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...
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 Char-
niaik, 2016; Dyer et al., 2016; Qian et al., 2021), which estimates the following joint distribution of a sentence \( X \) and its syntactic structure \( Y \):

\[
p(X, Y) = p(a_1, \ldots, a_n) = \prod_{t=1}^{n} p(a_t | a_{<t})
\]  

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 \( e_x \) onto the stack.
- `NT(X)`: Push a nonterminal embedding \( e_X \) 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 \( e_1, \ldots, e_k \), to compose subtrees into a single vector representation \( e_s \):

\[
e_s = \text{Composition}([e_1, \ldots, e_k]).
\]  

\( e_s \) is then pushed onto the stack.

After each action, CAGs employ the self-attention mechanism, which selectively attends to previous vectors in the stack \( e_1, \ldots, e_k \) by calculating the weight of attention to each vector with the query, key, and value vectors generated from \( e_1, \ldots, e_k \), in order to represent a partial parse at each time step \( t \):

\[
h_t = \text{SelfAttn}([e_1, \ldots, e_k]).
\]
Then, $h_t$ defines the next action distribution:

$$a_{t+1} \sim \text{softmax}(W_a h_t + b_a)$$

where $W_a$ and $b_a$ are the weights and biases of a fully connected layer that projects $h_t$ to logits for each action $a$, and $\text{softmax}$ 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 $w_1, \ldots, w_n$. 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 non-terminal 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-
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 ((+Composition)).

---

\(^1\)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.

CAGs are syntactic LMs (+ 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 $p(w_i|w_{<i})$:

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

(5)

where $Y_i$ 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

5Our 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).
The LMs with explicit syntactic supervision 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), suggesting that explicit syntactic supervision plays an important role to make LMs more human-like.

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

<table>
<thead>
<tr>
<th></th>
<th>− Syntax</th>
<th>+ Syntax</th>
<th>− Composition</th>
<th>+ Composition</th>
<th>[+ Syn] − [− Syn]</th>
<th>[+ C] − [− C]</th>
</tr>
</thead>
<tbody>
<tr>
<td>− SelfAttn</td>
<td>56.6 ± 4.3 (LSTM)</td>
<td>72.5 ± 1.8 (ActionLSTM)</td>
<td>81.1 ± 2.8 (RNNG)</td>
<td>75.4 ± 0.2 (PLM)</td>
<td>24.4 ± 1.8</td>
<td>31.5 ± 2.1</td>
</tr>
<tr>
<td>+ SelfAttn</td>
<td>48.1 ± 1.5 (Transformer)</td>
<td>75.4 ± 0.2 (PLM)</td>
<td>69.6 ± 0.9 (PLM-mask)</td>
<td>83.8 ± 1.4 (CAG)</td>
<td>24.4 ± 1.8</td>
<td>4.9 ± 1.0</td>
</tr>
<tr>
<td>[+ S. A.] − [− S. A.]</td>
<td>8.5 ± 3.7</td>
<td>2.9 ± 1.8</td>
<td>-11.5 ± 2.9</td>
<td>2.7 ± 3.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.
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. 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.

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. 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).

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.

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


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.
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, respectively. The standard deviations of the differences were calculated assuming that the accuracies were normally distributed.