SUBS: Subtree Substitution for Compositional Semantic Parsing

Anonymous ACL submission

Abstract

Although sequence-to-sequence models often achieve good performance in semantic parsing for i.i.d. data, their performance is still inferior in compositional generalization. Several data augmentation methods have been proposed to alleviate this problem. However, prior work only leveraged superficial grammar or rules for data augmentation, which resulted in limited improvement. We propose to use subtree substitution for compositional data augmentation, where we consider subtrees with similar semantic functions as exchangeable. Our experiments showed that such augmented data led to significantly better performance on SCAN and GEOQUERY, and reached new SOTA on compositional split of GEOQUERY.

1 Introduction

Semantic parsing transforms natural language utterances to formal language. Because meaning representations or programs are essentially compositional, semantic parsing is an ideal testbed for compositional generalization. Although neural seq2seq models could achieve state-of-the-art performance in semantic parsing for i.i.d. data, they failed at compositional generalization due to lack of reasoning ability. That is, they do not generalize well to formal language structures that were not seen at training time. For example, a model that observes at training time the questions "*What is the population of the largest state?*" and "*What is the largest city in USA?*" may fail to generalize to questions such as "*What is the population of the largest city in USA?*". This leads to large performance drops on data splits designed to measure compositional generalization (compositional splits), in contrast to the generalization abilities of humans.

To improve compositional generalization in semantic parsing (compositional semantic parsing), prior work focused on incorporating inductive biases directly to models or data augmentation. From the model perspective, some work used neural-symbolic models (Chen et al., 2020), generated intermediate discrete structures (Herzig and Berant, 2020; Zheng and Lapata, 2020), or conducted meta-learning (Lake, 2019). From the data perspective, Jia and Liang (2016) proposed to recombine data with simple synchronous context-free grammar (SCFG), despite not for compositional generalization. Andreas (2019) used some simple rules for data augmentation, where tokens with the same context were considered as exchangeable. Such techniques are still limited since they only leveraged superficial grammars or rules, and failed when there are linguistically rich phrases or clauses.

To fill this gap, we propose to augment the training data of semantic parsing with diverse compositional examples based on induced or annotated (semantic and syntactic) trees. Specifically, we propose to exchange subtrees where roots have similar meaning functions. Since we consider all hierarchies in all trees, deep structures and complex phrases or clauses are considered for data augmentation, which is key for compositional generalization. For instance, in Figure 1, if we exchange subtrees with "*largest*" as meaning function of its root, composition of "*population of the*" and "*largest city in the smallest state in the USA*" results in a new augmented structure "*population of the largest city in the smallest state in the USA*". Although certain substructure substitution methods were explored in other NLP tasks (Shi et al., 2021), subtree substitution with fine-grained meaning functions has been under-explored. Our experiments showed that such augmented data led to significantly better performance on SCAN (Lake and Baroni, 2018) and GEOQUERY, and reached new SOTA on compositional split of GEOQUERY.

2 Methods

Span trees Suppose training set is $\{(x^i, z^i)\}_{i=1}^N$, where x_i is a natural language utterance and z_i is the corresponding program. An utterance x can be

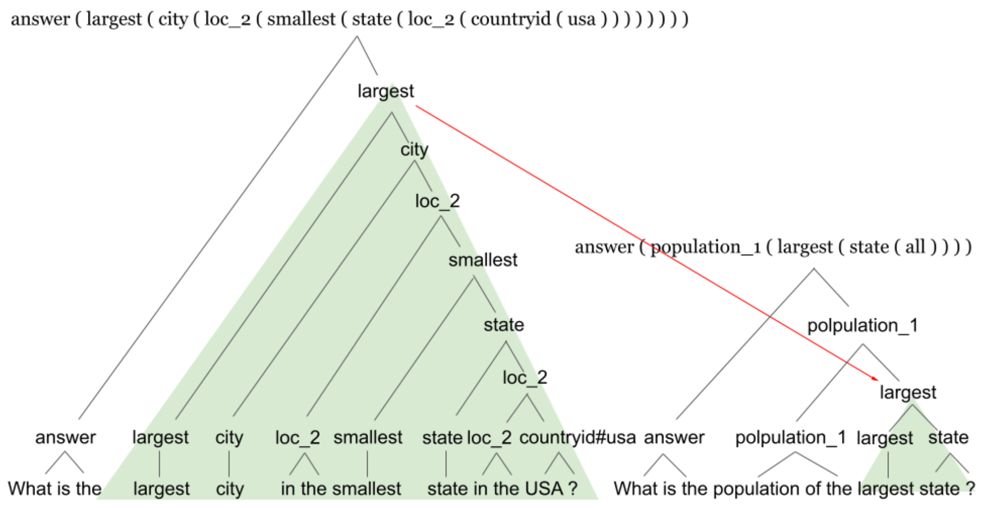

Figure 1: Subtree substitution results in an augmented example. Natural Language: *What is the population of the largest city in the smallest state in the USA ?* Formal Language: `answer ( population_1 ( largest ( city ( loc_2 ( smallest ( state ( loc_2 ( countryid ( usa ) ) ) ) ) ) ) ) )`.

mapped to a span tree T, such that program(T)= z, where the deterministic function program($\cdot$) maps span trees to programs (Herzig and Berant, 2020).

As shown in Figure 1, a span tree T is a tree where each node covers a span (i, j) with tokens $x_{i:j} = (x_i, x_{i+1}, \cdots, x_j)$. A span subtree can be viewed as a mapping from every span (i, j) to a single category $c \in C$, where C is a set of domain-specific categories representing domain constants, which include entities (e.g. *countryid#usa* in Figure 1) and predicates (e.g. *loc_2* in Figure 1). The final program can be computed from the span tree deterministically by the function program($\cdot$). Concretely, program(T) iterates over the nodes in T bottom-up, and generates a program $z_{i:j}$ for each node covering the span (i, j). For a terminal node, $z_{i:j} = c$. For an internal node, $z_{i:j}$ is determined by composing the programs of its children, $z_{i:s}$ and $z_{s:j}$ where s is the split point. As in Combinatory Categorical Grammar, composition is simply function application, where a domain-specific type system is used to determine which child is the function and which is the argument. Span trees can be induced by a hard-EM algorithm or semi-automatically annotated. We refer the reader to Herzig and Berant (2020) to see how to obtain span-trees.

2.1 Subtree Substitution (SUBS)

As shown in Figure 1, we consider span subtrees with similar semantic functions as exchangeable. Formally, func($\cdot$) maps a subprogram to a semantic category, and subtrees with the same semantic categories have similar semantic functions. For two data points (x^1, z^1) and (x^2, z^2), if func($z^1_{i_1:j_1}$) = func($z^2_{i_2:j_2}$), we obtain a new augmented (x', z'):

$$x' = x^1_{:i_1} + x^2_{i_2:j_2} + x^1_{j_1:}, z' = z^1 \backslash z^1_{i_1:j_1} / z^2_{i_2:j_2}$$

Definition of func($\cdot$) may vary in different dataset. One straightforward way is to extract the outside predicate in $z_{i:j}$ as its semantic category, which is used on GEOQUERY, such as func(`largest ( state ( all ) ) `)= `largest`.

	RIGHT	AROUNDRIGHT
LSTM	0.00	1.00 (2800 updates)
LSTM + SUBS	1.00	1.00 (800 updates)

Table 1: Accuracy of diagnostic experiments on SCAN.

2.2 Semantic Parsing

After getting augmented data by subtree substitution, we then combine augmented data and the original training data to train a seq2seq semantic parser, where we choose LSTM models with attention (Luong et al., 2015) and copying mechanism (Gu et al., 2016), or pretrained BART (Lewis et al., 2020) as the seq2seq model architecture.

3 Experiments and Results

Dataset We first use SCAN (Lake and Baroni, 2018) as a diagnostic dataset to test the performance of subtree substitution in compositional semantic parsing. SCAN is a synthetic dataset, which consists of simple English commands paired with sequences of discrete actions. We use the program

	Question	Query
Herzig and Berant (2020)	0.78	0.59
LSTM	0.75	0.58
+ SCFG (Jia et al., 2016)	0.80	0.68
+ GECA (Andreas, 2019)	0.77	0.60
+ SUBS (ours, induced tree)	0.79	**0.70**
+ SUBS (ours, gold tree)	**0.81**	**0.79**
BART	0.91	0.85
+ SUBS (ours, induced tree)	0.91	0.85
+ SUBS (ours, gold tree)	**0.93**	**0.88**

Table 2: Exact-match accuracy on i.i.d. (Question) and compositional (Query) splits of GEOQUERY dataset.

version of Herzig and Berant (2020). For instance, "*run right after jump*" corresponds to the program "`i_after ( i_run ( i_right ) , i_jump )`". Also, semi-automatically annotated span trees from Herzig and Berant (2020) are used for subtree substitution. To test compositional semantic parsing, we use the *Primitive right* (RIGHT) and *Primitive around right* (AROUNDRIGHT) compositional splits from Loula et al. (2018), where templates of the form *Primitive right* and *Primitive around right* (respectively) appear only in the test set. In these templates, *Primitive* stands for *jump*, *walk*, *run*, or *look*. For simplicity, func(·) is defined only on `i_right` and `i_left`, where func(`i_right`) = func(`i_left`) = `direction`. That is, all "`i_right`" and "`i_left`" appear as leaf nodes in span trees and they are exchangeable.

We use GEOQUERY dataset to test the performance of subtree substitution in both i.i.d. and compositional generalization for semantic parsing. GEOQUERY contains 880 questions about US geography (Zelle and Mooney, 1996). Following Herzig and Berant (2020), we use the variable-free FunQL formalism from Kate et al. (2005). The i.i.d. split (Question), which is randomly sampled from the whole dataset, contains 513/57/256 instances for train/dev/test set. The compositional split (Query) contains 519/54/253 examples for train/dev/test set, where templates created by anonymizing entities are used to split the dataset, to make sure that all examples sharing a template are assigned to the same set (Finegan-Dollak et al., 2018). As for span trees, we use semi-automatically annotated span trees (gold tree) released by Herzig and Berant (2020). Alternatively, we use the span trees induced by Herzig and Berant (2020)'s span-based semantic parsing, without any human labour.

3.1 Diagnostic Results

Results of diagnostic experiments on SCAN dataset are shown in Table 1, where we use LSTM parser without data augmentation as the baseline. We can see that on the RIGHT split, LSTM seq2seq semantic parser could only achieve zero exact-match accuracy without any data augmentation techniques, which means that the model's compositional generalizability on the RIGHT split is very poor. After adding our augmented data with subtree substitution, we achieve an exact-match accuracy of 100%. Actually, we got 6660 augmented examples besides the original 12180 training examples. Among all augmented examples, 3351 examples are in the test set, which means 74.87% of 4476 test examples are recovered by subtree substitution. On the AROUNDRIGHT split, using LSTM seq2seq semantic parser could already achieve 100% exact-match accuracy, which means that the model learned from *Primitive right* and *Primitive opposite right* generalize to *Primitive around right* well in our program format "`i_primitive ( i_around ( i_right ) )`". After adding our augmented examples, the parser converged to 100% exact-match accuracy faster, where our method requires around 800 updates to converge while baseline model requires 2800 updates with the same batch size 64.

3.2 Main Results

Table 2 shows the results of experiments on GEOQUERY dataset, where we examined both seq2seq LSTM and BART parsers. LSTM and BART parsers without any data augmentation are simplest baselines. We also compare with other two data augmentation methods as additional baselines, recombining data with simple SCFG (Jia and Liang, 2016) or using simple rules for Good Enough Data Augmentation (GECA) (Andreas, 2019), which were proven useful for compositional semantic parsing. We can see that on the Question split, adding augmented data from (gold) subtree substitution leads to improvements for both LSTM and BART seq2seq models, suggesting that subtree substitution as data augmentation helps i.i.d generalization for semantic parsing. On the Query split, (gold) subtree substitution achieves more substantial improvements over seq2seq baseline models (absolute 21% and 3% improvements of the exact-match accuracy for LSTM and BART respectively), achieving state-of-the-art results. Moreover, our methods are also better than the two data augmentation baselines. Therefore, subtree substitution

	training instances	augmented instances	avg att l	max att l	avg prg l	max prg l
GECA	519	804	8.85	18	15.96	29
SUBS	519	29039	10.43	26	19.33	43
	avg seg l	**max seg l**	**avg att seg l**	**max att seg l**	**avg prg seg l**	**max prg seg l**
GECA	1.93	4	-	-	-	-
SUBS	5.99	25	3.98	13	8.01	25

Table 3: Complexity of augmented examples on the Query split of GEOQUERY dataset, which is measured by maximal (max) and average (avg) lengths (l) of exchanged segments (seg) and resulted utterances(att)/programs(prg).

	50	100	200	519
BART	0.64	0.72	0.79	0.85
BART + SUBS	**0.67**	**0.79**	**0.85**	**0.88**

Table 4: Effect of numbers of training examples on compositional split of GEOQUERY.

is a simple yet effective compositional data augmentation method for compositional semantic parsing. With (induced) subtree substitution, SUBS still achieves improvements for LSTM models. Our SUBS could outperform Herzig and Berant (2020), although our induced tree are based on their model. That said, incorporating inductive biases to data and then to the model (seq2seq model finetuning) could achieve superior performance than directly incorporating inductive biases to model via latent variables (Herzig and Berant, 2020).

Analysis of Augmented Data We further examine why subtree substitution could achieve much better performance by analyzing its augmented data. As shown in Table 3, GECA only identifies and exchanges very simple structures, where the average and maximal length of exchanged segments are 1.93 and 4. A closer look at these augmented data shows that nearly all of these segments are simple entities (e.g. STATE: "Illinois", "Arizona" etc.) or other Nouns (e.g. "area", "population" etc.). In contrast, subtree substitution can identify and exchange much more complex structures, where the average and maximal length of exchanged segments are 5.99 and 25. For example, *largest city in the smallest state in the USA* and *largest state* are identified as exchangeable. As a result, subtree substitution could produce more complex utterance and program pairs, where the average and maximal length of these resulted utterances are 10.43 and 26, compared with the average (8.53) and maximal (18) length of utterances returned by GECA. Moreover, subtree substitution could generate much more augmented instances, because it can identify more complex structures besides those simple ones identified by GECA. Compared with SCFG, SUBS could also identify complex structures automatically with subtrees, while SCFG only handle simple phrases defined by rules.

Effect of Training Data Size Table 4 shows that with more training examples, models' performances improve. In all settings, using (gold) subtree substitution boosts the performance of BART. When there are 100 and 200 training examples, the improvement is more significant, demonstrating the effectiveness of SUBS in the few-shot setting.

4 Related Work

Several data augmentation methods have been introduced for (compositional) semantic parsing. Jia and Liang (2016) recombined data by SCFG, and Andreas (2019) used some simple rules to exchange tokens with the same context. However, they leveraged only superficial grammars or rules, which has limited capacity to identify complex structures. Akyürek et al. (2020) learned to recombine and resample data with a prototype-based generative model, instead of using rules. Certain substructure substitution methods have been explored for data augmentation in other NLP tasks (Shi et al., 2021). Dependency tree cropping and rotation within sentence was used in low-resource language POS tagging (Şahin and Steedman, 2019) and dependency parsing (Vania et al., 2019). Dependency tree swapping was explored in low-resource language dependency parsing (Dehouck and Gómez-Rodríguez, 2020). However, subtree substitution with fine-grained meaning functions has not been examined. To the best of our knowledge, we are the first to explore tree manipulation for semantic parsing.

5 Conclusion

This work proposed to use subtree substitution to compositionally augment the data of semantic parsing to help the compositional generalization. Our method achieved significant improvements over seq2seq models, other data augmentation methods and span-based semantic parsing.

References

Ekin Akyürek, Afra Feyza Akyürek, and Jacob Andreas. 2020. Learning to recombine and resample data for compositional generalization. *arXiv preprint arXiv:2010.03706*.

Jacob Andreas. 2019. Good-enough compositional data augmentation. *arXiv preprint arXiv:1904.09545*.

Xinyun Chen, Chen Liang, Adams Wei Yu, Dawn Song, and Denny Zhou. 2020. Compositional generalization via neural-symbolic stack machines. *arXiv preprint arXiv:2008.06662*.

Mathieu Dehouck and Carlos Gómez-Rodríguez. 2020. Data augmentation via subtree swapping for dependency parsing of low-resource languages. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3818–3830.

Catherine Finegan-Dollak, Jonathan K Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-sql evaluation methodology. *arXiv preprint arXiv:1806.09029*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1631–1640.

Jonathan Herzig and Jonathan Berant. 2020. Span-based semantic parsing for compositional generalization. *arXiv preprint arXiv:2009.06040*.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. *arXiv preprint arXiv:1606.03622*.

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

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pages 2873–2882. PMLR.

Brenden M Lake. 2019. Compositional generalization through meta sequence-to-sequence learning. *arXiv preprint arXiv:1906.05381*.

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.

Joao Loula, Marco Baroni, and Brenden M Lake. 2018. Rearranging the familiar: Testing compositional generalization in recurrent networks. *arXiv preprint arXiv:1807.07545*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*.

Gözde Gül Şahin and Mark Steedman. 2019. Data augmentation via dependency tree morphing for low-resource languages. *arXiv preprint arXiv:1903.09460*.

Haoyue Shi, Karen Livescu, and Kevin Gimpel. 2021. Substructure substitution: Structured data augmentation for nlp. *arXiv preprint arXiv:2101.00411*.

Clara Vania, Yova Kementchedjhieva, Anders Søgaard, and Adam Lopez. 2019. A systematic comparison of methods for low-resource dependency parsing on genuinely low-resource languages. *arXiv preprint arXiv:1909.02857*.

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

Hao Zheng and Mirella Lapata. 2020. Compositional generalization via semantic tagging. *arXiv preprint arXiv:2010.11818*.

A Training Details

We adapted OpenNMT (Klein et al., 2017) for LSTM model with attention and copying mechanism, while used fairseq (Ott et al., 2019) to implement BART model.

We manually tune the hyper-parameters. For LSTM models, we use one-layer bidirectional LSTM in the encoder side and one-layer unidirectional LSTM in the decoder side. We use dropout with 0.5 as dropout rate and Adam optimizer with a learning rate of 0.001. We use MLP attention and reuse attention scores as copying scores. On GeoQuery, the batch size is set to 1 sentence without augmented data and set to 64 sentences with augmented data. On Scan, all batch sizes are 64 sentences. For BART models, we use BART large models. We use Adam as optimizer with a learning rate 1e-5. We use dropout and attention dropout with 0.1 as dropout rate. Also, we use label smoothing with a rate 0.1. Batch sizes are 1024 tokens. Besides, we employ a weight-decay rate 0.01. All the parameters are manually tuned based on the dev performance.

We train all models on NVIDIA A100 SXM4 40 GB GPU. We set the max training epoch to be 100 and select the best performed epoch according to dev performance. Training process on each clause or whole sequence could be finished within 3 hours.

For baselines with other data augmentation methods, we reran GECA and SCFG on this FunQL formalism of GEOQUERY and these splits with annotated span trees. That's why our results are a little different from the reported results in the original paper. We got similar results with their source code and our code on our data, in order to make sure that there is no problem with our results and code.

We got the same denotation accuracy as reported by Herzig and Berant (2020), but we reported exact-match accuracy on Table 2 for fair comparison.