsampling to create test sets, we argue that a diverse subsample also makes for more comprehensive test sets by better covering the space of possible structures. This is analogous to why one would choose to evaluate on a test set with balanced classes even when the training data might have considerable imbalance.

3 Structurally Diverse Datasets

3.1 Substructures

Oren et al. (2021) and Bogin et al. (2022) used program templates and bigrams extracted from program ASTs. While we follow them in extracting substructures from programs alone, we choose to use different substructures as we believe templates and bigrams are not the right granularity to diversify over. Templates are too coarse; they share a lot of structure and may be large in number themselves, making it inefficient to even cover all of them. Bigrams on the other hand are too fine, and may be unable to capture many salient structural patterns. We thus use subtrees of the program AST up to a certain size $d$. The ASTs are constructed as in (Bogin et al., 2022): tokens in a program are categorized as either functions, values, or structural tokens (such as parentheses or commas) that define the hierarchical structure of the program. Figure 1 shows an example program along with the different types of substructures extracted from it, including template, bigrams and subtrees. Note that subtrees will additionally also include all isolated nodes in the AST.

3.2 Structurally Diverse Subsampling

As described in §2, the goal of structurally diverse subsampling is to sample instances that contain many different substructures in many different contexts to maximize the pressure on the model to learn the underlying compositional system. One
way to approach this would be to set it up as an optimization problem that directly selects an optimally diverse set of instances. This, however, has the challenge of having to define a measure of diversity that is also tractable to optimize given pools that may be very large, making it harder to experiment with different measures of diversity.

We thus choose to take the route of an iterative algorithm (pseudo-code in Algorithm 1) that picks instances from the pool, one-by-one until the requisite number of instances have been sampled. At a high level, in each iteration, it first selects a substructure based on some substructure-weighting scheme, $w_c$, and then selects an instance with that substructure using another instance-weighting scheme, $w_e$. It also keeps track of substructures that have been sampled ($C_{sample}$ and $T_{sample}$) and resets when all substructures have been sampled allowing it to cycle over them repeatedly. The weighting schemes can be used to encode our interpretation of diversity. Additionally, being iterative has the usual benefit of being able to stop at any time and return the currently sampled instances.

In the current state, Algorithm 1 is still just a recipe and instantiating it into a subsampling algorithm requires substructure definition and substructure and instance-weighting schemes to be specified. In the following sections we’ll describe our proposed subsampling algorithm that improves subtree diversity in the framework of this recipe and show that it also subsumes both template diversity and bigram diversity algorithms from Oren et al. (2021) and Bogin et al. (2022).

3.3 Subtree Diversity

Our subsampling algorithm diversifies over sub-trees as defined in § 3.1, i.e. $C(x, z)$ is the set of sub-trees of size $\leq d$ in the AST of $z$. In this work we use $d = 4$. Further, it uses the following substructure and instance weighting schemes:

Substructure Selection: We use the following substructure weighting function that prioritizes selecting more frequent unsampled substructures in the pool: $w_c(e) = |\{e \in D_{pool} : c \in C(e)\}|$ if $c \notin C_{sample}$ and 0 otherwise. Here $|\cdot|$ denotes set cardinality.

Instance Selection: We experiment with the following instance weighting schemes:

1. RANDEX samples an instance uniformly at random: $w_e(e) = 1$ is a constant function.

Algorithm 1 Iterative algorithm for Diverse Sub-sampling

Require: Sample pool $D_{pool}$; instance-to-substructure mapping $C$; template mapping $T$; substructure weighting function $w_c$; sample weighting function $w_e$; training budget $B$

$D_{sample} \leftarrow \phi$

$C_{sample} \leftarrow \phi$

$T_{sample} \leftarrow \phi$

$i \leftarrow 0$

while $i < B$ do

$c = \text{argmax}_{e \in C} w_c(e, D_{pool}, D_{sample}, C_{sample})$

$e = \text{argmax}_{e \in D_{pool}} w_e(e, D_{pool}, D_{sample}, T_{sample})$

s.t. $e \in C(e)$

$D_{pool} \leftarrow D_{pool} \setminus e$

$D_{sample} \leftarrow D_{sample} \cup e$

$C_{sample} \leftarrow C_{sample} \cup c$

$T_{sample} \leftarrow T_{sample} \cup \{T(e)\}$

$C \leftarrow \bigcup_{e \in D_{pool}} C(e)$

$T \leftarrow \{T(e') : e' \in D_{pool}\}$

if $C \setminus C_{sample} = \phi$ then

$C_{sample} \leftarrow \phi$

end if

if $T \setminus T_{sample} = \phi$ then

$T_{sample} \leftarrow \phi$

end if

end while

return $D_{sample}$

2. RANDNEWT samples an instance with unseen template: $w_c(e) = 1 \mathbb{I}[T(e) \notin T_{sample}]$ where $\mathbb{I}[\cdot]$ is the indicator function.

3. FREQNEWT samples instance with the most frequent unsampled template: $w_c(e) = |\{e' \in D_{pool} : T(e') = T(e)\}|$ if $T(e) \notin T_{sample}$ and 0 otherwise. This and RANDNEWT try to improve coverage over templates.

3.4 Template and Bigram Diversity

Template Diversity from Oren et al. (2021), henceforth referred to as TEMPLATE, can be implemented in the framework of Algorithm 1 as:

- $C(x, z)$ is a singleton set containing the template for $z$.
- $w_c(e) = 1 \mathbb{I}[c \notin C_{sample}]$ i.e. a uniformly random unsampled template.
- $w_e(e)$ is a constant function (RANDEX).

Bigram Diversity algorithm from Bogin et al. (2022), henceforth referred to as BIGRAM, can also
be implemented in the framework of Algorithm 1 as:

- $C(x, z)$ is the set of bigrams in $z$’s AST as defined by Bogin et al. (2022).
- Bogin et al. (2022)’s bigram diversity algorithm randomly samples from unsampled bigrams until there is still an unsampled bigram and then any random bigram. This can be formulated as: $w_c(e) = 1 \left[ c \notin \tilde{C}_{\text{sample}} \right]$ where $\tilde{C}_{\text{sample}} = \bigcup_{e \in D_{\text{sample}}} C(e)$ is the set of all sampled bigrams. Note that $\tilde{C}_{\text{sample}}$ is different from $C_{\text{sample}}$: the latter gets emptied to enable repeatedly cycling through substructures.
- $w_c(e)$ is a constant function (RANDEx).

Additionally we also experiment with using template and bigram diversity algorithms along with the substructure weighting scheme described in §3.3 that prioritizes more frequent substructures. These will be referred to as TEMPLATE[FREQ] and BIGRAM[FREQ] respectively.

4 Experiments

Given a dataset $D$ of utterance-program pairs, we first create three different types of splits, that we describe below, with each split consisting of a training pool $D_{\text{pool}}$ and test set $D_{\text{test}}$. We then compare the various subsampling algorithms described in §3 by using them to sample training sets $D_{\text{train}}$ of varying budget size $B$ from $D_{\text{pool}}$ and evaluating on $D_{\text{test}}$.

4.1 Splits

IID split For this we randomly sample some instances from $D$ to use as $D_{\text{test}}$ keeping the rest for $D_{\text{pool}}$.

Template split This is a type of compositional split proposed by (Finegan-Dollak et al., 2018). Here instances are grouped based on their program template which is obtained by applying an anonymization function to the programs that replaces certain program tokens such as strings and numbers with their abstract type. The split is then created by randomly splitting the set of templates into a train set and a test set and using examples for train or test templates as $D_{\text{pool}}$ or $D_{\text{test}}$ respectively. We follow the procedure of Bogin et al. (2022) to obtain solvable template splits i.e. there is no token in the test set but not in the training set.

Subtree split As discussed in §2, diversely sampled sets of instances may also make for more comprehensive test sets. To test this, we also evaluate on a third type of split: we use the SUBTREE[FREQNEWT] diverse subsampling algorithm to sample test sets $D_{\text{test}}$ from $D$, using the rest of the instances as $D_{\text{pool}}$.

4.2 Datasets

We use five different semantic parsing datasets for our analysis from a variety of domains with varied number of program templates and both synthetic as well as natural language input utterances. Tables 1 and 2 show a few examples and statistics regarding number of instances and different types of compounds respectively.

ATIS (Hemphill et al., 1990; Dahl et al., 1994): A dataset of natural language queries about aviation paired with $\lambda$-calculus programs.

Overnight (Wang et al., 2015): A dataset containing both synthetic and natural language utterances from 11 domains (e.g. socialnetwork, restaurants, etc.) paired with Lambda-DCS logical forms.

Schema2QA (Xu et al., 2020): Uses the ThingTalk language (Campagna et al., 2019). We use the synthetic instances from the people domain generated by Oren et al. (2021).

SM-CalFlow (Andreas et al., 2020): Consists of dialogs paired with LISP programs. Each instance is a single dialogue turn form one of two domains pertaining to creating calendar events or querying an org chart.

With the exception of SM-CalFlow which we took from Andreas et al. (2020), we used the pre-processed versions of the above datasets provided by Bogin et al. (2022) and used their code\footnote{https://github.com/benbogin/unobserved-local-structures} to anonymize programs and produce ASTs. For SM-CalFlow, we anonymized strings and numbers such as “staff meeting” and “IL” in Table 1 and used nltk\footnote{https://www.nltk.org/} to produce ASTs.
### Table 1: Examples of input utterance and target program pairs for the datasets used in this work.

| Dataset          | Input Utterance                                                                 | Target Program                                                                 |
|------------------|--------------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| ATIS             | What is the number of black dog that is chasing mouse?                         | count ( with_relation ( filter ( black , find ( dog ) ) , chasing , find ( mouse ) ) ) |
| COVR             | What is the number of black dog that is chasing mouse?                         | count ( with_relation ( filter ( black , find ( dog ) ) , chasing , find ( mouse ) ) ) |
| OVERNIGHT†      | † person whose height is 180 cm and whose birthdate is 2004                    | (listValue ( filter ( getProperty ( singleton en.person ) ( string !type ) ( string height ) ( string = ) ( number 180 en.cm ) ) ( string birthdate ) ( string = ) ( date 2004 -1 -1 ) ) ) |
| SCHEMA2QA        | what people are named aideliz li                                              | ( Person ) filter id = "aideliz li" |
| SM-CALFLOW       | When is my next staff meeting scheduled for?                                  | (Yield ( Event.start ( FindNumNextEvent ( Event.subject? ( ? = "staff meeting" ) ) ) ) ) |

### Table 2: Number of instances, bigrams, subtrees (size ≤ 4), and templates in the datasets used for structural diversity experiments.

| Dataset          | Instances | Bigrams | Subtrees | Templates |
|------------------|-----------|---------|----------|-----------|
| ATIS             | 5037      | 5091    | 41536    | 1149      |
| COVR             | 100000    | 298     | 4490     | 29141     |
| OVERNIGHT        | 4419      | 354     | 3015     | 87        |
| SM-CALFLOW       | 121910    | 43689   | 146386   | 21083     |
| SCHEMA2QA        | 1577860   | 223     | 3060     | 139       |

### 4.3 Model and Training

We use the pretrained BART-base model (Lewis et al., 2020) for our experiments, fine-tuning it on subsamples for each dataset, split and subsampling algorithm. For each dataset we use 5 different random seeds to create 5 splits of each type. Then for each split and training budget, we use 3 different seeds to subsample 3 training sets for each subsampling algorithm. See App. A for more details.

### 5 Results

#### 5.1 Structural Diversity in Train Sets

Figures 2 and 3 show the results on Template and the IID splits of the different datasets.

**Diverse subsampling is more sample-efficient than random subsampling.** Diverse subsampling outperforms random subsampling in both IID and template splits, being more efficient in 9 out of 10 dataset-split type combinations with over 10 point improvements in template splits and over 5 point improvement in IID splits of all datasets except SM-CALFLOW. The only splits where random subsampling performs better are IID splits of SM-CALFLOW. We believe this to be due to the fact that target programs in SM-CALFLOW contain free-form strings and which inflate the number of substructures (bigrams or subtrees) as shown in

**Subtree diversity is most consistent.** Among the diverse subsampling algorithms, SUBTREE diversity is the most consistent. It is more efficient than BIGRAM diversity in all datasets and splits and outperforms TEMPLATE diversity everywhere except in the template splits of ATIS, SCHEMA2QA and OVERNIGHT which all have very few templates (as shown in Table 2) and hence little to gain from diversifying over smaller substructures. These results suggest that for most tasks except for those with little structural diversity in the first place, diversification over more granular substructures would be more sample efficient.

**Prioritizing more frequent compounds improves efficiency.** The original Bigram and Template diversity algorithms (BIGRAM and TEMPLATE), while good in template splits, are worse than even random subsampling in IID splits. However, replacing their compound selection scheme with one that prioritizes frequent compounds (BIGRAM[FREQ] and TEMPLATE[FREQ]) improves both their efficiencies in IID splits with minor degradation in template splits. The only exception is the template diversity in COVR. This is expected given that it was generated from a synchronous grammar with production rules sampled uniformly at random and hence is dominated by instances with shorter templates. Thus, TEMPLATE[FREQ] will only pick these.

**Improving coverage over templates makes subtree diversity more consistent.** Comparing SUBTREE[RANDEx] and SUBTREE[RANDNEWT] we see that prioritizing examples with unseen templates improves efficiency in template splits with relatively minor reductions in IID splits.
5.2 Structural Diversity in Test Sets

Figure 4 compares the performance of random and diverse (SUBTREE[RANDNEWT]) subsamples in IID and Subtree splits. It is clear that Subtree splits are harder than IID splits. More importantly, while random training sets perform very poorly on them, diverse subsamples quickly climb close to their performance on IID splits in all datasets except ATIS. Note that this is not merely because of distributional change: these splits were created using a different variant of Subtree diverse subsampling (SUBTREE[FREQNEWT]) and bigram and template diversity algorithms also regain much of their IID performance (see App. B). We thus believe that this demonstrates comprehensiveness of diverse samples as test sets.

6 Information-theoretic Analysis

Having seen that structurally diverse datasets indeed improve generalization, we now attempt to explain why they do so. Our hypothesis is that they improve generalization by weakening the spurious correlations between substructures that would exist in any dataset of bounded size. To test this, we define a measure for correlations and compare its value for random and diverse subsamples.

Measure for Correlations

We measure correlations between substructures in a set of instances $D$ as the sum of pairwise mutual information (MI) between them, normalized to take into account the larger number of distinct substructures in diverse subsamples. Formally, as earlier, we view each instance as a bag of substructures, specifically sub-trees of size $d \leq 4$ with $S = s_1, \ldots, s_m$ as the set of all subtrees. We define an indicator random variable $I_s$ for each subtree $s$ that is 1 if $s$ is present in a random instance, 0 otherwise. MI between two subtrees, $s_i$ and $s_j$ is then defined as,

$$\text{MI}(s_i, s_j) = - \sum \frac{\tilde{p}(I_{s_i}, I_{s_j}) \log \tilde{p}(I_{s_i}, I_{s_j})}{\tilde{p}(I_{s_i})\tilde{p}(I_{s_j})}$$

where $\tilde{p}(I_s = 1)$ is the empirical probability of $s$ occurring in an instance and $\tilde{p}(I_{s_i} = 1, I_{s_j} = 1)$ is the empirical probability that $s_i$ and $s_j$ co-occur in an instance, i.e.

$$\tilde{p}(s = 1) = \frac{|\{s \in e: e \in D\}|}{|D|}$$

$$\tilde{p}(s_i = 1, s_j = 1) = \frac{|\{s_i, s_j \in e: e \in D\}|}{|D|}$$

Other probabilities are defined analogously. Finally, our measure of correlations is the average mutual information (AMI) across all subtree pairs.

$$\text{AMI}(D) = \frac{1}{|S|^2} \sum_{i,j} \text{MI}(s_i, s_j)$$

Results

For each dataset, we took 3 samples using random and subtree subsampling algorithms, treating the entire dataset as pool and computed

| Split Type | Budget | COVR | Overnight | Schema2QA |
|------------|--------|------|-----------|-----------|
| IID        | 100    | -0.50| -0.83     | -0.39     |
|            | 300    | -0.70| -0.76     | -0.56     |
| Template   | 100    | -0.57| -0.58     | -0.68     |
|            | 300    | -0.60| -0.56     | -0.64     |

Table 3: Spearman correlations of NCMI and Accuracy for train sets sampled randomly and using SUBTREE[RANDNEWT] and SUBTREE[RANDEX] algorithms averaged across 4 split seeds (from § 4).
their NCMI. Figure 5 shows that diverse subsamples have lower NCMI than random subsamples across different training budgets and the different split types, confirming our hypothesis that diverse datasets have lower correlation. Moreover, examining Spearman correlations of NCMI and accuracy (Table 3) for train sets subsampled using random, SUBTREE[RANDEX] and SUBTREE[RANDNEWT] subsampling algorithms from the experiments in §4, we find that they are significantly negatively correlated, giving further credence to our measure. These results substantiate our hypothesis that reduction of spurious correlations is indeed one way in which diverse subsamples improve generalization.

7 Related Work

Benchmarks Multiple benchmarks for compositional generalization have been created. These include those created manually specifically for the purpose of measuring generalization (Lake and Baroni, 2018; Bastings et al., 2018; Bahdanau et al., 2019; Kim and Linzen, 2020; Ruis et al., 2020; Bogin et al., 2021a). Compositional splits have also been created automatically from existing semantic parsing datasets by splitting by output length (Lake and Baroni, 2018), holding out program templates (Finegan-Dollak et al., 2018), and by maximizing compound divergence between the training and test sets (Keysers et al., 2020; Shaw et al., 2021).

Improving Generalization Many approaches to examining and improving compositional generalization have been proposed, including specialized architectures with inductive bias for compositional generalization (Herzig and Berant, 2021; Bogin et al., 2021b; Chen et al., 2020; Gordon et al., 2020; Yin et al., 2021), data augmentation (Andreas, 2020; Akyürek et al., 2021; Guo et al., 2021), modifications to training methodology (Oren et al., 2020; Csordás et al., 2021) and meta learning (Conklin et al., 2021; Lake, 2019). Data-based approaches have the advantage of being model-agnostic and hence can be used in conjunction with pretrained models. However without a more informed choice of what instances are added, given the infinite productivity of language, they would require large amounts of data to induce compositional generalization. Selecting structurally diverse instances is one such choice that leads to improved generalization and sample efficiency.

Data Selection Our work is most closely related to that of Oren et al. (2021) and Bogin et al. (2022), both of which take into account the structure of output programs in semantic parsing and select selecting structurally diverse instances for training to improve compositional generalization and sample efficiency. Our work, especially that on sampling diverse test sets, is also related to work on creating compositional splits from existing datasets (Keysers et al., 2020; Shaw et al., 2021; Bogin et al., 2022) and reducing biases in datasets via adversarial filtering or other means (Bras et al.,
and representation debiasing (Li and Vasconcelos, 2019).

8 Conclusion

In this work we broadly explored the efficacy of structural diversity for the task of semantic parsing. We proposed a novel subtree diversity algorithm that diversifies over subtrees in program abstract syntax trees. Evaluating on multiple datasets with varying complexities and logical formalisms and on both IID and compositional splits, we demonstrated that diversity almost always, and often greatly, improves generalization and sample efficiency with our proposed algorithm. We further demonstrated that structurally diverse sets of instances also make for more comprehensive test sets and showed that reduction in spurious correlations might be one reason why diverse train sets are more sample efficient.

We hope that our experiments demonstrating benefits of diversity in a wide variety of datasets would encourage future research in finding other potentially better substructures in other structure prediction tasks and sampling algorithms with better sample efficiency. In particular, the current algorithms assume the availability of a pool of labeled instances. In many semantic parsing scenarios it is possible to sample a large pool of possible outputs from which we could select diverse programs to be annotated with corresponding utterances, but this is one of the only scenarios where our current algorithms could be applied. Future research could thus look at the setup of applying diversity when there is no pool available, e.g., when generating a synthetic dataset from a synchronous grammar or by preferring structural diversity in the input space instead of the output space (if a large collection of unlabeled inputs can be obtained or generated). Another important direction of future research is to apply principles of diversity to natural language utterances for which a hierarchical representation may not be as readily available and which will also have to deal with language variability. Additionally, future work may also look at better measures of spurious correlations that we believe may lead to more efficient diverse sampling algorithms.

Finally, and quite importantly, in the context of compositional generalization, our results indicate that the apparent lack of it in NLP models may be due in part to the lack of diversity in datasets, suggesting the need to create more diverse compositional generalization benchmarks.

References

Ekin Akyürek, Afra Feyza Akyurek, and Jacob Andreas. 2021. Learning to recombine and resample data for compositional generalization. ArXiv, abs/2010.03706.

Jacob Andreas. 2020. Good-enough compositional data augmentation. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7556–7566, Online. Association for Computational Linguistics.

Jacob Andreas, John Bufe, David Burkett, Charles Chen, John Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, Hao Fang, Alan Guo, David Hall, Kristin Hayes, Kellie Hill, Diana Ho, Wendy Iwazuk, Smriti Jha, Dan Klein, Jayant Krishnamurthy, Theo Lanman, Percy Liang, Christopher H. Lin, Ilya Lintsbakh, Andy McGovern, Aleksandr Nisnevich, Adam Pauls, Dmitrij Petters, Brent Read, Dan Roth, Subroto Roy, Jesse Russak, Beth Short, Div Slomian, Ben Snyder, Stephon Striplin, Yu Su, Zachary Tellman, Sam Thomson, Andrei Vorobev, Izabela Witoszko, Jason Wolfe, Abby Wray, Yuchen Zhang, and Alexander Zotov. 2020. Task-oriented dialogue as dataflow synthesis. Transactions of the Association for Computational Linguistics, 8:556–571.

Dzmitry Bahdanau, Harm de Vries, Timothy J. O’Donnell, Shikhar Murty, Philippe Beaudoin, Yoshua Bengio, and Aaron C. Courville. 2019. CLOSURE: assessing systematic generalization of CLEVR models. CoRR, abs/1912.05783.

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.

Ben Bogin, Shivanshu Gupta, and Jonathan Berant. 2022. Unobserved local structures make compositional generalization hard. CoRR, abs/2201.05899.

Ben Bogin, Shivanshu Gupta, Matt Gardner, and Jonathan Berant. 2021a. COVR: A test-bed for visually grounded compositional generalization with real images. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 9824–9846, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ben Bogin, Sanjay Subramanian, Matt Gardner, and Jonathan Berant. 2021b. Latent compositional representations improve systematic generalization in grounded question answering. Transactions of the
Advances in Neural Information Processing Systems, 9:195–210.

Ronan Le Bras, Swabha Swayamdipta, Chandra Bhagavatula, Rowan Zellers, Matthew E. Peters, Ashish Sabharwal, and Yejin Choi. 2020. Adversarial filters of dataset biases. CoRR, abs/2002.04108.

Giovanni Campagna, Silei Xu, Mehrad Moradshahi, Richard Socher, and Monica S. Lam. 2019. Genie: A generator of natural language semantic parsers for virtual assistant commands. In Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2019, page 394–410, New York, NY, USA. Association for Computing Machinery.

Xinyun Chen, Chen Liang, Adams Wei Yu, Dawn Song, and Denny Zhou. 2020. Compositional generalization via neural-symbolic stack machines. In Advances in Neural Information Processing Systems, volume 33, pages 1690–1701. Curran Associates, Inc.

Henry Conklin, Bailin Wang, Kenny Smith, and Ivan Titov. 2021. Meta-learning to compositionally generalize. 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 3322–3335, Online. Association for Computational Linguistics.

Róbert Csordás, Kazuki Irie, and Juergen Schmidhuber. 2021. The devil is in the detail: Simple tricks improve systematic generalization of transformers. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 619–634, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunnicke-Smith, David Pallett, Christine Pao, Alexander Rudnickiy, and Elizabith Shirberg. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994.

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

Jerry A Fodor and Zenon W Pylyshyn. 1988. Connectionism and cognitive architecture: A critical analysis. Cognition, 28(1-2):3–71.

Matt Gardner, William Merrill, Jesse Dodge, Matthew Peters, Alexis Ross, Sameer Singh, and Noah A. Smith. 2021. Competency problems: On finding and removing artifacts in language data. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 1801–1813, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jonathan Gordon, David Lopez-Paz, Marco Baroni, and Diane Bouchacourt. 2020. Permutation equivariant models for compositional generalization in language. In International Conference on Learning Representations.

Demi Guo, Yoon Kim, and Alexander Rush. 2020. Sequence-level mixed sample data augmentation. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 5547–5552, Online. Association for Computational Linguistics.

Yinou Guo, Hualei Zhu, Zeqi Lin, Bei Chen, Jiantong Lou, and Dongmei Zhang. 2021. Revisiting iterative back-translation from the perspective of compositional generalization. In Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021, pages 7601–7609. AAAI Press.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The atis spoken language systems pilot corpus. In Proceedings of the Workshop on Speech and Natural Language, HLT ’90, page 96–101, USA. Association for Computational Linguistics.

Jonathan Herzig and Jonathan Berant. 2021. Span-based semantic parsing for compositional generalization. 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 908–921, Online. Association for Computational Linguistics.

Daniel Keysers, Nathanael Schärfli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarikov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. 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.

Brenden 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
Brenden M. Lake. 2019. Compositional generalization through meta sequence-to-sequence learning. CoRR, abs/1906.05381.

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. 2017. Building machines that learn and think like people. Behavioral and brain sciences, 40.

Yi Li and Nuno Vasconcelos. 2019. REPAIR: removing representation bias by dataset resampling. CoRR, abs/1904.07911.

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.

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.

Inbar Oren, Jonathan Herzig, Nitish Gupta, Matt Gardner, and Jonathan Berant. 2020. Improving compositional generalization in semantic parsing. In Findings of the Association for Computational Linguistics: EMNLP 2020, pages 2482–2495, Online. Association for Computational Linguistics.

Linlu Qiu, Peter Shaw, Panupong Pasupat, Pawel Krzysztof Nowak, Tal Linzen, Fei Sha, and Kristina Toutanova. 2021. Improving compositional generalization with latent structure and data augmentation. CoRR, abs/2112.07610.

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, volume 33, pages 19861–19872. Curran Associates, Inc.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winograd: An adversarial winograd schema challenge at scale. Commun. ACM, 64(9):99–106.

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.

Bailin Wang, Wenpeng Yin, Xi Victoria Lin, and Caiming Xiong. 2021. Learning to synthesize data for semantic parsing. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 2760–2766, Online. Association for Computational Linguistics.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1332–1342, Beijing, China. Association for Computational Linguistics.

Silei Xu, Giovanni Campagna, Jian Li, and Monica S. Lam. 2020. Schema2QA: High-Quality and Low-Cost Q&A Agents for the Structured Web, page 1685–1694. Association for Computing Machinery, New York, NY, USA.

Pengcheng Yin, Hao Fang, Graham Neubig, Adam Pauls, Emmanouil Antonios Plataniotis, Yu Su, Sam Thomson, and Jacob Andreas. 2021. Compositional generalization for neural semantic parsing via span-level supervised attention. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 2810–2823, Online. Association for Computational Linguistics.
| Budget | #Epochs |
|--------|---------|
| 50     | 160     |
| 300    | 128     |
| 600    | 96      |
| 1000   | 80      |

Table 4: Number of training epochs for COVR, ATIS, Schema2QA and Overnight depending on train set size.

A Training

Models for COVR, ATIS, Schema2QA and Overnight datasets were trained for different number of epochs depending on train set size as shown in Table 4. Models for Schema2QA were all trained for 240 epochs. Training was run with batch sizes ranging from 8 to 20, depending on the maximum number of example tokens in each dataset and a learning rate of $3e^{-5}$ with polynomial decay. Each experiment was run with a Nvidia Titan RTX GPU. Following (Bogin et al., 2022), we do early stopping using the test set. As our goal is to estimate the train set quality and not the model, we argue this is an acceptable choice in our setting.

B Full Results

Tables 5, 6, 7, 8 and 9 show results for all the subsampling algorithm, split type and budget size combinations we tried for COVR, ATIS, Schema2QA, Overnight and SM-CalIFLow respectively.
| Split Type   | Budget 50 | 100 | 300 | SUBTREE TEMPLATE | 50 | 100 | 300 | 50 | 100 | 300 | 50 | 100 | 300 |
|-------------|-----------|-----|-----|------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| RANDOM      | 56.31     | 79.61 | 95.90 | 40.27            | 69.80 | 95.96 | 32.87 | 64.25 | 92.02 |
| BIGRAM      | 47.22     | 71.98 | 96.70 | 49.80            | 80.28 | 97.99 | 35.51 | 67.29 | 94.48 |
| BIGRAM IMPROVED | 62.18 | 79.43 | 96.79 | 58.70            | 82.60 | 97.45 | 45.86 | 67.70 | 93.80 |
| TEMPLATE    | 27.59     | 46.08 | 61.56 | NaN              | NaN   | NaN   | NaN   | 50.75 | 75.76 |
| TEMPLATE IMPROVED | 43.37 | 50.27 | 63.54 | 19.95            | 24.54 | 42.15 | 6.17   | 11.58 | 41.02 |
| SUBTREE RAND | 61.22 | 86.04 | 98.37 | 41.88            | 76.77 | 98.95 | 39.09 | 73.78 | 97.25 |
| SUBTREE RAND NEW | 61.81 | 85.50 | 98.39 | 42.87            | 78.90 | 98.48 | 38.87 | 75.06 | 97.61 |
| SUBTREE FREQ NEW | 48.95 | 59.60 | 77.25 | 25.90            | 38.91 | 87.90 | 10.48 | 24.66 | 60.73 |

Table 5: Complete subsampling results for COVR.
| Split Type       | Budget 1000 | 3000 | 6000 | 1000 | 3000 | 6000 | 1000 | 3000 | 6000 |
|------------------|-------------|------|------|------|------|------|------|------|------|
|                  | HD          |      |      |      |      |      |      |      |      |
| RANDOM           | 36.38       | 48.06| 52.97| 18.54| 35.63| 44.43| 12.77| 26.49| 33.02|
| BIGRAM           | 15.49       | 29.78| 37.30| 15.78| 32.61| 40.78| 8.94 | 22.90| 30.93|
| BIGRAM IMPROVED  | 29.71       | 43.28| 47.21| 24.55| 40.50| 46.86| 15.46| 29.22| 35.68|
| TEMPLATE         | 21.04       | 32.29| 37.61| NaN  | NaN  | NaN  | 16.42| 30.80| 37.34|
| TEMPLATE IMPROVED| 31.08       | 40.74| 43.46| NaN  | NaN  | NaN  | 14.64| 31.05| 37.69|
| SUBTREE [RANDEX]| 31.78       | 43.84| 48.70| 25.48| 42.98| 48.94| 16.56| 31.19| 37.46|
| SUBTREE [RANDNEW]| 32.20       | 42.84| 47.15| 25.32| 43.11| 49.32| 16.89| 32.17| 38.59|
| SUBTREE [FREQNEW]| 29.78       | 41.66| 47.12| 27.76| 47.51| 51.60| 15.98| 31.71| 38.43|

Table 9: Complete subsampling results for SM-CalFlow.