Composition, Attention, or Both?*

Ryo Yoshida and **Yohei Oseki**
The University of Tokyo
{yoshiryo0617, oseki}@g.ecc.u-tokyo.ac.jp

Abstract

In this paper, we propose a novel architecture called **Composition Attention Grammars** (CAGs) that recursively compose subtrees into a single vector representation with a composition function, and selectively attend to previous structural information with a self-attention mechanism. We investigate whether these components—the composition function and the self-attention mechanism—can both induce human-like syntactic generalization. Specifically, we train language models (LMs) with and without these two components with the model sizes carefully controlled, and evaluate their syntactic generalization performance against six test circuits on the SyntaxGym benchmark. The results demonstrated that the composition function and the self-attention mechanism both play an important role to make LMs more human-like, and closer inspection of grammatical phenomena implied that the composition function allowed syntactic features, but not semantic features, to percolate into subtree representations.

1 Introduction

Recently, language models (LMs) trained on large datasets have achieved remarkable success in various Natural Language Processing (NLP) tasks (cf. Wang et al., 2019a,b). The literature of targeted syntactic evaluations has shown that these models implicitly learn syntactic structures of natural language, even though they do not receive explicit syntactic supervision (Warstadt et al., 2020; Hu et al., 2020).

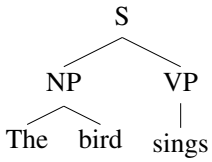
However, previous work has also shown that there is still a benefit for LMs to receive explicit syntactic supervision. Recurrent Neural Network Grammars (RNNGs; Dyer et al., 2016), the integration of Recurrent Neural Networks (RNNs; Elman, 1990) with an explicit syntactic bias, have achieved better syntactic generalization performance than vanilla RNNs (Kuncoro et al., 2018; Wilcox et al., 2019; Hu et al., 2020). In addition, previous work has recommended RNNGs as a cognitively plausible architecture, showing that RNNGs can successfully predict human reading times (Yoshida et al., 2021) or brain activities (Hale et al., 2018). The key difference between RNNGs and RNNs is a **composition function**, which recursively composes subtrees into a single vector representation.

On the other hand, Transformer architectures (Vaswani et al., 2017) have been shown to outperform RNN architectures in various NLP tasks (Devlin et al., 2019). The key difference between Transformers and RNNs here is a **self-attention mechanism**, which selectively attends to previous vectors to obtain sentence representations. Recently, an attempt was made to investigate whether Transformer architectures with the self-attention mechanism also benefit from explicit syntactic supervision (Qian et al., 2021), but their "Parsing as Language Modeling (PLM)" approach (Choe and Charniak, 2016) does not employ the composition function, which is essential for RNNGs. Therefore, it is reasonable to hypothesize that their approach may not achieve the full benefit of explicit syntactic supervision.

In this paper, we propose a novel architecture called **Composition Attention Grammars** (CAGs) that recursively compose subtrees into a single vector representation with the composition function, and selectively attend to previous structural information with the self-attention mechanism. We investigate whether these components—the composition function and the self-attention mechanism—can both induce human-like syntactic generalization. Specifically, we train LMs with

*While writing this paper, we noticed that Sartran et al. (2022) was submitted to the arXiv, proposing Transformer Grammars (TGs) that incorporate recursive syntactic composition via an attention mask. Their work and ours are similar in spirit, but different in how to obtain a vector representation of subtrees, making them complementary. We discuss the details in Section 5.

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 5851 - 5863



t	1	2	3	4	5	6	7	8	9
a_t	NT(S)	NT(NP)	GEN(The)	GEN(bird)	REDUCE	NT(VP)	GEN(sings)	REDUCE	REDUCE
Symbol	(S	(NP	The	bird	)	(VP	sings	)	)

Figure 1: An example of actions to jointly generate the sentence and its syntactic structure in a top-down, left-to-right fashion.

and without these two components, with the model sizes carefully controlled, and evaluate their syntactic generalization performance against six test circuits (Hu et al., 2020) on the SyntaxGym benchmark (Gauthier et al., 2020). The results demonstrated that the composition function and the self-attention mechanism both play an important role to make LMs more human-like, and closer inspection of grammatical phenomena implied that the composition function allowed syntactic features, but not semantic features, to percolate into subtree representations.

In addition, the methodological innovation of this paper is a strictly controlled experimental design, as practiced in cognitive sciences. In NLP research, evaluations are often conducted on models with different model sizes, leading to uncertainty regarding which component of these models affects the results. This paper conducts strictly controlled experiments in order to isolate the effects of individual components such as the composition function and the self-attention mechanism.

2 Composition Attention Grammar

In this section, we introduce a novel architecture called Composition Attention Grammars (CAGs).

2.1 Syntactic language model

CAGs are a type of syntactic LM (Choe and Charniak, 2016; Dyer et al., 2016; Qian et al., 2021), which estimates the following joint distribution of a sentence X and its syntactic structure Y:

$$p(X, Y) = p(a_1, \cdots, a_n) = \prod_{t=1}^{n} p(a_t | a_{<t}) \quad (1)$$

where a_t is an action by which CAGs jointly generate the sentence and its syntactic structure in a top-down, left-to-right fashion. Each a_t can be one of the three actions below:

- `GEN(x)`: Generate a terminal symbol "x".
- `NT(X)`: Open a nonterminal symbol "X".
- `REDUCE`: Close a nonterminal symbol that was opened by `NT(X)`.

See Figure 1 for an example of actions to jointly generate the sentence and its syntactic structure in a top-down, left-to-right fashion.

2.2 Architecture

To estimate the joint distribution in Equation 1, CAGs utilize (i) the composition function to recursively compose subtrees into a single vector representation, and (ii) the self-attention mechanism to selectively attend to previous structural information. The architecture of CAGs is summarized in Figure 2. Following previous work (Kuncoro et al., 2017; Noji and Oseki, 2021), CAGs rely on a stack data structure, and each action in Section 2.1 changes the stack state as follows:

- `GEN(x)`: Push a terminal embedding $\mathbf{e}_x$ onto the stack.
- `NT(X)`: Push a nonterminal embedding $\mathbf{e}_X$ onto the stack.
- `REDUCE`: First, repeatedly pop vectors from the stack until a nonterminal embedding is popped. Then, apply the composition function based on bidirectional LSTMs (Schuster and Paliwal, 1997) to these popped vectors $\mathbf{e}_l, \ldots, \mathbf{e}_m$, to compose subtrees into a single vector representation $\mathbf{e}_s$:

$$\mathbf{e}_s = \text{Composition}([\mathbf{e}_l, \ldots, \mathbf{e}_m]). \quad (2)$$

  $\mathbf{e}_s$ is then pushed onto the stack.

After each action, CAGs employ the self-attention mechanism, which selectively attends to previous vectors in the stack $\mathbf{e}_1, \ldots, \mathbf{e}_k$ by calculating the weight of attention to each vector with the query, key, and value vectors generated from $\mathbf{e}_1, \ldots, \mathbf{e}_k$, in order to represent a partial parse at each time step t:

$$\mathbf{h}_t = \text{SelfAttn}([\mathbf{e}_1, \ldots, \mathbf{e}_k]). \quad (3)$$

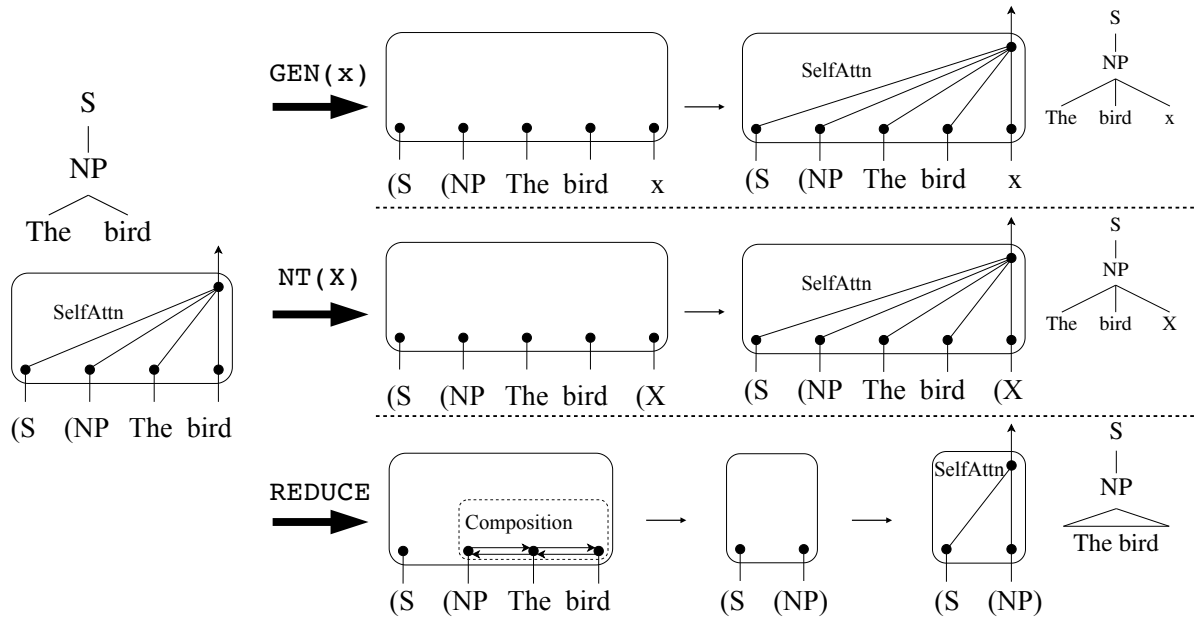


Figure 2: The architecture of Composition Attention Grammars (CAGs). CAGs utilize (i) the composition function to recursively compose subtrees into a single vector representation, and (ii) the self-attention mechanism to selectively attend to previous structural information.

Then, $\mathbf{h}_t$ defines the next action distribution:

$$a_{t+1} \sim \mathrm{softmax}(\mathbf{W}_a \mathbf{h}_t + \mathbf{b}_a) \quad (4)$$

where $\mathbf{W}_a$ and $\mathbf{b}_a$ are the weights and biases of a fully connected layer that projects $\mathbf{h}_t$ to logits for each action a, and $\mathrm{softmax}$ is a softmax function that projects the logits to the next action distribution.

2.3 Differences from other syntactic LMs

In this subsection, we focus on the differences between CAGs and other syntactic LMs.

Difference from RNNGs CAGs and RNNGs both utilize the composition function to recursively compose subtrees into a single vector representation. CAGs differ from RNNGs in that, in order to represent the partial parse at each time step, CAGs utilize the self-attention mechanism which selectively attends to previous structural information, whereas RNNGs utilize stack-LSTMs (Dyer et al., 2015). We hypothesize that CAGs have the advantage of selective attention to previous structural information over RNNGs.

Difference from PLMs CAGs and PLMs both utilize the self-attention mechanism which selectively attends to previous structural information. CAGs differ from PLMs in that CAGs utilize the composition function to recursively compose subtrees into a single vector representation, whereas PLMs treat actions $a_1, \ldots, a_n$ flatly as vanilla Transformers treat words $w_1, ..., w_n$. We hypothesize that CAGs have the advantage of recursive composition of subtrees over PLMs.

In order to incorporate composition-like characteristics, Qian et al. (2021) proposed PLM-masks, namely, PLMs with a dynamic masking mechanism, which specializes two attention heads: one to attend to the inside of the most recently opened nonterminal symbol, and another to attend to the outside. We will perform a comparison between CAGs and PLM-masks in order to investigate whether recursive composition of subtrees has additional advantages over the dynamic masking mechanism in inducing human-like syntactic generalization.

3 Experiment

We designed a strictly controlled experiment for testing whether the two components—the composition function and the self-attention mechanism—can both induce human-like syntactic generalization. Specifically, we train LMs with and without these two components with the model sizes carefully controlled, and evaluate their syntactic generalization performance against six test circuits on the SyntaxGym benchmark. We also train and evaluate two vanilla LMs with and without the self-

	− Syntax	+ Syntax	
		− Composition	+ Composition
− SelfAttn	LSTM (Hochreiter and Schmidhuber, 1997)	ActionLSTM (Choe and Charniak, 2016)	RNNG (Dyer et al., 2016)
+ SelfAttn	Transformer (Radford et al., 2018)	PLM (Qian et al., 2021)	PLM-mask ((+) Composition; Qian et al., 2021)
			CAG (This work)

Table 1: LMs investigated in this paper. ± Syntax means whether LMs receive explicit syntactic supervision. ± Composition means whether LMs utilize the composition function, and ± SelfAttn means whether LMs are based on Transformer architectures with the self-attention mechanism. PLM-masks do not utilize the composition function, but use the local subtree information with the dynamic masking mechanism ((+) Composition).

	#Layer	#Hidden dimension	#Input dimension	#Head	#Model size
LSTM	2	301	301	N/A	**16.59M**
ActionLSTM	2	301	301	N/A	**16.58M**
RNNG	2	276	276	N/A	**16.61M**
Transformer	3	272	272	4	**16.62M**
PLM	3	272	272	4	**16.63M**
PLM-mask	3	272	272	4	**16.63M**
CAG	3	256	256	4	**16.57M**

Table 2: Hyperparameters of LMs investigated in this paper. We controlled the hyperparameters in order to make model sizes maximally comparable.

attention mechanisms as a baseline. The following subsections describe the experimental settings in further detail.

3.1 Language models

This subsection describes LMs investigated in this paper (Table 1). We controlled the hyperparameters in order to make model sizes maximally comparable (Table 2).

LSTM LSTMs (Hochreiter and Schmidhuber, 1997) are a vanilla LM (− Syntax) based on RNN architectures (− SelfAttn). LSTMs were adopted as a baseline for syntactic LMs without the self-attention mechanism. Our LSTMs were implemented with the PyTorch package.[1]

ActionLSTM ActionLSTMs (Choe and Charniak, 2016) are a syntactic LM (+ Syntax) based on RNN architectures (− SelfAttn). ActionLSTMs treat actions flatly without the composition function (− Composition). Our ActionLSTMs were implemented with the PyTorch package.

RNNG RNNGs are a syntactic LM (+ Syntax) based on RNN architectures (− SelfAttn). RNNGs recursively compose subtrees into a single vector representation with the composition function (+ Composition). The implementation with the PyTorch package by Noji and Oseki (2021) was employed.[2]

Transformer Transformers (Radford et al., 2018) are a vanilla LM (− Syntax) based on Transformer architectures (+ SelfAttn). Transformers were adopted as a baseline for syntactic LMs with the self-attention mechanism. Our Transformers were implemented with Huggingface's Transformer package (Wolf et al., 2020).[3]

PLM PLMs are a syntactic LM (+ Syntax) based on Transformer architectures (+ SelfAttn). PLMs treat actions flatly without the composition function (− Composition). The implementation with Huggingface's Transformer package by Qian et al. (2021) was employed.[4]

PLM-mask PLM-masks are a syntactic LM (+ Syntax) based on Transformer architectures (+ SelfAttn). PLM-masks do not utilize the composition function, but use the local subtree information with the dynamic masking mechanism

[1] https://github.com/pytorch/pytorch

[2] https://github.com/aistairc/rnng-pytorch

[3] https://github.com/huggingface/transformers

[4] https://github.com/IBM/transformers-struct-guidance

((+) Composition). The implementation with Huggingface's Transformer package by Qian et al. (2021) was employed.

CAG CAGs are a syntactic LM (+ Syntax) based on Transformer architectures (+ SelfAttn). CAGs recursively compose subtrees into a single vector representation with the composition function (+ Composition). Our CAGs were implemented with the PyTorch and Huggingface's Transformer packages.[5]

3.2 Training

All LMs were trained on the BLLIP-LG dataset, which comprises 1.8M sentences and 42M tokens sampled from the Brown Laboratory for Linguistic Information Processing 1987-89 Corpus Release 1 (BLLIP; Charniak et al., 2000). We followed the train-dev-test split of Hu et al. (2020). Following Qian et al. (2021), we split the sentences into subwords using a Byte Pair Encoding tokenizer (Sennrich et al., 2016) from Huggingface's Transformer package. The baseline vanilla LMs used only terminal subwords, whereas the syntactic LMs used terminal subwords and syntactic structures. We utilized syntactic structures re-parsed by Hu et al. (2020) with a state-of-the-art constituency parser (Kitaev and Klein, 2018). All LMs were trained at the sentence level with a learning rate of 10^{-3}, a dropout rate of 0.1, Adam optimizer, and a mini-batch size of 256 for 15 epochs. We selected the checkpoint with the lowest loss on the development set for evaluation. The experiment was conducted three times with different random seeds.

3.3 Targeted syntactic evaluation

In order to evaluate whether LMs learn human-like syntactic generalization, we employed six test circuits (Hu et al., 2020) on the SyntaxGym benchmark (Gauthier et al., 2020). Specifically, each test circuit deals with the following grammatical phenomenon: Agreement, Licensing, Garden-Path Effects, Gross Syntactic State, Center Embedding, and Long-Distance Dependencies. Each circuit is further subcategorized into suites; for example, the Agreement circuit contains a suite on a specific type of Agreement, such as "subject-verb number agreement with prepositional phrase". Each test suite consists of items designed to probe the specific grammatical phenomenon, and LMs succeed when they meet a success criterion, which defines inequalities among conditional probabilities on a grammatically critical position that should hold if they have learned the appropriate syntactic generalization. For example, to succeed on an item of the "subject-verb number agreement with prepositional phrase" suite, LMs should assign a higher probability to the underlined critical position of (1a) than (1b):

(1) a. The author next to the senators is good.

b. *The author next to the senators are good.

Following Qian et al. (2021), we employed word-synchronous beam search (Stern et al., 2017) to derive the probability of a grammatically critical position from syntactic LMs. Word-synchronous beam search retains a collection of the most likely syntactic structures that are predicted given an observed partial sentence $w_1, \cdots, w_i$ and marginalizes their probabilities to approximate $p(w_i|w_{<i})$:

$$p(w_i|w_{<i}) = \frac{p(w_1, \cdots, w_i)}{p(w_1, \cdots, w_{i-1})} \sim \frac{\sum_{Y_i \in \mathcal{Y}_i} p(w_1, \cdots, w_i, Y_i)}{\sum_{Y_{i-1} \in \mathcal{Y}_{i-1}} p(w_1, \cdots, w_{i-1}, Y_{i-1})} \quad (5)$$

where $\mathcal{Y}_i$ denotes the collection of syntactic structures given $w_1, \cdots, w_i$. Following Qian et al. (2021), we set the action beam size to 100, word beam size to 10, and fast-track to 5.

4 Results and discussion

4.1 Overall accuracies

Overall accuracies of our controlled experiment are summarized in Figure 3. The average accuracies across the SyntaxGym test suites and different random seeds (the vertical axis) are plotted against the LMs investigated in this paper (the horizontal axis), with the accuracies of PLM-masks and GPT-2 (Radford et al., 2019) from Qian et al. (2021). The accuracies of PLM-masks and GPT-2 from Qian et al. (2021) are reference points as their model sizes are significantly larger than the other models investigated in this paper. Each dot denotes the accuracy of a specific seed. The results demonstrate that CAGs achieved the highest overall accuracy, suggesting that the composition function and the

[5] Our implementation is available at https://github.com/osekilab/composition-attention-grammar. The implementation is based on the PyTorch implementation of RNNG by Noji and Oseki (2021).

	− Syntax	+ Syntax			
		− Composition	+ Composition	[+ Syn.] − [− Syn.]	[+ C.] − [− C.]
− SelfAttn	56.6 ± 3.3 (LSTM)	72.5 ± 1.8 (ActionLSTM)	81.1 ± 2.8 (RNNG)	20.2 ± 4.7	8.6 ± 3.3
+ SelfAttn	48.1 ± 1.5 (Transformer)	75.4 ± 0.2 (PLM)	69.6 ± 0.9 (PLM-mask) **83.8 ± 1.4 (CAG)**	24.4 ± 1.8 31.5 ± 2.1	-5.8 ± 0.9 8.4 ± 1.4
[+ S. A.] − [− S. A.]	-8.5 ± 3.7	2.9 ± 1.8	-11.5 ± 2.9 2.7 ± 3.1		

Table 3: Overall accuracy of each LM and the difference in the accuracy between minimally different LMs. [+ S. A.] − [− S. A.] denotes the difference in the accuracy between LMs with + SelfAttn and − SelfAttn. [+ Syn.] − [− Syn.] and [+ C.] − [− C.] denote the differences in the accuracy between LMs with + Syntax and − Syntax, and between LMs with + Composition and − Composition, respectively. The standard deviations of the differences were calculated, assuming that the accuracies were normally distributed.

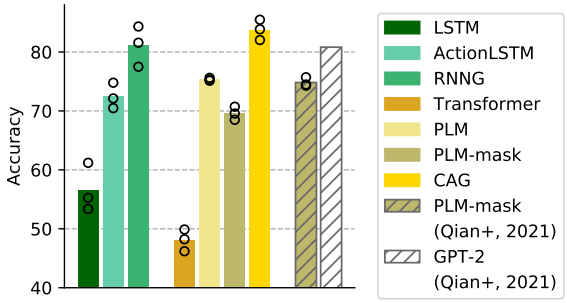


Figure 3: Overall accuracies of our controlled experiment. The average accuracies across the SyntaxGym test suites and different random seeds (the vertical axis) are plotted against the LMs investigated in this paper (the horizontal axis), with the accuracies of PLM-masks and GPT-2 from Qian et al. (2021). The accuracies of PLM-masks and GPT-2 from Qian et al. (2021) are reference points as their model sizes are significantly larger than the other models investigated in this paper. Each dot denotes the accuracy of a specific seed.

self-attention mechanism both play an important role to make LMs more human-like. Notice importantly that CAGs (83.8%) outperformed GPT-2 (80.8%) trained on 250× data with a 7× model size.

In the rest of this subsection, we discuss the effects of model components on the overall accuracy. In order to isolate the effects of individual components, Table 3 shows the overall accuracy of each LM and the difference in the accuracy between minimally different LMs.

+Syntax vs. −Syntax The LMs with explicit syntactic supervision outperformed the LMs without it, both without the self-attention mechanism (LSTM: 56.6%, the average accuracy of ActionLSTM and RNNG: 76.7%; +20.2%) and with the self-attention mechanism (Transformer: 48.1%, the average accuracy of PLM and PLM-mask: 72.5%; +24.4%, and the average accuracy of PLM and CAG: 79.4%; +31.5%). This result corroborates previous work (Kuncoro et al., 2017; Wilcox et al., 2019; Hu et al., 2020), suggesting that explicit syntactic supervision plays an important role to make LMs more human-like.

+Composition vs. −Composition The LMs with the composition function outperformed the LMs without it, both without the self-attention mechanism (ActionLSTM: 72.5%, RNNG: 81.1%; +8.6%) and with the self-attention mechanism (PLM: 75.4%, CAG: 83.8%; +8.4%), suggesting that the composition function induces human-like syntactic generalization (Kuncoro et al., 2017; Wilcox et al., 2019).

+SelfAttn vs. −SelfAttn Without explicit syntactic supervision, the LMs with the self-attention mechanism underperformed the LMs without it (LSTM: 56.6%, Transformer: 48.1%; -8.5%). In contrast, with explicit syntactic supervision, the LMs with the self-attention mechanism outperformed the LMs without it, both without the composition function (ActionLSTM: 72.5%, PLM: 75.4%; +2.9%) and with the composition function (RNNG: 81.1%, CAG: 83.8%; +2.7%). This result suggests that it is important to selectively attend to previous structural information not just words.

+Composition vs. (+)Composition CAGs with the composition function outperformed PLM-masks with the dynamic masking mechanism (PLM-mask: 69.6%, CAG: 83.8%; +14.2%). This result suggests that recursive composition of subtrees has additional advantages over the local subtree information in inducing human-like syntactic generalization. Note incidentally that our PLM-masks achieved a lower accuracy (69.6%) than PLM-masks from Qian et al. (2021) (74.8%), which may be caused by the difference in balance between specialized and vanilla attention heads: we

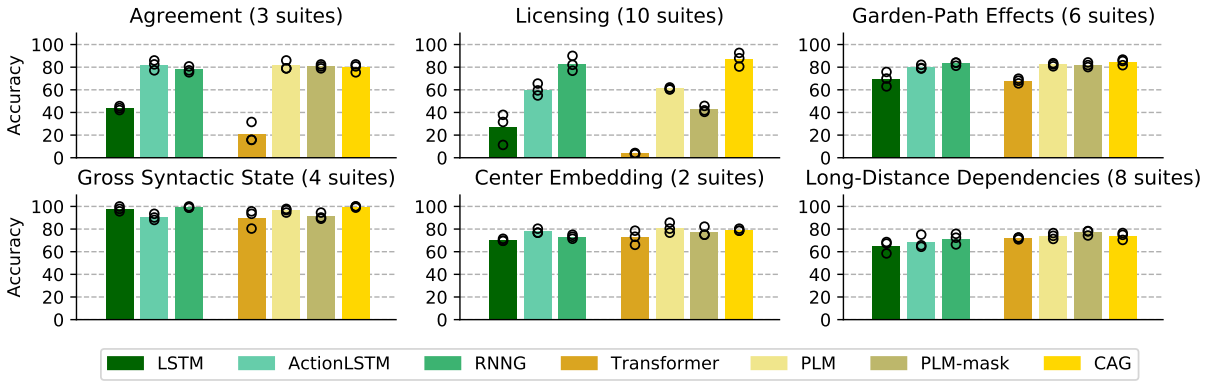


Figure 4: Circuit accuracies of our controlled experiment. The average accuracies across the SyntaxGym test suites and different random seeds on each test circuit (the vertical axis) are plotted against the LMs investigated in this paper (the horizontal axis). Each dot denotes the accuracy of a specific seed.

specialized two out of 4 attention heads, whereas Qian et al. (2021) specialized two out of 12. Nevertheless, given that CAGs (83.8%) outperformed the PLM-masks from Qian et al. (2021) (74.8%) by a large margin, it is safe to conclude that recursive composition of subtrees has additional advantages over the local subtree information.

4.2 Circuit accuracies

Circuit accuracies of our controlled experiment are summarized in Figure 4. The average accuracies across the SyntaxGym test suites and different random seeds on each test circuit (the vertical axis) are plotted against the LMs investigated in this paper (the horizontal axis). Each dot denotes the accuracy of a specific seed. The results demonstrate that with explicit syntactic supervision, the LMs with the self-attention mechanism marginally outperformed the LMs without it on most of the test circuits, but the LMs with the composition function outperformed or underperformed the LMs without it depending on the test circuits.

In the rest of this subsection, we investigate the pros and cons of the composition function through closer inspection of grammatical phenomena.

Syntactic features may percolate into the subtree representations. The LMs with the composition function outperformed the comparable LMs without it on three out of six circuits (Licensing, Garden-Path Effects, and Gross Syntactic State). Specifically, RNNGs and CAGs both outperformed ActionLSTMs and PLMs by a large margin (+23.0% and +26.0%, respectively) on Licensing, which includes items like (3):

(3) a. The author next to the senators hurt herself.

b. *The authors next to the senator hurt herself.

To successfully assign a higher probability to (3a) than (3b), LMs should understand that the reflexive pronoun must agree with the subject of the sentence in number. The subject NP "The author/authors next to the senators/senator" is composed into a single NP vector, as confirmed by the fact that RNNGs and CAGs both correctly assigned the following structure "(NP The author/authors (ADVP next (PP to (NP the senators/senator))))" to the subject NP.[6] Given that RNNGs and CAGs successfully assigned a higher probability to an acceptable sentence through this subject NP vector, we can hypothesize that the syntactic features such as number may properly percolate into the subject NP vector.

Semantic features may not percolate into the subtree representations. In contrast, the LMs with the composition function underperformed the comparable LMs without it on the other circuits (Agreement, Center Embedding, and Long-Distance Dependencies). Specifically, RNNGs and CAGs both underperformed ActionLSTMs and PLMs most significantly on Center Embedding (-4.76% and -1.79%, respectively), which includes items like (4):

(4) a. The shirt that the man bought ripped.

b. *The shirt that the man ripped bought.

[6]RNNGs and CAGs both achieved considerably high bracketing F1 (RNNG: 96.0, CAG: 98.2) against acceptable test sentences parsed with the state-of-the-art constituency parser (Kitaev and Klein, 2018) on the Licensing circuit.

To successfully assign a higher probability to (4a) than (4b), LMs should understand that the verb that can take the inanimate subject "shirt" should appear at the end of the sentence. The subject NP "The shirt that the man bought/ripped" is composed into a single NP vector, as confirmed by the fact that RNNGs and CAGs both correctly assigned the following structure "(NP The shirt (SBAR (WHNP that)(S (NP the man)(VP bought/ripped))))" to the subject NP.[7] Given that RNNGs and CAGs failed to assign a higher probability to an acceptable sentence through this subject NP vector, we can hypothesize that the semantic features such as animacy may not properly percolate into the subject NP vector.

What kind of features percolates? The important implication here is that, with the composition function, the syntactic features may percolate into the subtree representations, but the semantic features may not. The detailed analysis of this implication (e.g., an analysis of the inner mechanics of feature percolation at the single neuron level; Lakretz et al., 2019) will remain for future work.

4.3 Overall accuracy and perplexity

In this subsection, we compare the SyntaxGym overall accuracy against perplexity, the standard evaluation metric for LMs. The relationship between the overall accuracy and perplexity is summarized in Figure 5: the overall accuracy (vertical axis) is plotted against perplexity (horizontal axis; lower is better). Following Qian et al. (2021), we calculated the perplexity on the BLLIP held-out test set and derived the perplexity from the syntactic LMs, given the syntactic structures of the test sentences equal to the gold structures. Figure 5 demonstrates that explicit syntactic supervision generally improves both the overall accuracy and perplexity, but among the syntactic LMs, the overall accuracy is not linearly correlated with perplexity: PLMs and PLM-masks achieved worse overall accuracy, but better perplexity than RNNGs and CAGs. This result corroborates Hu et al. (2020) that suggests a dissociation between perplexity and human-like syntactic generalization performance.

Recently, the relationship between perplexity and LMs' cognitive plausibility has attracted considerable attention. Besides LMs' human-like syntactic generalization performance, previous work on the correlation between perplexity and LMs' psychometric predictive power has typically reported that LMs with the better perplexity are more cognitively plausible (Fossum and Levy, 2012; Goodkind and Bicknell, 2018; Wilcox et al., 2020), but more recently, the counter-argument that lower perplexity is not always human-like has been widely discussed (Hao et al., 2020; Oh et al., 2021; Kuribayashi et al., 2021). Given these recent trends, it is possible that the evaluation solely on perplexity may be orthogonal to the goal of human-like LMs (cf. Linzen, 2020).

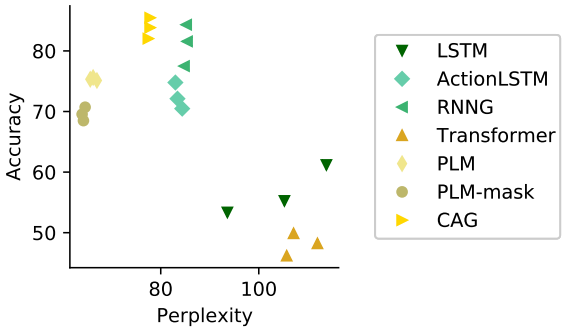


Figure 5: The relationship between the overall accuracy and perplexity: the overall accuracy (vertical axis) is plotted against perplexity (horizontal axis; lower is better).

5 Related work

While writing this paper, we noticed that Sartran et al. (2022), which is similar in spirit to our work, was submitted to the arXiv: they proposed Transformer Grammars (TGs) that incorporate recursive syntactic composition. TGs obtain a single vector representation of subtrees with the self-attention mechanism via an attention mask, but in contrast, CAGs obtain the representation with the composition function based on bidirectional LSTMs. While TGs are superior to CAGs in computational efficiency (see Limitations section), CAGs achieved better syntactic generalization performance on SyntaxGym (83.8%) than TGs (82.5%) that were trained with a 12× model size, suggesting that the composition function based on bidirectional LSTMs is advantageous in obtaining a vector representation of subtrees. Thorough comparisons between CAGs and TGs will remain for future work.

[7] RNNGs and CAGs both achieved high bracketing F1 (RNNG: 96.7, CAG: 95.2) on the Center Embedding circuit. In addition, these scores are higher than ActionLSTMs and PLMs (ActionLSTM: 96.1, PLM: 94.2), respectively, indicating that the lower accuracy of RNNGs and CAGs than ActionLSTMs and PLMs on this circuit is not due to failure in parsing.

6 Conclusion

In this paper, we proposed a novel architecture called **Composition Attention Grammars** (CAGs) that recursively compose subtrees into a single vector representation with the composition function, and selectively attend to previous structural information with the self-attention mechanism. We investigated whether these components—the composition function and the self-attention mechanism—can both induce human-like syntactic generalization. Specifically, we trained LMs with and without these two components with the model sizes carefully controlled, and evaluated their syntactic generalization performance against six test circuits on the SyntaxGym benchmark. The results demonstrated that the composition function and the self-attention mechanism both play an important role to make LMs more human-like, and closer inspection of grammatical phenomena implied that the composition function allowed syntactic features, but not semantic features, to percolate into subtree representations.

Limitations

Although it is not a central research question in this paper, a limitation with CAGs is their computational cost. While TGs (Sartran et al., 2022) process all inputs simultaneously during training as in vanilla Transformers, CAGs must be trained recursively because the internal state of the stack changes dynamically due to the composition function. In fact, although we utilized effective batching for LMs with the composition function (Noji and Oseki, 2021) and prevented CAGs from recomputing pre-computed attention keys and values, training of CAGs on the BLLIP-LG dataset (1.8M sentences and 42M tokens) for 15 epochs took two weeks on eight GPUs (NVIDIA V100). In addition, the self-attention mechanism consumes a large amount of memory, making it difficult to train CAGs with larger model sizes. The model size in this paper is the maximum that can be trained on V100 with 32GB memory. In order to address these limitations, we plan to introduce a computationally efficient self-attention mechanism (cf. Tay et al., 2020) to CAGs in future work.

Acknowledgement

We would like to thank Peng Qian for sharing the re-parsed BLLIP-LG dataset, which is used to train syntactic LMs in Hu et al. (2020), and for answering various questions. We are also grateful to three anonymous reviewers for valuable comments and suggestions. This work was supported by JST PRESTO Grant Number JPMJPR21C2.

References

Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, John Hale, and Mark Johnson. 2000. Bllip 1987-89 wsj corpus release 1. *Linguistic Data Consortium, Philadelphia*, 36.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336, Austin, Texas. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14:179–211.

Victoria Fossum and Roger Levy. 2012. Sequential vs. hierarchical syntactic models of human incremental sentence processing. In *Proceedings of the 3rd Workshop on Cognitive Modeling and Computational Linguistics (CMCL 2012)*, pages 61–69, Montréal, Canada. Association for Computational Linguistics.

Jon Gauthier, Jennifer Hu, Ethan Wilcox, Peng Qian, and Roger Levy. 2020. SyntaxGym: An online platform for targeted evaluation of language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 70–76, Online. Association for Computational Linguistics.

Adam Goodkind and Klinton Bicknell. 2018. Predictive power of word surprisal for reading times is a linear function of language model quality. In *Proceedings of the 8th Workshop on Cognitive Modeling and Computational Linguistics (CMCL 2018)*, pages 10–18, Salt Lake City, Utah. Association for Computational Linguistics.

John Hale, Chris Dyer, Adhiguna Kuncoro, and Jonathan Brennan. 2018. Finding syntax in human encephalography with beam search. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2727–2736, Melbourne, Australia. Association for Computational Linguistics.

Yiding Hao, Simon Mendelsohn, Rachel Sterneck, Randi Martinez, and Robert Frank. 2020. Probabilistic predictions of people perusing: Evaluating metrics of language model performance for psycholinguistic modeling. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pages 75–86, Online. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural computation*, 9(8):1735–80.

Jennifer Hu, Jon Gauthier, Peng Qian, Ethan Wilcox, and Roger Levy. 2020. A systematic assessment of syntactic generalization in neural language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1725–1744, Online. Association for Computational Linguistics.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What do recurrent neural network grammars learn about syntax? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1249–1258, Valencia, Spain. Association for Computational Linguistics.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Melbourne, Australia. Association for Computational Linguistics.

Tatsuki Kuribayashi, Yohei Oseki, Takumi Ito, Ryo Yoshida, Masayuki Asahara, and Kentaro Inui. 2021. Lower perplexity is not always human-like. 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 5203–5217, Online. Association for Computational Linguistics.

Yair Lakretz, German Kruszewski, Theo Desbordes, Dieuwke Hupkes, Stanislas Dehaene, and Marco Baroni. 2019. The emergence of number and syntax units in LSTM language models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 11–20, Minneapolis, Minnesota. Association for Computational Linguistics.

Tal Linzen. 2020. How can we accelerate progress towards human-like linguistic generalization? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5210–5217, Online. Association for Computational Linguistics.

Hiroshi Noji and Yohei Oseki. 2021. Effective batching for recurrent neural network grammars. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4340–4352, Online. Association for Computational Linguistics.

Byung-Doh Oh, Christian Clark, and William Schuler. 2021. Surprisal estimators for human reading times need character models. 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 3746–3757, Online. Association for Computational Linguistics.

Peng Qian, Tahira Naseem, Roger Levy, and Ramón Fernandez Astudillo. 2021. Structural guidance for transformer language models. 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 3735–3745, Online. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Laurent Sartran, Samuel Barrett, Adhiguna Kuncoro, Miloš Stanojević, Phil Blunsom, and Chris Dyer. 2022. Transformer grammars: Augmenting transformer language models with syntactic inductive biases at scale.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Mitchell Stern, Daniel Fried, and Dan Klein. 2017. Effective inference for generative neural parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1695–1700, Copenhagen, Denmark. Association for Computational Linguistics.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020. Efficient transformers: A survey.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Proceedings of NIPS*, pages 5998–6008.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019b. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. BLiMP: The benchmark of linguistic minimal pairs for English. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Ethan Wilcox, Jon Gauthier, Jennifer Hu, Peng Qian, and Roger Levy. 2020. On the predictive power of neural language models for human real-time comprehension behavior. In *Proceedings of the 42th Annual Meeting of the Cognitive Science Society - Developing a Mind: Learning in Humans, Animals, and Machines, CogSci 2020, virtual, July 29 - August 1, 2020*. cognitivesciencesociety.org.

Ethan Wilcox, Peng Qian, Richard Futrell, Miguel Ballesteros, and Roger Levy. 2019. Structural supervision improves learning of non-local grammatical dependencies. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3302–3312, Minneapolis, Minnesota. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Ryo Yoshida, Hiroshi Noji, and Yohei Oseki. 2021. Modeling human sentence processing with left-corner recurrent neural network grammars. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2964–2973, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

A Effect of individual components on circuit accuracies.

As with the overall accuracy, in order to isolate the effect of individual components on circuit accuracies, Table 4 shows the circuit accuracy of each LM and the difference in accuracy between LMs with minimal differences.

	− Syntax	+ Syntax			
		− Composition	+ Composition	[+ Syn.] − [− Syn.]	[+ C.] − [− C.]
− SelfAttn	43.9 ± 1.4 (LSTM)	**81.9 ± 3.6 (ActionLSTM)**	77.8 ± 2.2 (RNNG)	38.0 ± 3.9	-4.09 ± 4.2
+ SelfAttn	21.1 ± 7.4 (Transformer)	81.3 ± 3.3 (PLM)	80.7 ± 1.4 (PLM-mask) 79.5 ± 3.0 (CAG)	59.9 ± 8.2 59.3 ± 8.6	-0.585 ± 3.6 -1.75 ± 4.5
[+ S. A.] − [− S. A.]	-22.8 ± 7.6	-0.585 ± 4.9	2.92 ± 2.6 1.75 ± 3.7		

(a) Agreement

	− Syntax	+ Syntax			
		− Composition	+ Composition	[+ Syn.] − [− Syn.]	[+ C.] − [− C.]
− SelfAttn	26.9 ± 11.3 (LSTM)	60.0 ± 4.3 (ActionLSTM)	83.0 ± 5.4 (RNNG)	33.1 ± 12	23.0 ± 6.9
+ SelfAttn	3.68 ± 0.4 (Transformer)	61.1 ± 0.8 (PLM)	42.7 ± 2.2 (PLM-mask) **87.0 ± 5.0 (CAG)**	48.2 ± 2.4 70.4 ± 5.1	-18.3 ± 2.4 26.0 ± 5.0
[+ S. A.] − [− S. A.]	-23.2 ± 11	1.05 ± 4.4	-40.3 ± 5.9 4.04 ± 7.4		

(b) Licensing

	− Syntax	+ Syntax			
		− Composition	+ Composition	[+ Syn.] − [− Syn.]	[+ C.] − [− C.]
− SelfAttn	69.6 ± 5.2 (LSTM)	80.1 ± 1.5 (ActionLSTM)	83.1 ± 1.3 (RNNG)	10.5 ± 5.4	3.01 ± 2.0
+ SelfAttn	67.9 ± 1.7 (Transformer)	82.2 ± 1.1 (PLM)	82.0 ± 1.7 (PLM-mask) **84.6 ± 2.1 (CAG)**	14.2 ± 2.6 15.5 ± 2.9	-0.198 ± 2.1 2.38 ± 2.4
[+ S. A.] − [− S. A.]	-1.72 ± 5.5	2.18 ± 1.9	-1.03 ± 2.1 1.55 ± 2.5		

(c) Garden-Path Effects

	− Syntax	+ Syntax			
		− Composition	+ Composition	[+ Syn.] − [− Syn.]	[+ C.] − [− C.]
− SelfAttn	97.8 ± 1.8 (LSTM)	90.6 ± 2.2 (ActionLSTM)	99.3 ± 0.5 (RNNG)	-7.25 ± 2.9	8.70 ± 2.3
+ SelfAttn	89.9 ± 6.7 (Transformer)	96.4 ± 1.4 (PLM)	91.3 ± 2.3 (PLM-mask) **99.6 ± 0.5 (CAG)**	3.95 ± 7.2 8.1 ± 6.9	-5.07 ± 2.7 3.26 ± 1.4
[+ S. A.] − [− S. A.]	-7.97 ± 7.0	5.80 ± 2.6	-7.97 ± 2.4 0.362 ± 0.72		

(d) Gross Syntactic State

	− Syntax	+ Syntax			
		− Composition	+ Composition	[+ Syn.] − [− Syn.]	[+ C.] − [− C.]
− SelfAttn	70.2 ± 0.8 (LSTM)	78.0 ± 1.7 (ActionLSTM)	73.2 ± 1.5 (RNNG)	7.74 ± 1.9	-4.76 ± 2.2
+ SelfAttn	72.6 ± 5.1 (Transformer)	**81.0 ± 3.7 (PLM)**	77.4 ± 3.4 (PLM-mask) 79.2 ± 0.8 (CAG)	6.6 ± 7.2 7.5 ± 6.4	-3.57 ± 5.0 -1.79 ± 3.8
[+ S. A.] − [− S. A.]	2.38 ± 5.2	2.98 ± 4.0	4.17 ± 3.7 5.95 ± 1.7		

(e) Center Embedding

	− Syntax	+ Syntax			
		− Composition	+ Composition	[+ Syn.] − [− Syn.]	[+ C.] − [− C.]
− SelfAttn	64.7 ± 4.5 (LSTM)	68.4 ± 4.8 (ActionLSTM)	71.5 ± 3.9 (RNNG)	3.63 ± 6.6	3.17 ± 6.2
+ SelfAttn	71.9 ± 0.7 (Transformer)	73.9 ± 2.1 (PLM)	**76.9 ± 1.8 (PLM-mask)** 73.9 ± 2.6 (CAG)	3.5 ± 2.9 2.0 ± 3.4	2.93 ± 2.8 0.00992 ± 3.3
[+ S. A.] − [− S. A.]	7.20 ± 4.5	5.56 ± 5.2	5.32 ± 4.3 2.40 ± 4.7		

(f) Long-Distance Dependencies

Table 4: Circuit accuracy of each LM and the difference in the accuracy between LMs with minimal differences. [+ S. A.] − [− S. A.] denotes the difference in the accuracy between LMs with + SelfAttn and − SelfAttn. [+ Syn.] − [− Syn.] and [+ C.] − [− C.] denote the difference in the accuracy between LMs with + Syntax and − Syntax , and the difference in the accuracy between LMs with + Composition and − Composition, respectively. The standard deviations of the differences were calculated assuming that the accuracies were normally distributed.