Tree-Planted Transformers: Unidirectional Transformer Language Models with Implicit Syntactic Supervision

Ryo Yoshida and **Taiga Someya** and **Yohei Oseki**
The University of Tokyo
{yoshiryo0617, taiga98-0809, oseki}@g.ecc.u-tokyo.ac.jp

Abstract

Syntactic Language Models (SLMs) can be trained efficiently to reach relatively high performance; however, they have trouble with inference efficiency due to the explicit generation of syntactic structures. In this paper, we propose a new method dubbed **tree-planting**: instead of explicitly generating syntactic structures, we "plant" trees into attention weights of unidirectional Transformer LMs to implicitly reflect syntactic structures of natural language. Specifically, unidirectional Transformer LMs trained with tree-planting will be called **Tree-Planted Transformers (TPT)**, which inherit the training efficiency from SLMs without changing the inference efficiency of their underlying Transformer LMs. Targeted syntactic evaluations on the SyntaxGym benchmark demonstrated that TPTs, despite the lack of explicit generation of syntactic structures, significantly outperformed not only vanilla Transformer LMs but also various SLMs that generate hundreds of syntactic structures in parallel. This result suggests that TPTs can learn human-like syntactic knowledge as data-efficiently as SLMs while maintaining the modeling space of Transformer LMs unchanged.

https://github.com/osekilab/TPT

1 Introduction

Recent years have witnessed remarkable success in Large Language Models (LLMs) based on Transformer LMs (Vaswani et al., 2017). However, despite their success, Transformer LMs have some drawback in *training efficiency*—especially when compared with humans. For example, GPT-3 (Brown et al., 2020) is trained on around $2,000\times$ larger data than a 12-year-old human would have experienced (Warstadt et al., 2023), indicating that Transformer LMs lack sufficient inductive bias for language acquisition.

On another strand, previous work has revealed that Syntactic Language Models (SLMs), defined as a generative models of a token sequence and its syntactic structures, can achieve high syntactic performance under data-constrained settings (Dyer et al., 2016; Noji and Oseki, 2021; Qian et al., 2021; Sartran et al., 2022; Yoshida and Oseki, 2022; Murty et al., 2023). For example, Sartran et al. (2022) showed that some SLMs can achieve comparable syntactic performance to an LLM-like model[1] that is trained with medium—around $250\times$ larger—data, suggesting that syntactic supervision is essential for LMs to achieve high training efficiency. However, despite their training efficiency, SLMs have trouble with *inference efficiency*—they require hundreds of syntactic structures generated via beam search (Stern et al., 2017; Crabbé et al., 2019) or an external parser to precisely approximate marginal distribution over a token sequence, which naturally incurs the costs hundreds of times higher than their underlying sequential models.

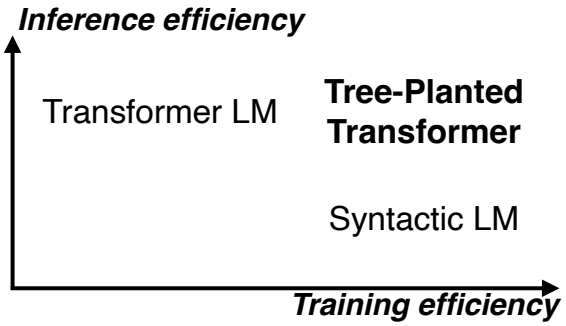


Figure 1: Two types of efficiency: training efficiency and inference efficiency. Our Tree-Planted Transformers (TPT) inherit the training efficiency from Syntactic LMs without changing the inference efficiency of their underlying Transformer LMs.

In this paper, we propose a new method dubbed **tree-planting**: instead of explicitly generating syn-

[1]Due to the rapid advances in recent years, what were once considered LLMs are no longer deemed "large" by current standards. We will refer to Transformer LMs larger than or equal to GPT-2 (Radford et al., 2018) as *LLM-like* models.

Findings of the Association for Computational Linguistics: ACL 2024, pages 5120–5134

tactic structures, we "plant" trees into attention weights of unidirectional Transformer LMs to implicitly reflect syntactic structures of natural language.[2] Specifically, unidirectional Transformer LMs trained with tree-planting will be called **Tree-Planted Transformers (TPT)**, which inherit the training efficiency from SLMs without changing the inference efficiency of their underlying Transformer LMs (Figure 1).

Previous studies have also explored the syntactic supervision of attention weights, mainly targeting bidirectional Transformer Encoders (Wu et al., 2018; Nguyen et al., 2020; Bugliarello and Okazaki, 2020; Bai et al., 2021; Sachan et al., 2021; Slobodkin et al., 2022; *inter alia*) (§2). These encoder-oriented approaches assume the entire sentence as input and typically aim to reflect the syntactic relationship between input words in a bottom-up manner. In contrast, tree-planting is uniquely designed for unidirectional Transformer LMs that can be used for text generation—and they have recently increased prevalence because of their compatibility with instruct tuning (Zhang et al., 2024) or Reinforcement Learning from Human Feedback (RLHF, Ouyang et al., 2022). Specifically, tree-planting considers syntactic structures involving the next word to be generated, by focusing on the *syntactic distance* (Shen et al., 2018, 2019; Du et al., 2020) between the next word and the previous words in the context (§3).

Targeted syntactic evaluations on the SyntaxGym benchmark (Gauthier et al., 2020) demonstrated that TPTs, despite the lack of explicit generation of syntactic structures, significantly outperformed not only vanilla Transformer LMs but also various SLMs that generate hundreds of syntactic structures in parallel. This result suggests that TPTs can learn human-like syntactic knowledge as data-efficiently as SLMs while maintaining the modeling space of Transformer LMs unchanged. Furthermore, closer inspection of syntactic phenomena implied that tree-planting shows high compatibility with dependency structures (§4).

Additionally, we analyzed the two hyperparameters introduced by tree-planting: (i) the number of heads where the tree-planting loss is applied, and (ii) the balance between the next-word prediction loss and the tree-planting loss. Our results demonstrated that the highest accuracy was generally achieved when a single head was adopted as a tree-planted head while the balancing parameter emerged as a crucial hyperparameter; excessively high and low weights on the tree-planting loss both led to ineffective outcomes (§5).

[2]The term "tree-planting" coincidentally bears a resemblance to the term used in Mueller and Linzen (2023), but this work diverges from ours in its motivation. Specifically, Mueller and Linzen (2023) investigated biases that enable syntactic generalization in Transformer LMs, from the perspectives of architectural features (depth, width, and number of parameters), as well as the genre and size of training corpus.

2 Related work

2.1 Syntactic Language Model

Syntactic Language Models (SLMs) are a generative model of a token sequence $\mathbf{x}$ and its syntactic structure $\mathbf{y}$. Formally, SLMs are defined as:

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{z}) = \prod_{t=1}^{n} p(z_t | z_{<t}), \qquad (1)$$

where $\mathbf{z}$ denotes the sequence of actions to generate both the token sequence and syntactic structure. For example, in top-down and left-to-right SLMs, each z_t could be either generating a token or opening/closing a constituent.

Recently, several SLMs based on the Transformer architecture have been proposed, achieving higher syntactic performance than medium LLM-like models (Qian et al., 2021; Sartran et al., 2022; Murty et al., 2023). However, because SLMs generate both a syntactic structure and token sequence, they cannot be directly utilized as LMs, or a generative model of a token sequence. To precisely approximate the marginal distribution over a token sequence, i.e., $p(\mathbf{x}) = \sum_{\mathbf{y} \in \mathcal{Y}} p(\mathbf{x}, \mathbf{y})$ (where $\mathcal{Y}$ represents the set of possible syntactic structures behind $\mathbf{x}$), they require hundreds of syntactic structures generated via beam search (Stern et al., 2017; Crabbé et al., 2019) or an external parser. Although actual costs would depend on the hardware and metrics, the calculation of $p(\mathbf{x}, \mathbf{y})$ for each structure in $\mathcal{Y}$ naturally multiplies inference costs by $|\mathcal{Y}|$ compared to the sequential models that directly calculates $p(\mathbf{x})$. Furthermore, additional costs would be incurred by the beam search procedure or the external parser itself.

2.2 Constraints on attention weights

As discussed in §2.1, the bottleneck that impairs SLMs' inference efficiency is their modeling space of the joint probability. To achieve the architecture that inherits the training efficiency of SLMs without changing the inference efficiency of their underlying Transformer LMs, it is necessary to

	Parser-free inference	Syntactic supervision	Unidirectional LM	Parallel computation
Wu et al. (2018);Nguyen et al. (2020); Bugliarello and Okazaki (2020);Bai et al. (2021); Sachan et al. (2021);Slobodkin et al. (2022)		✓		✓
Wang et al. (2019)	✓			✓
Strubell et al. (2018);Chen et al. (2023)	✓	✓		✓
Peng et al. (2019)	✓	✓	✓	
Tree-planting (ours)	✓	✓	✓	✓

Table 1: Comparison of our tree-planting with the previous work that constrains attention weights according to syntactic structures, based on the requirements for the architecture that inherits the training efficiency of SLMs without changing the inference efficiency of their underlying Transformer LMs: (i) parser-free inference, (ii) syntactic supervision, (iii) unidirectional LM, and (iv) parallel computation.

introduce syntactic supervision without changing the modeling space of the sequential models. For our goal, we will build upon another line of approach that constrains attention weights according to syntactic structures—mainly targeting bidirectional Transformer Encoders like BERT (Devlin et al., 2019).[3] Table 1 summarizes the previous work in this line of approach, comparing our tree-planting (§3) against others based on the requirements for our goal: (i) parser-free inference, (ii) syntactic supervision, (iii) unidirectional LM, and (iv) parallel computation.

First, the majority of these approaches are purely motivated to explicitly restrict attention weights with syntactic structures from external parsers, under the assumption that these parsers would be available during inference (Wu et al., 2018; Nguyen et al., 2020; Bugliarello and Okazaki, 2020; Bai et al., 2021; Sachan et al., 2021; Slobodkin et al., 2022). These studies achieved successful performance in their respective downstream tasks, but not only are their approaches all not directly applicable to unidirectional LMs, they also require external parsers during inference, rendering them not aligned with our goal of the inference efficient architecture.

Second, several approaches have been proposed that eliminate the need for external parsers during inference, but they still fall short of meeting all the requirements. Wang et al. (2019) aimed at an unsupervised approach, where a hierarchical architectural bias widens the range of neighboring tokens eligible to attend from lower to upper layers, yet this method is still not aligned with our goal of achieving higher training efficiency via syntactic supervision. Additionally, Strubell et al. (2018) and Chen et al. (2023) designed the loss functions that implicitly encourage the attention to syntactic parents or children for each token, satisfying the 3/4 requirements for our goal.[4] However, these approaches are potentially encoder-oriented and not suitable for unidirectional LMs; they assume the entire sentence as input and reflect the syntactic relationship between input words in a bottom-up manner.

Finally, another approach also closely aligned with the spirit of this research is a hybrid Parser and neural Language Model (PaLM; Peng et al., 2019). PaLM is the integration of an unidirectional RNN LM with an additional attention layer, which would be supervised to attend the constituent spans among the spans ending at time $t-1$: $\{w_1, \cdots, w_{t-1}\}, \cdots, \{w_{t-2}, w_{t-1}\}$. Although PaLM also meets the 3/4 requirements, it was by nature proposed for RNN LMs. The challenge arises when adapting PaLM to Transformer LMs; the generation of embeddings for the spans cannot be parallelized in a manner that is compatible with a self-attention mechanism of Transformer LMs.

To sum up, none of the previous approaches fully satisfy the requirements for our goal, highlighting the necessity for new methodologies.

3 Proposed method: tree-planting

In this paper, we propose a new method dubbed **tree-planting**: we "plant" trees into attention weights of unidirectional Transformer LMs to re-

[3] Beyond studies that constrain attention weights according to syntactic structures, there are also investigations that aimed at incorporating various types of information, such as word alignment, into the attention mechanisms (e.g., Yin et al., 2021). This subsection, however, specifically focuses on studies that target syntactic biases.

[4] Other than the approach to constrain attention weights, Tziafas et al. (2023) proposed a method to train Transformer Encoders in a multi-task setting of masked language modeling and categorial grammar supertagging. As an anonymous reviewer correctly pointed out, this method also satisfied the 3/4 requirements: (i), (ii), and (iv).

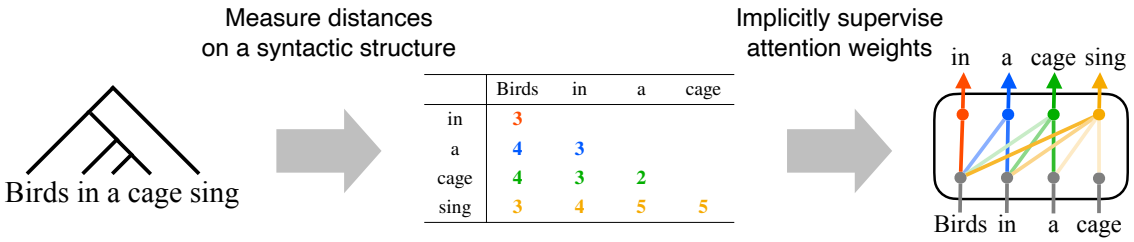


Figure 2: Overview of the proposed method: tree-planting

flect syntactic structures of natural language (Figure 2). Specifically, unidirectional Transformer LMs trained with tree-planting will be called **Tree-Planted Transformers (TPT)**, which inherit the training efficiency from SLMs without changing the inference efficiency of their underlying Transformer LMs. Tree-planting is uniquely designed for unidirectional Transformer LMs, satisfying all the requirements discussed in §2.2: (i) parser-free inference, (ii) syntactic supervision, (iii) unidirectional LM, and (iv) parallel computation.

3.1 Supervision of attention weights

A self-attention mechanism of unidirectional Transformer LMs computes a representation for predicting the next token through a weighted sum of each token in the context. Specifically, when predicting the $i+1$-th token, the attention weights from the i-th token to the j-th token is computed as follows:

$$A_{ij} = \frac{\exp\left(\frac{\mathbf{Q}_i \mathbf{K}_j^T}{\sqrt{d_K}}\right)}{\sum_{k=1}^{i} \exp\left(\frac{\mathbf{Q}_i \mathbf{K}_k^T}{\sqrt{d_K}}\right)}, \quad (2)$$

where $\mathbf{Q}_i$ and $\mathbf{K}_j$ represent the query vector of the i-th token and the key vector of the j-th token, respectively, and d_K denotes the dimension of the key vector. As Equation 2 shows, the computation for the $i+1$-th token prediction does not depend on any computation for the $1, \cdots, i$-th token predictions, which enables parallel computation.

In producing the supervision of attention weights, we extend the notion of *syntactic distance array* (Shen et al., 2018, 2019; Du et al., 2020), a 1D array of the number of edges on syntactic structures between two *consecutive* words, to a 2D matrix between *all pairs* of words:

$$D_{ij} = \mathrm{CountEdge}(w_i, w_j), \quad (3)$$

where w_i and w_j represent the i-th and j-th words, respectively, and CountEdge is the function that maps a pair of words to the number of edges on syntactic structures between them. This notion of *syntactic distance matrix* could be applied to any kind of syntactic structure, as long as the number of edges can be counted on it.[5]

Then, the syntactic distance matrix $\mathbf{D}$ is converted to the supervision of attention weights $\mathbf{S}$ as follows:

$$S_{ij} = \begin{cases} \frac{\exp(-D_{i+1,j})}{\sum_{k=1}^{i} \exp(-D_{i+1,k})} & (i \geq j) \\ 0 & (i < j) \end{cases}, \quad (4)$$

where S_{ij} represents the supervision of the attention weight from the i-th word to the j-th word when predicting the $i+1$-th word. In short, this supervision expects the attention weight of each word to decrease exponentially with the number of edges between the predicted word.[6]

This supervision design is highly oriented towards unidirectional Transformer LMs; it considers syntactic structures involving the next word to be generated in a manner compatible with parallel computation of the self-attention mechanism. This only successfully satisfies the 3/4 requirements for our purpose: (ii) syntactic supervision, (iii) unidirectional LM, and (iv) parallel computation. To fulfill the remaining requirement of (i) parser-free inference, we adopt a strategy similar to that of Strubell et al. (2018); Chen et al. (2023), designing the loss function to implicitly supervise attention.

3.2 Loss function

The supervision in §3.1 is produced at the word level but Transformer LMs typically take their input at the subword level. To bridge this gap, we first

[5]Note importantly that compatibility with the syntactic distance matrix varies across types of syntactic structures because some kind of information would be lost during the conversion from syntactic structures to the matrix. For example, when applied to dependency structures, the information on the direction of syntactic dependency will be lost.

[6]We adopt an exponential function as Lin and Tegmark (2017) reported that the mutual information between words would decay exponentially with respect to the number of edges on the syntactic structure between them.

convert the subword-level attention weight matrix $\mathbf{A}$ from a targeted Transformer LM to the word-level attention weight matrix $\mathbf{W}$ as follows:

$$W_{ij} = \frac{C_{ij}}{\sum_{k=1}^{i} C_{ik}}, \tag{5}$$

$$C_{ij} = \sum_{l=\mathrm{START}(w_{i+1})}^{\mathrm{END}(w_{i+1})} \sum_{m=\mathrm{START}(w_j)}^{\mathrm{END}(w_j)} A_{lm}, \tag{6}$$

where W_{ij} represents the word-level attention weight from the i-th word to the j-th word. C_{ij} is defined as the sum of the subword-level attention weights over the subword inside w_j when predicting the subword inside w_{i+1}, with A_{lm} representing the subword-level attention weight from the l-th subword to the m-th subword and START and END being the functions that map words to their start and end subword index, respectively. We employ $\mathbf{A}$ from specific attention heads called tree-planted heads.[7]

To implicitly supervise the word-level attention weight matrix $\mathbf{W}$ with the supervision $\mathbf{S}$, we introduce a tree-planting loss $\mathcal{L}_{\mathrm{TREE}}$ employing a Kullback–Leibler (KL) Divergence loss D_{KL}:[8]

$$\mathcal{L}_{\mathrm{TREE}} = \frac{\sum_{i=1}^{n-1} D_{\mathrm{KL}}(\mathbf{S}_i || \mathbf{W}_i)}{n-1}, \tag{7}$$

where n represents the length of a word sequence $\mathbf{w}$. In short, the tree-planting loss is the average KL Divergence loss in predicting each word except the beginning of $\mathbf{w}$.

During the training, $\mathcal{L}_{\mathrm{TREE}}$ is averaged over tree-planted heads and balanced with the next word prediction loss $\mathcal{L}_{\mathrm{NWP}}$:

$$\mathcal{L} = \mathcal{L}_{\mathrm{NWP}} + \lambda \frac{\sum_{h \in \mathcal{H}} \mathcal{L}_{\mathrm{TREE}}^{(h)}}{H}, \tag{8}$$

where $\mathcal{L}_{\mathrm{TREE}}^{(h)}$ represents a tree-planting loss for each tree-planted head h, H is the total number of tree-planted heads, and λ is a weight that balances the importance of the next word prediction loss and the average tree-planting loss. Unidirectional Transformer LMs trained with this loss function will be called Tree-Planted Transformers (TPT).

[7] Qian et al. (2021) also proposed the architecture which constrains some attention heads based on syntactic structures, or PLM-mask. PLM-mask and our tree-planting are similar in spirit, but they are quite different in their implementation: PLM-mask is a type of SLM that jointly generates a word sequence and its syntactic structure, but tree-planting builds TPTs, a type of LM. Furthermore, PLM-mask explicitly masks the attention weights based on the local parser state but tree-planting implicitly guides attention weights to reflect the whole syntactic structure.

[8] This loss function is inspired by Ma et al. (2023), which guides attention weights to focus on relevant texts in a document-level relation extraction task.

4 Experiment

To investigate whether TPTs can learn human-like syntactic knowledge as data-efficiently as SLMs while maintaining the modeling space of Transformer LMs unchanged, we conduct training on a small treebank and targeted syntactic evaluations on a syntactic knowledge benchmark.

4.1 Settings

Training data We used `LG` dataset of Hu et al. (2020), which comprises approximately 1.8M sentences from BLLIP corpus (Charniak et al., 2000). Implicit syntactic supervision with each of three types of syntactic structures was investigated: (i) dependency structures (`[dep.]`), (ii) constituency structures (`[cons.]`), and (iii) binarized constituency structures (`[bin.]`). The (i) dependency structures were parsed with the `en_core_web_sm` model from the `spacy` library (Montani et al., 2023).[9] The (ii) constituency structures were re-parsed with the Berkeley Neural Parser (Kitaev and Klein, 2018)[10] by Hu et al. (2020). The (iii) binarized constituency structures were obtained by the binarization of the (ii) constituency structures with the `chomsky_normal_form` function from the `nltk` library (Bird et al., 2009).[11] We removed 43,986 sentences that the dependency parser analyzed as multiple sentences, but the constituency parser analyzed as a single sentence.

Models We used the same architecture and BPE tokenizer as GPT-2 small (124M; Radford et al., 2018). The implementation of `GPT2LMHeadModel` and `GPT2Tokenizer` from the `transformers` library (Wolf et al., 2020)[12] were employed but all parameters of `GPT2LMHeadModel` were randomly initialized. For the tree-planted head and the weight of the tree-planting loss, we adopted a single attention head on the last layer and $\lambda = 0.5$, respectively. The choice of the tree-planted head and the weight was based on preliminary experiments and the detailed effects of them will be described in §5.

[9] https://spacy.io

[10] https://github.com/nikitakit/self-attentive-parser

[11] https://www.nltk.org

[12] https://huggingface.co/docs/transformers

As baselines, we trained three models: (i) a model with zero weight for the tree-planting loss (`[zero]`), (ii) a model supervised with random syntactic distances that were generated from the distribution same as the dependency structures (`[rand.]`), and (iii) a model supervised with sequential distances (`[seq.]`). Note importantly, (i) is equivalent to a Transformer LM. Hyperparameters are shown in Appendix A.

Evaluation data We evaluated syntactic knowledge of the models via targeted syntactic evaluations on the SyntaxGym benchmark (Gauthier et al., 2020). The SyntaxGym benchmark comprises six syntactic *circuits*: `Agreement`, `Center Embedding`, `Garden-Path Effects`, `Gross Syntactic States`, `Licensing`, and `Long-Distance Dependencies`. Each syntactic circuit consists of 2–10 syntactic *suites* on a specific type of syntactic phenomenon; for example, the `Agreement` circuit contains syntactic suites such as "subject-verb number agreement with a prepositional phrase". Each syntactic suite contains 20–30 syntactic *items* with different vocabulary; for example, the "subject-verb number agreement with a prepositional phrase" suite includes syntactic items as follows:

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

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

LMs' predictions are evaluated against *success criterion*, which specifies the inequality between conditions within an item; for example, the underlined position of the grammatical sentence (1a) should be assigned the higher conditional probability than the ungrammatical one (1b).

All models were trained and evaluated two times with different random seeds. We report average accuracies with a standard deviation, along with word-level perplexity on the BLLIP test set.

4.2 Overall accuracies

Table 2 shows the overall accuracies of TPTs and their baselines on the SyntaxGym benchmark (SG), along with word-level perplexity on the BLLIP test set (PPL). The overall accuracies were calculated across the syntactic suites. We also report the accuracies of several SLMs that were also trained on the same BLLIP-`LG` dataset: PLM, PLM-mask (Qian et al., 2021), and TG (Sartran et al., 2022). Only unmarked PLM and PLM-mask can be fairly comparable with TPTs as their evaluation was conducted generating a single syntactic structure via greedy search, to align inference costs with TPTs.[13] † and ‡ represent the reference points from Sartran et al. (2022) as their inference methods are more costly than TPTs: † and ‡ employed word-synchronous beam search (Stern et al., 2017) of action beam size 100[14] and the external parser (Dyer et al., 2016) to generate 300 candidate structures, respectively. The accuracies of several LLM-like models are also reported from Sartran et al. (2022): GPT-2 (Radford et al., 2018), Gopher (Rae et al., 2022), and Chinchilla (Hoffmann et al., 2022). They are also the reference points as these LLM-like models were trained on 250× to 1000× larger corpora (denoted by ¶). Perplexity can be directly comparable only within the same mark, either ♠ or ♡, due to differ-

	SG (↑)	PPL (↓)
Baselines:		
TPT`[zero]`	71.7 ± 0.3	47.5 ± 0.1♠
TPT`[rand.]`	69.0 ± 1.0	47.4 ± 0.1♠
TPT`[seq.]`	70.1 ± 3.5	**47.3 ± 0.2**♠
TPTs (ours):		
TPT`[dep.]`	**77.1 ± 0.2**	47.7 ± 0.1♠
TPT`[cons.]`	75.8 ± 0.0	**45.5 ± 0.0**♡
TPT`[bin.]`	73.0 ± 1.8	45.6 ± 0.2♡
SLMs (comparable):		
PLM	42.2 ± 1.2	-
PLM-mask	42.5 ± 1.5	-
SLMs (reference):		
PLM†	73.2 ± 0.6	49.3 ± 0.3♡
PLM-mask†	74.6 ± 1.0	49.1 ± 0.3♡
TG‡	82.5 ± 1.6	30.3 ± 0.5♡
LLM-like models (reference):		
GPT-2¶	78.4	-
Gopher¶	79.5	-
Chinchilla¶	79.7	-

Table 2: Overall accuracies of TPTs and their baselines on the SyntaxGym benchmark (SG), along with word-level perplexity on the BLLIP test set (PPL). The overall accuracies were calculated across the syntactic suites. † and ‡ represent the reference points as their inference methods are more costly than TPTs. ¶ are also the reference points as they were trained on significantly larger corpora than TPTs. Perplexity can be directly comparable only within the same mark, either ♠ or ♡, due to differences in the tokenization of the constituency parser and dependency parser.

[13] The fair comparison of TG was not performed because their trained parameters were not publicly available.

[14] Word beam size was 10 and fast track size was 5.

ences in the tokenization of the dependency parser and constituency parser.

There are some important observations in the overall accuracies on the SyntaxGym benchmark:

- TPT[zero], which is equivalent to a Transformer LM, underperformed all TPTs with some implicit syntactic supervision, suggesting that tree-planting can induce data-efficient syntactic generalization.
- TPTs[rand.][seq.] also underperformed all TPTs with some implicit syntactic supervision, indicating that not KL Divergence loss itself but the loss based on *syntactic structures* is necessary.
- Among TPTs with some implicit syntactic supervision, TPT[dep.] achieved the best performance. We further investigate this point in §4.3.
- Most importantly, despite the lack of explicit generation of syntactic structures, TPTs[dep.][cons.] significantly outperformed not only the comparable SLMs (unmarked PLM and PLM-mask) but also the various SLMs that generate hundreds of syntactic structures in parallel (PLM† and PLM-mask†).

Even though the best TPT[dep.] underperformed the reference points of TG, which consumed at least 300× higher inference cost, and LLM-like models, which were trained on at least 250× larger data, these observations adequately suggest that TPTs can learn human-like syntactic knowledge as data-efficiently as SLMs while maintaining the modeling space of Transformer LMs unchanged.

Regarding perplexity, although TPT[dep.] numerically underperformed its comparable baselines, they all achieved similar perplexity with no significant differences.

4.3 Circuit accuracies

In this subsection, we investigate the reason for the high compatibility between tree-planting and **dependency structures** through the lens of circuit accuracies. Figure 3 shows the circuit accuracies of TPTs with some implicit syntactic supervision and the baseline model with zero weight for the tree-planting loss on the SyntaxGym benchmark. The circuit accuracies calculated across the syntactic suites (the vertical axis) are plotted against the models (the horizontal axis), with each dot denoting the accuracy of a specific seed.

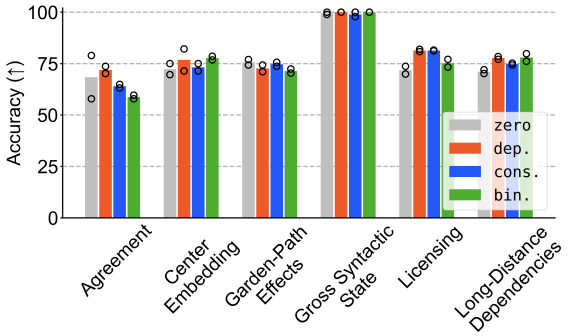


Figure 3: Circuit accuracies of TPTs with some implicit syntactic supervision and the baseline model with zero weight for the tree-planting loss on the SyntaxGym benchmark. The circuit accuracies calculated across the syntactic suites (the vertical axis) are plotted against the models (the horizontal axis), with each dot representing the accuracy of a specific seed.

vs. zero supervision TPT[dep.] outperformed TPT[zero] on 5/6 circuits, suggesting that tree-planting with dependency structures is generally advantageous over zero supervision. However, the Garden-Path Effects circuit presents an exception, where LMs are evaluated for the ability to be surprised in a human-like manner, through comparisons between sentences minimally different not in *grammaticality* but in *local ambiguity* (Hu et al., 2020). The underperformance of TPT[dep.] may suggest that due to the syntactic knowledge introduced by tree-planting with dependency structures, TPT[dep.] was no longer surprised by locally ambiguous but grammatical sentences. We further investigate this point in Appendix B.

vs. constituency structures Surprisingly, on 5/6 circuits, TPT[dep.] outperformed TPT[cons.]. The only exception is the Garden-Path Effects circuit, where the potential disadvantage of tree-planting with dependency structures exists, as mentioned above. Specifically, TPT[dep.] most significantly outperformed TPT[cons.] on the Agreement circuit, which includes the syntactic items such as (1) from §4.1: "The author next to the senator is/*are good". For these syntactic items, only the head of the subject NP (*author*) is always nearest to the main verb (*is*/*are*) on dependency structures, but the same does not hold on constituency structures: in constituency structures, the determiner of the subject NP (*the*) and

the head of the post-modifying PP (*to*) are as nearest to the main verb as the head of the subject NP (cf. Appendix C). Given that tree-planting utilizes the number of edges as implicit syntactic supervision, the property of dependency structures may potentially be more desirable for tree-planting than constituency structures.

vs. binarized constituency structures TPT[dep.] outperformed TPT[bin.] on 3/6 circuits, with similar performance (a difference less than -1.0%) on the other 3 circuits. Notably, TPT[dep.] achieved significantly better performance (a difference more than $+5.0\%$) on the Agreement and Licensing circuits. Noji and Oseki (2023) reported that deep syntactic supervision is not always optimal; rather mild syntactic supervision is sufficient for addressing long-distance dependencies between elements within and outside complex NP subjects. Given that (i) the Agreement and Licensing circuits consist only of syntactic suites that exemplify this condition[15] and (ii) the average syntactic distance in the training data is significantly shorter for dependency structures (4.8) than binarized constituency structures (13.1), it could be argued that tree-planting would be more "good enough" syntactic supervision with dependency structures, rather than with binarized constituency structures.[16]

5 Analysis

In this section, we report the effects of (i) the number of tree-planted heads and (ii) the weight of a tree-planting loss, using TPT[dep.].

5.1 Number of tree-planted heads

Our TPTs are based on a 12-layer, 12-head Transformer LM. In §4, out of 12×12 heads, we adopted a single attention head on the last layer as a tree-planted head. In this subsection, we explore two alternatives: (i) head-direction extension and (ii) layer-direction extension. For the head-direction extension, $H = 0, 1, 3, 6, 9, 12$ heads on the last layer were adopted as tree-planted heads. For the layer-direction extension, one attention head from each of the bottom $H = 0, 1, 3, 6, 9, 12$ layers was adopted as tree-planted heads.

In the left two columns of Figure 4, the results of the head-direction and layer-direction extension are shown: the overall accuracies on the SyntaxGym benchmark (SG) and the word-level perplexity on the BLLIP test set (PPL) (the vertical axis) are plotted against the number of tree-planted heads (the horizontal axis). Each dot denotes the accuracy or perplexity of a specific seed. For both settings, $H = 0, 1$ are equivalent to TPT[zero] and TPT[dep.], respectively.

Considering the overall accuracies on the SyntaxGym benchmark, in both the head-direction and layer-direction extension, the highest accuracy was achieved when only a single head was adopted as a tree-planted head, while it is noteworthy that all the models with tree-planted heads outperformed the model without them. Incidentally, it should be mentioned that the result of the layer-direction extension exhibited significantly more variability. Although the exact reason why a single tree-planted head would work well is unclear, the adoption of multi tree-planted heads inherently induces the handling of redundant information across heads, which might potentially hinder the management of non-syntactic information of natural languages (e.g., lexical information). Regarding perplexity, no consistent trend emerged.

5.2 Weight of a tree-planting loss

In §4, we adopted $\lambda = 0.5$ as the weight of the tree-planting loss. Here, we extend λ to 0.0, 0.25, 0.50, 0.75, and 1.00. $\lambda = 0, 0.50$ are equivalent to TPT[zero] and TPT[dep.], respectively.

The rightmost column of Figure 4 shows the results of the weight extension. The overall accuracies on the SyntaxGym benchmark display a single-peaked pattern, with the maximum reached for $\lambda = 0.50$. Interestingly, this result suggests that by overtly focusing on reflecting syntactic structures, TPTs paradoxically become unable to learn syntactic knowledge efficiently. On the other hand, we observed that the perplexity got worse monotonically as the weight increased. From these observations, we may deduce that to acquire syntactic knowledge efficiently, TPTs should learn not only to reflect syntactic structures in their attention weights but also to precisely predict the next word. Therefore, the weight of the tree-planting loss emerges as a critical hyperparameter, indicat-

[15] Among the other syntactic circuits, the Center Embedding circuit also consist only of syntactic suites that exemplify this condition.

[16] The average syntactic distance of constituency structures is 10.0. This suggests that tree-planting would also be more "good enough" syntactic supervision with dependency structures rather than with constituency structures, besides the points discussed in the "vs. constituency structures" paragraph.

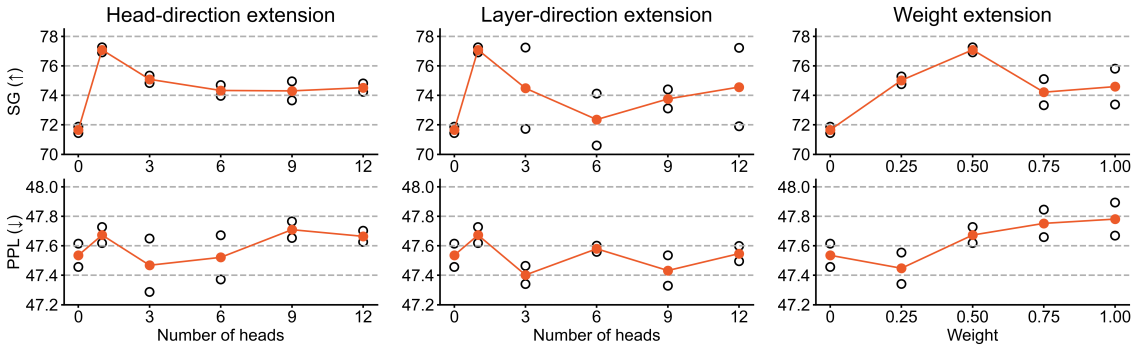


Figure 4: The results of the head-direction, layer-direction, and weight extension. For the head-direction and layer-direction extension, the overall accuracies on the SyntaxGym benchmark and the perplexity on the BLLIP test set (the vertical axis) are plotted against the number of tree-planted heads (the horizontal axis). For the weight extension, the horizontal axis indicates the weight of the tree-planting loss.

ing that the search for the optimal balance between the next-word prediction loss and tree-planting loss is vital for developing more human-like TPTs.

6 Conclusion

In this paper, we propose a new method dubbed **tree-planting**: instead of explicitly generating syntactic structures, we "plant" trees into attention weights of unidirectional Transformer LMs to implicitly reflect syntactic structures of natural language. Specifically, unidirectional Transformer LMs trained with tree-planting will be called **Tree-Planted Transformers (TPT)**, which inherit the training efficiency from SLMs without changing the inference efficiency of their underlying Transformer LMs. Targeted syntactic evaluations on the SyntaxGym benchmark demonstrated that TPTs, despite the lack of explicit generation of syntactic structures, significantly outperformed not only vanilla Transformer LMs but also various SLMs that generate hundreds of syntactic structures in parallel. This result suggests that TPTs can learn human-like syntactic knowledge as data-efficiently as SLMs while maintaining the modeling space of Transformer LMs unchanged.

Limitations

This paper has at least three limitations. First, we only conducted sentence-level tree-planting. Typically, Transformer LMs are trained at the document level, but SLMs are trained at the sentence level (Dyer et al., 2016; Kuncoro et al., 2017; Noji and Oseki, 2021; Yoshida and Oseki, 2022), because on treebanks the annotations are assigned at the sentence level. Because of this constraint, we also employed sentence-level experimental design and verified the effectiveness of the proposed method first and foremost. Recent research in SLMs, however, has begun to extend treebank annotations to the document level and train document-level SLMs on them (Sartran et al., 2022; Murty et al., 2023). When constructing TPTs for practical use, it might be beneficial to follow these recent studies and perform tree-planting with document-level annotations.

Second, we only evaluated TPTs on the syntactic knowledge benchmark and perplexity. Recently, Murty et al. (2023) evaluated the performance of SLMs on text classification tasks; to the best of our knowledge, this is the first work that evaluated SLMs on tasks other than the targeted syntactic evaluations. More recently, Hu et al. (2024) evaluated unsupervised SLMs on text generation tasks as well as text classification tasks. Murty et al. (2023) and Hu et al. (2024) both suggested that syntactic supervision could also be beneficial to solving them; this indicates that there is also room for a broader evaluation of our methodology.

Finally, there might still be room for further improvement of tree-planting and TPTs. For instance, an in-depth comparison between TPT and TG, which achieved much better accuracy on the `Center Embedding` and `Garden-Path Effects` circuits (cf. Sartran et al., 2022), could provide insights for improving the design of tree-planting. Additionally, since the modeling space of TPT is identical to that of underlying Transformer LMs, TPTs are theoretically capable of continual learning on standard text corpora. In future work, we plan to develop a novel method to scale TPTs on large text corpora without compromising syntactic knowledge.

Ethical considerations

A significant feature of TPT lies in the training efficiency and inference efficiency, which can potentially contribute to reducing computational resources. One minor concern is the possibility of bias in the models utilized in this paper, attributed to the training data (i.e., the BLLIP corpus), although this experimental setting follows conventional practices in the literature on SLMs. We employed ChatGPT and Grammarly for writing assistance and utilized ChatGPT and Copilot for the development of experimental code. These tools were used in compliance with the ACL 2023 Policy on the Use of AI Writing Assistance.

Acknowledgements

We appreciate the insightful reviews provided by the three anonymous ARR reviewers. We would also like to thank Peng Qian for supplying the reparsed BLLIP-LG dataset, which was used to train TPT[cons.]. Special thanks to Laurent Sartran for answering various questions regarding Sartran et al. (2022). We are grateful to Kohei Kajikawa, Shinnosuke Isono, Yushi Sugimito, and Taketeru Yamakoshi for their valuable comments and suggestions. This work was supported by JSPS KAKENHI Grant Number 24H00087, Grant-in-Aid for JSPS Fellows JP24KJ0800, JST PRESTO Grant Number JPMJPR21C2, and JST SPRING Grant Number JPMJSP2108.

References

Jiangang Bai, Yujing Wang, Yiren Chen, Yaming Yang, Jing Bai, Jing Yu, and Yunhai Tong. 2021. Syntax-BERT: Improving Pre-trained Transformers with Syntax Trees. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3011–3020, Online. Association for Computational Linguistics.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. " O'Reilly Media, Inc.".

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Emanuele Bugliarello and Naoaki Okazaki. 2020. Enhancing Machine Translation with Dependency-Aware Self-Attention. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1618–1627, Online. Association for Computational Linguistics.

Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, John Hale, and Mark Johnson. 2000. BLLIP 1987-89 WSJ Corpus Release 1.

Angelica Chen, Ravid Schwartz-Ziv, Kyunghyun Cho, Matthew L. Leavitt, and Naomi Saphra. 2023. Sudden Drops in the Loss: Syntax Acquisition, Phase Transitions, and Simplicity Bias in MLMs.

Benoit Crabbé, Murielle Fabre, and Christophe Pallier. 2019. Variable beam search for generative neural parsing and its relevance for the analysis of neuroimaging signal. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1150–1160, Hong Kong, China. 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.

Wenyu Du, Zhouhan Lin, Yikang Shen, Timothy J. O'Donnell, Yoshua Bengio, and Yue Zhang. 2020. Exploiting Syntactic Structure for Better Language Modeling: A Syntactic Distance Approach. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6611–6628, Online. 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.

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.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training Compute-Optimal Large Language Models.

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.

Xiang Hu, Pengyu Ji, Qingyang Zhu, Wei Wu, and Kewei Tu. 2024. Generative Pretrained Structured Transformers: Unsupervised Syntactic Language Models at Scale.

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.

Henry W. Lin and Max Tegmark. 2017. Critical Behavior in Physics and Probabilistic Formal Languages. *Entropy*, 19(7):299.

Youmi Ma, An Wang, and Naoaki Okazaki. 2023. DREEAM: Guiding Attention with Evidence for Improving Document-Level Relation Extraction. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1971–1983, Dubrovnik, Croatia. Association for Computational Linguistics.

Ines Montani, Matthew Honnibal, Matthew Honnibal, Adriane Boyd, Sofie Van Landeghem, and Henning Peters. 2023. Explosion/spaCy: V3.7.2: Fixes for APIs and requirements. Zenodo.

Aaron Mueller and Tal Linzen. 2023. How to plant trees in language models: Data and architectural effects on the emergence of syntactic inductive biases. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11237–11252, Toronto, Canada. Association for Computational Linguistics.

Shikhar Murty, Pratyusha Sharma, Jacob Andreas, and Christopher Manning. 2023. Pushdown Layers: Encoding Recursive Structure in Transformer Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3233–3247, Singapore. Association for Computational Linguistics.

Xuan-Phi Nguyen, Shafiq R. Joty, Steven C. H. Hoi, and Richard Socher. 2020. Tree-structured attention with hierarchical accumulation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

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.

Hiroshi Noji and Yohei Oseki. 2023. How Much Syntactic Supervision is "Good Enough"? In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2300–2305, Dubrovnik, Croatia. Association for Computational Linguistics.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

Hao Peng, Roy Schwartz, and Noah A. Smith. 2019. PaLM: A Hybrid Parser and Language Model. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3644–3651, Hong Kong, China. 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, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018. Language models are unsupervised multitask learners.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang,

Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Ed Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. 2022. Scaling Language Models: Methods, Analysis & Insights from Training Gopher.

Devendra Sachan, Yuhao Zhang, Peng Qi, and William L. Hamilton. 2021. Do Syntax Trees Help Pre-trained Transformers Extract Information? In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2647–2661, Online. Association for Computational Linguistics.

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.

Yikang Shen, Zhouhan Lin, Chin-Wei Huang, and Aaron C. Courville. 2018. Neural language modeling by jointly learning syntax and lexicon. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron C. Courville. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Aviv Slobodkin, Leshem Choshen, and Omri Abend. 2022. Semantics-aware Attention Improves Neural Machine Translation. In *Proceedings of the 11th Joint Conference on Lexical and Computational Semantics*, pages 28–43, Seattle, Washington. 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.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-Informed Self-Attention for Semantic Role Labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038, Brussels, Belgium. Association for Computational Linguistics.

Georgios Tziafas, Konstantinos Kogkalidis, Gijs Wijnholds, and Michael Moortgat. 2023. Improving BERT pretraining with syntactic supervision. In *Proceedings of the 2023 CLASP Conference on Learning with Small Data (LSD)*, pages 176–184, Gothenburg, Sweden. Association for Computational Linguistics.

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 *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Yaushian Wang, Hung-Yi Lee, and Yun-Nung Chen. 2019. Tree Transformer: Integrating Tree Structures into Self-Attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1061–1070, Hong Kong, China. Association for Computational Linguistics.

Alex Warstadt, Aaron Mueller, Leshem Choshen, Ethan Wilcox, Chengxu Zhuang, Juan Ciro, Rafael Mosquera, Bhargavi Paranjabe, Adina Williams, Tal Linzen, and Ryan Cotterell. 2023. Findings of the BabyLM Challenge: Sample-Efficient Pretraining on Developmentally Plausible Corpora. In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 1–34, Singapore. 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.

Wei Wu, Houfeng Wang, Tianyu Liu, and Shuming Ma. 2018. Phrase-level Self-Attention Networks for Universal Sentence Encoding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3729–3738, Brussels, Belgium. Association for Computational Linguistics.

Pengcheng Yin, Hao Fang, Graham Neubig, Adam Pauls, Emmanouil Antonios Platanios, Yu Su, Sam Thomson, and Jacob Andreas. 2021. Compositional Generalization for Neural Semantic Parsing via Span-level Supervised Attention. In *Proceedings of the*

2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 2810–2823, Online. Association for Computational Linguistics.

Ryo Yoshida and Yohei Oseki. 2022. Composition, Attention, or Both? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5822–5834, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, and Guoyin Wang. 2024. Instruction Tuning for Large Language Models: A Survey.

Optimizer	AdamW
Learning rate	5e-5
Number of epochs	10
Dropout rate	0.1
Batch size	256

Table 3: Hyperparameters for our experiments

A Hyperparameters

Hyperparameters for our experiments are shown in Table 3, which primarily followed default settings. All models were trained and evaluated on 8× NVIDIA V100 (16GB). The total computational cost for all experiments in this paper amounted to about 1,300 GPU hours.

B Further investigation of the `Garden-Path Effects` circuit

In §4.3, we suggest the probability that syntactic knowledge introduced by tree-planting with dependency structures may prevent TPT`[dep.]` from being surprised by locally ambiguous but grammatical sentences. To inspect this, we break down the `Garden-Path Effects` circuit into the syntactic suites: "main verb / reduced relative clause" (MVRR) and "NP/Z garden-paths" (NP/Z).

Figure 5 shows the suite accuracies of TPTs with some implicit syntactic supervision and the baseline model with zero weight for the tree-planting loss on the `Garden-Path Effects` circuit, with the reference point of the more inference-costly SLM, or PLM-mask† (Qian et al., 2021). We find that the deficiency of TPT`[dep.]` is attributed to its inadequate performance on the MVRR circuit, which includes the syntactic items as follows:

(2) a. The dog seen on the beach <u>chased</u> after a bird.

b. !The dog walked on the beach <u>chased</u> after a bird.

The success criterion on these suites defines that the underlined position of the unambiguous sentence (2a) should be assigned a higher conditional probability than the locally ambiguous one (2b). We speculate that TPT`[dep.]` might lose its sensitivity to the local ambiguity introduced by the participle verb (*seen/walked*), as it is guided to focus more intently on the head of the subject NP (*dog*) when predicting the main verb (*chased*), than the unrestricted baseline.

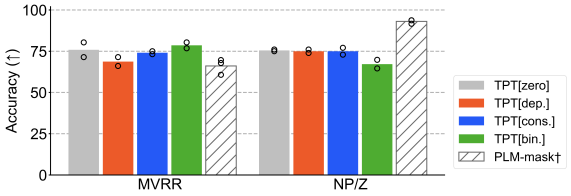


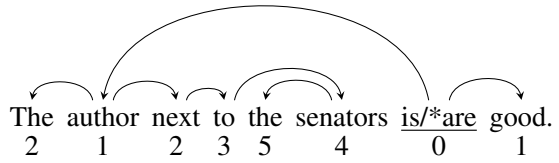
Figure 5: Suite accuracies of TPTs with some implicit syntactic supervision and the baseline model with zero weight for the tree-planting loss on the `Garden-Path Effects` circuit, with the reference point of the more inference-costly SLM, or PLM-mask† (Qian et al., 2021)

Conversely, TPT`[cons.][bin.]` did not underperform TPT`[zero.]` on the MVRR suites. This result could be straightforwardly understood, given that on these structures, the participle verb (*seen/walked*) and the head of the subject NP (*dog*) are equidistant from the main verb (*chased*). However, it is worth noting that the determiner of the subject NP (*the*) also shares this distance, which may not always be a desirable property for tree-planting (cf. §4.3).

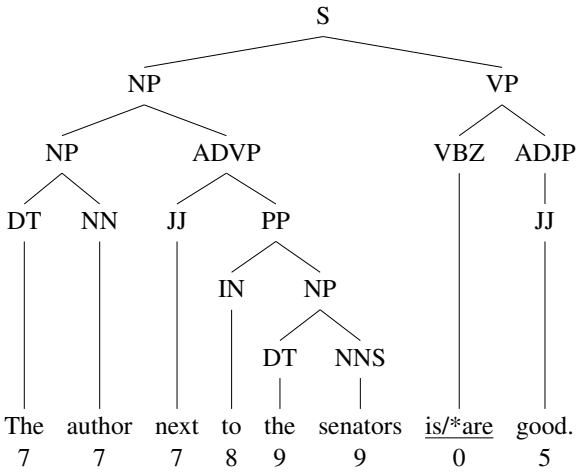
Finally, PLM-mask†, the more inference-costly SLM, also underperformed TPT`[zero]` on the MVRR suites. This suggests that the models with explicit syntactic supervision may also struggle with losing sensitivity to the local ambiguity as PLM`[dep.]`.

C Dependency/constituency structures of (1) from §4.1

To assist the discussion in §4.3, the dependency and constituency structures of (1) from §4.1 were displayed in Figure 6a and 6b, respectively. Numbers below each word represent the number of edges from the underlined position. To parse (1), the parsers referenced in §4.1 were employed.



(a) Dependency structure



(b) Constituency structure

Figure 6: Dependency/constituency structures of (1) from §4.1

D Begin/End Of Sentence Tokens

Sentences in the BLLIP corpus do not include Begin/End of Sentence (BOS/EOS) tokens, which are essential for sequences processed by LMs. To integrate these tokens, we implemented the following modifications:

- For dependency structures, we introduced BOS/EOS tokens by defining new edges from the ROOT to these tokens.
- For constituency structures, we introduced BOS/EOS tokens by modifying the tree structure to encapsulate the original structure within a new root node, specifically by adding a BOS token and an EOS token as the first and the last child of this new root, respectively.

E License of the data/tools

We summarize the license of the data/tools employed in this paper in Table 4. All data and tools were used under their respective license terms.

Data/tool	License
BLLIP (Charniak et al., 2000)	BLLIP 1987-89 WSJ Corpus Release 1 License Agreement
SyntaxGym (Gauthier et al., 2020)	MIT
`spacy` (Montani et al., 2023)	MIT
`nltk` (Bird et al., 2009)	Apache 2.0
`transformers` (Wolf et al., 2020)	Apache 2.0
Berkeley Neural Parser (Kitaev and Klein, 2018)	MIT
PLM/PLM-mask (Qian et al., 2021)	MIT

Table 4: License of the data/tools