Learning to Substitute Spans towards Improving Compositional Generalization

Zhaoyi Li[1,2], **Ying Wei**[3*] and **Defu Lian**[1,2*]
[1]School of Computer Science and Technology, University of Science and Technology of China
[2]State Key Laboratory of Cognitive Intelligence, Hefei, Anhui, China
[3]Department of Computer Science, City University of Hong Kong
lizhaoyi777@mail.ustc.edu.cn, yingwei@cityu.edu.hk, liandefu@ustc.edu.cn

Abstract

Despite the rising prevalence of neural sequence models, recent empirical evidences suggest their deficiency in compositional generalization. One of the current de-facto solutions to this problem is compositional data augmentation, aiming to incur additional compositional inductive bias. Nonetheless, the improvement offered by existing handcrafted augmentation strategies is limited when successful systematic generalization of neural sequence models requires multi-grained compositional bias (i.e., not limited to either lexical or structural biases only) or differentiation of training sequences in an imbalanced difficulty distribution. To address the two challenges, we first propose a novel compositional augmentation strategy dubbed **Span Sub**stitution (SpanSub) that enables multi-grained composition of substantial substructures in the whole training set. Over and above that, we introduce the **L**earning **to** **S**ubstitute **S**pan (L2S2) framework which empowers the learning of span substitution probabilities in SpanSub in an end-to-end manner by maximizing the loss of neural sequence models, so as to outweigh those challenging compositions with elusive concepts and novel surroundings. Our empirical results on three standard compositional generalization benchmarks, including SCAN, COGS and GeoQuery (with an improvement of at most 66.5%, 10.3%, 1.2%, respectively), demonstrate the superiority of SpanSub, L2S2 and their combination.

1 Introduction

The secret for human beings to learning so quickly with little supervision has been demonstrated to be associated with the powerful ability of *systematic generalization*, being capable of producing an infinite number of novel combinations on the basis of known components (Chomsky, 1957). In stark contrast, a large body of recent evidence suggests that current state-of-the-art neural sequence models

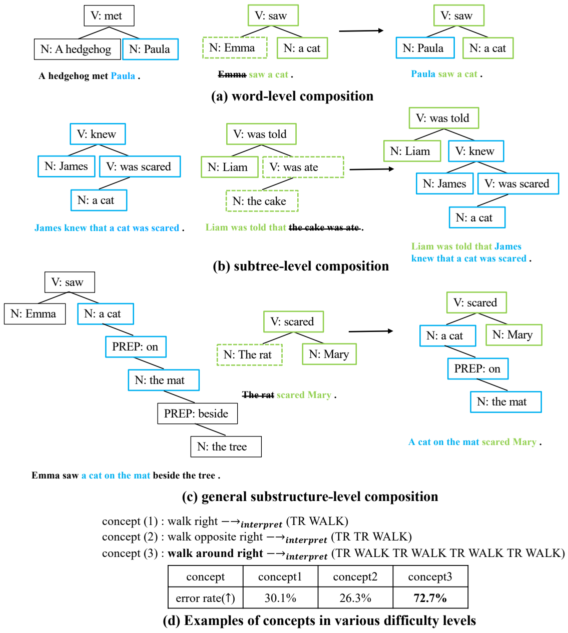

Figure 1: (a), (b) and (c) illustrate three distinct compositional generalization types in COGS (Kim and Linzen, 2020), which require word-level, subtree-level and general substructure-level recombinations of training data, respectively. Besides, (d) shows concepts in distinct difficulty in the SCAN (Lake and Baroni, 2018) dataset, where the interpretation of *walk around right* is much more complex than that of the other two concepts.

lack of adequate power for compositional generalization (*a.k.a.,* systematic generalization) (Lake and Baroni, 2018; Furrer et al., 2020). For instance, a model which has observed the two training sentences of "*look opposite right* twice and jump right thrice" and "*walk around right* and run twice" likely fails to understand the testing sentence of "*walk around right* twice and jump right thrice". Sharpening the compositional generalization ability of neural sequence models is beyond important to fill the gap with human-like natural language understanding, catalyzing not only better performances but also fewer expensive annotations.

Inspired by the tight relationship between compositionality and group-equivariance of neural mod-

*Corresponding authors

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 2791–2811

els (Gordon et al., 2020; Akyürek and Andreas, 2022; Basu et al., 2022), a series of compositional data augmentation solutions have made great strides via injecting compositional inductive bias into neural sequence models (Andreas, 2020; Guo et al., 2020a; Akyürek and Andreas, 2022; Yang et al., 2022; Jiang et al., 2022). The key idea behind compositional data augmentation is to substitute a part in one original training example with a part from another training example, thus composing a novel example that complements the training data with compositional bias. Introducing comprehensive enough comositional bias to embrace a diversity of testing tasks, however, is not trivial. First, the "part"[1] to be substituted out and in is expected to be in multiple levels, ranging from words (Akyürek and Andreas, 2022) in Fig. 1(a), to complete substrees (Yang et al., 2022) in Fig. 1(b), to more general substructures in Fig. 1(c). How to develop an augmentation method that flexibly accommodates multiple levels of parts remains an open question. Second, the "parts" are uneven in their difficulty levels. As shown in Fig. 1(d), though the numbers of both training and testing sentences containing the three concepts in the SCAN MCD split are comparable and we have applied compositional data augmentation via the proposed SpanSub (which will be detailed later), the predicted error rates of testing sentences grouped by the three concepts still differ significantly, which is in alignment with the observations in (Bogin et al., 2022). There is an urgent need to augment with difficulty awareness and allow more compositions on the challenging concepts (e.g., concept 3 in Fig. 1(d)).

To conquer the two challenges, we first propose a novel compositional data augmentation scheme SpanSub that substitutes a *span* in a training sentence with one in another sentence, where a span refers to a consecutive fragment of tokens that subsumes all multi-grained possibilities of a word, a subtree, as well as a more general substructure. The core of SpanSub lies in extraction of such spans and identification of exchangeable spans, towards which we define the exchangeability of spans by the exchageability or syntactic equivalence of their first and last tokens. On top of this, we propose the L2S2 framework made up of a L2S2 augmenter, which is a differentiable version of SpanSub with all substitution actions equipped with probabilities. By training down-stream neural sequence models to evaluate the difficulty of various spans and maximizing their losses, the L2S2 framework seeks to train the L2S2 augmenter to tip the scales of those substitution actions contributing challenging compositions by elusive spans and novel surroundings.

In summary, the main contributions of this paper are three-fold.

- SpanSub is the first to explore span-based compositional data augmentation, thus flexibly supporting multi-grained compositional bias;
- L2S2 as a differentiable augmentation framework first empowers difficulty-aware composition, being compatible with various down-stream models.
- We have empirically demonstrated the superiority of SpanSub, L2S2, and their combination on three standard benchmarks (SCAN, COGS and GeoQuery) with improvements of at most 66.5%, 10.3% and 1.2% over prior part, respectively.[2]

2 Related Work

Compositional generalization in neural sequence models A large body of literature pursues various ways of introducing compositional inductive bias into neural sequence models, in a bid to improve systematic generalization. The first category of studies, e.g., CGPS (Li et al., 2019), SyntAtt (Russin et al., 2020), GroupEqu (Gordon et al., 2020), customizes neural architectures that promote lexical generalization via explicit disentanglement of the meaning of tokens. The second strand aims to align words or substructures in the input sequences with their counterparts in the output sequences by auxiliary tasks (e.g., IR-Transformer (Ontanon et al., 2022)), additional architectural modules (e.g., LexLearn (Akyurek and Andreas, 2021)), as well as extra objectives imposed on attention layers (e.g., SpanAtt (Yin et al., 2021)). Third, the works of Meta-seq2seq (Lake, 2019), Comp-MAML (Conklin et al., 2021), and MET (Jiang et al., 2022) resorts to the meta-learning paradigm to directly encourage compositional generalization of neural models. Last but not least, compositional data augmentation that composes in-distribution data to accommodate out-of-distribution compositional sequences has been empirically demonstrated to enjoy not only the

[1] We use the words of "part", "concept", and "span" later interchangeably.

[2] Code available at https://github.com/Joeylee-rio/Compgen_l2s2

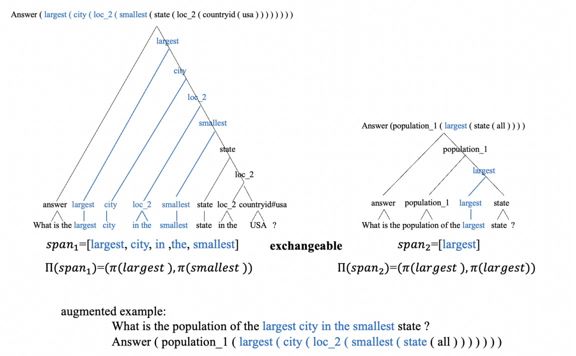

Figure 2: An augmentation example by SpanSub. SpanSub substitutes a span "largest" with another span "largest city in the smallest", and augments a new question "What is the population of the largest city in the smallest state?".

performance but also the model-agnostic benefits. The explored principles for augmentation include exchangeability of tokens in the same context (e.g., GECA (Andreas, 2020)), token-level mixup (Zhang et al., 2018) (e.g., SeqMix (Guo et al., 2020a)), group-equivariance of language models (Basu et al., 2022) by substituting training tokens (e.g., LexSym (Akyürek and Andreas, 2022), Prim2PrimX (Jiang et al., 2022)) or subtrees (e.g., SUBS (Yang et al., 2022)) with virtual or off-the-shelf tokens or substrees. Note that the aforementioned approaches guarantee the validity of composed sequences by following the widely accepted alignment practices in NLP, e.g., SpanTree (Herzig and Berant, 2021) and FastAlign (Dyer et al., 2013). Our work further pushes ahead with compositional data augmentation by (1) substituting spans, which offers more diverse and flexible generalization than substituting monotonous tokens or subtrees, and (2) endowing the augmentation strategy to be differentiable and learnable in an end-to-end manner, which dynamically adapts to the difficulty of down-stream neural sequence tasks.

3 Span Substitution

We propose SpanSub to generate novel examples through exchanging multi-grained spans, which refer to consecutive fragments in input sequences, of the same equivalence class between training examples as shown in Fig. 2. Before proceeding to the details of SpanSub, we first introduce two preprocessing prerequisites for SpanSub, including extraction of span alignment and inference of the equivalence class of a word. On top of these, we present our substitution strategy that dictates the equivalence and exchangeability between spans.

3.1 Preprocessing

The techniques of extracting span alignment from paired linguistic data and identifying syntactically equivalent words (e.g., Part-of-Speech tagging) have been well studied in the NLP community. Following the practice in a wealth of literature on compositional augmentation (Akyürek and Andreas, 2022; Yang et al., 2022; Jiang et al., 2022), we also directly adapt the off-the-shelf techniques, which we introduce as below for self-contained purpose, to preprocess rather than delving into them. More details and results of preprocessing for all the datasets are available in Appendix A.2.

Extraction of span alignment Span alignment refers to establish the correspondence between spans in the input sequence (e.g., "largest city in the smallest") and their counterparts (e.g., "largest(city(loc_2(smallest())))") in the output sequence of a training example. For the SCAN dataset, we extract span alignment by extending SimpleAlign (Akyurek and Andreas, 2021) that targets single words (e.g., *jump* $\rightarrow$ *JUMP right* $\rightarrow$ *TURN_RIGHT*) to support alignment of consecutive fragments (e.g., *jump right* $\rightarrow$ *TURN_RIGHT JUMP*). As there always exists a deterministic function program (Ontanon et al., 2022; Yang et al., 2022) that transforms the output sequence y to a tree for COGS and GeoQuery, we resort to the intermediate representation (Herzig et al., 2021) of COGS from (Ontanon et al., 2022) and the span tree of GeoQuery from (Herzig and Berant, 2021) to map the input sequence x to the tree form T, respectively. The tree T, in such a way, serves as a bridge to align the input and output.

Inference of the equivalence class of a word The aim is to infer the equivalence class of a word w, i.e., $\pi(w)$, according to the cluster it belongs to. Exemplar clusters include verbs and nouns. Fortunately, the COGS dataset has intrinsic clusters of words by their tree structure representations. As for SCAN and GeoQuery, we follow (Akyürek and Andreas, 2022; Jiang et al., 2022) to assign those words sharing the context into a single cluster. For example, the words of "largest" and "smallest" fall into the same cluster in Fig. 2.

3.2 Substitution Strategy

The equivalence or exchangeability of spans, which a substitution strategy aims to establish, boils

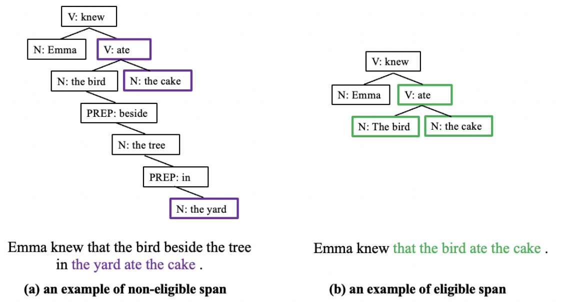

Figure 3: Examples of non-eligible and eligible spans in COGS. (a) shows a non-eligible span which corresponds to an union set of disconnected fragments of the tree.

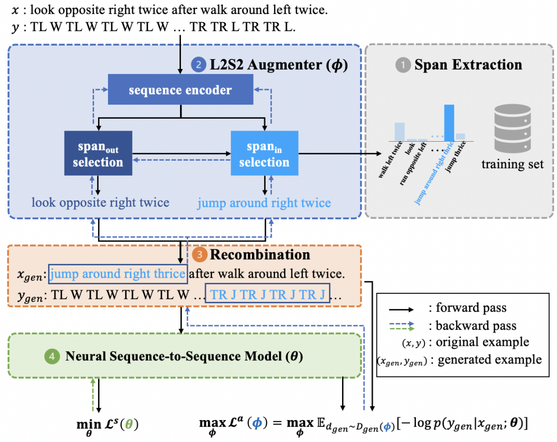

Figure 4: Illustration of L2S2 framework.

down to answering the following two questions: (1) what is an eligible span? (2) how to define the equivalence? First, given a consecutive span $s = [w_p, w_{p+1}, ..., w_{p+k}]$ where w_{p+i} $(0 \leq i \leq k)$ represents a semantic unit (i.e., a word with semantic meaning), we define the span to be eligible if and only if it is semantically self-contained and unitary. Fig. 3 shows a non-eligible span example "the yard ate the cake" which corresponds to an union set of two disconnected fragments of the tree and has an ambiguity (the subject of "ate" should be "the bird" rather than "the yard".). Such constraints imposed on eligible spans prevent substitutions with duplicate or missing parts. Due to page limit, we leave the formal mathematical definition of an eligible span into Appendix C.1.

Second, we formalize a heuristic rule to define the equivalence class of an eligible span s as the combined equivalence classes of its first and last token, i.e.,

$$\Pi(s) = \Pi([w_p, w_{p+1}, ..., w_{p+k}]) = (\pi(w_p), \pi(w_{p+k})), \quad (1)$$

where π indicates the equivalence class of a single word as defined in Section 3.1. By defining as above, it is legal to substitute a span s_1 with another span s_2 if and only if (1) both s_1 and s_2 are eligible according Definition 1 in Appendix C.1 and (2) $\Pi(s_1) = \Pi(s_2)$. Detailed pseudo codes of SpanSub is also available (i.e., Alg. 1) in Appendix C.1.

When dealing with tree structured tasks like GeoQuery and COGS, there are two special cases that need to be considered:

- $s = [w_p]$ (e.g., "largest" in Fig. 2) degenerates to a single word: we specify that s can only be substituted with another span s' (either degenerated or undegenerated) with $\Pi(s') = [\pi(w_p), \pi(w_p)]$.
- s is a subtree with its root token w_r: we specify that s can exchange with either another subtree s' with $\Pi(s') = [\pi(w_r), \pi(w_r)]$ or another span s' with $\Pi(s') = [\pi(w_p), \pi(w_{p+k})]$).

4 Learning to Substitute Spans (L2S2)

Beyond the benefit of multi-grained compositional bias introduced by SpanSub, the following three observations lead us to take a step further towards augmentation with attention on challenging spans. (1) The distinct combinations for a linear number of distinct spans could be as many as the super-linear number (Oren et al., 2021). (2) The spans constitute both easy-to-comprehend and elusive ones, while oftentimes elusive ones are so rare that those combinations by them account for a very small portion. (3) It is imperative to increase the percentage of these minority combinations to improve the compositional generalization in a broad range of down-stream tasks. Concretely, we introduce an online and optimizable L2S2 framework consisting of a L2S2 augmenter that inherits the idea of span substitution with SpanSub. More importantly, through maximizing the loss of down-stream neural sequence models, we learn span substitution probabilities in the upstreaming L2S2 augmenter to put high values on those chanllenging compositions of elusive spans and novel surroundings. The overview of the L2S2 framework is shown in Fig. 4.

4.1 Parameterizing the L2S2 Augmenter

Given a training example $\boldsymbol{d} = (x, y)$, the objective of the L2S2 augmenter is to synthesize a new example $\boldsymbol{d}_{gen} = (x_{gen}, y_{gen})$ via a sequence of two actions $\boldsymbol{a} = (a_{out}, a_{in})$: (1) a_{out} which selects the span s_{out} to be swaped out from the span set

$\mathcal{S}_1 = \{s_1^i\}_{i=1}^u$ extracted from x[3], and (2) a_{in} which selects the span s_{in} to be swapped in from the span set $\mathcal{S}_2 = \{s_2^i\}_{i=1}^v$ extracted from the whole training dataset, following a_{out}. Note that the preprocessing and span set extraction procedures are similar with Section 3, and $\mathcal{S}_1 \subset \mathcal{S}_2$. Once s_{out} and s_{in} are selected, we have $\boldsymbol{d}_{gen}$ via recombination, i.e.,

- x_{gen} = x.replace(s_{out},s_{in}),
- y_{gen} = y.replace($align(s_{out})$,$align(s_{in})$),

where replace(p, q) denotes p is replaced with q.

The probability of generating an ideal d_{gen} based on d is intuitively factorized as follows:

$$p(\boldsymbol{d}_{gen}|\boldsymbol{d};\boldsymbol{\phi}) = p(\boldsymbol{a}|\boldsymbol{d};\boldsymbol{\phi}) = p((a_{out}, a_{in})|\boldsymbol{d};\boldsymbol{\phi}) \\ = p(a_{out}|\boldsymbol{d};\boldsymbol{\phi}) \cdot p(a_{in}|a_{out}, \boldsymbol{d};\boldsymbol{\phi}) \quad (2)$$

where $\boldsymbol{\phi}$ denotes the parameters of the L2S2 augmenter. In the following, we will detail how to model the two probabilities, during which we will introduce the the three parts that constitute $\boldsymbol{\phi}$.

Parameterizing $p(a_{out}|\boldsymbol{d};\boldsymbol{\phi})$ for selection of spans to be substituted out Whether a span should be swapped out conditions on the equivalence class and the surroundings of the span, which are dictated by the representation of the span and that of the original training sequence x, respectively. To this end, we formulate the probability distribution $p(a_{out}|\boldsymbol{d};\boldsymbol{\phi})$ over all u candidate spans in S_1 as follows,

$$p(a_{out}|\boldsymbol{d};\boldsymbol{\phi}) = \tau(\mathcal{M}(\boldsymbol{\phi}_e(x), \boldsymbol{\phi}_o(\mathcal{S}_1))), \quad (3)$$

where $\boldsymbol{\phi}_e$ as the first part of $\boldsymbol{\phi}$ represents the parameters of a sequence encoder $\mathcal{R}(\cdot)$, and $\boldsymbol{\phi}_o$ (the second part of $\boldsymbol{\phi}$) denotes the embedding module for each candidate span in the span set $\mathcal{S}_1$. $\mathcal{M}(\cdot,\cdot)$ is a similarity function that measures the distance between two vectors. τ refers to the gumbel-softmax function (Jang et al., 2017), which guarantees sampling of the span with the largest probability, i.e., $a_{out}^* \sim p(a_{out}|\boldsymbol{d};\boldsymbol{\phi})$, to be differentiable. Implementation of the sampled action a_{out}^* results in the selected span s_{out}^* to be substituted out.

Parameterizing $p(a_{in}|a_{out};\boldsymbol{d};\boldsymbol{\phi})$ for selection of spans to be substituted in The factors that govern the selection of a span to be swapped in from the whole span set $\mathcal{S}_2$ include the representations of (1) the span itself, (2) the input sentence x for augmentation, and (3) the previously selected swap-out span s_{out}^*, so that those elusive spans that share the equivalence class with s_{out}^* but contribute novel compositions via recombination with surroundings in x are prioritized. Consequently, the probability distribution $p(a_{in}|a_{out},\boldsymbol{d};\boldsymbol{\phi})$ over all v candidate spans in S_2 follows,

$$\mathbf{c} = [\boldsymbol{\phi}_e(x); \boldsymbol{\phi}_o(s_{out}^*)]), \\ p(a_{in}|a_{out},\boldsymbol{d};\boldsymbol{\phi}) = \tau(\mathcal{M}(\boldsymbol{\phi}_f(\mathbf{c}), \boldsymbol{\phi}_i(\mathcal{S}_2))), \quad (4)$$

where $\boldsymbol{\phi}_f$ and $\boldsymbol{\phi}_i$ altogether act as the third part of $\boldsymbol{\phi}$. Specifically, $\boldsymbol{\phi}_i$ is the embedding module for all spans in the span set $\mathcal{S}_2$ and $\boldsymbol{\phi}_f$ aligns the concatenated representation of the sentence and the swap-out span, i.e., **c**, with $\boldsymbol{\phi}_i(\mathcal{S}_2)$ into the commensurable space. Being consistent with the previous paragraph, we leverage the similarity function $\mathcal{M}(\cdot,\cdot)$ and the gumbel-softmax trick τ to sample $a_{in}^* \sim p(a_{in}|a_{out}^*,\boldsymbol{d};\boldsymbol{\phi})$. It is noteworthy that we manually set the probability $a_{in} \to 0$ if $\Pi(s_{in}) \neq \Pi(s_{out}^*)$ to excluse those potentially illegal synthesized examples. The action a_{in}^* finalizes the span s_{in}^* to be substituted in.

4.2 Training Procedures for L2S2

Training L2S2 boils down to two alternating procedures: first, the generated examples by the L2S2 augmenter pass forward to train the downstream neural sequence-to-sequence model parameterized by $\boldsymbol{\theta}$; second, the performance of the neural sequence model serves as feedback to update the upstream augmenter parameterized by $\boldsymbol{\phi} = \{\boldsymbol{\phi}_e, \boldsymbol{\phi}_o, \boldsymbol{\phi}_i, \boldsymbol{\phi}_f\}$.

Training objective for the seq-to-seq model The objective of training the seq-to-seq model is to minimize the expected negative log-likelihood of producing the output sequence y_{gen} from the input one x_{gen} conditioned on the its parameters $\boldsymbol{\theta}$, i.e.,

$$\min_{\boldsymbol{\theta}} \mathcal{L}^s(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{d}_{gen}\sim\mathcal{D}_{gen}}[-\log p(y_{gen}|x_{gen};\boldsymbol{\theta})] \\ \approx \min_{\boldsymbol{\theta}} -\frac{1}{NT}\sum_{n=1}^{N}\sum_{t=1}^{T}\log p(y_{gen}^{n,t}|x_{gen}^{n,t};\boldsymbol{\theta}). \quad (5)$$

We would highlight that the empirical estimation samples over not only N examples but also T sequences of actions for each example, thus avoiding the randomness and high variance induced by the gumbel softmax trick. Thus, $(x_{gen}^{n,t}, y_{gen}^{n,t})$ denotes a generated example from the n-th original training example by following the t-th sampled action

[3]We can also identify spans in the y. This depends on the task type.

sequence $(a_{out}^{n,t}, a_{in}^{n,t})$. $\mathcal{D}_{gen}$ represents the distribution of all generated samples by the augmenter.

Training objective for the L2S2 augmenter Our main purpose is to encourage the upstream L2S2 augmenter to outweigh those challenging compositions by the elusive spans and novel surroundings. To achieve this goal, we evaluate the difficulty of a newly composed example $\boldsymbol{d}_{gen}$ by the feedback from the down-stream seq-to-seq model, i.e., the negative log-likelihood of predicting it; the larger the negative log-likelihood is, the more challenging the generated example is. Intuitively, we solve the following optimization problem to train the L2S2 augmenter to maximize the difficulty of synthesized examples.

$$\max_{\phi} \mathcal{L}^a(\phi) = \max_{\phi} \mathbb{E}_{\boldsymbol{d}_{gen} \sim \mathcal{D}_{gen}}[-\log p(y_{gen}|x_{gen}; \boldsymbol{\theta})]$$
$$\approx \max_{\phi} -\frac{1}{NT} \sum_{n=1}^{N} \sum_{t=1}^{T} p(\boldsymbol{d}_{gen}^{n,t}|\boldsymbol{d}^{n,t}; \phi) \log p(y_{gen}^{n,t}|x_{gen}^{n,t}; \boldsymbol{\theta}), \tag{6}$$

where $p(\boldsymbol{d}_{gen}^{n,t}|\boldsymbol{d}^{n,t}; \phi)$ refers to the gumbel softmax probability distribution of the t-th sampled action sequence $(a_{out}^{n,t}, a_{in}^{n,t})$ that translates $\boldsymbol{d}^{n,t}$ into $\boldsymbol{d}_{gen}^{n,t}$. To keep the L2S2 augmenter timely posted of the training state of the neural seq-to-seq model, we alternatingly optimize these two parts. We present the pseudo codes for training L2S2 in Alg. 2 in the Appendix. C.2.

5 Experiments

5.1 Datasets and Splits

We evaluate our proposed methods on the following three popular and representative semantic parsing benchmarks which target for challenging the compositional generalization capacity of neural sequence models. These benchmarks contain not only synthetic evaluations deliberately designed for diverse categories of systematic generalization but also non-synthetic ones additionally requiring capabilities of neural models in handling natural language variations (Shaw et al., 2021). More detailed descriptions of these datasets can be found in Appendix A.

SCAN Introduced by (Lake and Baroni, 2018), SCAN contains a large set of synthetic paired sequences whose input is a sequence of navigation commands in natural language and output is the corresponding action sequence. Following previous works (Andreas, 2020; Akyurek and Andreas, 2021; Jiang et al., 2022), we evaluate our methods on the two splits of ***jump*** (designed for evaluating a novel combination of a seen primitive, i.e., *jump*, and other seen surroundings) and ***around right*** (designed for evaluating a novel compositional rule). Notably, we also consider the more complex and challenging Maximum Compound Divergence (MCD) splits of SCAN established in (Keysers et al., 2020), which distinguish the compound distributions of the training and the testing set as sharply as possible.

COGS Another synthetic COGS dataset (Kim and Linzen, 2020) contains 24,155 pairs of English sentences and their corresponding logical forms. COGS contains a variety of systematic linguistic abstractions (e.g., active → passive, nominative → accusative and transtive verbs → intranstive verbs), thus reflecting compositionality of natural utterance. It is noteworthy that COGS with its testing data categorized into 21 classes by the compositional generalization type supports fine-grained evaluations.

GeoQuery The non-synthetic dataset of GeoQeury (Zelle and Mooney, 1996) collects 880 anthropogenic questions regarding the US geography (e.g., "what states does the mississippi run through ?") paired with their corresponding database query statements (e.g., "answer (state (traverse_1 (riverid (mississippi))))"). Following (Herzig and Berant, 2021; Yang et al., 2022), we also adopt the FunQl formalism of GeoQuery introduced by (Kate et al., 2005) and evaluate our methods on the compositional template split (***query*** split) from (Finegan-Dollak et al., 2018) where the output query statement templates of the training and testing set are disjoint and the *i.i.d.* split (***question*** split) where training set and testing set are randomly separated from the whole dataset.

5.2 Experimental Setup

Baselines We compare our methods with the following prior state-of-the-art baselines for compositional generalization. (1) Data augmentation methods: GECA (Andreas, 2020) and LexSym (Akyürek and Andreas, 2022) on all the three benchmarks, Prim2PrimX+MET (Jiang et al., 2022) which is a data augmentation methods further boosted by mutual exclusive training on SCAN and COGS, and SUBS (Yang et al., 2022) as the current state-of-the-art on GeoQuery. Besides, we additionally compare our methods with GECA+MAML (Conklin et al., 2021)(boost

Method	Jump	Around Right	MCD1	MCD2	MCD3
CGPS (Li et al., 2019)	98.8%± 1.4%	83.2%± 13.2%	1.2%± 1.0%	1.7%± 2.0%	0.6%± 0.3%
GECA+MAML (Conklin et al., 2021)	–	–	58.9%± 6.4%	34.5%± 2.5%	12.3%± 4.9%
Comp-IBT (Guo et al., 2020b)	99.6%	37.8%	64.3%	80.8%	52.2%
T5-11B (Raffel et al., 2020)	98.3%	49.2%	7.9%	2.4%	16.2%
LSTM	1.3%± 0.4%	10.2%± 4.6%	8.9%± 1.6%	11.9%± 9.4%	6.0%± 0.9%
+GECA (Andreas, 2020)	95.2%± 8.0%	84.3%± 6.3%	23.4%± 9.1%	25.5%± 8.8%	10.9%± 4.6%
+LexLearn (Akyurek and Andreas, 2021)	91.2%± 11.9%	95.3%±1.6%	12.5%± 2.0%	19.3%± 1.9%	11.6%± 0.9%
+LexSym (Akyürek and Andreas, 2022)	100.0%± 0.0%	84.0%±7.1%	47.4%± 7.1%	30.8%± 8.4%	13.7%± 3.6%
+Prim2PrimX+MET (Jiang et al., 2022)	7.3%± 5.6%	97.6%± 1.0%	31.5%± 4.1%	33.5%± 2.7%	11.6%± 1.0%
+GECA+MAML (Conklin et al., 2021)	95.8%± 6.9%	86.2%± 5.6%	28.2%± 9.6%	31.8%± 8.5%	11.2%± 4.2%
+SpanSub (**Ours**)	**100.0**%± 0.0%	99.9%±0.1%	63.4%± 13.1%	72.9%± 10.1%	74.0%± 10.2%
+SpanSub+L2S2 (**Ours**)	**100.0**%± 0.0%	**100.0**%± 0.0%	**67.4**%± 12.1%	**73.0**%± 10.1%	**80.2**%± 1.8%

Table 1: Test accuracy on SCAN Jump, Around Right and MCD splits.

Method	COGS
MAML (Conklin et al., 2021)	64.1%±3.2%
IR-Transformer (Ontanon et al., 2022)	78.4%
Roberta+Dangle (Zheng and Lapata, 2022)	87.6%
T5-Base (Raffel et al., 2020)	85.9%
LSTM	55.4%±4.2%
+GECA (Andreas, 2020)	48.0%±5.0%
+LexLearn (Akyurek and Andreas, 2021)	82.0% ±0.0%
+LexSym (Akyürek and Andreas, 2022)	81.4%±0.5%
+Prim2PrimX+MET (Jiang et al., 2022)	81.1%±1.0%
+SpanSub (**Ours**)	91.8%±0.1%
+SpanSub+L2S2 (**Ours**)	**92.3**%±0.2%

Table 2: Overall test accuracy on COGS dataset.

Method	question	query
SpanParse (Herzig and Berant, 2021)	78.9%	76.3%
LSTM	75.2%	58.6%
+GECA (Andreas, 2020)	76.8%	60.6%
+LexSym (Akyürek and Andreas, 2022)	81.6%	80.2%
+SUBS (Yang et al., 2022)	80.5%	77.7%
+SpanSub (**Ours**)	**82.4**%	**81.4**%
BART (Lewis et al., 2020)	90.2%	71.9%
+GECA (Andreas, 2020)	87.9%	83.0%
+LexSym (Akyürek and Andreas, 2022)	90.2%	87.7%
+SUBS (Yang et al., 2022)	**91.8**%	88.3%
+SpanSub (**Ours**)	90.6%	**89.5**%

Table 3: Test accuracy on GeoQuery question (i.i.d.) and query (compositional) splits.

GECA with meta-learning) and Comp-IBT (Guo et al., 2020b) which is also a data augmentation method requiring to access 30% testing inputs and outputs in advance. (2) Methods that incorporate the alignment of tokens or substructures: LexLearn (Akyurek and Andreas, 2021) on SCAN and COGS, IR-Transformer (Ontanon et al., 2022) on COGS, as well as SpanParse (Herzig and Berant, 2021) on GeoQuery. (3) Methods that design specialized architectures: CGPS (Li et al., 2019) on SCAN and Roberta+Dangle (Zheng and Lapata, 2022) on COGS. (4) We also report the results on SCAN and COGS from powerful pretrained T5 (Raffel et al., 2020) as reference.

Base Models In alignment with the previous works (Andreas, 2020; Akyurek and Andreas, 2021; Akyürek and Andreas, 2022), we adopt the LSTM-based seq-to-seq model (Sutskever et al., 2014) with the attention (Bahdanau et al., 2014) and copy (See et al., 2017) mechanisms as our base model on the SCAN and COGS benchmarks. For the non-synthetic dataset of GeoQuery, we follow SpanParse (Herzig and Berant, 2021) and SUBS (Yang et al., 2022) by using not only LSTM but also a more capable pre-trained language model BART (Lewis et al., 2020) as our base models. Detailed experimental settings are available in Appendix B.

Evaluation Metric Grounded on the semantic parsing task, we adopt the evaluation metric of exact-match accuracy in all of our experiments.

5.3 Main Results

The results of our experiments on SCAN, COGS and GeoQuery benchmarks are shown in Table 1, Table 2 and Table 3 respectively. Note that **"+SpanSub"** means that we directly use SpanSub to generate additional training data and train our base models on the original training data and the additional training data generated by SpanSub as well; **"+SpanSub+L2S2"** means that we (1): firstly augment the original training data with additionally generated data using SpanSub, (2): train the L2S2 framework (using Algorithm 2) on the augmented training data, and (3): get the trained base models from the L2S2 framework. We run each experiment on the 5 different seeds and report the mean and the standard deviation. We also do ablation studies and control experiments (in Appendix. D.2) to separately verify the effectiveness of SpanSub and L2S2 and their combination.

SCAN Results On all of the 5 splits (jump, around right, MCD1, MCD2 and MCD3) which we study in the SCAN benchmarks, SpanSub and the combination of it and L2S2 both lead to significant improvements for our base models. For easier/classic ***jump*** and ***around right*** splits, the performance of our base model improves to solving these two tasks completely. For more chanllenging ***MCD*** splits, when we leverage SpanSub to generate additional training data for our base model, the performance of it improves around 64% on average. Moreover, the adoption of L2S2 further boosts the performance by at most 6.2% on the basis of only using SpanSub. Using our methods obviously outperforms using the majority of other baseline methods, except for Comp-IBT on MCD2 split. Nonetheless, Comp-IBT requires to access 30% inputs and outputs in the testing set, so it is not directly comparable with ours.

COGS Results On COGS task, the performance of our base model(LSTM) increase from 55.4% to 91.8% when we use SpanSub to generate additional training data for it. SpanSub has approximately 10% lead compared with our baseline methods (LexLearn, LexSym, Prim2PrimX+MET) implemented on the same base model. Even compared with methods that leverage powerful pretrained models (e.g., Roberta+Dangle and T5-Base), LSTM+SpanSub still has some advantages. Furthermore, through adopting L2S2 on the basis of SpanSub, we can improve the performance of our base model from 91.8% to 92.3%.

GeoQuery Results On the compositional template ***query*** split, SpanSub leads to substantial and consistent improvement over other baseline data augmentation methods (GECA, LexSym and SUBS) on both of implementations based on LSTM and BART, achieving new state-of-the-art results (pushing forward the previously state-of-the-art results by 1.2%). As for the *i.i.d* ***question*** split, SpanSub still has advantages over baseline methods when based on LSTM model. When we adopt BART as our base model, SpanSub boosts the performance of BART by 0.4% which is ahead of GECA and LexSym, falling behind SUBS.

5.4 Analysis and Discussion

In this section, we aim to further answer the following four questions:

- Does the SpanSub help with fully exploring of augmentation space as supposed in Section 1?
- Does the L2S2 learn to realize the hardness-aware automatic data augmentation as supposed in Section 1?
- Ablation Studies and Control Experiments: Do the L2S2 and the SpanSub separately help with compositional generalization? Can their combination further improve generalization capactiy? Does the up-stream learnable augmentation module play an necessary role?
- Can the proposed L2S2 methods generalize to more types of down-stream neural sequence models (other than LSTM-based models, e.g., Transformers (Vaswani et al., 2017))?

Method	lex	s1	s2	s3
LSTM	69.3%	0.0%	0.0%	0.9%
+LexSym	95.3%	0.0%	0.0%	0.7%
+SpanSub	99.1%	91.8%	45.0%	7.2%
+SpanSub+L2S2	99.4%	93.7%	45.1%	10.7%

Table 4: Test accuracy of different generalization types in COGS task. "lex" refers to lexical generalization test; "s1","s2" and "s3" refer to "obj_pp_to_subj_pp", "pp_recursion", "cp_recursion" respectively, which are 3 different types of structural generalization tests.

Analysis of performances with SpanSub To further analyze the improvement of performance brought by SpanSub and L2S2, we break down the the performance on COGS task into four different part, including lexical generalization performance and three different types of structural generalization performances. Results are shown in Table 4. Compared with LexSym, which only enable single-grained substitutions (i.e., substituting for single words), we find that SpanSub can not only improve generalization on testing cases of different structural types, but also further boost the lexical level generalization.

Analysis of performances with L2S2 For results on SCAN(MCDs) tasks: We investigate the concrete substitution probabilities generated by L2S2 augmentor on MCD1 (where the complex concept "<verb> around <direction>" never co-occur with "twice" in the training set) split of SCAN task (training only with L2S2 framework). Given an example "run right thrice after walk opposite left twice", we keep on observing the probabilities of L2S2 augmentor selecting the span "walk opposite left" to be swapped out and selecting the spans

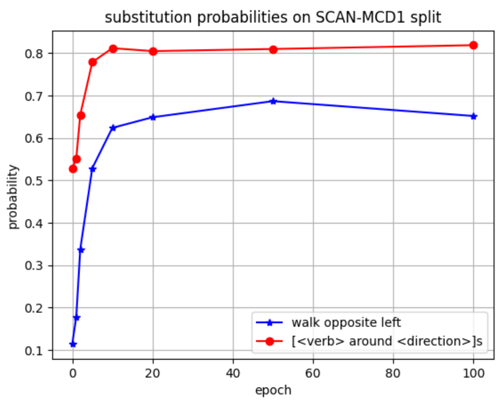

Figure 5: The variation curves of substitution probabilities with the training process going on. Given an training example "run right thrice after walk opposite left twice", The blue curve represents the variation curve of probabilities of swapping "walk opposite left" out and the red curve represents the variation curve of probabilities of swapping spans like "<verb> around <direction>s" in.

like "<verb> around <direction>" to be swapped in, with the training process going on. The results are shown in Fig 5.[4] As the training process goes on, L2S2 augmentor learns to compose spans like "<verb> around <direction>" and novel surrounding "twice". This exactly verify our hypothesis that L2S2 framework can automatically learn to put high value on the compositions of elusive concepts and novel surroundings. As a comparison with imbalanced prediction error rates shown in Fig 1(d), we put the results of additionally using L2S2 and RandS2 (which is the controlled version of L2S2, by substituting the learned parameters in the L2S2 with random ones.) in Table 6. We can conclude that L2S2 can effectively help with the performance of down-stream neural seq-to-seq models on the prediction of harder examples.[5]

For results on the COGS task: as shown in Table 4, we find that the utilization of L2S2 framework training can help SpanSub better generalize on testing cases of "cp_recursion" type. As shown in Fig 6, in SpanSub, "cp_recursion" type generalization cases require the compositions of concepts of sentential complements (e.g., "John knew **that** the cake was ate .") and novel surroundings (with deep recursion of **that**-structure). L2S2 framework training improves SpanSub on "cp_recursion"

[4]In this figure we count "epoch" (x-axis) after the end of the warm-up stage.

[5]Note that Fig 1(d) shows the results on SCAN-MCD1, and Table 6 shows the results on SCAN-MCD3. This slight mismatch does not change our conclusion here.

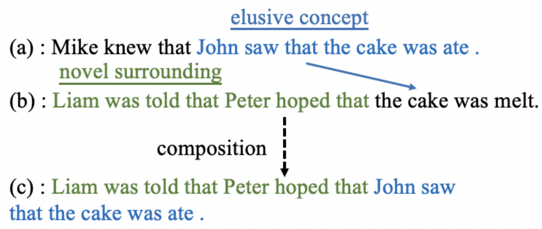

Figure 6: A composition that helps to improve "cp_recursion" generalization in SpanSub. The composition of "John saw that the cake was ate" and "Liam was told that Peter was hoped that" results in examples with deeper recursion of **that**-structure.

generalization through encouraging such compositions.

Ablation Study Except for the performance analysis provided above, we also do ablation study and control experiments to separately verify the effectiveness of SpanSub, L2S2 and their combination. Due to the page limit, our detailed experiment setting and results are shown in Table 8 in Appendix D.

Generalizing L2S2 to more based models Since we claim that our proposed L2S2 method is model-agnostic, here we generalize it to three different kind of base models[6]: one-layer LSTM used in (Andreas, 2020), two-layer LSTM used in (Akyurek and Andreas, 2021) and Transformer used in (Jiang et al., 2022). The experiments results are shown in Table 7 in Appendix D.

6 Conclusion

In this paper, (1) we present a novel substitution-based compositional data augmentation scheme, SpanSub, to enable multi-grained compositions of substantial substructures in the whole training set and (2) we introduce an online, optimizable and model-agnostic L2S2 framework containing a L2S2 augmentor which automatically learn the span substitution probabilities to put high values on those challenging compositions of elusive spans and novel surroundings and thus further boost the systematic generalization ability of down-stream nerual sequence models especially on those hard-to-learn compositions. Empirical results demonstrate the effectiveness and superiority of SpanSub, L2SS and their combination.

[6]here the term "base model" refers to down-stream neural seq-to-seq models in Fig 2.

7 Limitations

The techniques in SpanSub are constructed on the basis prior works of extracting span alignments and clustering words in the training data according to their syntactic role. There is no generic solution for these problem applicable for all of the datasets (this is mainly because the output formats and structures are diverse) at present, which requires users to spend efforts looking for preprocessing techniques applicable for their own datasets. However, the methodology of the proposed SpanSub is rather general to many different datasets and tasks (e.g., Semantic Parsing and Machine Translation). Besides, although we define eligible spans to try to alleviate additionally introducing noisy augmented data, our experiment result on GeoQuery (i.i.d. split) shows that SpanSub can still slightly hurt generalization performance (in comparison with other state-of-the-art methods). Hence we regard that relieving the potentially negative influence of noisy augmentation is important to further improve this work.

8 Acknowledgement

We sincerely thank the anonymous reviewers for giving useful feedback and constructive suggestions to the initial version of the paper. This work was supported by grants from the National Key R&D Program of China (No. 2021ZD0111801) and the National Natural Science Foundation of China (No. 62022077).

References

Ekin Akyurek and Jacob Andreas. 2021. Lexicon learning for few shot sequence modeling. In *ACL*, pages 4934–4946.

Ekin Akyürek and Jacob Andreas. 2022. Compositionality as lexical symmetry. *CoRR*, abs/2201.12926.

Jacob Andreas. 2020. Good-enough compositional data augmentation. In *ACL*, pages 7556–7566.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate.

Sourya Basu, Prasanna Sattigeri, Karthikeyan Natesan Ramamurthy, Vijil Chenthamarakshan, Kush R. Varshney, Lav R. Varshney, and Payel Das. 2022. Equi-tuning: Group equivariant fine-tuning of pretrained models. *ArXiv*, abs/2210.06475.

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.

Noam Chomsky. 1957. *Syntactic Structures*. Mouton and Co., The Hague.

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.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *North American Chapter of the Association for Computational Linguistics*.

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

Daniel Furrer, Marc van Zee, Nathan Scales, and Nathanael Schärli. 2020. Compositional generalization in semantic parsing: Pre-training vs. specialized architectures. *CoRR*, abs/2007.08970.

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. 2020a. 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.

Yinuo Guo, Hualei Zhu, Zeqi Lin, Bei Chen, Jian-Guang Lou, and Dongmei Zhang. 2020b. Revisiting iterative back-translation from the perspective of compositional generalization. In *AAAI Conference on Artificial Intelligence*.

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. *ArXiv*, abs/2104.07478.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*.

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.

Yichen Jiang, Xiaoping Zhou, and Mohit Bansal. 2022. Mutual exclusivity training and primitive augmentation to induce compositionality. *ArXiv*, abs/2211.15578.

Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. 2005. Learning to transform natural to formal languages. In *AAAI Conference on Artificial Intelligence*.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Najoung Kim and Tal Linzen. 2020. COGS: A compositional generalization challenge based on semantic interpretation. In *EMNLP*, pages 9087–9105.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

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

Brenden M. Lake. 2019. *Compositional Generalization through Meta Sequence-to-Sequence Learning*. Curran Associates Inc., Red Hook, NY, USA.

Brenden M. Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2879–2888. PMLR.

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

Yuanpeng Li, Liang Zhao, Jianyu Wang, and Joel Hestness. 2019. Compositional generalization for primitive substitutions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4293–4302, Hong Kong, China. Association for Computational Linguistics.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, Berlin, Germany. Association for Computational Linguistics.

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

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

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. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. 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.

Jacob Russin, Jason Jo, Randall O'Reilly, and Yoshua Bengio. 2020. Compositional generalization by factorizing alignment and translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 313–327, Online. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks.

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.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Jingfeng Yang, Le Zhang, and Diyi Yang. 2022. Subs: Subtree substitution for compositional semantic parsing. In *North American Chapter of the Association for Computational Linguistics*.

Pengcheng Yin, Hao Fang, Graham Neubig, Adam Pauls, Emmanouil Antonios Platanios, Yu Su, Sam Thomson, and Jacob Andreas. 2021. Compositional generalization for neural semantic parsing via span-level supervised attention. In *North American Chapter of the Association for Computational Linguistics*.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI, Vol. 2*.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*.

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.

A Datasets and Preprocessing

A.1 Datasets

SCAN Introduced by (Lake and Baroni, 2018), SCAN contains a large set of synthetic paired sequences whose input is a sequence of navigation commands in natural language and output is the corresponding action sequence. Following previous works (Andreas, 2020; Akyurek and Andreas, 2021; Jiang et al., 2022), we evaluate our methods on the two splits of ***jump*** (designed for evaluating a novel combination of a seen primitive, i.e., *jump*, and other seen surroundings) and ***around right*** (designed for evaluating a novel compositional rule). Notably, we also consider the more complex and challenging Maximum Compound Divergence (MCD) splits of SCAN established in (Keysers et al., 2020), which distinguish the compound distributions of the training and the testing set as sharply as possible.

COGS Another synthetic COGS dataset (Kim and Linzen, 2020) contains 24,155 pairs of English sentences and their corresponding logical forms. COGS contains a variety of systematic linguistic abstractions (e.g., active $\rightarrow$ passive, nominative $\rightarrow$ accusative and transtive verbs $\rightarrow$ intranstive verbs), thus reflecting compositionality of natural utterance. It is noteworthy that COGS with its testing data categorized into 21 classes by the compositional generalization type supports fine-grained evaluations.

GeoQuery The non-synthetic dataset of GeoQeury (Zelle and Mooney, 1996) collects 880 anthropogenic questions regarding the US geography (e.g., "what states does the mississippi run through ?") paired with their corresponding database query statements (e.g., "answer (state (traverse_1 (riverid (mississippi))))"). Following (Herzig and Berant, 2021; Yang et al., 2022), we also adopt the FunQl formalism of GeoQuery introduced by (Kate et al., 2005) and evaluate our methods on the compositional template split (***query*** split) from (Finegan-Dollak et al., 2018) where the output query statement templates of the training and testing set are disjoint and the *i.i.d.* split (***question*** split) where training set and testing set are randomly separated from the whole dataset.

We provide examples of the above three datasets as follows for readers' reference:

```
// a SCAN example
```

```
scan["input"] =
    "walk around right twice and jump left
        thrice"
scan["target"] =
    "TR W TR W TR W TR W TR W TR W
    TR W TR W TL J TL J TL J"
// a COGS example
cogs["input"] =
    "Amelia gave Emma a strawberry ."
cogs["target"] =
    "give . agent ( x _ 1 , Amelia ) AND give .
        recipient ( x _ 1 , Emma )
    AND give . theme ( x _ 1 , x _ 4 ) AND
        strawberry ( x _ 4 )"
// a GeoQuery example
geoquery["input"] =
    "what is the tallest mountain in america ?"
geoquery["target"] =
    "answer ( highest ( mountain ( loc_2 (
        countryid ( 'usa' ) ) ) ) )"
```

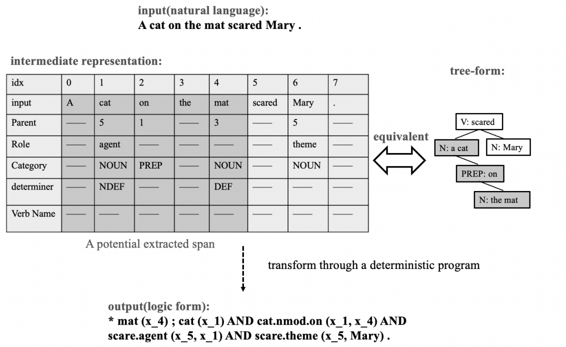

Figure 7: An instance for an intermediate representation, its corresponding tree-form and a potential extracted span for COGS.

A.2 Proprocessing of Datasets

Extraction of span alignments For SCAN dataset, since there is no off-the-shelf technique to map sequential data in SCAN dataset to tree-form, we slightly the modify algorithm SimpleAlign from (Akyurek and Andreas, 2021) to extract consecutive span alignments for our experiments on SCAN. We denote the input sequence as x, the output sequence as y, the span, which is going to be extracted from the input sequence, as v and its counterpart in the output sequence as w. Basically, we extract a pair of span alignment (v, w) following the maximally restrictive criterion:

$$
\begin{aligned}
&nec.(v, w) = \forall xy.(w \in y) \rightarrow (v \in x) \\
&suff.(v, w) = \forall xy.(v \in x) \rightarrow (w \in y) \qquad (7) \\
&C_1(v, w) = nec.(v, w) \wedge suff.(v, w)
\end{aligned}
$$

Both v and w are supposed to be consecutive fragments in the input sequence and output sequence respectively.

We additionally apply appropriate relaxations on the top of criterion(7) to enable the extraction of more spans: we tolerate many-to-one mapping and one-to-many mapping to some extent to avoid discarding of "<verb>s around <direction>s" and "<verb>s <direction>s"(e.g., both of interpretations of "walk around right" and "walk right" cover "TR W"). Besides, we manually set the maximum number of words in v to 3 and the maximum number of words in w to 8.

For COGS, we directly use the intermediate representation from (Ontanon et al., 2022). An instance of intermediate representation is shown in Fig 7. We search for every consecutive fragments in the intermediate presentations of COGS to extract eligible spans according to Definition 1. The naive implementation of the above search algorithm has the time complexity of $\mathcal{O}(n \cdot m^3)$, where n is the number of sentences in the training set and m is the maximal length of a single sentence in the training set.

For GeoQuery, following (Yang et al., 2022), we directly adopt the span trees (*gold trees*) extracted and aligned by (Herzig and Berant, 2021). And we refer the readers to get more detailed information about how to construct such span trees from the original paper (Herzig and Berant, 2021).

Note that we slightly correct several denotations in the original *gold trees* from (Herzig and Berant, 2021), for they are slightly differing from the ground-truth. To clarify it, we put an example of modification here (given that the others are similar, we do not present the others here):

```
geoquery["input"] =
    "what is the population of washington dc ?"
geoquery["program"] =
    "answer ( population_1 ( cityid (
        'washington', 'dc' ) ) )"
// the original gold_spans
geoquery["gold_spans"] =
    {"span": [5, 5], "type":
        "cityid#'washington'"}
// after correction
geoquery["gold_spans"] =
    {"span": [5, 6], "type":
        "cityid#'washington'"}
    // this is just one of the spans
    // washington dc is the capital city of USA;
    // washington is a state of USA;
```

To ensure a fair comparison with previous substitution-based data augmentation methods (Akyürek and Andreas, 2022; Yang et al., 2022), we rerun their methods on the modified

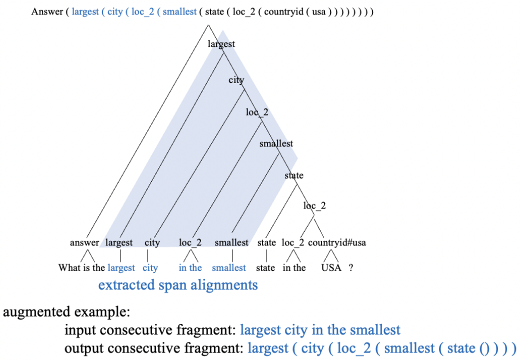

Figure 8: An instance for a constructed span tree and extracting a consecutive span from the span tree.

gold trees.

Inferring the equivalence class of words For COGS, we directly leverage the information in the intermediate representations to infer the equivalence class of the words (e.g., NOUN, VERB or PREP). For SCAN and GeoQuery, we use the technique of inferring the types of words form (Akyürek and Andreas, 2022), which cluster the words according to their shared contexts in the training set.

For GeoQuery, we additionally adopt context2vec methods (Melamud et al., 2016) (where we train a simple one-layer LSTM-based mask-reconstruction model) to boost the exploration of potentially syntactically-equivalent words (i.e., candidates to fill in the masked blank). We put the final result of word-clustering on GeoQuery here as follows:(We cluster the words in the target side)

```
/*
word clustering result for GeoQuery:
words not included are not syntactically
equivalent to any other words
*/
cluster1 = ['highest','major','largest',
            'smallest','shortest','lowest',
            'longest']
cluster2 = ['mountain','state','city',
            'river','place','lake']
cluster3 = ['loc_2','traverse_2']
cluster4 = ['countryid','cityid','stateid',
            'placeid']
cluster5 = ['traverse_1','loc_1','capital_2']
cluster6 = ['largest_one','smallest_one']
cluster7 = ['area_1','density_1','population_1']
cluster8 = ['size','high_point_1']
cluster9 = ['most','fewest']
```

B Training Details and Hyper-parameter Selection of Algorithms

In this section, we detailedly describe the training details of our models in our framework(up-stream L2S2 Augmentor and down-stream neural seq-to-seq model) and the selection of hyper-parameters in our Algorithms(SpanSub and L2S2).

B.1 L2S2 Augmentor

For both of SCAN and COGS experiments, we use an two layer bidirectional LSTM (with 128 hidden units and an embedding size of 128, a dropout rate of 0.5) as our sequence encoder. We separately use an embedding layer with an embedding size of 512 for the embedding module for spans to be swapped out and another embedding layer with an embedding size of 512 for the embedding module for spans to be swapped in. We use (cosine-similarity$\cdot$2) $\in [-2, 2]$ as all of our similarity functions in L2S2 augmentor. We set all of the temperatures for gumbel-softmax sampling in L2S2 augmentor to 1. Besides, we use a Adam optimizer (Kingma and Ba, 2014) to optimize our L2S2 augmentor with an learning rate of 1e-3. The above hyper-parameters are commonly used for LSTM-based models in NLP community and hence we did not spend extra efforts to tune them in our experiments.

B.2 Neural Seq-to-Seq Models

We keep this part of hyper-parameters aligned with previous baselines. For *jump* and *around right* splits of SCAN and COGS experiments, we keep the hyperparameters of our LSTM in align with (Akyurek and Andreas, 2021; Akyürek and Andreas, 2022; Jiang et al., 2022). We use a 2-layer encoder-decoder LSTM (with attention (Bahdanau et al., 2014) and copy (See et al., 2017) mechanisms) with 512 hidden units and an embedding size of 512, a droupout rate of 0.4. For *MCD*1, *MCD*2 and *MCD*3 splits of SCAN experiments, the hyperparameters of our LSTM are adopted form (Andreas, 2020). We use a 1-layer bidirectional encoder-decoder LSTM (with attention and copy mechanisms) with 512 hidden units and an embedding size of 64, a droupout rate of 0.5. For all of these above experiments, we train our model with an Adam optimizer with an initial learning rate of 1e-3. We use an ReduceLROnPlateau scheduler (implemented in PyTorch) with a scale factor of 0.5 to automatically reduce our learning rate. We set all of the batch size to 128.

For GeoQuery tasks, in align with SUBS (Yang et al., 2022), we also directly use OpenNMT (Klein et al., 2017) to implement our LSTM-based model with attention and copy mechanisms and we utilize fairseq (Ott et al., 2019) to implement our BART-based model. For LSTM-based experiments, we use one-layer bidirectional LSTM in the encoder side and one-layer unidirectional LSTM in the decoder side. We use dropout with a rate of 0.5 and Adam optimizer with a learning rate of 1e-3. We use MLP attention and directly use the attention scores as copying scores and we set the batch size for experiments based on LSTM to 64. For BART-based experiments, we use BART-base models updated by Adam optimizer with a learning rate of 1e-5. We set the rate for both dropout and attention dropout to 0.1 and we use label smoothing with a rate of 0.1. We set the batch size for all of the experiments based on BART to 1024 tokens. Besides, we set the rate of the weight-decay to 0.01.

B.3 Hyper-parameters in SpanSub(Algorithm 1)

For *jump* and *around right* splits of SCAN and GeoQuery experiments, we set the iterative depth K in SpanSub augmentation scheme to 1. For *MCD* splits of SCAN experiments, we set the iterative depth K in SpanSub augmentation scheme to 2. For COGS experiments, we set the iterative depth K in SpanSub augmentation scheme to 4. For SCAN experiments, we set the number of generated examples N (without de-duplicating) to 1e5. For COGS experiments, we set the number of generated examples N (without de-duplicating) to 4e5. For GeoQuery experiments, we simply searching for every potential augmentations in the training set (because the training set for GeoQuery contains merely 519 examples, we try to make the best use of each example.), and the size of augmented set is shown in Table 5. Following (Jia and Liang, 2016; Qiu et al., 2022), we also ensure approximately equal number of the original examples and the augmented examples being used for training in SpanSub experiments, giving consideration to both of i.i.d. generalization and compositional generalization.

We decide the iterative depth K through observing that from which iteration there are nearly no more novel data generated. For N, we simply set a number which is large enough compared with the size of the original dataset, and then we de-duplicate the augmented dataset.

B.4 Hyper-parameters in Training L2S2 framework(Algorithm 2)

One crucial hyper-parameter in Training L2S2 framework is the warm-up epochs / update steps. In most cases, we need to set an appropriate value to warm-up update steps to guarantee the down-stream sequence model to be fully aware of the distribution (hardness) of the original training examples while not over-fit to them. For most of our experiments(*jump*, *around right*, *MCD1* and *MCD2* splits of SCAN experiments, COGS experiments), we set the warm-up epoch to 5, and then we alternatively train the up-stream module and down-stream module in the L2S2 framework to 150 epochs in total. For *MCD2* split of SCAN experiments, we first train our neural seq-to-seq model for 80 epochs, and then we alternatively train the up-stream L2S2 augmentor and the down-stream neural seq-to-seq model for 70 epochs[7]. For experiments with L2S2 framework, we set the number of sampled actions T for each example to 4. All of this part of hyper-parameters are decided by cross-validation.

Other Training Details We conduct all of our experiments on NVIDIA GeForce RTX2080Ti GPUs. For *jump* and *around right* splits of SCAN, COGS and GeoQuery, we select our model for testing with the best development accuracy. For all *MCD* splits of SCAN, we use the train/dev/test splits from the original paper (Keysers et al., 2020)[8], we also select our model for testing with the best accuracy on dev set.

C Definitions and Algorithms

In this section, we mainly describe the pseudo-code of SpanSub and L2S2, and the formal description of the term "span".

[7] In our initial experiments, we found that L2S2 method only slightly works on the *MCD2* split of SCAN dataset when using 1 layer LSTM-based model as the down-stream sequence model. However, in the following experiments in Table 7, we found that it works well on other 2 down-stream sequence models (we set warm-up epoch number to 5 for other down-stream seq-to-seq models).

[8] The official github repo is https://github.com/google-research/google-research/tree/master/cfq#scan-mcd-splits, and one can download the dataset from https://storage.cloud.google.com/cfq_dataset/scan-splits.tar.gz

C.1 SpanSub

Different from (Yang et al., 2022), we extract any consecutive fragments as our spans. An instance for the constructed span tree and extracting a consecutive span from the span tree is shown in Fig 8. And we give the formal description of the term "span" used throughout this paper.

Definition 1 *(**Eligible Span**) Given a sentence or a program sequence $S = [e_0, e_1, ..., e_n]$, there exists one and only one multi-way tree T corresponding to S, the in-order traversal sequence*[9] *Λ of which is $v_0 \to v_1 \to ... \to v_n$ (node v_i corresponds to token e_i, $0 \leq i \leq n$). Any span $S' = [e_p, e_{p+1}, ..., e_{p+k}] \subseteq S$, where $0 \leq p \leq p + k \leq n$, corresponds to a sub-sequence Λ' of Λ (i.e., $v_p \to v_{p+1} \to ... \to v_{p+k}$). Moreover, an eligible span S' also corresponds to a connected substructure T' of T, which meet the following 2 requirements:*

- *there is at most one node $v_i \in \Lambda'$ which is the child node of node $v \in \Lambda \backslash \Lambda'$*[10]*;*

- *there is at most one node $v_o \in \Lambda'$ which is the parent node of node $v \in \Lambda \backslash \Lambda'$;*

Note that each node in the tree T has one parent node and at least one child node. Specially, the parent node of the root node and the child node(s) of the leaf node(s) are special imaginary nodes.

Plus, we append the pseudo-code of SpanSub here in Algorithm 1. Note that:

For SCAN task, we only substitute spans in the both the input side and target side simultaneously when there is no confusion:

- If there are repetitively matched spans in either input side or output side, we substitute all of those repetitive ones at the same time. For example, input "walk and walk twice" is supposed to be interpreted as the target "W W W". If we are going to substitute "walk" with "jump" in the input side and its counterpart "W" with "J" in the target side, we are supposed to simultaneously substitute all of the matched spans, resulting in "jump and jump twice" $\to$ "J J J".

- If there are more than one kinds of span-matchs (in either input side or target side) and there is(are) overlap(s) between these matchs, we discard this example to alleviate the introduction of imprecise substitution. For example, input "walk around right thrice" is supposed to be interpreted as the target "<SOS> TR W TR W TR W TR W TR W TR W TR W TR W TR W TR W TR W TR W <EOS>" (supposing that we have already extracted the span "walk around right" $\to$ "TR W TR W TR W TR W"). However, we can not simultaneously substitute the "walk around right" in the input side and "TR W TR W TR W TR W" in the target side for there are many kinds of match (e.g., both of index[1, 5] and index[3, 7] are "TR W TR W TR W TR W".) in the target side and there exist overlaps between them.

Since GeoQuery is a highly realistic dataset (hence there are not always one-to-one mappings between words in the input sentences and words in the target programs, which potentially results generation of many noisy data.), we additionally impose two constraints to help with filter these generated noisy data: 1) if a modifier word in the target side(e.g., "largest_one") could be mapped to several different words in the input side(e.g.,"largest", "most", ...), we need to pay attention when substituting the words(e.g., "area_1") modified by this modifier or the modifier itself : we discard the synthetic new data covering the novel <modifier, modified word> combinations (e.g., "largest area" $\to$ "largest_one (area_1)", while "most area" makes no sense.); 2) if a modified word in the input side(e.g., "largest") could be mapped to several different words in the target side(e.g., "largest", "largest_one" and "longest"), we can induce that words in the target side like "river" can only follow after "longest" if there is no case in the training set showing that "river" can follow after other interpretation of "largest" (i.e., "largest" and "largest_one"). Hence we can directly discard those synthetic examples covering "largest (river (.." or "largest_one (river (..".

C.2 L2S2 framework

Here we also append the pseudo-code of training L2S2 framework in Algorithm 2.

[9] In our case in-order traversal of a multi-way tree is to traverse the most left child, traverse the root node and then traverse left childs from right to left in order.

[10] If there is no such node, we specifiy that the first node in the in-order traversal sequence is v_i.

Algorithm 1: SpanSub

Input: Original dataset $\mathcal{D}$, the number of generated examples N, Span-Alignments extraction algorithm $\mathcal{A}$, Span-Classification function Π, Iterative Depth K.

Output: Augmented dataset $\mathcal{D}_{aug}$.

$align, spans \leftarrow$ Run $\mathcal{A}$ on $\mathcal{D}$;
$\mathcal{D}_{train} \leftarrow \mathcal{D}$;
for $i \leftarrow 1$ *to* K **do**
  $\mathcal{D}_{aug} \leftarrow \{\}$;
  for $j \leftarrow 1$ *to* N **do**
    Uniformly draw $d \in \mathcal{D}_{train}$;
    $(inp, out) \leftarrow d$;
    Uniformly draw span s from inp;
    Uniformly draw span $s' \in \{v | v \in spans, \Pi(v) = \Pi(s)\}$;
    $inp_{aug} \leftarrow$ substitute s with s' in inp;
    $out_{aug} \leftarrow$ substitute $align(s)$ with $align(s')$ in out;
    $d_{aug} \leftarrow (inp_{aug}, out_{aug})$;
    $\mathcal{D}_{aug} \leftarrow \mathcal{D}_{aug} \cup \{d_{aug}\}$ ▷ dedup
  $\mathcal{D}_{train} \leftarrow \mathcal{D}_{aug} \cup \mathcal{D}_{train}$;
return $\mathcal{D}_{aug}$

Algorithm 2: Training L2S2 framework

Input: Original dataset $\mathcal{D}$,
L2S2 generator initialized parameters ϕ_0,
Seq-to-Seq Model initialized parameters θ_0,
Warm-up update number m,
Sampled action number for each given example T.

Output: L2S2 generator parameters ϕ_f,
Seq-to-Seq Model parameters θ_f.

$\theta \leftarrow \theta_0; \phi \leftarrow \phi_0$
for *step* $\leftarrow 1$ *to* m **do**
  Sample $\mathcal{B} \sim \mathcal{D}$;
  Optimize θ on $\mathcal{B}$ through Objective 5
while *not converged* **do**
  Sample $\mathcal{B} \sim \mathcal{D}$;
  for $t \leftarrow 1$ *to* T **do**
    Sample $\mathcal{B}_{gen,t} \sim p(\mathcal{B}_{gen} | \mathcal{B}, \phi)$;
  Optimize ϕ on $\{B_{gen,t}\}_{t=1}^{T}$ through Objective 6
  Sample $\mathcal{B} \sim \mathcal{D}$;
  Sample $\mathcal{B}_{gen} \sim p(\mathcal{B}_{gen} | \mathcal{B}, \phi)$;
  Optimize θ on B_{gen} through Objective 5
return ϕ, θ

w/o Aug	GECA	LexSym	SUBS	SpanSub
519	2,028	28,520	20,564	99,604

Table 5: The maximum numbers of distinct augmented examples on the query split of GeoQuery dataset with different augmentation methods. w/o Aug refers to the number of original training examples.

D Additional Experiments

In this section, we mainly provide additional experiment results to support the conclusions in the main text(Section D).

D.1 The maximum numbers of distinct augmented examples with different augmentation methods on GeoQuery task

As we discussed in Section 1, we hypothesize that SpanSub enables multi-grained compositions of substantial substructures in the whole training set and thus lead to improvement for various kinds of compositional generalization. We provide a statistic on the maximum number of augmented examples (after deduplication) on the query split of GeoQuery dataset with different augmentation methods, including GECA, LexSym, SUBS and SpanSub in Table 5. SpanSub overwhelmingly outweigh other augmentation methods and even their summation, which reflects its superiority of exploring potential compositions of substantial substructures in the whole training set.

D.2 Ablation Studies and Control Experiments

In this section, we investigate the effect of SpanSub, L2S2 framework training and their combination. Besides, we also investigate the effectiveness of the optimizable L2S2 augmentor in the L2S2 framework through control experiments. Our results are shown in Table 8.

Effectiveness of SpanSub and L2S2 framework training Through observing the experiment results of "LSTM"-group, "+L2S2"-group, "+SpanSub"-group and "+SpanSub+L2S2"-group on SCAN MCD(1,2,3) and COGS tasks, we can induce a consistent conclusion that : (1) both of the SpanSub data augmentation method and the L2S2 framework training method can improve the performance of our base model and (2) the combination

Error Type	walk right	walk opposite right	walk around right
RandS2	51.2%	28.1%	76.8%
L2S2	37.4%	14.6%	40.2%

Table 6: Comparision of the error rates(↓) of examples with different concepts (i.e., spans) between RandS2 and L2S2. Results are attained using the same LSTM architecture with (Andreas, 2020) on SCAN-MCD3 split.

of them, SpanSub+L2S2, can further boost the performance of our base model. These empirically verify the effectiveness of both SpanSub and L2S2 parts.

Effectiveness of L2S2 augmentor in L2S2 framework Furthermore, to verify the the effectiveness of the optimizable L2S2 augmentor part in the L2S2 framework, we design control experiments where the L2S2 augmentor is substituted with a non-differentiable random augmentor (The function of random augmentor is to randomly substitute a span in the given example with another span in the span set.) and everything else is maintained (We name it "RandS2"). Through observing the results of "+SpanSub", "+SpanSub+RandS2" and "+SpanSub+L2S2", we can draw a conclusion that RandS2 is not capable of functioning as L2S2 when being combined with SpanSub and in some cases RandS2 even has slight negative influence on SpanSub. Through observing the results of "+RandS2" and "+L2S2", we can similarly induce that RandS2 can not work as well as L2S2 on SCAN-MCD splits when being utilized alone . The reason for RandS2 can also improve the performance of based models is that RandS2 can be viewed as an online version SpanSub here. To conclude, we empirically verify the effectiveness of L2S2 augmentor in L2S2 framework through comparing the effect of it with the effect of a random augmentor.

D.3 Experiments with different kinds of Base Models

A significant advantage of our SpanSub and L2S2 is their model-agnostic [11] property so that we can easily apply these techniques to various base models with different architectures. In this section, we aim to answer the question that whether our proposed SpanSub and L2S2 methods can consistently help improve the compositional generalization of standard base models with different architectures(e.g., LSTM seq-to-seq models with different architectures, and Transformer (Vaswani et al., 2017)) or not?

[11]Here the term of model means the down-steam sequence-to-sequence model.

Method	MCD1	MCD2	MCD3
LSTM$_1$	8.9%± 1.6%	11.9%± 9.4%	6.0%± 0.9%
+RandS2	46.6%± 8.9%	52.3%± 2.4%	58.8%± 3.1%
+L2S2	**55.1%**± 17.6%	**54.3%**± 8.0%	**70.8%**± 5.0%
+SpanSub	63.4%± 13.1%	72.9%± 10.1%	74.0%± 10.2%
+SpanSub+RandS2	63.3%± 11.7%	66.2%± 6.6%	71.2%± 13.9%
+SpanSub+L2S2	**67.4%**± 12.1%	**73.0%**± 10.1%	**80.2%**± 1.8%
LSTM$_2$	6.8%± 3.5%	9.6%± 3.0%	9.3%± 2.5%
+RandS2	41.4%± 4.2%	64.1%± 7.6%	70.1%± 5.4%
+L2S2	**44.3%**± 6.7%	**65.9%**± 6.7%	**76.5%**± 4.3%
+SpanSub	52.7%± 5.1%	71.0%± 6.4%	78.9%± 2.3%
+SpanSub+RandS2	55.1%± 6.4%	73.4%± 6.5%	78.5%± 6.2%
+SpanSub+L2S2	**55.4%**± 8.6%	**74.1%**± 5.5%	**80.8%**± 7.4%
Transformer	1.7%± 0.7%	4.3%± 1.3%	4.4%± 1.2%
+RandS2	11.2%± 2.2%	37.0%± 7.1%	48.1%± 2.6%
+L2S2	**19.3%**± 2.2%	**68.1%**± 1.7%	**57.8%**± 2.2%
+SpanSub	24.8%± 1.7%	79.4%± 1.5%	61.3%± 0.9%
+SpanSub+RandS2	21.0%± 1.9%	**80.2%**± 2.3%	60.3%± 1.3%
+SpanSub+L2S2	**27.0%**± 4.4%	**80.2%**± 1.9%	**63.3%**± 2.3%

Table 7: Experiments on SCAN-MCDs splits with three standard seq-to-seq models with different architectures. Note that ***LSTM***$_1$ is the LSTM-based seq-to-seq model in align with (Andreas, 2020) (base on one-layer LSTM and embedding dimension of 64) and ***LSTM***$_2$ is the LSTM-based seq-to-seq model in align with (Akyürek and Andreas, 2022) (based on two-layer LSTM and embedding dimension of 512). ***Transformer*** is the standard seq-to-seq model introduced by (Vaswani et al., 2017). Here we use a transformer adopted from (Jiang et al., 2022), with a three-layer encoder and a three-layer decoder (both encoder layers and decoder layers contain self-attention layers and fully-connected layers).

Firstly, we have empirically demonstrated the effectiveness of both proposed SpanSub and L2S2 methods on SCAN (standard splits and MCD splits) tasks with LSTM-based seq-to-seq model (in line with (Andreas, 2020))and COGS task with another distinct LSTM architecture (in line with (Akyürek and Andreas, 2022)) respectively in Section 5.3. Moreover, here we conduct more experiments on SCAN-MCD splits with LSTM architecture (in line with (Akyürek and Andreas, 2022)) and Transformer to demonstrate that Span and L2S2 can consistently help improve the compositional generalization of standard base models with different architectures. Our results are shown in Table 7. Through observing these results, we find that our previous conclusions consistently hold with these three different standard seq-to-seq models (i.e., ***LSTM***$_1$, ***LSTM***$_2$ and ***Transformer***), which stands for that both SpanSub and L2S2 can help various down-stream sequence models better compositionally generalize.

Method	MCD1	MCD2	MCD3	COGS
LSTM	8.9%± 1.6%	11.9%± 9.4%	6.0%± 0.9%	55.4%± 4.2%
+RandS2 (**Control**)	46.6%± 8.9%	52.3%± 2.4%	58.8%± 3.1%	**89.7**%± 0.2%
+L2S2 (**Ours**)	**55.1**%± 17.6%	**54.3**%± 8.0%	**70.8**%± 5.0%	**89.7**%± 0.2%
+SpanSub (**Ours**)	63.4%± 13.1%	72.9%± 10.1%	74.0%± 10.2%	91.8%± 0.1%
+SpanSub+RandS2(**Control**)	63.3%± 11.7%	66.2%± 6.6%	71.2%± 13.9%	91.9%± 0.1%
+SpanSub+L2S2 (**Ours**)	**67.4%**± 12.1%	**73.0%**± 10.1%	**80.2%**± 1.8%	**92.3**%± 0.2%

Table 8: Ablation studies of SpanSub and L2S2 and comparison with control group(RandS2).

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section 7

☐ A2. Did you discuss any potential risks of your work?
Not applicable. Left blank.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
very first of our paper and Section1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section3,Section4

☑ B1. Did you cite the creators of artifacts you used?
Section3,Section4

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Not applicable. Left blank.

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Section3,Section4

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 5 and Appendix A

C ☑ Did you run computational experiments?

Section 5 and Appendix D

☐ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Not applicable. Left blank.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section5, AppendixB

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section5

☐ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Not applicable. Left blank.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.