Simple and effective data augmentation for compositional generalization

Anonymous ACL submission

Abstract

Compositional generalization, the ability to predict complex meanings from training on simpler sentences, poses challenges for powerful pretrained seq2seq models. In this paper, we show that data augmentation methods that sample MRs and backtranslate them can be effective for compositional generalization, but only if we sample from the right distribution. Remarkably, sampling from a uniform distribution performs almost as well as sampling from the test distribution, and greatly outperforms earlier methods that sampled from the training distribution. We further conduct experiments to investigate the reason why this happens and where the benefit of such data augmentation methods come from.

1 Introduction

Compositional generalization is the ability of a system to correctly predict the meaning of complex sentences when trained only on simpler sentences (Lake and Baroni, 2018; Keysers et al., 2020). It has been studied in particular detail in the context of semantic parsing, the task of mapping sentences to symbolic meaning representations. Recent findings suggest that even powerful pretrained seq2seq models such as BART (Lewis et al., 2020) and T5 (Raffel et al., 2020), which excel at broad-coverage semantic parsing (Bevilacqua et al., 2021), perform very poorly on compositional generalization (Yao and Koller, 2022).

One promising method for compositional generalization is data augmentation (Andreas, 2020; Yang et al., 2022; Qiu et al., 2022). The idea is to generate additional training data by sampling from an *augmentation distribution*, in the hope that a model trained on the augmented data will generalize better to the out-of-distribution test data. Data augmentation for semantic parsing is complicated by the fact that it needs to recombine matching pieces of the sentence and of the meaning representation, but this matching is not made explicit in the training data. Many approaches therefore use somewhat complex methods to e.g. induce synchronous grammars (Qiu et al., 2022). As a simpler alternative, Wang et al. (2021) proposed to learn only a grammar for generating meaning representations, and then to use backtranslation to map the sampled meaning representations into sentence-MR pairs.

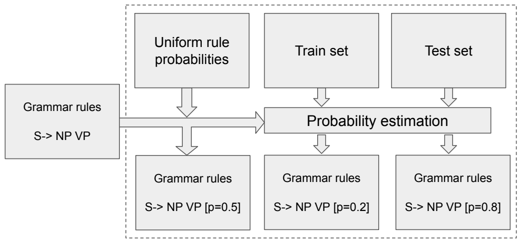


Figure 1: A diagram to show data augmentation from different distributions with PCFG.

The effectiveness of a data augmentation regime depends on the distribution from which the augmented data is sampled. Wang et al. sample from the *training* distribution and find that this improves semantic parsing accuracy on out-of-distribution text-to-SQL tasks. However, it is not clear that augmenting from the training distribution is universally helpful, especially on compositional generalization tasks where the test instances are deliberately designed to be unlikely under the training distribution.

In this paper, we investigate the impact that the choice of augmentation distribution has on the ability of a semantic parser to generalize compositionally. We compare Wang et al.'s approach (fit a grammar for meaning representations to the *training* data) to an approach where we fit the MR grammar to the *test* data (as an upper bound). Finally, we look at an MR grammar with *uniform rule weights*. Figure 1 shows the difference between these three methods. In an evaluation across four compositional generalization datasets (COGS, CFQ, GeoQuery, SCAN), we find that augmentation based on the test data strongly outperforms augmentation based on the training data; but surprisingly, augmentation with the uniform grammar is almost as effective as augmentation from the test data. This can be partially explained by the ability of the uniform grammar to contribute unseen local structures (Bogin et al., 2022) and assign low perplexity to the test MRs. Our findings point to a remarkably simple method for effective data augmentation for compositional generalization: obtain a grammar for the meaning representations (a formal language), set uniform rule weights, sample, and backtranslate.

2 Related work

Compositional generalization Compositional generalization has been shown challenging for neural sequence-to-sequence models. For example, Lake and Baroni (2018) shows that LSTM (Hochreiter and Schmidhuber, 1997) fails to generalize to new combinations or longer sequences of symbolic commands; Kim and Linzen (2020) shows that both LSTM and Transformers (Vaswani et al., 2017) cannot generalize to complex linguistic structures; Yao and Koller (2022) find that structural generalization, a difficult compositional generalization type, is consistently hard for BART and T5; Bogin et al. (2022) find that unobserved local structures can explain the difficulty of compositional generalization across multiple tasks.

Data augmentation The idea of augmenting the training data with synthetic instances originates in low-resource NLP tasks. For semantic parsing, Jia and Liang (2016) induced a synchronous CFG from the training set using domain-specific heuristics. Yu et al. (2018) and Zhong et al. (2020) generate new sentence-SQL pairs by identifying complex SQL patterns in the training set and filling their slots with different table or column names.

Data augmentation also successfully improves compositional generalization. Andreas (2020) propose a heuristic for sampling new parallel data by replacing tokens in training samples with similar tokens sharing the same context; Yang et al. (2022); Li et al. (2023) extend this idea by exchanging subtrees and spans to leverage linguistically rich phrases. Compared to their methods, we sample arbitrary meaning representations that can be derived from our hand-written grammar.

Qiu et al. (2022) propose a data augmentation procedure based on inducing probabilistic quasi-synchronous grammars from the training data. Although their system achieves promising results, it requires a complicated algorithm to induce clean grammar rules. Oren et al. (2021) also propose to sample structurally diverse synthetic data from a manually designed synchronous context-free grammar. Compared to these works, our method only considers a grammar of the meaning representation, which is easy to access.

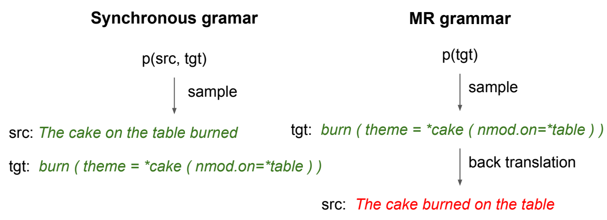


Figure 2: Comparison of different data augmentation methods based on COGS meaning representation.

Similar to our method, Guo et al. (2021) adopt iterative back-translation for compositional semantic parsing, but they directly use a subset of meaning representations from development or test set as augmented meaning representations. Our work instead shows that meaning representations generated from a probabilistic grammar still work.

Closest in spirit to this paper is the work of Wang et al. (2021), who also sample only meaning representations and generate input sentences through backtranslation. Figure 2 illustrates their method. The key difference to our work is that we explore the impact of augmentation distributions.

3 Methodology

Our method consists of two steps: sample meaning representations and then backtranslate them into natural language sentences. It exploits the fact that in many realistic use cases of a semantic parser, one can generate arbitrary amounts of symbolic *meaning representations* from a grammar: These are from a formal language, and the developer of a semantic parser either has access to a grammar for this formal language or can easily write one.

3.1 Data augmentation

Context-free grammar For a semantic parsing task, we assume as given a context-free grammar that describes all possible meaning representations. Figure 3 shows an example. Figure 3a shows part of our grammar for the GeoQuery dataset, which consists of multiple production rules. Based on these rules, we can parse a meaning representation *answer (loc_1 (cityid (houston, _)))* as shown in Figure 3b. In a probabilistic context-free grammar (PCFG), each production rule has a rule probability. The probability of a parse tree can be calculated as the product of the probability of each production rule that constitutes the parse tree.

Parameter estimation To estimate the probability of each production rule, we can use maximum likelihood estimation, which is based on counting the rule occurrences in parse trees. Given a sequence of meaning representations $y_1, \ldots, y_n$, the probability of a grammar rule $N \rightarrow \zeta$ can be calculated by the equation below, where $Count()$ denotes counting the occurrences of a rule in $y_1, \ldots, y_n$.

$$P = \frac{Count(N \rightarrow \zeta)}{\sum_\gamma Count(N \rightarrow \gamma)}$$

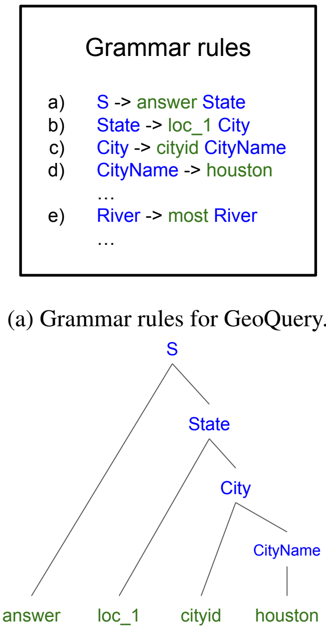
Data augmentation After estimating the rule probabilities, we can sample novel meaning representations from the resulted grammar. We then backtranslate (Sennrich et al., 2016) each sampled meaning representation to obtain the synthetic natural language text. Specifically, we train another sequence-to-sequence model on the in-distribution train set, which takes as input a meaning representation and outputs a sentence.

To utilize the generated parallel data, we can either concatenate it with the original training data, or we can first pretrain the baseline parser on the generated data and then fine-tune the parser on the original training set. Since Wang et al. (2021) show that concatenation can hurt the performance of the parser, we experiment with both methods and report results of the best method for each dataset.

3.2 Augmentation distribution for compositional generalization

Our data generation method differs from Wang et al. (2021) in that we consider different distributions for sampling the augmentation data. We hypothesize that this will be advantageous for compositional generalization, for two reasons.

First, test sets for such tasks are generally designed to contain structures that are not observed in the train set; these are difficult to sample from the training distribution. For example, in Figure 3, the rule *(e)* will be estimated to have zero probability, so the generated meaning representations will never contain the pattern *most River*. Second, the test set may involve generalization to meaning representations with deep recursion depth or longer symbol sequence. These are unlikely under the training distribution when the train set only contains shallow recursions or short sequences, and will thus be rare in the sampled data.



(a) Grammar rules for GeoQuery.

(b) Parse tree of a GeoQuery meaning representation.

Figure 3: An example to show part of our grammar from GeoQuery. Blue color refers to non-terminal and green color refers to terminal symbols. Special symbols (e.g. brackets) are ignored for space.

We compare the effect of different augmentation distributions. Specifically, we look at augmentation PCFGs whose parameters are estimated from the *training data* (P_{train}); those estimated from the *test data* (P_{test}); and PCFGs with *uniform* rule distributions ($P_{uniform}$); i.e. each of the k rules for a nonterminal N has probability $1/k$. P_{test} represents an ideal case where the test distribution is accessible, which generally does not hold for realistic scenarios. In this paper we only use P_{test} as an upper bound, to show the importance of the choice of augmentation distribution.

4 Experiments

In this section, we introduce our datasets, experiment setup and results.

4.1 Datasets

COGS COGS (Kim and Linzen, 2020) is a semantic parsing dataset where the input is an English sentence and the output is a logical form. We use the variable-free meaning representation (e.g.

A girl in a house sneezed → *sneeze (agent = girl (nmod . in = house)))* of COGS following Qiu et al. (2022). COGS is generated with a PCFG, where the train set consists of data with simple linguistic structures and the generalization set consists of 21 generalization types to test different generalization abilities. This includes 18 lexical generalization types (i.e. a novel combination of a familiar structure with a familiar word) and 3 structural generalization types (i.e. a novel combination of two familiar structures). Here "familiar" means the structure or word is observed in the train set.

We focus on the three challenging structural generalization types *obj_pp_to_subj_pp*, *pp_recursion* and *cp_recursion*, which were highlighted as particularly difficult by Yao and Koller (2022). The original train set comprises instances with prepositional phrase (PP) and clauses (CP) recursion depths limited to 2, while *pp_recursion* and *cp_recursion* instances range from depths 3 to 12. In *obj_pp_to_subj_pp* instances, PP structure modifies subject nouns, which only modifies object nouns in the train set (e.g. *Emma ate the ring beside a bed* → *A girl in a house sneezed*).

CFQ CFQ (Keysers et al., 2020) is a semantic parsing dataset where the input is an English sentence and the output is a SPARQL query (e.g. *Did M1 acquire a company* → *select count (*) where{(x0 a employer) . (M1 company_acquired x0)}*). Previous works (Herzig et al., 2021) shows that preprocessing leads to a large difference for CFQ results. Thus we use the RIR meaning representations in Herzig et al. (2021) and additionally normalize reversible relation tokens following Zheng and Lapata (2022). We use three MCD splits generated by maximizing the similarity of atom distribution and the divergence of compound distribution between train and test sets together.

Geoquery For GeoQuery, we focus on the FunQL formalism (Kate et al., 2005), where the input is an English sentence and the output is a FunQL query (e.g. *what is the tallest mountain in america* → *answer highest mountain loc_2 countryid usa*). We use the dataset created by Herzig and Berant (2021) and follow Lindemann et al. (2023) to remove special symbols in the meaning representation. We use *template* (Finegan-Dollak et al., 2018) and *length* splits created based on the program template and length respectively.

SCAN SCAN (Lake and Baroni, 2018) is a semantic parsing dataset where the input is a command and the output is a sequence of actions (e.g. *jump twice* → *JUMP JUMP*). SCAN provides many primitive-based splits and length split. We use *turnleft* and *length* split, which have been shown challenging in Qiu et al. (2022).

4.2 Set up

Models. We address all our semantic parsing tasks with a sequence-to-sequence model. Given its strong performance on semantic parsing and sentence generation tasks, we fine-tune T5 (Raffel et al., 2020) as our baseline semantic parser as well as for backtranslation. Training details are reported in Appendix B. All our results are averaged over 5 random runs and we report standard deviation in Appendix C. Exact match accuracy is used as the evaluation metric for all datasets. For GeoQuery, the same input sentence can be mapped into multiple correct programs, so we also report execution accuracy following Herzig and Berant (2021).

Grammars. To apply our data augmentation method to a dataset, we need a context-free grammar that can generate its meaning representations. For COGS, we adopt the official grammar provided by authors. For CFQ, GeoQuery and SCAN, we manually write a context-free grammar to apply our method. We use T5+P_{train} to refer to the model trained with the union of original train set and the data sampled from P_{train} and so for the other distributions. For all three augmentation distributions, we sample the same number of unique meaning representations. Details of grammar design and sampling are described in Appendix D. For COGS and SCAN, we directly concatenate the synthesized data with the original data set. For CFQ and GeoQuery, we find that concatenation hurts the performance and thus pretrain the model on the synthesized data first and then fine-tune it on the original train set. We report detailed results for both settings in Appendix C.

4.3 Results

COGS Table 1 shows exact match accuracies on COGS. We observe that the distribution of the augmented meaning representations makes a large difference on the performance: the grammar estimated on the test set (e.g. P_{test}) substantially improves performance (+8.3) and achieves near-perfect accuracy overall, while the grammar estimated on

Models	Obj	PP	CP	All
T5 (Qiu et al., 2022)	-	-	-	89.8
LeAR ♠ (Liu et al., 2021)	92.5	**100**	**98.5**	98.9
SpanSub ‡(Li et al., 2023)	-	-	-	92.3
T5+CSL ‡ (Qiu et al., 2022)	-	-	-	**99.5**
T5	88.2	24.1	32.3	91.0
$+P_{train}$ ‡	89.4	51.2	43.5	92.9
$+P_{test}$ ‡	**94.6**	96.7	95.1	99.3
$+P_{uniform}$ ‡	92.9	87.8	50.7	95.9

Table 1: Results on COGS. *Obj, PP, CP* refers to structural generalization types *obj_pp_to_subj_pp*, *pp_recursion* and *cp_recursion* respectivly. ‡refers to parsers using data augmentation method. ♠refers to structured parsers.

Models	MCD1	MCD2	MCD3	Avg
T5 (Herzig et al., 2021)	85.8	64.0	53.6	67.8
T5-large (Herzig et al., 2021)	88.6	79.2	72.7	80.2
T5-3B (Herzig et al., 2021)	88.4	85.3	77.9	83.8
LeAR ♠ (Liu et al., 2021)	91.7	89.2	91.7	90.9
Least-to-Most (Drozdov et al., 2022)	**94.3**	**95.3**	**95.5**	**95.0**
T5	89.9	75.3	72.2	79.1
$+P_{train}$ ‡	89.9	77.9	75.8	81.2
$+P_{test}$ ‡	90.4	79.1	75.5	81.7
$+P_{uniform}$ ‡	91.2	78.8	74.3	81.4
+dev MRs ‡	87.1	89.5	89.3	88.6

Table 2: Results on CFQ. *+dev MRs* refers to using meaning representations from development set for our data augmentation method.

	Template		Length	
Models	EM	Exe	EM	Exe
BART (Herzig and Berant, 2021)	-	67.0	-	19.3
Span+lexicon ♠ (Herzig and Berant, 2021)	-	82.2	-	63.6
LeAR ♠ (Liu et al., 2021)	-	84.1	-	-
SUBS (gold tree) ‡(Yang et al., 2022)	88.3	-	-	-
SpanSub (gold tree) ‡(Li et al., 2023)	**89.5**	-	-	-
T5	73.9	79.9	35.8	50.5
$+P_{train}$ ‡	74.1	84.3	56.1	72.1
$+P_{test}$ ‡	80.1	**88.2**	60.1	**74.1**
$+P_{uniform}$ ‡	79.3	87.6	**60.4**	73.7

Table 3: Results on GeoQuery. *EM* denotes exact match accuracy and *Exe* denotes execution accuracy.

Models	Turnleft	Length
T5 (Qiu et al., 2022)	62.0	14.4
T5+GECA ‡ (Qiu et al., 2022)	57.6	10.5
T5+CSL ‡ (Qiu et al., 2022)	**100**	**100**
T5	61.2	4.4
$+P_{train}$ ‡	92.9	8.1
$+P_{test}$ ‡	92.9	60.5
$+P_{uniform}$ ‡	92.9	60.5

Table 4: Results on SCAN.

the train set (e.g. P_{train}) only slightly improves the performance (+1.9). We consider this is because the grammar estimated on train set tends to produce simple structures, which does not help improve complex structure predictions. Noticeably, the uniform grammar $P_{uniform}$ yields a much higher improvement than P_{train}. This suggests that the importance of the distribution of meaning representations for compositional generalization.

CFQ Table 2 shows exact match accuracies on CFQ. All three augmentation strategies are roughly on par with each other. We attribute this limitation to the fact that the CFQ dataset is generated by mapping intermediate logical forms into SPARQL, which incorporates variables and conjuncts. Such complex relationships are difficult to capture accurately using context-free grammars, resulting in many sampled meaning representations containing nonsensical elements (e.g., redundant conjuncts).

To verify our hypothesis, we further experiment with a setting where instead of sampling MRs from estimated PCFG, we directly backtranslate MRs from development set as augmented data. Since the development set of CFQ shares the same distribution as the test set, this setting represents what a perfect method for augmenting from the test distribution would achieve, illustrating that the issue really comes from our flawed grammar.

We also observe that our T5 baseline outperforms the T5 model from Herzig et al. (2021). We attribute this to the additional preprocessing steps we adopted from Zheng and Lapata (2022).

GeoQuery Table 3 shows exact match accuracies and execution accuracy on GeoQuery. On the template split, P_{test} gives the best performance (+6.2 EM and +8.3 Exe). On the length split, all three strategies substantially improve the performance. $P_{uniform}$ achieves on-par performance with P_{test} and outperforms P_{train} on both splits, which is consistent with the results on COGS.

SCAN Table 4 shows exact match accuracies on SCAN. We observe P_{test} and $P_{uniform}$ substantially improve the performance on both splits, whereas the P_{train} only performs well on *turnleft* split. All three strategies achieves the same performance on *turnleft* split. This is because the meaning representation space of SCAN is too small and thus all possible meaning representations can be sampled by three strategies, which results in the

Datasets		English			Meaning representations		
		Avg length	Bigrams(%)	Instance(%)	Avg length	Bigrams(%)	Instance(%)
COGS	T5	7.5	30.5	0	13.8	88.7	0
	$+P_{train}$	7.5	37.8	0	13.8	93.2	0
	$+P_{test}$	8.4	53.4	11.7	17.5	99.5	11.8
	$+P_{uniform}$	8.5	40.5	0	17.0	99.3	0.2
CFQ MCD1	T5	13.5	91.2	0	44.3	98.9	6.5
	$+P_{train}$	13.7	99.6	0.4	46.9	99.6	7.0
	$+P_{test}$	14.0	99.7	0.6	45.9	100	7.5
	$+P_{uniform}$	7.1	99.4	0.2	36.4	100	7.0
GeoQuery Template	T5	8.3	66.5	0	6.1	74.8	0
	$+P_{train}$	9.6	76.5	27.2	8.3	85.9	45.5
	$+P_{test}$	10.3	78.2	29.8	8.8	100	60.3
	$+P_{uniform}$	9.4	76.7	25.0	8.4	100	26.5
SCAN Length	T5	7.0	100	0	10.8	100	0
	$+P_{train}$	7.1	100	8.6	11.1	100	20.7
	$+P_{test}$	7.1	100	9.6	12.2	100	100
	$+P_{uniform}$	7.1	100	9.6	12.2	100	100

Table 5: Dataset statistics for different augmentation strategies. *T5* denotes the statistics of the original train set. $+P_{train}$, $+P_{test}$, $+P_{uniform}$ denote augmented datasets based on different PCFGs. We report the statsitics for both input sentence side and output meaning representation side. Thus, *Avg length* under *English* tab refers to the average length of input sentences. We report three statistics: average length (*Avg length*), the coverage (expressed as a percentage) of bigrams in the test set by the training set (*Bigrams*) and the coverage of entire instance in the test set by the training set (*Instance*).

same train set. The same case happens for P_{test} and $P_{uniform}$ on the length split. Noticeably, our method outperforms GECA, which generates parallel data for data augmentation using templates. This suggests that sampling meaningful and useful meaning representations proves more effective than sampling limited parallel data in certain scenarios.

5 Discussion

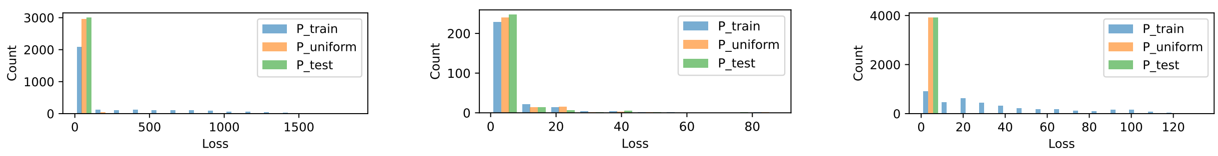
The surprising finding so far is that across all four compositional generalization datasets, augmenting from $P_{uniform}$ performs on par with P_{test}. This seems counterintuitive: the uniform augmentation strategy has no knowledge of the test data's distribution, and one would expect that augmentation data sampled from a grammar-based approximation to the test distribution should perform much better. We therefore investigate this finding in detail.

Augmentation data statistics We present statistics of the generated augmentation data in Table 5. For each corpus and augmentation method, we show the average sequence length, bigram coverage, and instance (i.e. exact sequence match) coverage for both input sentences and output MRs. The bigram coverage is determined by dividing the number of observed bigrams in the test set that also exist in the training set by the total count of possible bigrams in the test set. Instance coverage is calculated analogously.

As expected, P_{test} always yields the highest coverage values on the meaning representations, suggesting that the MR grammar approximates the test distribution effectively. On the other hand, instance-level coverage on the English side does not grow very high for any dataset. This indicates that the backtranslation model, which is trained on the original in-distribution data, still struggles to produce novel recombinations of the English sentences.

$P_{uniform}$ is on par with P_{test} on many measures and datasets, and considerably outperforms P_{train}. This suggests that novel structural combinations are judged unlikely based on the training distribution, or are simply assigned a probability of zero because structures were entirely unobserved.

It is remarkable that P_{test} and $P_{uniform}$ produce meaning representations of similar length on COGS and could therefore be capable of generating augmentation data of similar structural complexity. At the same time, P_{test} achieves a significantly higher parsing accuracy on the PP and CP recursion generalization types. A plot of the distribution of the augmentation instances according to recursion depth (Fig. 5) reveals that while P_{test} generates augmentation instances evenly across all recursion depths, $P_{uniform}$ emphasizes moderately



(a) COGS structural types (b) GeoQuery Template split (c) SCAN Length split

Figure 4: Count of test instances with regard to different loss values.

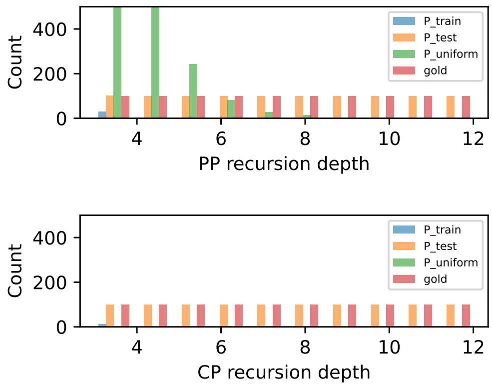


Figure 5: Depth distribution of train set for COGS.

(PP) or extremely (CP) shallow instances.[1] This explains the difference in parsing accuracy, and further emphasizes that compositional generalization is not just challenging because transformers struggle when generalizing to longer inputs (Hupkes et al., 2020), but also to structurally more complex inputs of similar length.

Perplexity analysis We further investigate whether $P_{uniform}$ produces useful augmentation data simply because it produces arbitrary instances of higher complexity than P_{train}, or if $P_{uniform}$ actually models the test distribution in some way. To this end, we measure the perplexity of the meaning representations of the test set across four corpus variants under each model (Table 6; see Appendix E.1 for details).

We find that across three of the four datasets, P_{test} and $P_{uniform}$ are close together, considerably outperforming P_{train} and the T5 baseline. An exception is CFQ, where the grammar introduces so much noise into the sampling process that all models are mostly on par. We consider this is because although $P_{uniform}$ has no particular knowledge of the test distribution built in, sampling from it covers enough MR n-grams that the test data becomes predictable.

	COGS	CFQ	GeoQuery	SCAN
Models		MCD1	Template	Length
T5	1.131	1.007	1.254	1.427
$+P_{train}$ ‡	1.133	1.005	1.252	1.124
$+P_{test}$ ‡	1.001	1.005	1.166	1.006
$+P_{uniform}$ ‡	1.007	1.005	1.184	1.006

Table 6: Perplexity of models with different augmentation strategies on test set.

The increased perplexity of P_{train} in comparison to the other models is not evenly distributed across the test instances. In Fig. 4, we plot a count of test instances for each loss value. Compared to P_{test} and $P_{uniform}$, the loss of P_{train} on some instances becomes exceptionally high, which results in higher perplexity and lower accuracy on such instances. Looking into the dataset, we find that such issue generally occurs on meaning representations with complex structures (e.g. deeper recursions for COGS and unseen program templates for GeoQuery). These structures are more predictable for models trained on P_{test} and $P_{uniform}$ augmentation data, which contains such structures more frequently.

Structure coverage According to Bogin et al. (2022), a key feature that makes compositional generalization difficult is the presence of unobserved local structures (i.e. a connected sub-graph that occurs in the meaning representation) in the test set. Is the better performance and perplexity of P_{test} and $P_{uniform}$ actually because they cover more structures in the test set?

To answer this question, we further plot the accuracy of our models against the structure coverage on COGS and GeoQuery in Figure 6. Here "structure coverage" refers to dividing the number of observed structure in the test set that also exist in the training set by the total count of possible structures in the test set. For GeoQuery, we consider the template split and follow Bogin et al. (2022) in

[1]We hypothesize that the difference between PP and CP in the $P_{uniform}$ case is due to the fact that each level of CP recursion requires the use of two production rules, rather than just one for PP, making the generation of deeper structures comparatively less likely.

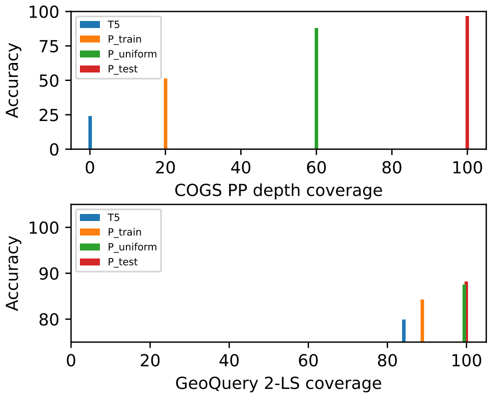


Figure 6: Performance against the local structure coverage for different augmentation distributions.

Input	what is the length of the river that runs through the most states ?
Gold	`len` **`most river`** `traverse_2 state all`
T5	`len intersection riverid most state all`
$+P_{train}$	`len intersection river traverse_2 most state all`
$+P_{test}$	`len most river traverse_2 state all`

Table 7: Examples from GeoQuery test set.

	COGS	CFQ	GeoQuery	SCAN
Metric	Struct	MCD1	Template	Length
Exact Match	30.9	4.8	19.6	8.6
BLEU	78.5	42.9	61.6	51.7

Table 8: Results of our backtranslation model on the test sets for each task. *Struct* under *COGS* means we only calculate the metric on structural generalization types.

defining the local structure of a meaning representation as all pairs of parent nodes and their children in its parse tree (i.e. 2-LS). For COGS, we focus on the PP recursion generalization type. Instead of considering local structures, we observe that the accuracy on such data is related to the maximal recursion depth observed in the train set. Thus we use PP recursion depth as a representative of global structures to calculate the structure coverage.

Our results show that P_{test} and $P_{uniform}$ yields a larger coverage of structures that occur in the test set than P_{train}. Furthermore, larger coverage is associated with higher accuracy. This is consistent with Bogin et al. (2022). Although Gupta et al. (2022) and Oren et al. (2021) also show the benefit of introducing more complex structures into the train set, our results further suggest that synthesized meaning representations with back-translated sentences can still help.

Qualitative error analysis Finally, we conducted a qualitative analysis to identify specific cases in which our approach led to improvements. In Table 7, the grammar rule *River* $\rightarrow$ *most River* is not observed by baseline and P_{train}, and thus the model struggles generating the bigram *most river* corresponding to this rule, which leads to a large loss value (i.e. 26.1) for this instance. In contrast, P_{test} covers all local structures, which allows the model to predict the instance correctly with a substantially lower loss (i.e. 0.1).

Sentence generation We report the performance of our backtranslation model in Table 8. Both exact match accuracy and BLEU score (Papineni et al., 2002) are used as evaluation metrics. All models achieve good BLEU scores, indicating the effectiveness of our backtranslation models. However, none of the model yields high accuracy, which suggests that our model can still learn to utilize such noisy data to achieve better performance.

6 Conclusion

We investigated the impact of the choice of augmentation distribution on compositional generalization. We found that a PCFG for the meaning representations with uniform rule weights supports much more effective data augmentation than one that is trained on the training data, and almost on part with one that is trained on the test data. A detailed analysis revealed that this is because the uniform grammar both achieves low perplexity on the test meaning representations and greatly improves structural coverage.

Thus, sampling meaning representations from a uniform PCFG and backtranslating them into natural-language sentences can serve as a simple and efficient data augmentation strategy for compositional generalization. It would be interesting to investigate the space of augmentation distributions in more detail in future work to see, for instance, how the generation of structurally even more diverse augmentation instances can be encouraged.

7 Limitations

Our work assumes that the language of all possible meaning representations can be described with a context-free grammar, and that such a grammar is available or can be easily reconstructed by hand. Given that MRs are a formal language, this seems realistic, but can involve some manual effort. When the meaning representations are generated out of a knowledge base through a process that is not publicly accessible, such as in CFQ, hand-crafting a grammar for MRs can introduce noise.

In our evaluation, we use corpora that are either synthetic (COGS, CFQ, SCAN) or very small (GeoQuery). Thus, one should interpret conclusions on data augmentation for such corpora with care. To our knowledge, there are no compositional generalization datasets that use naturally occurring language. The robustness of our results across corpora still suggests the generality of our findings.

References

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.

Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. 2021. One SPRING to rule them both: Symmetric AMR semantic parsing and generation without a complex pipeline. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI-21)*, volume 35, pages 12564–12573. AAAI Press.

Ben Bogin, Shivanshu Gupta, and Jonathan Berant. 2022. Unobserved local structures make compositional generalization hard. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2731–2747, Abu Dhabi, United Arab Emirates. 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.

Andrew Drozdov, Nathanael Schärli, Ekin Akyürek, Nathan Scales, Xinying Song, Xinyun Chen, Olivier Bousquet, and Denny Zhou. 2022. Compositional semantic parsing with large language models.

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.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

Jiaqi Guo, Qian Liu, Jian-Guang Lou, Zhenwen Li, Xueqing Liu, Tao Xie, and Ting Liu. 2020. Benchmarking meaning representations in neural semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1520–1540, Online. Association for Computational Linguistics.

Yinuo Guo, Hualei Zhu, Zeqi Lin, Bei Chen, Jian-Guang Lou, and Dongmei Zhang. 2021. Revisiting iterative back-translation from the perspective of compositional generalization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7601–7609.

Shivanshu Gupta, Sameer Singh, and Matt Gardner. 2022. Structurally diverse sampling for sample-efficient training and comprehensive evaluation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4966–4979, Abu Dhabi, United Arab Emirates. 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.

Jonathan Herzig, Peter Shaw, Ming-Wei Chang, Kelvin Guu, Panupong Pasupat, and Yuan Zhang. 2021. Unlocking compositional generalization in pre-trained models using intermediate representations. *Computing Research Repository (CoRR)*, arXiv:2104.07478.

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

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.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany. Association for Computational Linguistics.

Rohit J Kate, Yuk Wah Wong, Raymond J Mooney, et al. 2005. Learning to transform natural to formal languages. In *AAAI*, volume 5, pages 1062–1068.

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 (ICLR)*.

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.

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

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2873–2882, Stockholmsmässan, Stockholm Sweden. 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.

Zhaoyi Li, Ying Wei, and Defu Lian. 2023. Learning to substitute spans towards improving compositional generalization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2791–2811, Toronto, Canada. Association for Computational Linguistics.

Matthias Lindemann, Alexander Koller, and Ivan Titov. 2023. Compositional generalisation with structured reordering and fertility layers. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2172–2186, Dubrovnik, Croatia. Association for Computational Linguistics.

Chenyao Liu, Shengnan An, Zeqi Lin, Qian Liu, Bei Chen, Jian-Guang Lou, Lijie Wen, Nanning Zheng, and Dongmei Zhang. 2021. Learning algebraic recombination for compositional generalization. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1129–1144, Online. 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.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Linlu Qiu, Peter Shaw, Panupong Pasupat, Pawel Nowak, Tal Linzen, Fei Sha, and Kristina Toutanova. 2022. Improving compositional generalization with latent structure and data augmentation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4341–4362, Seattle, United States. Association for Computational Linguistics.

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

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for 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.

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.

Jingfeng Yang, Le Zhang, and Diyi Yang. 2022. SUBS: Subtree substitution for compositional semantic parsing. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 169–174, Seattle, United States. Association for Computational Linguistics.

Yuekun Yao and Alexander Koller. 2022. Structural generalization is hard for sequence-to-sequence models.

Dataset	lr	batch_size	weight_decay	steps
COGS	1e-5	2048	0	25k
CFQ	7.4e-5	2048	1e-3	50k
GeoQuery	1e-5	4096	1e-3	10k
SCAN	1e-5	1024	1e-3	25k

Table 9: Hyperparmeters of baseline models used in our experiments. Batch size is quantified in terms of input tokens. *batch_size* refers to the batch size during training. *weight_decay* refers to the weight decay used in the optimizer. *lr* refers to the learning rate. *steps* refers to the training steps we used to train the model.

Dataset	Time (hours) *w.o.* aug	Time (hours) *w.* aug
COGS	7	8
CFQ	10	10
GeoQuery	0.5	1
SCAN	20	4

Table 10: Training time for our model on each dataset (1 run) in our experiments.

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.

Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. 2018. SyntaxSQLNet: Syntax tree networks for complex and cross-domain text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1653–1663, Brussels, Belgium. Association for Computational Linguistics.

Hao Zheng and Mirella Lapata. 2022. Disentangled sequence to sequence learning for compositional generalization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4256–4268, Dublin, Ireland. Association for Computational Linguistics.

Victor Zhong, Mike Lewis, Sida I. Wang, and Luke Zettlemoyer. 2020. Grounded adaptation for zero-shot executable semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6869–6882, Online. Association for Computational Linguistics.

A Dataset details

We report dataset statistics in Table 11. COGS provides both an in-distributional test set (i.e. test) and an out-of-distributional test set (i.e. gen). For the splits of CFQ, GeoQuery and SCAN we used no in-distributional test set is provided.

B Training details

B.1 Evaluation metrics

For all tasks, we report exact match accuracy of our model, which means that the output sequence is correct only if each output token is correctly predicted. For GeoQuery, we additionally report execution accuracy, which means we execute generated FunQL code and calculate the accuracy of the outputs. This metric can better measure the generalization ability of our model since one input sentence can be mapped into multiple correct FunQl queries. For example, *how long is the rio grande river* can be parsed into either *answer (len (river (riverid (rio grande))))* or *answer (len (intersection (riverid (rio grande), (river (all)))))*. Both queries return the correct value. We use the code from (Herzig and Berant, 2021) to calculate the execution accuracy.

B.2 Hyperparameters

Baseline. We use *t5-base*[2] (220 million parameters) as our baseline for all experiments. We use the default subword vocabulary and do not extend it with new words. We use Adam (Kingma and Ba, 2015) as our optimizer. Since (Csordás et al., 2021) shows that early stopping based on in-distribution validation set leads to low performance on out-of-distribution test set, we do not apply early stopping for COGS, GeoQuery and SCAN and only use the checkpoint at the end of training, following (Herzig et al., 2021). CFQ provides out-of-distribution development set, so we use exact match accuracy on the development set as the validation metric. No learning rate scheduler is used for all experiments. During evaluation, we use beam search with beam size 4. Task-specific hyperparameters are present in Table 9.

Data augmentation. We maximally sample 21k, 100k, 30k and 10k unique meaning representations for COGS, CFQ, GeoQuery and SCAN respectively. For SCAN, we find that the size of possible meaning representations is small (i.e. 9228 unique meaning representations) and thus we sample all possible unique meaning representations from our PCFG.

B.3 Other details

We use Allennlp (Gardner et al., 2018) for our implementation. Experiments are run on Tesla A100

[2] https://huggingface.co/t5-base

Dataset	Split	# train	# dev.	# test	# gen	Vocab. size	Train len.	Test len.	Gen len.
COGS	-	24155	3000	3000	21000	809	22/48	19/40	61/144
CFQ	MCD1	95743	11968	11968	-	171	29/133	30/103	-
	MCD2	95743	11968	11968	-	171	29/133	30/103	-
	MCD3	95743	11968	11968	-	171	29/133	30/103	-
GeoQuery	template	544	60	276	-	324	23/17	19/12	-
	length	540	60	280	-	324	13/7	23/17	-
SCAN	Turnleft	21890	-	1208	-	19	9/48	8/27	-
	length	16990	-	3920	-	19	9/22	9/48	-

Table 11: Statistics for all our datasets. # denotes the number of instances in the dataset. Vocab.size denotes the size of vocabulary for the dataset, which consists of input tokens and output tokens. Train.len denotes the maximum length of the input tokens and output tokens in the train set. Test.len and Gen.len denote the maximum length in the test and generalization set.

GPU cards (80GB). Table 10 shows the training time cost.

C Detailed results

We report detailed experimental results in Table 15. Both means and standard deviations are reported over 5 runs for each model. As discussed in Section 3, we experiment with two ways to utilize synthesized data: concatenating them with the original train set and pretrain the model on them first and then finetune on the original train set. We report numbers for both settings, with *Concat* refers to concatenation and *Pretrain* refers to pretraining the model first and then fine-tuning it.

D Grammar details

We use PCFG provided in Kim and Linzen (2020) for COGS and hand-written grammars for CFQ, GeoQuery and SCAN. In this section, we mainly introduce details of our used grammar for these three hand-written grammars.

D.1 Grammar design

GeoQuery The meaning represnetation of GeoQuery is based on FunQL. Following the definations of FunQL [3], we can easily write a context-free grammar for it. We adopted the FunQL grammar used in (Guo et al., 2020) and extends it with some rules to fit our dataset. A selection of our context-free grammar rules are shown in Table 12.

SCAN SCAN is a synthetic dataset generated by Lake and Baroni (2018). They generate the dataset by generating commands (i.e. input sentences) first and then translating commands into action sequences (i.e. meaning representations) with a translation function. Instead, we write a context-free grammar for meaning representations. A selection of our context-free grammar rules are shown in Table 13.

CFQ CFQ is a synthetic dataset generated by Keysers et al. (2020). They generate the natural language sentences and corresponding intermediate logical forms first, and then apply multiple rules to obtain the SPARQL meaning representations.

S → answer (Var)
Var → City
Var → Place
Var → State
City → CityNonterm
City → CityTerm
CityNonterm → city (City)
CityNonterm → loc_2 (State)
CityTerm → city (all)
CityTerm → capital (all)

Table 12: Part of our FunQL grammar.

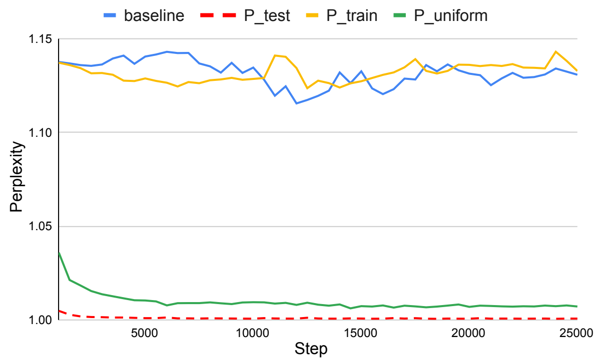
S → Command
Command → Walk_command
Walk_command → Walk_actions
Walk_actions → LWalk
LWalk → Turn_left Walk
Turn_left → i_turn_left
Walk → i_walk

Table 13: Part of our SCAN grammar.

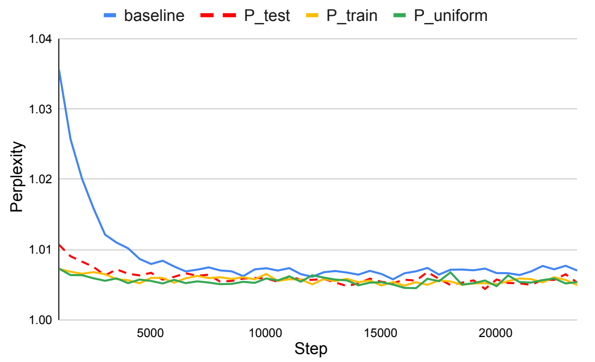
S → Prefix Main
Main → lb Conjuncts rb
Conjuncts → Conjuncts . Conjunct
Conjuncts → Conjunct
Conjunct → Unary_relation
Unary_relation → (Var a Film_unary_arg)
Film_unary_arg → film.film
Var → M0

Table 14: Part of our SPARQL grammar.

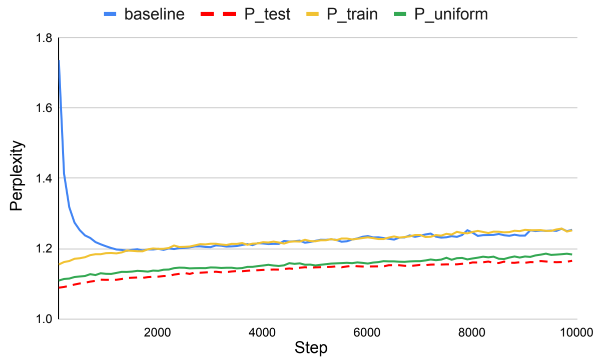
[3] https://www.cs.utexas.edu/~ml/wasp/geo-funql.html



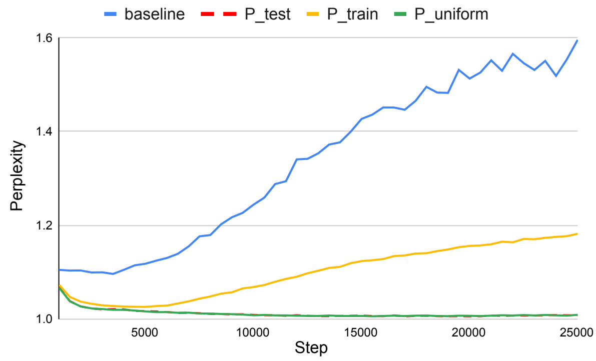
(a) COGS



(b) CFQ MCD1

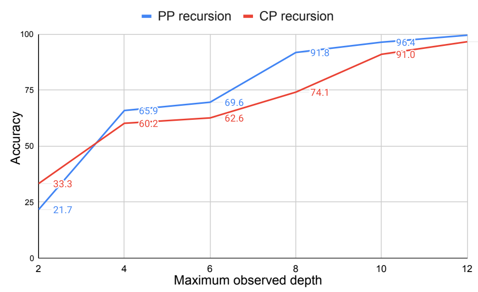


(c) GeoQuery Template split

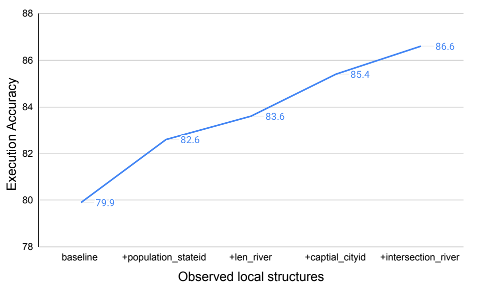


(d) SCAN Length split

Figure 7: Perplexity of models with different augmentation strategies on test set. The x-axis refers to training steps.



(a) Exact match accuracy of T5 on COGS generalization set with different maximum structure depths observed in the train set.



(b) Execution accuracy of T5 on GeoQuery test set (template split) with increasingly observed rule combinations in the train set.

Figure 8: Performance curve with regard to train sets with incrementally added structures.

Designing a context-free grammar for SPARQL is hard because it contains variables and each relation only accepts specific typed variables as arguments. For example, the object of *film.writer.film* relation should be a film. In our grammar, we consider all variable strings are produced by the nonterminal *Var* and we do post-process to filter out samples that do not follow type constraints described above. A selection of our context-free grammar rules are shown in Table 14.

In our experiments, we find this setting still generates most noisy meaning representations due to redundant conjuncts (e.g. *SELECT DISTINCT ?x0 WHERE { (FILTER (?x0 != M0)) . (M5 (film.editor.film) (?x1)) }*). A better solution might be to construct the PCFG based on the graph structure.

D.2 Parameter estimation

To estimate parameters of a grammar on a dataset based on maximum likelihood estimation, we first parse meaning representations in the dataset with our grammar rules described above. We implement this with NLTK package[4]. We binarize our gram-

[4] https://www.nltk.org/

mar rules to adopt parsing methods in NLTK.

Ambiguous trees For CFQ, all meaning representations can be parsed into unambiguous trees. For GeoQuery and SCAN, parsing results in ambiguous trees for some cases. For a meaning representation with N ambiguous parse trees, we simply use a count $1/N$ as the count for rules in each tree to estimate their parameters.

E Additional experiments

E.1 Perplexity curve

We plot the perplexity curve of different models on test set for each task in Figure 7. For CFQ and CFQ, the perplexity at the beginning is already very small. This is because on these two datasets we pretrain the model on synthesized data first, since direct concatenating the synthesized data only hurts the performance. We can observe that for COGS, GeoQuery and SCAN, the perplexity of P_{test} always achieves the lowest perplexity and $P_{uniform}$ gives lower perplexity than P_{train}. On CFQ, all three augmentation distributions achieves lower perplexity than baseline T5 and performs on par. This pattern holds during the entire training process, which serves as further evidence for the discussion in Section 5.

E.2 Breakdown performance improvements

We also conduct a more detailed analysis to investigate how the performances evolve as more complex structures get observed. Specifically, we address the PP and CP recursion generalization types on COGS and GeoQuery template split. For COGS, we incrementally augment the train set with more complex data (i.e. deeper recursions) in increments of 100 instances per depth. For GeoQuery, we manually select four local structures *population_1 stateid*, *len river*, *capital cityid*, *intersection river* that pose challenges for the baseline parser's predictions yet are present in the P_{test} set. We incrementally introduce each pattern into the train set. As shown in Figure 8, as more complex MR structures being observed by the model, its performance gets better improved.

	Model	COGS Obj to Subj PP	CP recursion	PP recursion	Overall
	T5	88.2 ± 3.6	32.3 ± 3.7	24.1 ± 6.4	91.0 ± 0.5
	+P_{train}	89.4 ± 2.3	43.5 ± 8.7	51.2 ± 7.5	92.9 ± 0.9
Concat	+P_{test}	94.6 ± 0.1	95.7 ± 2.8	96.7 ± 5.0	99.3 ± 0.4
	+$P_{uniform}$	94.8 ± 0.0	50.7 ± 2.4	87.7 ± 1.0	95.9 ± 0.1
	+P_{train}	85.8 ± 6.5	41.3 ± 11.3	51.6 ± 5.8	92.8 ± 0.6
Pretrain	+P_{test}	94.8 ± 0.0	43.1 ± 5.7	85.0 ± 4.0	99.2 ± 0.2
	+$P_{uniform}$	94.6 ± 0.1	94.8 ± 0.2	92.7 ± 4.9	95.6 ± 0.5

	Model	CFQ MCD1	MCD2	MCD3	Average
	T5	89.8 ± 0.8	74.7 ± 1.8	74.0 ± 0.9	79.4 ± 2.4
	+P_{train}	49.5 ± 1.9	47.1 ± 1.3	51.2 ± 2.9	49.2 ± 1.0
Concat	+P_{test}	39.0 ± 1.3	44.7 ± 2.3	42.3 ± 0.7	42.0 ± 0.9
	+$P_{uniform}$	57.5 ± 4.1	59.4 ± 2.4	55.2 ± 3.3	57.4 ± 1.8
	+P_{train}	89.9 ± 1.2	77.9 ± 2.9	75.8 ± 1.0	81.2 ± 2.1
Pretrain	+P_{test}	90.4 ± 0.7	79.1 ± 1.7	75.5 ± 2.7	81.7 ± 3.6
	+$P_{uniform}$	91.2 ± 1.1	78.8 ± 1.7	74.3 ± 1.7	81.4 ± 1.1

	Model	GeoQuery Template	Length
	T5	73.9 ± 2.6	46.1 ± 1.5
	+P_{train}	39.0 ± 0.9	20.7 ± 0.6
Concat	+P_{test}	52.3 ± 1.3	35.6 ± 1.7
	+$P_{uniform}$	22.4 ± 1.7	5.1 ± 0.3
	+P_{train}	74.1 ± 1.6	56.1 ± 2.1
Pretrain	+P_{test}	80.1 ± 1.7	60.4 ± 2.4
	+$P_{uniform}$	79.3 ± 1.3	60.1 ± 0.6

	Model	SCAN Turnleft	Length
	T5	73.9 ± 2.6	4.4 ± 0.9
	+P_{train}	92.9 ± 14.4	8.1 ± 1.3
Concat	+P_{test}	92.9 ± 14.4	60.5 ± 2.5
	+$P_{uniform}$	92.9 ± 14.4	60.5 ± 2.5
	+P_{train}	75.5 ± 5.4	15.5 ± 1.5
Pretrain	+P_{test}	75.5 ± 5.4	15.9 ± 1.3
	+$P_{uniform}$	75.5 ± 5.4	15.9 ± 1.3

Table 15: Detailed results in our experiments.