Layer-Wise Representation Fusion for Compositional Generalization

Yafang Zheng[1,2*], **Lei Lin**[1,2,3*], **Shuangtao Li**[1,2], **Yuxuan Yuan**[1,2], **Zhaohong Lai**[1,2],
Shan Liu[1,2], **Biao Fu**[1,2], **Yidong Chen**[1,2], **Xiaodong Shi**[1,2†]

[1]Department of Artificial Intelligence, School of Informatics, Xiamen University
[2] Key Laboratory of Digital Protection and Intelligent Processing of Intangible Cultural Heritage
of Fujian and Taiwan (Xiamen University), Ministry of Culture and Tourism, China
[3]Kuaishou Technology, Beijing, China
{zhengyafang, linlei}@stu.xmu.edu.cn, {ydchen, mandel}@xmu.edu.cn

Abstract

Existing neural models are demonstrated to struggle with compositional generalization (CG), i.e., the ability to systematically generalize to unseen compositions of seen components. A key reason for failure on CG is that the syntactic and semantic representations of sequences in both the uppermost layer of the encoder and decoder are entangled. However, previous work concentrates on separating the learning of syntax and semantics instead of exploring the reasons behind the representation entanglement (RE) problem to solve it. We explain why it exists by analyzing the representation evolving mechanism from the bottom to the top of the Transformer layers. We find that the "shallow" residual connections within each layer fail to fuse previous layers' information effectively, leading to information forgetting between layers and further the RE problems. Inspired by this, we propose LRF, a novel **L**ayer-wise **R**epresentation **F**usion framework for CG, which learns to fuse previous layers' information back into the encoding and decoding process effectively through introducing a *fuse-attention module* at each encoder and decoder layer. LRF achieves promising results on two realistic benchmarks, empirically demonstrating the effectiveness of our proposal. Codes are available at https://github.com/thinkaboutzero/LRF.

Introduction

The remarkable progress of sequence-to-sequence (seq2seq) models in language modeling has been primarily attributed to their ability to learn intricate patterns and representations from vast amounts of data (Sutskever, Vinyals, and Le 2014; Dong and Lapata 2016; Vaswani et al. 2017). However, a critical challenge that remains unsolved for neural sequence models is the ability to understand and produce novel combinations from known components (Fodor and Pylyshyn 1988; Lake et al. 2017), i.e., *compositional generalization* (CG). For example, if a person knows "the doctor has lunch" [Der Arzt hat Mittagessen] and "the lawyer" [Der Anwalt] where the segment in "[]" denotes the German translation, then it is natural for the person to know the translation of "the lawyer has lunch" is [Der Anwalt hat Mittagessen] even though they
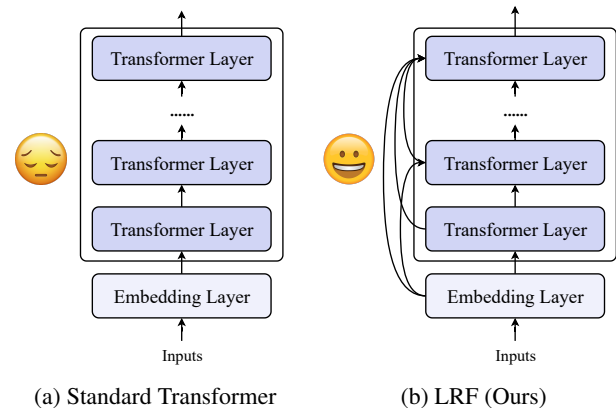

Figure 1: Comparison between (a) Standard Transformer and (b) LRF (ours). (a) illustrates the standard architecture of the Transformer model and (b) illustrates the architecture of our method, which fuses representations from previous layers at each layer effectively.

have never seen it before. Such nature is beneficial for models to perform robustly in real-world scenarios, as even huge training data can not cover a potentially infinite number of novel combinations.

Recent studies have demonstrated that a key reason for failure on CG is that the syntactic and semantic representations of sequences in both the uppermost layer of the encoder and decoder are entangled, i.e., encoder or decoder RE problem (Russin et al. 2019; Li et al. 2019; Raunak and et al 2019; Liu et al. 2020a, 2021; Mittal et al. 2022; Zheng and Lapata 2022; Thrush 2020; Akyurek and Andreas 2021; Yao and Koller 2022; Yin et al. 2022). To alleviate it, one line of research on CG concentrates on utilizing separate syntactic and semantic representations. Specifically, they either produce two separate syntactic and semantic representations and then compose them appropriately (Li et al. 2019; Russin et al. 2019; Jiang and Bansal 2021), or design external modules and then employ a multi-stage generation process (Liu et al. 2020b, 2021; Ruis and Lake 2022; Li et al. 2022; Cazzaro et al. 2023).

However, they focus on separating the learning of syn-

---

tax and semantics. Different from them, we explore the reasons behind the RE problem to solve it. We get insights from the findings that the residual connections are "shallow", and conclusions on analyzing the Transformer (Peters et al. 2018; He, Tan, and Qin 2019; Voita, Sennrich, and Titov 2019; Belinkov et al. 2020) show that the bottom layers of the Transformer contain more syntactic information while the top ones contain more semantic information. As shown in Figure 1a, the residual connections within each layer have been "shallow" themselves, and only pass through simple, one-step operations (Yu et al. 2018), which make the model "forget" distant layers and fail to fuse information from previous layers effectively (Bapna et al. 2018; Dou et al. 2018; Wang et al. 2018, 2019). Furthermore, it is clear that the nature of the RE problems are lacking effective fusion of syntactic and semantic information stored in different layers, since the representations are gradually entangled from bottom to top of the Transformer layers (Raunak and et al 2019; Russin et al. 2019; Zheng and Lapata 2022).

Based on the above findings, we hypothesize that "shallow" residual connections within each layer are one of the reasons resulting in the RE problems. To this end, we propose LRF, a novel **L**ayer-wise **R**epresentation **F**usion framework for CG, which learns to fuse previous layers' information at each layer effectively. Specifically, we extend the base model by fusing previous layers' information back into the encoding and decoding process by introducing a fuse-attention module in each encoder and decoder layer.

Experimental results on CFQ (Keysers et al. 2020) (semantic parsing) and CoGnition (Li et al. 2021) (machine translation, MT) empirically show that our method achieves better generalization performance, outperforming competitive baselines and other techniques. Notably, LRF achieves **20.0%** and **50.3%** (about **30%**, **20%** relative improvements) for instance-level and aggregate-level error reduction rates on CoGnition. Extensive analyses demonstrate that fusing previous layers' information at each layer effectively leads to better generalization results, outperforming competitive baselines and more specialized techniques.

Methodology

We adopt the Transformer (Vaswani et al. 2017) as an example for clarification, however, **our proposed method is applicable to any seq2seq models**. In the following, we first introduce the Transformer, and then our proposed LRF.

Transformer

Given a sequence of a source sentence and a target sentence $X = \{x_1, ..., x_S\}, Y = \{y_1, ..., y_T\}$, where S, T denote the number of source and target tokens, respectively. The Transformer encoder first maps X to embedding matrix H^0, and then takes H^0 as input and outputs a contextualized representation $H^L \in \mathbb{R}^{d \times S}$, where d, L denote the hidden size and the number of encoder layers respectively. Similarly, the Transformer decoder maps Y to embedding matrix $\hat{H}^0$ first, and then takes $\hat{H}^0$ as input and outputs a contextualized representation $\hat{H}^L \in \mathbb{R}^{d \times T}$.
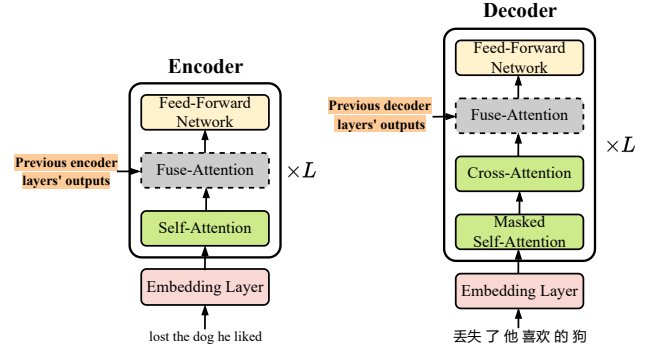

Figure 2: Architecture of LRF based on the Transformer. The dotted boxes denote the fuse-attention module.

Attention Mechanism. An attention function can be described as a query (Q) and a set of key-value (K-V) pairs mapped to an output. Formally, given Q, K, and V, the scaled dot product attention mechanism is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{Q^\top K}{\sqrt{d_k}})V, \qquad (1)$$

where d_k is the dimension of K.

A typical extension of the above is multi-head attention (MHA), where multiple linear projections are executed in parallel. The calculation process is as follows:

$$\text{MHA}(Q, K, V) = [head_1; ...; head_h]W^O, \qquad (2)$$

$$head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V), \qquad (3)$$

where $W_i^Q \in \mathbb{R}^{d \times d_k}$, $W_i^K \in \mathbb{R}^{d \times d_k}$, $W_i^V \in \mathbb{R}^{d \times d_v}$ and $W_i^O \in \mathbb{R}^{hd_v \times d}$ are model parameters. h denotes the number of heads.

Layer Structure. The Transformer encoder has L identical layers, and each layer consists of two sub-layers (i.e., self-attention and feed-forward networks). The Transformer decoder has L identical layers, and each layer consists of three sub-layers (i.e., masked self-attention, cross-attention and feed-forward networks). In the l-th self-attention layer of the encoder, the query, key and value are all the hidden states outputted by the previous layer H^{l-1}. The self-attention mechanism in the decoder operates in a similar manner. The formal expression is as follows:

$$H_a^l = \text{MHA}(H^{l-1}, H^{l-1}, H^{l-1}), \qquad (4)$$

$$\hat{H}_a^l = \text{MHA}(\hat{H}^{l-1}, \hat{H}^{l-1}, \hat{H}^{l-1}). \qquad (5)$$

In the l-th cross-attention layer of the decoder, the query is the hidden states outputted by the l-th self-attention layer $\hat{H}_a^l$ and the key and value are all hidden states outputted by the uppermost layer of the encoder H^L. The computation process is as follows:

$$\hat{H}_{ca}^l = \text{MHA}(\hat{H}_a^l, H^L, H^L). \qquad (6)$$

The feed-forward sub-layer is a two-layer transformation with a ReLU activation function:

$$H_l = W_2^l \text{ReLU}(W_1^l H_a^l + b_1^l) + b_2^l, \qquad (7)$$

$$\hat{H}_l = W_4^l \text{ReLU}(W_3^l \hat{H}_{ca}^l + b_3^l) + b_4^l, \qquad (8)$$

where $W_1^l, b_1^l, W_2^l, W_3^l, b_2^l, W_4^l, b_3^l$ and b_4^l are all trainable model parameters. We omit the layer normalization and residual connection for brevity.

Layer-Wise Representation Fusion (LRF)

Our proposed LRF extends the Transformer by introducing a fuse-attention module under the feed-forward module in each encoder and decoder layer, which learns to fuse information from previous layers at each encoder and decoder layer effectively.

Fuse-Attention Module. In the l-th encoder layer, all previous layers' outputs are stacked as $H_{plo} \in \mathbb{R}^{d \times l}$, where l is the number of previous layers (include the embedding layer). The fuse-attention module fuses different aspects of language information from previous layers effectively via the multi-head attention mechanism:

$$H_p^l = \text{MHA}(H_a^l, H_{plo}, H_{plo}), \qquad (9)$$

where $H_{plo} = \{H_0, ..., H_{l-1}\}$, and the localness of fuse-attention module is implemented by mask mechanisms. The output H_p^l is fed into the l-th feed-forward sub-layer of the encoder (Eq. 7).

In the l-th decoder layer, all previous layers' outputs are stacked as $\hat{H}_{plo} \in \mathbb{R}^{d \times l}$, where l is the number of previous layers (include the embedding layer). The fuse-attention module fuses different aspects of language information from previous layers effectively via the multi-head attention mechanism:

$$\hat{H}_p^l = \text{MHA}(\hat{H}_{ca}^l, \hat{H}_{plo}, \hat{H}_{plo}), \qquad (10)$$

where $\hat{H}_{plo} = \{\hat{H}_0, ..., \hat{H}_{l-1}\}$, and the localness of fuse-attention module is implemented by mask mechanisms. The output $\hat{H}_p^l$ is fed into the l-th feed-forward sub-layer of the decoder (Eq. 8).

The differences between LRF and Transformer are illustrated by the dotted boxes in Figure 2. By introducing a fuse-attention module in every encoder and decoder layer, each layer of the encoder and decoder are able to access and fuse previous layers' information effectively.

Training. Formally, $\mathcal{D} = \{(X, Y)\}$ denotes the training corpus, $\mathcal{V}$ denotes the vocabulary of $\mathcal{D}$. LRF aims to estimate the conditional probability $p(y_1, ..., y_T | x_1, ..., x_S)$, where $(x_1, ..., x_S)$ is an input sequence and $(y_1, ..., y_T)$ is its corresponding output sequence:

$$p(Y|X; \{\theta^{(0)} \cup \theta_+\}) = \prod_{t=1}^{T+1} p(y_t|y_{<t}, X; \{\theta^{(0)} \cup \theta_+\}), \quad (11)$$

where t is the index of each time step, $y_{<t}$ denotes a prefix of Y, $\theta^{(0)}$ denotes initial parameters of the Transformer model, θ_+ denotes parameters of the fuse-attention modules and each factor $p(y_t|X, y_1, ..., y_{t-1}; \{\theta^{(0)} \cup \theta_+\})$ is defined as a softmax distribution of $\mathcal{V}$.

During training, the model is optimized using cross-entropy (CE) loss, which is calculated as follows:

$$L_{CE}(\{\theta^{(0)} \cup \theta_+\}) = -\sum_{t=1}^{T+1} \log p(y_t|y_{<t}, X; \{\theta^{(0)} \cup \theta_+\}). \quad (12)$$

CFQ	
x	"Did M0 direct M1"
y	SELECT count (*) WHERE { M0 film.director.film M1 }
CoGnition	
x	"the dog he liked practiced all weekend long ."
y	他 喜欢 的 狗 整个 周末 都 在 练习。

Figure 3: Examples of CFQ and CoGnition.

Experiments

We mainly evaluate LRF on two comprehensive and realistic benchmarks for measuring CG, including semantic parsing (*CFQ*) and machine translation (*CoGnition*).

Experimental Settings

Datasets. *CoGnition* is an English → Chinese (En→Zh) translation dataset, which is used to systematically evaluate CG in MT scenarios. It consists of a training set of 196,246 sentence pairs, a validation set and a test set of 10,000 samples. In particular, it also has a dedicated synthetic test set (i.e., CG-test set) consisting of 10,800 sentences containing novel compounds, so that the ratio of compounds that are correctly translated can be computed to evaluate the model's ability of CG directly. *CFQ* is automatically generated from a set of rules in a way that precisely tracks which rules (atoms) and rule combinations (compounds) of each example. In this way, we can generate three splits with *maximum compound divergence* (MCD) while guaranteeing a small atom divergence between train and test sets, where large compound divergence denotes the test set involves more examples with unseen syntactic structures. We evaluate our method on all three splits. Each split dataset consists of a training set of 95,743, a validation set and a test set of 11,968 examples. Figure 3 shows examples of them.

Data Preprocessing. We follow the same settings of (Li et al. 2021) and (Keysers et al. 2020) to preprocess CoGnition and CFQ datasets separately. For CoGnition, we use an open-source Chinese tokenizer[1] to preprocess Chinese and apply Moses tokenizer[2] to preprocess English, which is the same in (Lin, Li, and Shi 2023). For CFQ, we use the GPT2-BPE tokenizer[3] to preprocess source and target English text.

Setup. For CoGnition, we follow the same experimental settings and configurations of (Li et al. 2021). We build our model on top of Transformer (Vaswani et al. 2017). We use one GeForce GTX 2080Ti for training with 100,000 steps and decoding. For CFQ, we follow the same experimental settings and configurations of (Zheng and Lapata 2022). We build our model on top of RoBERTa (Liu et al. 2019). We use the base RoBERTa with 12 encoder layers, which is combined with a Transformer decoder that has 2 decoder layers with hidden size 256 and feed-forward dimension 512. We use one GeForce GTX 2080Ti for training with 45,000 steps

---

[1]https://github.com/fxsjy/jieba

[2]https://github.com/moses-smt/
mosesdecoder/blob/master/scripts/tokenizer/tokenizer.perl

[3]https://github.com/facebookresearch/fairseq/blob/main/
examples/roberta/multiprocessing_bpe_encoder.py

Model	Params	Compound Translation Error Rate (CTER) ↓					BLEU ↑
		NP	VP	PP	Total	Δ	
Transformer	35M	24.7%/55.2%	24.8%/59.5%	35.7%/73.9%	28.4%/62.9%	-/-	59.5
Transformer-Rela	35M	30.1%/58.1%	27.6%/61.2%	38.5%/74.1%	32.1%/64.5%	+3.7%/+1.6%	59.1
Transformer-Small	25M	25.1%/56.9%	25.6%/60.3%	39.1%/75.0%	29.9%/64.5%	+1.5%/+1.6%	59.0
Transformer-Deep	46M	21.4%/51.4%	23.2%/57.6%	32.5%/71.5%	25.7%/60.2%	-2.7%/-2.7%	60.2
Bow	35M	22.2%47.9%	24.8%/55.6%	35.0%/73.2%	27.3%/58.9%	-1.1%/-3.0%	-
SeqMix	35M	24.5%/49.7%	26.9%/58.9%	34.4%/73.1%	28.6%/60.6%	+0.2%/-2.3%	-
Dangle	35M	-/-	-/-	-/-	24.4%/55.5%	-5.0%/-7.4%	59.7
Proto-Transformer	42M	14.1%/36.5%	22.1%/50.9%	28.9%/68.2%	21.7%/51.8%	-6.7%/-11.1%	60.1
Transformer+CReg	25M	-/-	-/-	-/-	20.2%/**48.3%**	-8.2%/-**14.6%**	61.3
DLCL	35M	-/-	-/-	-/-	28.4%/67.9%	+0.0%/+5.0%	59.2
LRF	48M	**12.0%/34.5%**	20.9%/50.7%	27.2%/65.8%	**20.0%**/50.3%	**-8.4%**/-12.6%	61.8
LRF-Rela	48M	12.9%/36.2%	**20.5%/50.6%**	**26.9%/65.0%**	20.1%/50.6%	-8.3%/-12.3%	**62.0**
LRF-Small	33M	14.1%/38.8%	21.4%/51.7%	27.9%/65.6%	21.1%/52.1%	-7.3%/-10.8%	60.9

Table 1: CTERs (%) on CoGnition. We report instance-level and aggregate-level CTERs in the CG-test set, separated by "/". In addition, we also report the commonly used metric BLEU score in MT tasks. "-" denotes that the results are not provided in the original paper. Results are averaged over 6 random runs.

and decoding. We implement all comparison models and LRF with an open source Fairseq toolkit (Ott et al. 2019).

Evaluation Metrics. For CoGnition, we use compound translation error rate (CTER (Li et al. 2021)) to measure the model's ability of CG. Specifically, *instance-level* CTER (i.e., $\text{CTER}_{\text{Inst}}$) denotes the ratio of samples where the novel compounds are translated incorrectly, and *aggregate-level* CTER (i.e., $\text{CTER}_{\text{Aggr}}$) denotes the ratio of samples where the novel compounds suffer at least one incorrect translation when aggregating all 5 contexts. To calculate CTER, (Li et al. 2021) manually construct a dictionary for all the atoms based on the training set, since each atom contains different translations. We also report character-level BLEU scores (Papineni et al. 2002) using SacreBLEU (Post 2018) as a supplement. For CFQ, we use exact match accuracy to evaluate model performance, where natural language utterances are mapped to meaning representations.

Model Settings

Machine Translation. We compare our method with competitive systems: (1) Transformer (Vaswani et al. 2017); (2) Transformer-Rela: only replaces absolute positional embedding with a relative one; (3) Transformer-Small: only decreases the number of encoder layers and decoder layers to 4, 4 respectively; (4) Transformer-Deep: only increases the number of encoder layers and decoder layers to 8, 8 respectively; (5) Bow (Raunak and et al 2019): uses bag-of-words pre-training; (6) SeqMix (Guo, Kim, and Rush 2020): synthesizes examples to encourage compositional behavior; (7) Dangle (Zheng and Lapata 2022): adaptively re-encodes (at each time step) the source input.[4] (8) Proto-Transformer (Yin et al. 2022): integrates prototypes of token representations over the training set into the source encoding; (9) Transformer+CReg (Yin et al. 2023): promotes rep-

resentation consistency across samples and prediction consistency for a single sample; (10) DLCL (Wang et al. 2019): proposes an approach based on dynamic linear combination of layers (DLCL), and is one of the very popular Enocder-Fusion work. Our method is built on top of (1)-(3), i.e., LRF, LRF-Rela and LRF-Small.

Semantic Parsing. We compare our method with competitive systems: (1) LSTM+attention; (2) Transformer (Vaswani et al. 2017); (3) Universal Transformer (Dehghani et al. 2019); (4) Evolved Transformer (So, Le, and Liang 2019); (5) CGPS (Li et al. 2019): leverages prior knowledge of compositionality with two representations, and adds entropy regularization to the encodings; (6) NSEN (Freivalds, Ozolins, and Sostaks 2019): is derived from the Shuffle-Exchange network; (7) T5-11B (Raffel et al. 2020): T5-11B is a T5 model with 11B parameters finetuned on CFQ; (8) T5-11B-mod (Furrer et al. 2020): uses masked language model (MLM) pre-training together with an intermediate representation; (9) RoBERTa (Liu et al. 2019): makes use of the RoBERTa-base model as the encoder and the randomly initialized Transformer decoder trained from scratch, where we use the same experimental settings of (Zheng and Lapata 2022); (10) HPD (Guo et al. 2020): proposes a novel hierarchical partially ordered set (poset) decoding paradigm; (11) Dangle (Zheng and Lapata 2022); (12) RoBERTa+CReg (Yin et al. 2023); (13) LRF: builds on (9) with our method.

Results on CoGnition

The main results on CoGnition are shown in Table 1. We observe that: **(1)** LRF gives instance-level and aggregate-level CTERs of **20.0%** and **50.3%** respectively, with a significant improvement of **8.4%** and **12.6%** accordingly compared to the Transformer. Moreover, LRF outperforms most baseline models under the almost same parameter settings significantly, indicating fusing the syntactic and semantic information of sequences stored in different lay-

---

[4]We use the same variant reported by (Zheng and Lapata 2022) (i.e., Dangle-EncDec (abs)) with absolute positional embedding.

Model	MCD1	MCD2	MCD3	MCD-Mean
LSTM+attention	28.9	5.0	10.8	14.9
Transformer	34.9	8.2	10.6	17.9
Universal Transformer	37.4	8.1	11.3	18.9
Evolved Transformer	42.4	9.3	10.8	20.8
CGPS	13.2	1.6	6.6	7.1
NSEN	5.1	0.9	2.3	2.8
T5-11B	61.4	30.1	31.2	40.9
T5-11B-mod	61.6	31.3	33.3	42.1
RoBERTa	60.6	33.6	36.0	43.4
HPD	72.0	**66.1**	**63.9**	**67.3**
Dangle	**78.3**	59.5	60.4	66.1
RoBERTa+CReg	74.8	53.3	58.3	62.1
LRF	68.9	43.4	44.7	52.4

Table 2: Exact-match accuracy on different MCD splits of CFQ. Results are averaged over 3 random runs.

ers effectively is more beneficial to CG. Although Transformer+CReg achieves slightly better performance and contains fewer parameters, it is more complex and costly compared with LRF; **(2)** LRF, LRF-Rela and LRF-Small can deliver various performance improvements, demonstrating the general effectiveness of our method; **(3)** Since LRF brings some extra parameters (approximately 12M), we investigate to what extent the performance improvement is derived from the increase of model parameters. Transformer-Deep performs slightly better than Transformer on the CG-test set, indicating that only increasing model parameters is slightly useful but far from sufficient. In addition, LRF-Small contains fewer parameters and better performance than Transformer. Compared to SeqMix, the improvement of LRF is more significant (2.3% vs 12.6% aggregate-level CTER). SeqMix utilizes linear interpolation in the input embedding space to reduce representation sparsity, and we suppose that the samples synthesized randomly may be unreasonable and harmful to model training; **(4)** It can be seen that Transformer is even slightly better than DLCL, indicating the inappropriate composition instead brings noise to significantly affect the model's CG performance. Instead, we use the attention mechanism to fuse the most relevant parts of previous layers' information at each layer in a flexible manner.

Results on CFQ

The main results on CFQ are presented in Table 2. We observe that: **(1)** RoBERTa is comparable to T5-11B, T5-11B-mod and outperforms other baseline systems without pre-training except HPD, indicating that pre-trained language models are key to achieving good performance on CFQ, which is consistent with the conclusions in (Furrer et al. 2020); **(2)** LRF substantially boosts the performance of RoBERTa (**43.4 → 52.4**), about **21%** relative improvements, and is in fact superior to T5-11B and T5-11B-mod. It also outperforms other baseline systems without pre-training except HPD. This result demonstrates that pre-training as a solution to CG also has limitations, and also indicates that LRF is complementary to pre-trained models; **(3)** HPD performs better than Dangle, RoBERTa+CReg and LRF, achieving

Model	CTER$_{\text{Inst}}$ ↓	CTER$_{\text{Aggr}}$ ↓
Transformer	28.4%	62.9%
+ Source FA	21.2% (-7.2%)	53.3% (-9.6%)
+ Target FA	25.8% (-2.6%)	58.8% (-4.1%)
+ Source & Target FA	**20.0% (-8.4%)**	**50.3% (-12.6%)**

Table 3: CTERs (%) against the fuse-attention (FA) modules at different sides on the CG-test set.

67.3 exact match accuracy, which is highly optimized for the CFQ dataset. On the contrary, LRF, RoBERTa+CReg and Dangle are generally applicable to any seq2seq models for solving any seq2seq tasks including MT, as stated before. However, compared with the impressive performance on CoGnition, the improvements brought by LRF is relatively moderate, and even worse than Dangle. The underlying reason is connected a recent finding that compositionality in natural language is much more complex than the rigid, arithmetic-like operations (Raunak and et al 2019; Li et al. 2021; Zheng and Lapata 2022; Dankers and et al. 2022). MT is paradigmatically close to the tasks typically considered for testing compositionality in natural language, while our approach is more suitable for dealing with such scenarios.

Analysis

In this section, we conduct in-depth analyses of LRF to provide a comprehensive understanding of the individual contributions of each component. For all experiments, we train a LRF (6-6 encoder and decoder layers) on the CoGnition dataset, unless otherwise specified.

Do We Need to Introduce the Fuse-Attention Module in Both Side?

We argue that "shallow" residual connections with each layer of encoder or decoder lead to encoder or decoder RE problem respectively. Therefore, we are curious about whether LRF can alleviate both the encoder and decoder RE problems, and which one is more severe or affects the model's ability of CG to a greater extent that is ignored in previous work. In this experiment, we investigate its influence on CoGnition. As shown in Table 3, we observe certain improvements (**-7.2%** and **-2.6%** CTER$_{\text{Inst}}$, **-9.6%** and **-4.1%** CTER$_{\text{Aggr}}$) when separately applying the fuse-attention modules at the encoder or decoder side. It suggests that our proposed method can alleviate the encoder and decoder RE problem respectively, and fusing information from previous layers back into the encoding or decoding process at each layer effectively can improve CG performance. Furthermore, their combination brings further improvement (**-8.4%** CTER$_{\text{Inst}}$, **-12.6%** CTER$_{\text{Aggr}}$), which illustrates that LRF can alleviate both the encoder and decoder RE problems and have cumulative gains. In addition, it is clear that the improvement from introducing the fuse-attention module at the encoder is more significant than that at the decoder, indicating that the encoder RE problem is more severe. The underlying reason is related to a recent finding that the encoder has a greater impact on performance than
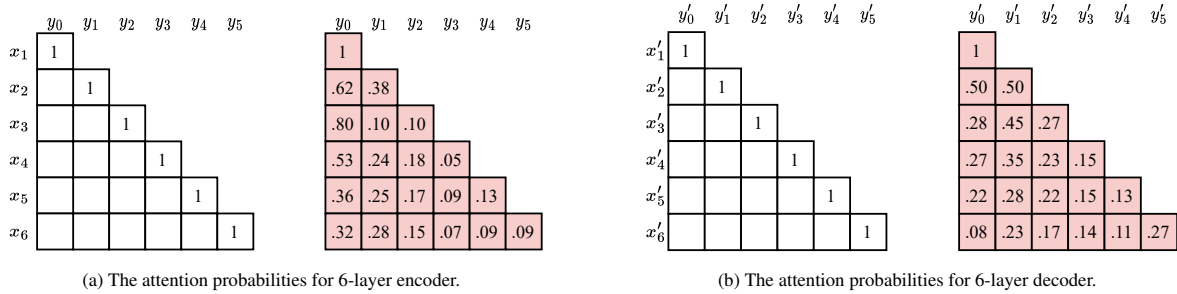
(a) The attention probabilities for 6-layer encoder.

(b) The attention probabilities for 6-layer decoder.

Figure 4: The differences between LRF and the Transformer in the process of fusing previous layers' information. $\{y_0, ..., y_l\}$ and $\{y'_0, ..., y'_l\}$ are the output of the encoder and decoder layers $0 \sim l$ respectively. $\{x_1, ..., x_l\}$ and $\{x'_1, ..., x'_l\}$ are the input of the encoder and decoder layers $1 \sim l$ respectively. y_0, y'_0 and $y_{<i}, y'_{<i}$ denote the input embedding and previous layers' outputs in the i-th encoder and decoder layer respectively. Red denotes the attention probabilities are learned by LRF.

Model	CTER$_{\text{Inst}}$ ↓	CTER$_{\text{Aggr}}$ ↓
Transformer	28.4%	62.9%
Transformer-accu	37.3% (+8.9%)	71.4% (+8.5%)
LRF-onlytop	22.1% (-6.3%)	52.7% (-10.2%)
LRF	**20.0% (-8.4%)**	**50.3% (-12.6%)**

Table 4: CTERs (%) on the CG-test set.

the decoder (Kasai et al. 2021; Xu et al. 2021a).

Do We Need to Fuse Previous Layers' Information at Each Layer Effectively?

We argue that it is important to introduce the fuse-attention module in **each layer**, since we view the emergence of RE problems as the process of gradually entangled representation from bottom to top of the Transformer layers. To validate this argument, we only introduce the fuse-attention module in the topmost layer of encoder and decoder (called LRF-onlytop). As shown in Table 4, LRF substantially boosts the performance of LRF-onlytop (-2.1% CTER$_{\text{Inst}}$, -2.4% CTER$_{\text{Aggr}}$). In addition, LRF-Small contains fewer parameters and better performance than LRF-onlytop. As mentioned above, we use the fuse-attention module to achieve the goal of fusing previous layers' information at each layer **effectively**. To validate this argument, we also conduct a toy experiment on CoGnition. Specifically, each layer of the encoder and decoder accumulates corresponding previous layers' information (called Transformer-accu),[5] rather than learn to fuse it adaptively like we do. Results are listed in Table 4. Transformer-accu even obtains worse generalization results than Transformer. It suggests that the "simple" combinations will instead bring noise to affect the model's CG performance.

To further demonstrate the fuse-attention module we introduced can fuse previous layers' information **effectively** and understand the individual contributions of previous layers' information for each layer, we visualize the attention

---

[5]The input of encoder and decoder layer i is $x_i = y_0 + \cdots + y_{i-1}, x'_i = y'_0 + \cdots + y'_{i-1}$, where $0 < i < L$ (see Figure 4).

probabilities of the fuse-attention modules in each encoder and decoder layer. Specifically, we train LRF on CoGnition and test on 680 (a batch) randomly selected examples of CG-test set, and then extract the attention probabilities of the fuse-attention modules. Ideally, the fuse-attention modules in different layer of the encoder and decoder should learn different and specific combinations of previous layers' information. In Figure 4, we observe that each layer of the encoder or decoder assigns different attention weights to information from previous layers as expected. This implies that the fuse-attention module in LRF can learn to fuse representations from previous layers at each layer effectively.

Effects on Compositional Generalization

Compound Length and Context Length. Longer compounds have more complex semantic information and longer contexts are harder to comprehend, making them more difficult to generalize (Li et al. 2021). We classify the test samples by compound length and context length, and calculate the CTER$_{\text{Inst}}$. In Figure 5, we can observe that LRF generalizes better as the compound and context grows longer compared to Transformer. In particular, LRF gives a lower CTER by **11.5%** over samples when the context leangth is longer than 13 tokens. It suggests that LRF can better captures the compositional structure of human language.

Complex Modifier. The postpositive modifier atom (MOD) is used to enrich the information of its preceding word (e.g., *he liked* in the phrase *lost the dog he liked*), which is challenging to translate due to word reordering from English to Chinese. We divide the test samples into two groups according to compounds with (w/) or without (wo/) MOD. In Figure 6, we observe that the advantage of LRF grows larger in translating the compounds with MOD, demonstrating its superiority in processing complex semantic composition.

Case Study. We present 2 source examples containing a novel compound with MOD and 4 atoms, and their translations in Table 5. For both samples, correct translations denote that the novel compounds are translated correctly. LRF correctly translates the novel compounds across different contexts for all samples, while Transformer suffers from omitting different atoms. For example, the translation

Source	Transformer	LRF
The waiter he liked wore each other's clothes.	他喜欢穿对方的衣服。 (**He liked** to wear each other's clothes.)	他喜欢的服务员穿着彼此的衣服。 (**The waiter he liked** wore each other's clothes.)
The waiter he liked came by and chased the bully off.	服务员来了，把那个恶霸赶走了。 (**The waiter** came by and chased the bully off.)	他喜欢的服务员过来把那个恶霸赶走了。 (**The waiter he liked** came by and chased the bully off.)

Table 5: Example translations of Transformer vs. LRF. The bold characters denote the novel compounds and corresponding translations. The English in parentheses represents the meaning of the model-generated Chinese.
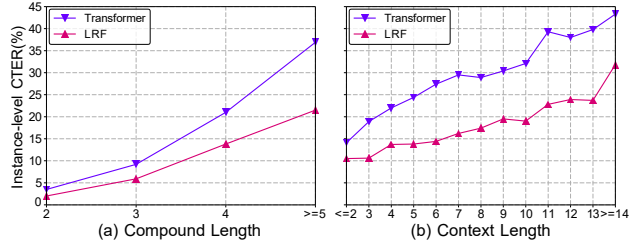
Figure 5: Instance-level CTER of LRF and Transformer over the different compound and context lengths.

Figure 6: $CTER_{Inst}$ on compounds w/o and w/ MOD.

of *the waiter* is omitted in the first example, *he liked* is omitted in the second example. Our results not only contain the correct compound translations but also achieve better translation quality, while Transformer makes various errors on unseen compositions.

Related Work

Compositional Generalization. CG has long played a prominent role in language understanding, explaining why we understand novel compositions of previously observed elements. Therefore, after realizing existing neural models still struggle in scenarios requiring CG (Lake and Baroni 2018; Keysers et al. 2020; Li et al. 2021), there have been various studies attempt to improve the model's ability of CG, including data augmentation (Andreas 2020; Akyürek, Akyürek, and Andreas 2021; Yang, Zhang, and Yang 2022; Li, Wei, and Lian 2023), modifications on model architecture (Li et al. 2019; Russin et al. 2019; Nye et al. 2020; Liu et al. 2020a, 2021; Zheng and Lapata 2021; Herzig and Berant 2021; Chaabouni, Dessì, and Kharitonov 2021; Wang, Lapata, and Titov 2021; Mittal et al. 2022; Lin et al. 2023), intermediate representations (Furrer et al. 2020; Herzig et al. 2021), meta-learning (Lake 2019; Conklin et al. 2021), explorations on pre-trained language models (Furrer et al. 2020; Zhou et al. 2023), auxiliary objectives (Jiang and Bansal 2021), and enriching semantic information at token-level (Thrush 2020; Akyurek and Andreas 2021; Zheng and Lapata 2022; Yao and Koller 2022). One line of research exploring how to alleviate the RE problems has attracted much attention. Our work is in line with it, we examine CG from a new perspective to solve it.

Neural Machine Translation. Recently, CG and robustness of Neural Machine Translation (NMT) have gained much attention from the r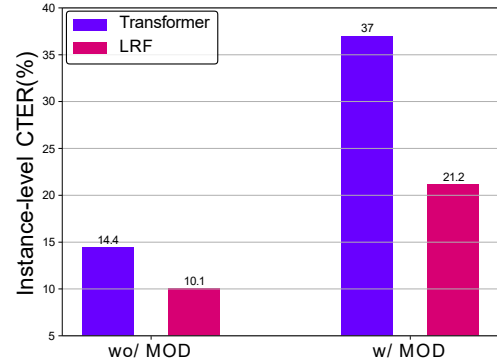esearch community (Cheng et al. 2020; Xu et al. 2021b; Lake and Baroni 2018; Li et al. 2021). (Raunak and et al 2019) propose bag-of-words pre-training for the encoder. (Guo, Kim, and Rush 2020) propose sequence-level mixup to create synthetic samples. Recently, (Li et al. 2021) introduce a practical benchmark dataset for analyzing fine-grained and systematic compositional ability of NMT, called CoGnition. (Dankers and et al 2022) argue that MT is a suitable and relevant testing ground to test CG in natural language. Based on this, (Zheng and Lapata 2022) propose to adaptively re-encode the source input at each time step. Different from them, our method introduce a fuse-attention module in each encoder and decoder layer to fuse previous layers' information back into the encoding and decoding process effectively, which is inspired by previous explorations about analyzing how Transformer encoder and decoder perform (Peters et al. 2018; He, Tan, and Qin 2019; Voita, Sennrich, and Titov 2019; Belinkov et al. 2020).

Conclusion

In this paper, we propose LRF, which learns to fuse previous layers' information back into the encoding and decoding process effectively through introducing a fuse-attention module at each encoder and decoder layer. Experiments on CoGnition and CFQ have shown the effectiveness of our proposal without any dataset or task-specific modification. To our knowledge, we are the first to explain why the RE problems exist and investigate how to fuse previous layers' information at each layer effectively to alleviate it, achieving better generalization results. We hope the work and perspective presented in this paper can inspire future related work.

Acknowledgments

References

Akyürek, E.; Akyürek, A. F.; and Andreas, J. 2021. Learning to Recombine and Resample Data For Compositional Generalization. In *Proc. of ICLR*.

Akyurek, E.; and Andreas, J. 2021. Lexicon Learning for Few Shot Sequence Modeling. In *Proc. of ACL*.

Andreas, J. 2020. Good-Enough Compositional Data Augmentation. In *Proc. of ACL*.

Bapna, A.; Chen, M.; Firat, O.; Cao, Y.; and Wu, Y. 2018. Training Deeper Neural Machine Translation Models with Transparent Attention. In *Proc. of EMNLP*.

Belinkov, Y.; Durrani, N.; Dalvi, F.; Sajjad, H.; and Glass, J. 2020. On the linguistic representational power of neural machine translation models. *Computational Linguistics*.

Cazzaro, F.; Locatelli, D.; Quattoni, A.; and Carreras, X. 2023. Translate First Reorder Later: Leveraging Monotonicity in Semantic Parsing. In *Proc. of EACL Findings*, 227–238.

Chaabouni, R.; Dessì, R.; and Kharitonov, E. 2021. Can Transformers Jump Around Right in Natural Language? Assessing Performance Transfer from SCAN. In *Proc. of BlackboxNLP*.

Cheng, Y.; Jiang, L.; Macherey, W.; and Eisenstein, J. 2020. AdvAug: Robust Adversarial Augmentation for Neural Machine Translation. In *Proc. of ACL*.

Conklin, H.; Wang, B.; Smith, K.; and Titov, I. 2021. Meta-Learning to Compositionally Generalize. In *Proc. of ACL*.

Dankers, V.; and et al. 2022. The Paradox of the Compositionality of Natural Language: A Neural Machine Translation Case Study. In *Proc. of ACL*.

Dehghani, M.; Gouws, S.; Vinyals, O.; Uszkoreit, J.; and Kaiser, L. 2019. Universal Transformers. In *Proc. of ICLR*.

Dong, L.; and Lapata, M. 2016. Language to Logical Form with Neural Attention. In *Proc. of ACL*.

Dou, Z.-Y.; Tu, Z.; Wang, X.; Shi, S.; and Zhang, T. 2018. Exploiting Deep Representations for Neural Machine Translation. In *Proc. of EMNLP*.

Fodor, J. A.; and Pylyshyn, Z. W. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition*.

Freivalds, K.; Ozolins, E.; and Sostaks, A. 2019. Neural Shuffle-Exchange Networks - Sequence Processing in O(n log n) Time. In *Proc. of NeurIPS*.

Furrer, D.; van Zee, M.; Scales, N.; and Schärli, N. 2020. Compositional generalization in semantic parsing: Pretraining vs. specialized architectures. *arXiv*.

Guo, D.; Kim, Y.; and Rush, A. 2020. Sequence-Level Mixed Sample Data Augmentation. In *Proc. of EMNLP*.

Guo, Y.; Lin, Z.; Lou, J.-G.; and Zhang, D. 2020. Hierarchical Poset Decoding for Compositional Generalization in Language. In *Proc. of NeurIPS*.

He, T.; Tan, X.; and Qin, T. 2019. Hard but robust, easy but sensitive: How encoder and decoder perform in neural machine translation. *ArXiv*.

Herzig, J.; and Berant, J. 2021. Span-based Semantic Parsing for Compositional Generalization. In *Proc. of ACL*.

Herzig, J.; Shaw, P.; Chang, M.; Guu, K.; Pasupat, P.; and Zhang, Y. 2021. Unlocking Compositional Generalization in Pre-trained Models Using Intermediate Representations. *arXiv*.

Jiang, Y.; and Bansal, M. 2021. Inducing Transformer's Compositional Generalization Ability via Auxiliary Sequence Prediction Tasks. In *Proc. of EMNLP*.

Kasai, J.; Pappas, N.; Peng, H.; Cross, J.; and Smith, N. A. 2021. Deep Encoder, Shallow Decoder: Reevaluating Non-autoregressive Machine Translation. In *Proc. of ICLR*.

Keysers, D.; Schärli, N.; Scales, N.; Buisman, H.; Furrer, D.; Kashubin, S.; Momchev, N.; Sinopalnikov, D.; Stafiniak, L.; Tihon, T.; Tsarkov, D.; Wang, X.; van Zee, M.; and Bousquet, O. 2020. Measuring Compositional Generalization: A Comprehensive Method on Realistic Data. In *Proc. of ICLR*.

Lake, B.; and Baroni, M. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proc. of ICML*.

Lake, B. M. 2019. Compositional generalization through meta sequence-to-sequence learning. In *Proc. of NeurIPS*.

Lake, B. M.; Ullman, T. D.; Tenenbaum, J. B.; and Gershman, S. J. 2017. Building machines that learn and think like people. *Behavioral and brain sciences*.

Li, Q.; Zhu, Y.; Liang, Y.; Wu, Y. N.; Zhu, S.; and Huang, S. 2022. Neural-Symbolic Recursive Machine for Systematic Generalization. *CoRR*, abs/2210.01603.

Li, Y.; Yin, Y.; Chen, Y.; and Zhang, Y. 2021. On Compositional Generalization of Neural Machine Translation. In *Proc. of ACL*.

Li, Y.; Zhao, L.; Wang, J.; and Hestness, J. 2019. Compositional Generalization for Primitive Substitutions. In *Proc. of EMNLP*.

Li, Z.; Wei, Y.; and Lian, D. 2023. Learning to Substitute Spans towards Improving Compositional Generalization. *arXiv preprint arXiv:2306.02840*.

Lin, L.; Li, S.; and Shi, X. 2023. LEAPT: Learning Adaptive Prefix-to-Prefix Translation For Simultaneous Machine Translation. In *Proc. of ICASSP*.

Lin, L.; Li, S.; Zheng, Y.; Fu, B.; Liu, S.; Chen, Y.; and Shi, X. 2023. Learning to Compose Representations of Different Encoder Layers towards Improving Compositional Generalization. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 1599–1614.

Liu, C.; An, S.; Lin, Z.; Liu, Q.; Chen, B.; Lou, J.; Wen, L.; Zheng, N.; and Zhang, D. 2021. Learning Algebraic Recombination for Compositional Generalization. In *Proc. of ACL Findings*.

Liu, Q.; An, S.; Lou, J.; Chen, B.; Lin, Z.; Gao, Y.; Zhou, B.; Zheng, N.; and Zhang, D. 2020a. Compositional Generalization by Learning Analytical Expressions. In *Proc. of NeurIPS*.

Liu, Q.; An, S.; Lou, J.; Chen, B.; Lin, Z.; Gao, Y.; Zhou, B.; Zheng, N.; and Zhang, D. 2020b. Compositional Generalization by Learning Analytical Expressions. In *Proc. of NeurIPS*.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv*.

Mittal, S.; Raparthy, S. C.; Rish, I.; Bengio, Y.; and Lajoie, G. 2022. Compositional Attention: Disentangling Search and Retrieval. In *Proc. of ICLR*.

Nye, M. I.; Solar-Lezama, A.; Tenenbaum, J.; and Lake, B. M. 2020. Learning Compositional Rules via Neural Program Synthesis. In *Proc. of NeurIPS*.

Ott, M.; Edunov, S.; Baevski, A.; Fan, A.; Gross, S.; Ng, N.; Grangier, D.; and Auli, M. 2019. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In *Proc. of NAACL*.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proc. of ACL*.

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep Contextualized Word Representations. In *Proc. of NAACL*.

Post, M. 2018. A Call for Clarity in Reporting BLEU Scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers, WMT 2018, Belgium, Brussels, October 31 - November 1, 2018*.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.*

Raunak, V.; and et al. 2019. On compositionality in neural machine translation. *ArXiv*.

Ruis, L.; and Lake, B. M. 2022. Improving Systematic Generalization Through Modularity and Augmentation. *CoRR*, abs/2202.10745.

Russin, J.; Jo, J.; O'Reilly, R. C.; and Bengio, Y. 2019. Compositional generalization in a deep seq2seq model by separating syntax and semantics. *arXiv*.

So, D. R.; Le, Q. V.; and Liang, C. 2019. The Evolved Transformer. In *Proc. of ICML*.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. *Proc. of NeurIPS*.

Thrush, T. 2020. Compositional Neural Machine Translation by Removing the Lexicon from Syntax. *arXiv*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Proc. of NeurIPS*.

Voita, E.; Sennrich, R.; and Titov, I. 2019. The Bottom-up Evolution of Representations in the Transformer: A Study with Machine Translation and Language Modeling Objectives. In *Proc. of EMNLP*.

Wang, B.; Lapata, M.; and Titov, I. 2021. Structured Reordering for Modeling Latent Alignments in Sequence Transduction. In *Proc. of NeurIPS*.

Wang, Q.; Li, B.; Xiao, T.; Zhu, J.; Li, C.; Wong, D. F.; and Chao, L. S. 2019. Learning Deep Transformer Models for Machine Translation. In *Proc. of ACL*.

Wang, Q.; Li, F.; Xiao, T.; Li, Y.; Li, Y.; and Zhu, J. 2018. Multi-layer Representation Fusion for Neural Machine Translation. In *Proc. of COLING*.

Xu, H.; van Genabith, J.; Liu, Q.; and Xiong, D. 2021a. Probing Word Translations in the Transformer and Trading Decoder for Encoder Layers. In *Proc. of NAACL*.

Xu, W.; Aw, A. T.; Ding, Y.; Wu, K.; and Joty, S. 2021b. Addressing the Vulnerability of NMT in Input Perturbations. In *Proc. of NAACL*.

Yang, J.; Zhang, L.; and Yang, D. 2022. SUBS: Subtree Substitution for Compositional Semantic Parsing. In *Proc. of NAACL*.

Yao, Y.; and Koller, A. 2022. Structural generalization is hard for sequence-to-sequence models. In *Proc. of EMNLP*.

Yin, Y.; Li, Y.; Meng, F.; Zhou, J.; and Zhang, Y. 2022. Categorizing Semantic Representations for Neural Machine Translation. In *Proc. of COLING*.

Yin, Y.; Zeng, J.; Li, Y.; Meng, F.; Zhou, J.; and Zhang, Y. 2023. Consistency Regularization Training for Compositional Generalization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1294–1308.

Yu, F.; Wang, D.; Shelhamer, E.; and Darrell, T. 2018. Deep layer aggregation. In *Proc. of CVPR*.

Zheng, H.; and Lapata, M. 2021. Compositional Generalization via Semantic Tagging. In *Proc. of EMNLP Findings*.

Zheng, H.; and Lapata, M. 2022. Disentangled Sequence to Sequence Learning for Compositional Generalization. In *Proc. of ACL*.

Zhou, D.; Schärli, N.; Hou, L.; Wei, J.; Scales, N.; Wang, X.; Schuurmans, D.; Cui, C.; Bousquet, O.; Le, Q. V.; and Chi, E. H. 2023. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models. In *Proc. of ICLR*.