Transformer Grammars: Augmenting Transformer Language Models with Syntactic Inductive Biases at Scale

Laurent Sartran[1] **Samuel Barrett**[2] **Adhiguna Kuncoro**[1,2]
Miloš Stanojević[1] **Phil Blunsom**[2] **Chris Dyer**[1]
[1]DeepMind, UK [2]University of Oxford, UK
{lsartran,akuncoro,stanojevic,cdyer}@deepmind.com
samuelbarrett1234@btinternet.com, phil.blunsom@cs.ox.ac.uk

Abstract

We introduce Transformer Grammars (TGs), a novel class of Transformer language models that combine (i) the expressive power, scalability, and strong performance of Transformers and (ii) recursive syntactic compositions, which here are implemented through a special attention mask and deterministic transformation of the linearized tree. We find that TGs outperform various strong baselines on sentence-level language modeling perplexity, as well as on multiple syntax-sensitive language modeling evaluation metrics. Additionally, we find that the recursive syntactic composition bottleneck which represents each sentence as a single vector harms perplexity on document-level language modeling, providing evidence that a different kind of memory mechanism—one that is independent of composed syntactic representations—plays an important role in current successful models of long text.

1 Introduction

Transformer language models (LMs) that are trained on vast amounts of data have achieved remarkable success at various NLP benchmarks (Peters et al., 2018; Devlin et al., 2019; Brown et al., 2020, *inter alia*). Intriguingly, this success is achieved by models that lack an explicit modeling of hierarchical syntactic structures, which were hypothesized by decades of linguistic research to be necessary for good generalization (Chomsky, 1957; Everaert et al., 2015). This naturally leaves a question: To what extent can we *further improve* the performance of Transformer LMs, through an inductive bias that encourages the model to explain the data by the means of recursive syntactic compositions? Although the benefits of modeling recursive syntax have been shown at the small data and model scales, such as in the case of recurrent neural network grammars (Dyer et al., 2016; Futrell et al., 2019; Kim et al., 2019; Hu et al., 2020; Noji and Oseki, 2021, *inter alia*), it remains an open question whether—and to what extent—a similar design principle is still beneficial for Transformer LMs at larger scales.

In this paper, we aim to answer these questions by introducing **Transformer Grammars** (TGs)—a novel class of Transformer language models that combine: (i) the expressive power, scalability, and strong performance of Transformer-XL (Dai et al., 2019); (ii) joint modeling of surface strings $\boldsymbol{x}$ and their corresponding phrase-structure trees $\boldsymbol{y}$, namely, $p(\boldsymbol{x}, \boldsymbol{y})$; and (iii) an inductive bias that constrains the model to explain the data through built-in recursive syntactic composition operations. By implementing these recursive compositions through a novel modification of the Transformer-XL attention mask, TGs retain the computational efficiency of standard Transformer-XLs, enabling them to avoid the limitations of LSTM-based recurrent neural network grammars (Dyer et al., 2016, RNNGs), which have been proven difficult to scale (Kuncoro et al., 2019; Noji and Oseki, 2021).

TGs are related to the recent work of Qian et al. (2021) that similarly aims to augment generative Transformer language models with a stronger modeling of syntactic structures, albeit with two key differences. First, whereas Qian et al. (2021) used syntactic structure to restrict the behavior of a subset of attention heads (Strubell et al., 2018; Astudillo et al., 2020), TGs incorporate a stronger form of syntactic inductive bias by using recursive syntactic compositions to create an explicit composed representation for each constituent, in a similar fashion as RNNGs. Hence, our approach sheds light into whether, and to

Transactions of the Association for Computational Linguistics, vol. 10, pp. 1423–1439, 2022. https://doi.org/10.1162/tacl_a_00526
Action Editor: David Chiang. Submission batch: 4/2022; Revision batch: 8/2022; Published 12/2022.

what extent, the recursive syntactic composition hypothesis—which has been shown to be valuable at the small data and model scale in the case of RNNGs—continues to offer additional benefits, beyond specializing a subset of attention heads for syntax. Second, in contrast to prior work, which has been limited to modeling sentences independently of document context, this work explores whether syntactic composition also benefits *document-level* language modeling.

We evaluate TGs against baseline models on three metrics: (i) perplexity, (ii) syntactic generalization, and (iii) parse reranking, and on two training datasets: (i) the small-scale Penn Treebank (Marcus et al., 1993, PTB) and (ii) the medium-scale BLLIP-LG (Charniak et al., 2000) datasets, with ∼1M and ∼40M words, respectively. We find that:

- Transformer Grammars (**TGs**) achieve better (i) single-sentence language modeling perplexity, (ii) syntactic generalization, and (iii) parse reranking performance than RNNGs—all the while being much faster to train than the batched RNNG of Noji and Oseki (2021).
- In single-sentence language modeling perplexity, the terminal-only Transformer XL baseline (**TXL (terminals)**) is outperformed by both TGs and a Transformer XL that predicts sentences as joint sequences of terminals and tree-building nonterminal symbols (**TXL (trees)**) but *without* the TG's attention restrictions (Choe and Charniak, 2016; Qian et al., 2021).
- Although modeling structure improves perplexity compared to terminal-only models, the TXL (trees) model slightly outperforms the more biased TG models. Using a regression analysis, we show that while TGs' recursive syntactic compositions benefit syntactic generalization, their implementation in terms of attention restriction interferes with Transformers' lexical copying ability, which turns out to play a role in obtaining low perplexities. This result indicates a partial dissociation between perplexity and syntactic generalization, both of which are important metrics for assessing LM success.
- On the benchmark of Hu et al. (2020) that is a carefully controlled test of syntactic generalization ability, TGs substantially outperform the syntax-free TXL (terminals) baseline, as well as the much stronger TXL (trees) model. Perhaps even more remarkably, TGs outperform the GPT-2-small (Radford et al., 2019), Gopher (Rae et al., 2021), and Chinchilla (Hoffmann et al., 2022) models, which are between $250\times$ and $1,000\times$ larger than the TG model, trained on vastly more data, and arguably represent the most sophisticated LMs in existence.
- When modeling full documents and evaluating perplexity, we again find that the TXL (trees) model outperforms the terminal-only TXL (terminals) baseline. However, TGs substantially underperform both the TXL (terminals) and TXL (trees) models. The failure of TGs, which represent prior-sentence context purely in terms of a composed syntactic representation, suggests that a different memory mechanism—that works in part independently of syntactic structures—may play a role in the processing of long-form text.

All in all, our findings show that—under comparable experimental conditions—LMs *with* notions of syntactic structures (both TXL (trees) & TG) outperform those *without* on multiple evaluation metrics. We further demonstrate that encouraging the model to explain the data through the means of recursive syntactic compositions—as is the case for TGs—is a valuable inductive bias for achieving an *even stronger* human-like syntactic competence, outperforming prior work that also incorporates syntactic biases, albeit without recursive compositions (Qian et al., 2021), in addition to some of the largest non-syntactic LMs to date. Lastly, our findings motivate the development of *scalable* LMs—that nevertheless incorporate stronger notions of syntactic structures—as a promising (albeit relatively under-explored) area of NLP research.

2 Model

TGs are syntactic language models: They jointly model the probability of syntactic phrase-structure trees $\boldsymbol{y}$ and strings of words $\boldsymbol{x}$, using the predicted structures to determine the structure of the computations of model states. Following a line

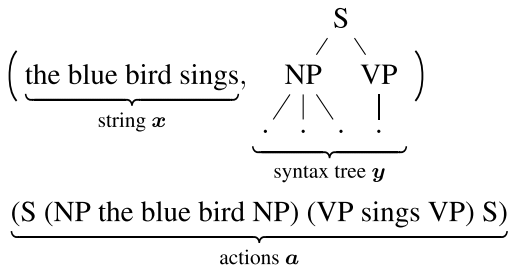


Figure 1: An example that represents a pair of string $\boldsymbol{x}$ and its phrase-structure tree $\boldsymbol{y}$, which are then represented as a sequence of actions that construct ($\boldsymbol{x}$, $\boldsymbol{y}$) in a top-down, left-to-right fashion (Dyer et al., 2016; Choe and Charniak, 2016).

of recent work in parsing and syntactic language models (Vinyals et al., 2015; Dyer et al., 2016; Choe and Charniak, 2016), the generation problem is decomposed into modeling a sequence of *actions* that construct ($\boldsymbol{x}$, $\boldsymbol{y}$) in a top-down, left-to-right fashion, by interleaving nonterminal nodes and their children, as shown in Figure 1. The linearized representation of ($\boldsymbol{x}$, $\boldsymbol{y}$) consists of three types of actions: (i) opening nonterminals (action type `ONT`), marking the opening of a new constituent; (ii) generating terminal symbols/leaf nodes (*i.e.,* words or subword tokens), henceforth denoted as `T`; and (iii) closing the most recent open constituent/incomplete nonterminal symbol, henceforth denoted as `CNT`.

Let $\boldsymbol{a} = (a_0, a_1, \ldots, a_{T-1})$ be a sequence of actions (of length T) that generates ($\boldsymbol{x}$, $\boldsymbol{y}$), where each action is part of the action vocabulary $\mathcal{V}$. TGs define a probability distribution over $\boldsymbol{a}$ through a left-to-right factorization, that is, $p(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{a}) = \prod_i p(a_i \mid \boldsymbol{a}_{<i})$.

2.1 Recursive Syntactic Composition via Attention

In Transformer language models, when generating a_i conditionally on $\boldsymbol{a}_{<i}$, attention is the only mechanism by which information from other positions $j < i$ is incorporated. The rules governing this information flow (*i.e.,* which positions can attend to which other positions) are defined by the *attention mask*. We design TGs to use recursive syntactic compositions, which have been shown to lead to better generalization in the LSTM-based RNNG model, and we implement them through the Transformer attention mechanism.

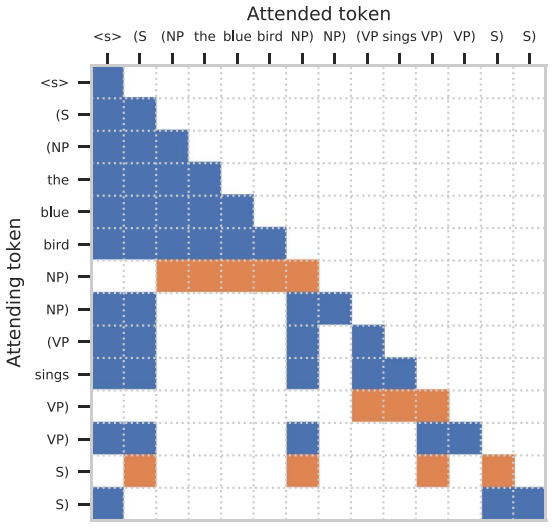
In TGs, the action sequence is generated from left-to-right, and each symbol a_i can be thought of as updating a stack of indices. When the current a_i is a closing nonterminal (*i.e.*, a constituent has just ended) its index i will be represented by a single-vector-sized *composed* representation obtained by attending to the child positions of the currently ending constituent. Subsequent positions ($> i$) may attend to this composed position, but they may not attend directly to the constituent positions, and this restriction imposes a syntactic bottleneck, since everything inside the constituent that influences subsequent predictions must become part of the composed representation. This bottleneck encourages the model to learn informative representations of composed phrases and is inspired by a similar design principle as RNNGs and other tree-structured architectures. In the stack, the restriction is instantiated by popping the indices for the child nodes and then pushing the index of the composed constituent. We refer to this process as COMPOSE attention.

In addition to COMPOSE attention, at each position i, we apply STACK attention, where i is pushed onto the stack attention, and attention is restricted to positions on the stack. Both STACK and COMPOSE attention use the same parameters and attention heads—what distinguishes them is only the rule for computing the set of positions that the model can attend to. Importantly, as we need to (i) perform *both* COMPOSE and STACK for a closing nonterminal (*e.g.,* to first compute a composed representation based on its parts/children, and then add the composed representation onto the stack), while (ii) performing exactly one attention operation per token, we transform the original sequence $\boldsymbol{a}$ by duplicating all closing nonterminals. This yields a sequence $\boldsymbol{a}'$ of length T', for example, `(S (NP the blue bird NP) NP) (VP sings VP) VP) S) S)`. The first closing nonterminal of each pair is given the type `CNT1`, and implements COMPOSE, whereas the second is given the type `CNT2`, and implements STACK. To keep the number of prediction events (*i.e.,* the number of times a probability distribution is emitted by the model) constant, no final prediction is made for COMPOSE positions (see Figure 2).

The exact procedure for STACK/COMPOSE is described in Algorithm 1. The positions that may be attended are represented as a binary attention mask $\mathbf{A} \in \mathbb{R}^{T' \times T'}$ (see Figure 2), such that $A_{ij} = 1$ *iff* the position j may be attended from i,

i	Input a'_i	Type	Attn. op.	Label
0	<s>	ONT	STACK	(S
1	(S	ONT	STACK	(NP
2	(NP	ONT	STACK	the
3	the	T	STACK	blue
4	blue	T	STACK	bird
5	bird	T	STACK	NP)
6	NP)	CNT1	COMPOSE	–
7	NP)	CNT2	STACK	(VP
8	(VP	ONT	STACK	sings
9	sings	T	STACK	VP)
10	VP)	CNT1	COMPOSE	–
11	VP)	CNT2	STACK	S)
12	S)	CNT1	COMPOSE	–
13	S)	CNT2	STACK	–

(a) Example of a (transformed) sequence with its corresponding token types, type of attention operations, and labels. No prediction is done for positions 6, 10, 12 (where COMPOSE is performed), nor for position 13 as no end-of-sequence token is required to model linearized trees.



(b) Attention mask with STACK/COMPOSE attention. STACK is represented in blue, and COMPOSE is de-noted in orange.

Figure 2: Processing of an example sentence: (S (NP the blue bird NP) (VP sings VP) S).

Algorithm 1 STACK/COMPOSE attention

Input: $\boldsymbol{a}'$ sequence of tokens
Output: $\mathbf{A} \in \mathbb{R}^{T' \times T'}$ attention mask

$S \leftarrow []$ ▷ Empty stack
$\mathbf{A} \leftarrow 0$
for $i \leftarrow 0$ to T' **do**
  if type$(\boldsymbol{a}'[i]) =$ CNT1 **then** ▷ COMPOSE
    $j \leftarrow i$
    while type$(\boldsymbol{a}'[j]) \neq$ ONT **do**
      $A_{ij} \leftarrow 1$
      $j \leftarrow S.pop()$
    end while
    $A_{ij} \leftarrow 1$
    $S.push(i)$
  else ▷ STACK
    if type$(\boldsymbol{a}'[i]) \neq$ CNT2 **then**
      $S.push(i)$
    end if
    for $j \in S$ **do**
      $A_{ij} \leftarrow 1$
    end for
  end if
end for
return A ▷ Attention mask

and 0 otherwise. Note that the computation of the attention mask is causal, that is, no information from positions $j > i$ is used to compute the positions that can be attended from i.

Relative Positional Encoding. In Transformer-XL, the positional information presented to the model is based on the difference between the attending position i and the attended position j (*i.e.*, $i - j$). This distance does not reflect nor use the topology of the tree. We thus generalize how relative positions are provided to the attention mechanism such that any matrix $\mathbf{R} \in \mathbb{Z}^{T' \times T'}$ can be used, where R_{ij} is the relative position between i and j. For TGs, we define $R_{ij} = \delta(i) - \delta(j)$, where $\delta(i)$ is the depth of the i-th token in the tree. Note that the relative distance R_{ij} will only be computed if $A_{ij} = 1$ (*i.e.*, j may be attended from i). For instance, for the action sequence in Figure 1, the relative distance between (the positions corresponding to) the words `sings` and `bird` is never computed, but it will be computed between `sings` and its sibling NP covering `the blue bird`.

2.2 Segmentation and Recurrence

In the same manner as Transformer-XL, TGs are recurrent neural networks that can process arbitrarily long sequences[1] as consecutive segments

[1]This desirable property of Transformer-XL is the main justification for our using it as baseline and starting point.

that contain a fixed number of tokens L, maintaining and updating a memory of temporal dimension M from one segment to the next. With $0 \leq \tau \leq \lceil \frac{T'}{L} \rceil$, $\boldsymbol{a}'_\tau = (a_{\tau L}, a_{\tau L+1}, \ldots, a_{\tau(L+1)-1})$ is the $\tau+1$-th segment. Token embeddings are obtained from an embedding matrix $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}| \times d}$ to form a sequence of L vectors in $\mathbb{R}^d$: $\mathbf{h}_\tau^{(0)} = \left(h_{\tau L}^{(0)}, \ldots, h_{\tau(L+1)-1}^{(0)}\right)$.

The core of the model is composed of K stacked recurrent layers, that is, for $1 \leq k \leq K$:

$$\mathbf{h}_\tau^{(k)}, \mathbf{m}_{\tau+1}^{(k)} = \text{Layer}^{(k)}(\mathbf{h}_\tau^{(k-1)}, \mathbf{m}_\tau^{(k)}, \mathbf{A}_\tau, \mathbf{R}_\tau)$$

where for each segment τ:

- $\mathbf{h}_\tau^{(k)} \in \mathbb{R}^{L \times d}$ is the sequence of hidden states, which forms the input for layer $k+1$,
- $\mathbf{m}_\tau^{(k)} \in \mathbb{R}^{M \times d}$ is the memory,
- $\mathbf{A}_\tau \in \mathbb{R}^{L \times (M+L)}$ is the attention mask from the current segment to the current segment and the memory,
- and $\mathbf{R}_\tau \in \mathbb{Z}^{L \times (M+L)}$ is the corresponding relative positions matrix.

All layers receive the same attention mask and relative positions matrix. Each layer k is composed of a multi-head self-attention (SelfAttn) sub-layer and a position-wise feed-forward network (FFN) sub-layer (with residual connections followed by layer normalization—omitted for clarity), as well as an update to the memory for the next segment:

$$\begin{aligned}
\mathbf{h}_\tau^{(k-\frac{1}{2})} &= \text{SelfAttn}_k(\mathbf{h}_\tau^{(k-1)}, \mathbf{m}_\tau^{(k)}, \mathbf{A}_\tau, \mathbf{R}_\tau) \\
\mathbf{h}_\tau^{(k)} &= \text{FFN}_k(\mathbf{h}_\tau^{(k-\frac{1}{2})}) \\
\mathbf{m}_{\tau+1}^{(k)} &= \text{MemoryUpdate}(\mathbf{h}_\tau^{(k-1)}, \mathbf{m}_\tau^{(k)})
\end{aligned}$$

The output of the last layer, $\mathbf{h}_\tau^{(K)}$, is multiplied by the transpose of the embedding matrix $\mathbf{E}^T$ to get the unnormalized next-token log probabilities.

Self-attention. Using the notation of Dai et al. (2019), let $\mathbf{W}_q$, $\mathbf{W}_{k,E}$, $\mathbf{W}_{k,R}$, $\mathbf{W}_v$, and u and v be the trainable model parameters. Let $[\cdot,\cdot]$ denote a concatenation operation along the time dimension. For a single head, we have:

$$\mathbf{q} = \mathbf{h}\mathbf{W}_q \quad \mathbf{k} = [\mathbf{m}, \mathbf{h}]\,\mathbf{W}_{k,E} \quad \mathbf{v} = [\mathbf{m}, \mathbf{h}]\,\mathbf{W}_v.$$

The attention score for an attending position i and an attended position j is

$$s_{ij} = (\mathbf{q}_i + u)^T \mathbf{k}_j + (\mathbf{q}_i + v)^T \mathbf{r}_{ij},$$

where $\mathbf{r}_{ij} \in \mathbb{R}^d$ is an embedding of the integer relative position R_{ij} (row from $\mathbf{W}_{k,R}$). Much like in Transformer-XL, the second term can be computed efficiently as the relative positions take values within a small interval $[R_{\min}, R_{\max}]$.

The mask $\mathbf{A}$ (§2.1) is applied element-wise on the scores, which sets masked entries to $-\infty$. The normalized attention weights are then obtained by applying a softmax activation function to the scores; the final attention outputs are the product of the attention weights and the values. In practice, we use multiple heads—the outputs of each are concatenated and passed onto a linear transformation.

Memory Update. In Transformer-XLs, the memory is updated by shifting the current input into it. Here we take advantage of the fact that positions within a subtree that have been COMPOSEd are never attended to in the future, and *a fortiori* in the following segments. Hence, only positions that may be attended need to be added or kept in the memory. This requires careful book-keeping of which position in the memory corresponds to which original position in the input sequence, both (i) to perform the update, and (ii) to compute the correct attention mask and relative positions.

2.3 Properties

Recursive Composition. Transformer Grammars accomplish recursive compositions via a custom attention mask that reflects the hierarchical phrase structures within natural language. Although the mask at a position $i+1$ depends on the mask at position i, during training the entire attention mask matrix can be precomputed in advance, and then applied independently to compute multiple syntactic compositions in parallel for the whole segment. For instance, in the example sequence from Figure 2, during training the representations of NP and VP are computed in parallel, even though their closing nonterminals are at different positions (6 and 10, respectively) in the sequence. Every following layer of TGs then takes the composed representations at

previous layers, and composes them further. For instance, at position 12, the second layer will form a composed representation of a sentence constituent `S)` by using as input the first layer representations of `NP)` and `VP)`. A consequence of this approach is that at least d layers are needed for tokens of depth d to affect the topmost composed representation, a property it shares with conventional Transformers applied to trees (Hahn, 2020).

Context-modulated Composition. TGs' composition steps use a COMPOSE attention mask at each closing nonterminal of type `CNT1`, and all other actions use a STACK attention mask. The stack mask makes available the representations of the completed constituents, words, and open nonterminals on the stack. Thus, in the example in Figure 2, the word `sings` can attend to the closed constituent `NP)`, as well as ancestor nonterminals `(S` and `(VP`. But, importantly, at `sings`, information about all preceding words is accessible only through the composed `NP)` representation, thus enforcing the *compressive* effect of syntactic composition.

At higher layers, STACK and COMPOSE attentions have a subtle interaction worth making explicit. The STACK attention that is used to compute the representation for `sings` can look at the composed representation of the preceding subject `NP)`, meaning that a certain amount of ''outside information'' can enter into the computation of the composed VP. The availability of outside information deviates from the strict bottom-up compositionality of RNNGs and similar models.

How does this outside information impact composition? In TGs (in contrast to Transformer-XLs), the influence of outside context on composed representations is indirect, and we therefore argue that the TG learner has a bias against capturing such outside information in the composed representation. Our argument relies on two facts: (i) that learning to compose a representation of a constituent ending at position i is driven by predictions/prediction failures of a subsequent symbol a_j, where $j > i$ and (ii) that if a_j's prediction *does* crucially depend information outside of the constituent ending at i, then there will *always* be a more direct attention path than the one via the composed representation at a_i. The existence of two paths with different numbers of operations—a more direct one (directly via attention) and a less direct one (via composition followed by attention)—explains the bias against including outside information in composed representations, and in favor of bottom-up information.

Finally, we remark that questions of whether and how contextual information plays a role in composition are complex and unresolved. Bowman et al. (2016) showed that allowing outside information to modulate compositional computations leads to better composed representations, and justified this design on the grounds that outside information may play a crucial disambiguating role in the composition function.

3 Experiments

We compare TGs with two Transformer-XL baselines: (i) one trained only on the terminal word sequences (**TXL (terminals)**), and (ii) another trained on the linearized tree sequence as done by Choe and Charniak (2016), henceforth denoted as **TXL (trees)**. We remark that model (i) is a word-level language model that estimates the probability of surface strings $p(\boldsymbol{x})$, whereas model (ii) is a syntactic language model that estimates $p(\boldsymbol{x}, \boldsymbol{y})$. We additionally compare against two prior syntactic LMs: (i) the ''generative parsing as language modeling'' approach of Qian et al. (2021), which operates in a similar fashion as the linearized Choe and Charniak (2016) baseline, albeit with two attention heads that are specialized for syntax (though differently from TGs' explicit recursive syntactic compositions); and (ii) the batched RNNG model of Noji and Oseki (2021).

Datasets. We conduct experiments on both the Penn Treebank (Marcus et al., 1993, PTB) dataset ($\approx$ 1M words), and the BLLIP-LG (Charniak et al., 2000) dataset according to the split by Hu et al. (2020) ($\approx$ 40M words). We use the parsed, pre-processed, sentence-level PTB dataset of Dyer et al. (2016), where unseen words and singletons on the training set are mapped according to a special set of unknown word symbols as proposed by Petrov and Klein (2007). For BLLIP-LG, we use the parse trees provided by Hu et al. (2020). Tokenization is performed with SentencePiece (Kudo and Richardson, 2018) using a unigram language model subword algorithm (Kudo, 2018) and a vocabulary of 32K word-pieces.

For BLLIP-LG, we consider two settings: (i) we model each sentence independently and (ii) we model each document—each of which is composed of multiple sentences—independently.

Experimental Details. To account for training variance, for each model (TGs, TXL (terminals), and TXL (trees)), we train 100 models of the same size with independent random initializations. On PTB, we use 16-layer models with 12M parameters; whereas on BLLIP-LG, we use 16-layer models with 252M parameters. We select for each training run the model checkpoint with the lowest validation loss, computed using with a single gold proposal tree for each sentence.

3.1 Language Modeling Perplexity

Experimental Setup Whereas the probability of a string $\boldsymbol{x}$ can be computed directly by left-to-right decomposition for models operating on strings, for models operating on the joint distribution of strings and syntax trees, we *define* $p(\boldsymbol{x})$ as the marginal distribution: $p(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}}} p(\boldsymbol{x}, \boldsymbol{y})$, where $\mathcal{Y}_{\boldsymbol{x}}$ is the set of possible trees for $\boldsymbol{x}$. As the cardinality of this set is infinite, exact computation of this probability is intractable. However, we can compute a lower bound on $p(\boldsymbol{x})$ by approximately marginalizing over a much smaller set of proposal trees $\mathcal{Y}'_{\boldsymbol{x}} = \{\boldsymbol{y}^{(1)}, \cdots, \boldsymbol{y}^{(N)}\} \subset \mathcal{Y}_{\boldsymbol{x}}$.

For a given $\boldsymbol{x}$, we would want $\mathcal{Y}'_{\boldsymbol{x}}$ to be the set of trees for which $p(\boldsymbol{y} \mid \boldsymbol{x})$ is largest. As this parsing distribution is unavailable, we approximate it with a proposal model $q(\boldsymbol{y} \mid \boldsymbol{x})$. The better this approximation is, the tighter the upper lower bound on $p(\boldsymbol{x})$ is. We use as $q(\boldsymbol{y} \mid \boldsymbol{x})$ a separately trained discriminative RNNG, and $\mathcal{Y}'_{\boldsymbol{x}}$ is a set of $N = 300$ trees, sampled *without replacement*, as an approximation to the set of N trees with largest $p(\boldsymbol{y} \mid \boldsymbol{x})$. Naturally, regardless of how $\mathcal{Y}'_{\boldsymbol{x}}$ is chosen, the approximate marginal $\sum_{\boldsymbol{y} \in \mathcal{Y}'_{\boldsymbol{x}}} p(\boldsymbol{x}, \boldsymbol{y})$ computed on a subset of $\mathcal{Y}_{\boldsymbol{x}}$ is a lower bound of the true probability $p(\boldsymbol{x})$.

We compute the word perplexity of the validation and test splits of the datasets under the models as $\text{PPL}(\mathcal{D}) = \left(\prod_{\boldsymbol{x} \in \mathcal{D}} p(\boldsymbol{x})\right)^{-\frac{1}{N_w}}$, where N_w is the total number of words in the dataset D. It is exact for the models operating on words, and a conservative approximation (an upper bound) for the models operating on the joint distribution of strings and syntax trees.

For the document-level language models, given a document that consists of N_s sentences, for each sentence i in the document, we need to marginalize over all possible syntax trees for every single $i-1$ *preceding* sentence in that document. We approximate this by greedily picking the single most likely syntax tree under the model for the first $i-1$ sentences, before concatenating this single-path prefix with the 300 tree proposals for the last sentence.

Discussion We report the mean and sample standard deviation of perplexity (first 3 columns) in Table 1, and plot their distributions in Figure 3.

Although all three models share the exact same number of model parameters and training dataset, our very first observation is that *both* the Transformer-XL baseline trained on the linearized tree sequences (TXL (trees)) and the proposed TG model achieve a lower perplexity—even though the reported perplexity is in fact only an upper bound—than a Transformer-XL trained only on terminals (TXL (terminals))—for which the perplexity calculation is exact—on PTB and the sentence-level BLLIP. This shows that joint modeling of syntactic structures and surface strings in Transformers—even *without* any explicit inductive bias for making use of the syntactic information (*e.g.,* TXL (trees))—is still helpful for improving perplexity. We conjecture that the next-token prediction task is made easier by the presence of nonterminals within the context, which restricts the word classes that may appear next. Although there are more such prediction events for the linearized tree sequences than for the words-only model, the predictions of the nonterminals are marginalized out at evaluation time. At training time, it might seem that the learning demands placed on the model are higher, and that having to predict the syntactic structures could produce an interference and reduce the available model capacity for predicting the words. Here we do not find this to be the case, as evidenced by both syntactic language models' better perplexity.

Comparing TGs to TXL (trees), perplexity suffers by about 0.5 points (1%-1.7% increase relative to the TXL (trees) model's perplexity) on the two sentence-level datasets, and by about 4 points (19% relative increase) at the document-level on BLLIP-LG. We investigate the causes of this degredation in the Analysis section (§4).

	Perplexity (↓)			SG (↑)	F_1 (↑)
	PTB	BLLIP sent.	BLLIP doc.	BLLIP sent.	PTB
TG†	61.8 ± 0.2	30.3 ± 0.5	26.3 ± 0.1	**82.5 ± 1.6**	**93.7 ± 0.1**
TXL (trees)†	**61.2 ± 0.3**	**29.8 ± 0.4**	**22.1 ± 0.1**	80.2 ± 1.6	93.6 ± 0.1
TXL (terminals)	62.6 ± 0.2	31.2 ± 0.4	23.1 ± 0.1	69.5 ± 2.1	n/a
RNNG$^{\diamondsuit}$ (Dyer et al., 2016)	105.2	n/a	n/a	n/a	93.3
PLM-Mask$^{\diamondsuit}$ (Qian et al., 2021)	n/a	49.1$^{\clubsuit,\dagger}$ ± 0.3	n/a	74.8	n/a
Batched RNNG$^{\diamondsuit}$ (Noji and Oseki, 2021)	n/a	62.9$^{\heartsuit,\dagger}$	n/a	81.4$^{\heartsuit}$ ± 2.7	n/a
GPT-2 (Radford et al., 2019)	n/a	n/a	n/a	78.4$^{\spadesuit}$	n/a
Gopher (Rae et al., 2021)	n/a	n/a	n/a	79.5	n/a
Chinchilla (Hoffmann et al., 2022)	n/a	n/a	n/a	79.7	n/a

Table 1: Results on the **test sets** obtained for 100 models of TG, TXL (trees) and TXL. The results marked $\diamondsuit$ are directly taken from prior work, which may not directly comparable due to differences in model sizes, inference procedures, etc. $\dagger$ are upper bounds of perplexities, $\clubsuit$ are from personal correspondence with Qian et al., and $\spadesuit$ was computed from results by Hu et al. (2020). Results with $\heartsuit$ and $\heartsuit$ are obtained by personal correspondence with Noji and Oseki; $\heartsuit$ we select the best perplexity results for the batched RNNG (35M parameters, beam size of 1,000), $\heartsuit$ model trained on 100M Wikipedia tokens. The PTB parsing results of the batched RNNG (Noji and Oseki, 2021, Table 2) are not directly comparable since they reported results with beam search inference, whereas we use a parse reranking setup. Perplexities of the large LMs (last 3 rows) are not reported because of likely test data contamination (Brown et al., 2020).

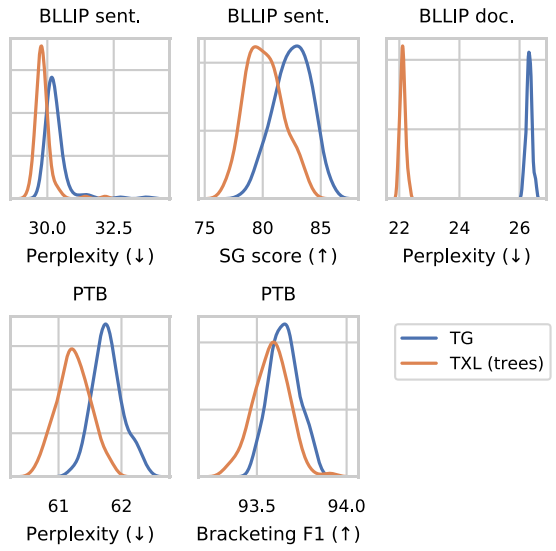


Figure 3: Distributions of the metrics of interest on the test sets, with 100 random initializations for each model. All the differences in means are statistically significant ($p < 10^{-3}$).

3.2 Parse Peranking

As human-annotated syntax trees are available for the PTB test split, we also compare models in their parsing accuracy by reranking the 300 candidate samples produced by RNNG for each sentence.

We report the mean and sample standard deviation of bracketing F_1 as computed with EVALB (Sekine and Collins, 1997) in Table 1, and plot its distribution in Figure 3. We observe that TG does slightly better (+0.1%) than TXL (trees) on this task; the small difference in mean is nevertheless statistically significant (two-sided Welch's unequal variances t-test, $p < 10^{-3}$).

3.3 Syntactic Generalization

Experimental Setup Hu et al. (2020) developed a series of test suites that probe the syntactic ability of language models on a large set of syntactic phenomena. The aim of this task is to comprehensively assess the ability of language models to syntactically generalize in a human-like fashion, which constitutes a key feature of human linguistic ability. A model succeeds on a given test case when the probabilities it assigns to specifically crafted examples obey an inequality (or conjunctions thereof) justified by how humans process language. We use models trained on independent sentences from BLLIP-LG, evaluate on parse trees provided by Hu et al. (2020) and generated using an RNNG proposal model, and we report the average syntactic generalization (**SG**) score across the same set of 31 test suites.

Discussion We report the mean and standard deviation of the average SG score in Table 1, and plot its distribution in Figure 3. Our first

observation is that the average SG score is substantially higher for models trained on linearized trees than on words alone—both TG and TXL (trees) outperform TXL (terminals). Interestingly, this also extends to models that are orders of magnitude larger and trained on much more data, such as GPT-2, Gopher (Rae et al., 2021, 280B params.), and Chinchilla (Hoffmann et al., 2022, 70B params). We believe that this result can be explained in three steps. First, the modeling of the structure via the nonterminals by TG and TXL (trees) can be seen, during training, as providing additional syntactic supervision. This enables them to pick, from a large number of candidate trees, good parses for a sentence. Second, as the SG score is computed from inequalities involving model surprisals on *words*, we perform an approximate marginalization step for TG and TXL (trees). In this approximate marginalization, valid parses are therefore heavily weighted. Finally, when the model has a strong preference for syntactically correct parses, the tasks from the test suite become easier, accounting for these models' higher scores. The results of the large LMs show model scale alone is insufficient to offset this effect.

Our second observation is that—comparing TGs to TXL (trees)—our approach is most beneficial on tasks that are most related to modeling structure, namely, parse reranking, in addition to the comprehensive SG test suite. On both tasks, TGs achieve higher bracketing F_1 and average SG scores with a statistically significant difference. We believe that this performance is explained by the restricted attention in TGs, thus preventing the model from attending to syntactically irrelevant parts of the input, and encouraging it to learn informative composed representations of subtrees.

In Figure 4, we present a breakdown of the SG results. As expected from the average SG score, the TXL (terminals) performs worse than both the TXL (trees) and TG, except for Gross Syntactic State, where it nearly reaches 100%. TG and TXL (trees) have very similar scores on all circuits except licensing, where TG substantially outperforms TXL (trees).[2] Altogether, these

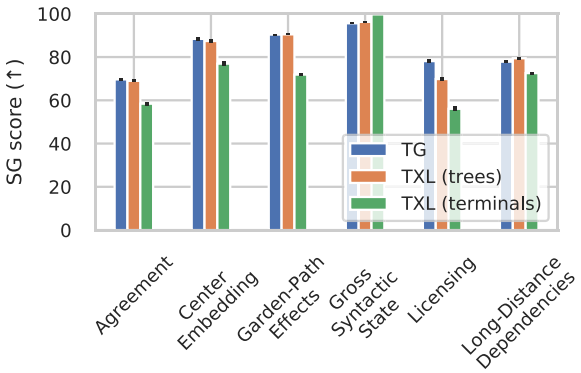


Figure 4: Per-circuit breakdown of the SG scores.

results demonstrate the benefits of recursive syntactic compositions for improving LM performance at syntax-sensitive benchmarks of human linguistic competence, even in the case of powerful Transformer-based language models that are trained at the medium data scale ($\approx$ 40M words). Furthermore, our findings shed more light on which syntactic constructions benefit the most from explicit syntactic compositions.

4 Analysis

To better understand what is causing the pattern of results in the previous section, we perform a series of analysis experiments: ablations, comparative analysis of emitted probabilities, and probing of the representations.

4.1 Ablation Experiments

As TGs jointly model the words and syntax tree, we perform ablations experiments where the words are preserved, but where the syntactic structure is transformed. The aim of this experiment is to better understand to what extent having access to the ''right'' syntactic structures at training time an important factor behind TGs' success. Would TGs still do just as well when they are trained with trivial or deterministically transformed syntax trees? We also study the effect of different kinds of position information on our metrics.

[2]One might wonder why Licensing and Gross syntactic state deviate from the pattern of results seen in other circuits. Licensing, where TGs excel, involves evaluating restrictions on pairs of words that are linearly separated, but ''structurally local'' (specifically, they stand in a c-command relationship). Since TGs are strongly biased to learn to make predictions in terms of such structurally local relations, it is unsurprising to find success in Licensing. On the other hand, success in Gross Syntactic State requires tracking whether an initial clause is subordinate (in which case a main clause should follow), or a main clause (in which case the sentence should end). Since subordinate clauses are introduced by one of a few subordinating conjunctions, and the content of a main clause is relatively independent of its subordinate, a syntax-free learner will easily learn that the subordinating conjunction determines the Gross syntactic state, explaining the result.

	Perplexity (↓)			SG (↑)	F_1 (↑)
	PTB	BLLIP sent.	BLLIP doc.	BLLIP sent.	PTB
TG†	76.1 ± 0.3	31.5 ± 0.6	26.9 ± 0.1	82.5 ± 1.6	$\mathbf{92.4 \pm 0.1}$
TG (no position info.)†	77.5 ± 0.3	31.9 ± 0.4	27.2 ± 0.1	$\mathbf{82.8 \pm 1.4}$	92.3 ± 0.1
TG (diff. in linear positions)	76.3 ± 0.3	31.5 ± 0.6	26.2 ± 0.1	82.6 ± 1.8	$\mathbf{92.4 \pm 0.1}$
TG (left-branching)†	134.0 ± 0.8	45.4 ± 0.6	41.3 ± 0.2	57.2 ± 2.2	n/a
TG (right-branching)†	91.2 ± 0.4	34.2 ± 0.6	29.2 ± 0.1	52.1 ± 3.8	n/a
TG (reversed trees)†	239.7 ± 11.5	49.3 ± 2.5	43.7 ± 0.6	53.9 ± 3.2	82.0 ± 0.2
TXL (trees)†	$\mathbf{74.9 \pm 0.3}$	$\mathbf{30.9 \pm 0.4}$	$\mathbf{22.3 \pm 0.1}$	80.2 ± 1.6	92.3 ± 0.1
TXL (trees, no position info.)†	80.4 ± 0.3	31.9 ± 0.3	31.5 ± 0.6	80.2 ± 1.8	92.0 ± 0.1
TXL (trees, left-branching)†	101.0 ± 0.9	34.3 ± 0.2	25.2 ± 0.2	64.3 ± 3.2	n/a
TXL (trees, right-branching)†	97.4 ± 0.4	33.6 ± 0.3	24.4 ± 0.1	53.4 ± 3.6	n/a
TXL (trees, reversed trees)†	129.3 ± 1.4	38.5 ± 2.6	28.3 ± 0.2	57.3 ± 3.4	87.3 ± 0.2
TXL (terminals)	77.3 ± 0.3	32.2 ± 0.5	23.3 ± 0.1	69.5 ± 2.1	n/a
TXL (terminals, no position info.)	80.5 ± 0.3	32.9 ± 0.2	25.7 ± 0.2	68.7 ± 2.0	n/a

Table 2: Results on the **validation** split of the datasets. †Perplexities reported for TG (all variants) and TXL (trees) are upper bounds, derived from approximately marginalizing over a set of proposal trees.

4.1.1 Transformed Structures

We transform a syntax tree into a *binary left-branching* one by moving the opening nonterminals to the left without reordering them, and, after the first two terminals, placing the required closing nonterminal after each terminal. For instance, `(S (NP the blue bird NP) (VP sings VP) S)` is transformed into `(S (NP (VP the blue VP) bird NP) sings S)`. Symmetrically, we form a *binary right-branching* tree from this example, `(S the (NP blue (VP bird sings VP) NP) S)`. Lastly, we define the *reversed trees* where the structure is reversed, that is, the children of a node are put in reverse order, but the order of the terminals is preserved, in this case `(S (VP the VP) (NP blue bird sings NP) S)`. In all three transformations, the apparent syntactic structure, as indicated by the nonterminals, is no longer the true syntactic structure of the sentence.

We train and evaluate (on the validation sets) TG and TXL (trees) on such transformed trees, and report our results in Table 2. For TG, perplexity, syntactic generalization, and bracketing F_1[3] are much worse, regardless of the transformation, compared to using the original trees. This is unsurprising considering that the operations performed by the model mechanically depend on the syntactic structure represented by the nonterminals. An apparent structure that does not correspond to the sentence will therefore lead to an unsuitable sequence of operations. More precisely, perplexity is most impacted on reversed trees than on left-branching trees, and right-branching trees have the least impact. Indeed, left-branching trees are comparatively easier to model, because these sequences are formed of a prefix of opening nonterminals, followed by an interleaved sequence of terminals and closing nonterminals. Right-branching trees are similarly easy, and furthermore, the COMPOSE operations specific to TG only happen when the closing nonterminals are encountered towards the end of the sequence, which is deterministically determined by its left context.

Unlike TG, TXLs (trees) have no constraints to use the syntactic information to model terminals, and thus they are free to use it or not. However, it is training data, and model capacity must be used to account for its distribution. This explains why performance degradation the TXL (trees) performance is harmed by the tree transformations, but less than the TG performance.

4.1.2 Positional Information

As observed by Haviv et al. (2022), not using positional information for TXL (terminals) and TXL (trees) has a negative but small impact on perplexity, which the authors conjecture is due to the ability of the model to learn positional information using the causal mask. Under this hypothesis, it follows that the impact on the syntactic generalization and the bracketing F_1 scores

[3]Here, we do not use the `DELETE_LABEL` directives from `COLLINS.prm`.

should be small, which is what we observe. The most impacted model is TXL (trees) on BLLIP-LG documents, which we conjecture is due to the long sequences of tokens, including repeated identical nonterminals, making access to a good positional signal important. For TG, the results are similar, and the same mechanism can be posited to be at play. Its attention mask is not only causal, but also very sparse. Because there are few tokens that can be attended to, position-based querying is at the same time less critical and easier to learn.

We train and evaluate a variant of TG using difference in linear position as relative position function, instead of difference in tree depth, and find almost no impact on performance. This is readily explained by the same reasons as above—as TG's attention mask is so sparse, position-based querying matters little. Given the same empirical performance, we solely ground our choice in its theoretical justification (see §2.1).

4.2 Regression Analysis of Probabilities

To determine when TGs are more or less successful than the unrestricted TXL (trees) model, we predict the differences in log probabilities of the true terminal a_i under the two models:

$$\Delta_i = \log p_{\text{TG}}(a_i \mid \boldsymbol{a}_{<i}) - \log p_{\text{TXL (trees)}}(a_i \mid \boldsymbol{a}_{<i}).$$

To reduce variance stemming from model initialization, we use an ensemble of 100 Transformer Grammars and an ensemble of 100 TXLs (trees).

4.2.1 Terminal Frequencies

We hypothesize that the syntactically restricted attention pattern of TGs—where subsequent predictions can only attend to composed representations—prevents it from learning the non-syntactic dimensions of the data distribution, such as rare co-occurrences, to the same extent as TXLs (trees). Based on this hypothesis, we expect the TGs' predictions to be worse for rare tokens.

We therefore compute the empirical unigram distribution of the terminals in the training split of BLLIP-LG documents, and partition terminals into high-frequency ($f \geq 10^{-3}$), medium-frequency ($10^{-5} \leq f < 10^{-3}$), and low-frequency ($f < 10^{-5}$) buckets. We then define three binary variables, indicating whether the terminal at a given position has a high, medium, or low frequency, and use these in an ordinary least squares model to predict the difference in log probabilities on the BLLIP-LG validation set: $\Delta \sim \text{HighFreq} + \text{MediumFreq} + \text{LowFreq}$.

We find an adjusted R^2 value of 0.039, and coefficients $\beta_{\text{HighFreq}} = -0.0488$, $\beta_{\text{MediumFreq}} = -0.2419$, $\beta_{\text{LowFreq}} = -0.5481$, all statistically different from 0 with a p-value $< 10^{-3}$.

This shows that—although TGs can predict the terminals appearing most frequently almost as well as TXLs (trees) do—they struggle to predict rarer ones. We hypothesize that lexical co-occurrences that cross syntactic units can be learnt directly by TXLs (trees), whereas this is more difficult to do for TGs. Indeed, a consequence of STACK/COMPOSE attention is that a terminal A can only attend to another terminal B *iff* B is in A's left-context, and B is A's sibling. Our result suggests that this is not happening sufficiently often for TGs to predict rare terminals as well as TXLs (trees) do.

4.2.2 Copying

Likewise, we hypothesize that TXL (trees) is better at copying words from the context than are TGs.

We define three binary variables, indicating (i) whether the true terminal to predict appears in the context in a previous sentence, but not in the current one; (ii) whether it appears in the context in the current sentence; or (iii) does not appear in the context at all. We use these in a new ordinary least squares model to predict the difference in log probabilities: $\Delta \sim \text{InContextPrevSentences} + \text{InContextCurSentence} + \text{NotInContext}$.

We find an adjusted R^2 value of 0.010, and coefficients $\beta_{\text{InContextPrevSentences}} = -0.2871$, $\beta_{\text{InContextCurSentence}} = -0.1003$, $\beta_{\text{NotInContext}} = -0.1340$, all statistically different from 0 with a p-value $< 10^{-3}$. This finding suggests that TGs perform worse than TXL (trees) on all three conditions, although the difference is most pronounced for terminals appearing in a previous sentence (but not in the current one). This observation suggests that TXLs (trees) benefit from a priming effect—previously seen tokens becoming more likely—whereas this effect is diminished in TGs.

4.3 Probing Analysis of Representations

Experimental Setup To quantify how well the representations learned by TG, TXL (trees), and

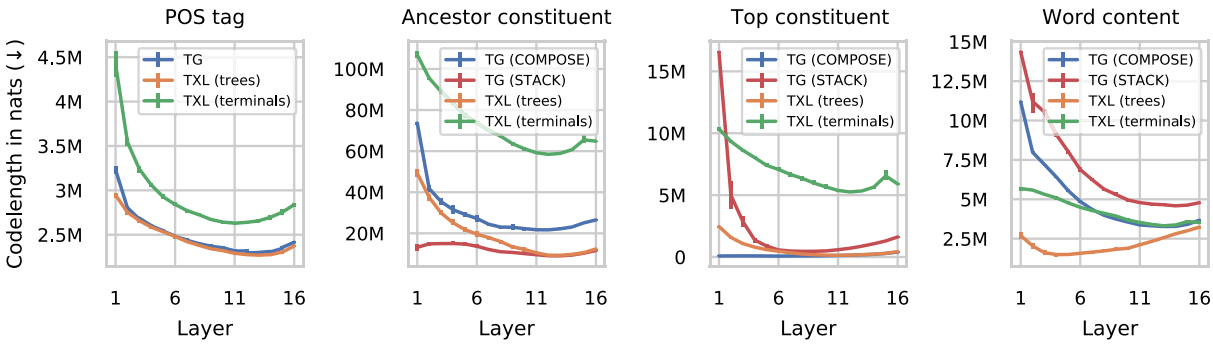


Figure 5: Code length of the labels given the model representations on probing tasks.

TXL (terminals) encode syntactic information, we use the information-theoretic probing framework developed by Voita and Titov (2020), capturing in a principled way how well the probes explain the labels given the representations, as well as how readily available the information is in the representations (*i.e.*, how complex the probes are). If the probe labels can be easily predicted from the model's hidden state activations, even with a simple probe, then this provides an indication that the probed phenomenon is more saliently encoded within the model's learnt vector representations. Here we used four probing tasks. The first two probing tasks are taken from Liu et al. (2019), where we predict (i) the part-of-speech tag of each terminal, and (ii) the grandparent constituent tag of each completed phrase/subtree, which requires an understanding of the relevant phrase-structure information. We then use two other probing tasks from Conneau et al. (2018): (iii) Predicting the top constituent of the syntax tree and (iv) word-content tagging. We train the probes on the activations output by each layer on the training set of BLLIP-LG, then evaluate the code length on the validation set. For the POS tagging task, we use a single representation for each word. For the three other tasks, we use the representation corresponding to the closing nonterminal for each subtree—for TG, two such representations are available due to the closing nonterminal duplication (§2.1), and we consider them separately; for TXL (terminals), we aggregate the representations of the first and last words of the phrase/subtree.

Discussion We report in Figure 5 the code length of the probing labels given the model representations on the four probing tasks. We observe that the representations from TG and TXL (trees) explain equally well the POS tags. Predicting the ancestor constituents is much easier using the representations from the closing nonterminals in the STACK positions in TG compared to TXL (trees), which is expected considering that the ancestors, by construction of the tree, are still on the stack when the subtree is closed. Conversely, predicting the top constituents (*i.e.*, the sequence of immediate children) is easy to do from the COMPOSE representations, which precisely attend over them. On these three syntactic tasks, TXL (terminals), which does not benefit from syntactic supervision, does predictably worse. Finally, it is much easier to predict whether a subtree contains a word from a given set from the representations from TXLs (trees) compared to either type of representations from TG, and even from the TXL (terminals), suggesting that TXL (trees) models retain which words appear in the context better than TGs do, echoing our regression analysis results.

5 Related Work

A variety of work has augmented language models with syntactic or hierarchical biases. The RNNG model (Dyer et al., 2016; Kuncoro et al., 2017) jointly models trees and strings and uses recursive networks to build representations of phrases (similar to the approach taken here); however, scaling RNNG training is nontrivial (Noji and Oseki, 2021). Other forms of structural bias that do not use observed syntax trees have come in the form of stack-structured memory (Yogatama et al., 2018), running RNNs at multiple scales (Chung et al., 2017), and structuring the ''forget'' gates in LSTMs to encourage hierarchy (Shen et al., 2019).

With the advent of Transformers and large-scale pretraining, the question of whether syntax and hierarchy ''is still needed'' received renewed interest. While bidirectional encoders do learn a great deal about syntax from pretraining (Manning et al., 2020), long-tail syntactic phenomena continue to pose a problem, and may be implicated in systematic semantic failures (Ettinger, 2020). Prior work has devised multiple strategies for injecting syntactic inductive biases into Transformers (Wang et al., 2020; Sundararaman et al., 2019; Kuncoro et al., 2020; Sachan et al., 2021; Bai et al., 2021). However, improved syntactic awareness is not found to be beneficial for some language understanding tasks (Warstadt et al., 2020; Kuncoro et al., 2020; Pruksachatkun et al., 2020; Sachan et al., 2021).

Our approach to injecting syntactic biases into generative transformer language models combines two modeling traditions: (i) syntactic language models that estimate the joint probability of strings and trees (Jurafsky et al., 1995; Chelba and Jelinek, 2000; Roark, 2001; Henderson, 2004; Mirowski and Vlachos, 2015; Choe and Charniak, 2016; Kim et al., 2019), and (ii) constraining attention patterns in accordance with syntactic structures (Strubell et al., 2018; Wang et al., 2019; Peng et al., 2019; Zhang et al., 2020; Nguyen et al., 2020; Astudillo et al., 2020). TGs are perhaps most closely related to the model proposed in Qian et al. (2021), who similarly combined syntactic language modeling with syntax-based attention constraints. We differ from this model in two primary ways. First, TGs use a new kind of typed-attention mask with duplicated closing nonterminal symbols that implement recursive syntactic compositions, which was identified as a critical component of RNN-based syntax models (Dyer et al., 2016; Kim et al., 2019; Wilcox et al., 2019; Futrell et al., 2019). Second, this paper explores an extension of sentence-level syntactic models to models of full documents. Modeling of multi-sentence sequences has been a key feature behind recent language modeling successes (Radford et al., 2019; Brown et al., 2020), and thus understanding how syntax interacts with this modeling problem is of considerable interest.

6 Conclusion

Transformer Grammars are a new syntactic language model that implements recursive syntactic composition of phrase representations through attention. Experiments show that TGs outperform prior work on two syntax-sensitive language modeling evaluation metrics. On sentence-level language modeling, TGs outperform a strong Transformer-XL that operates only on the word sequences, although we find that they perform worse at the document-level, when restricted to use a single composed representation for each previous sentence. We also find that the presence of structural information is strictly better on all metrics than in Transformers trained on words alone. While just as efficient to sample from as any autoregressive language model, TGs however do not provide probability estimates as easily, and using it where these are needed requires accepting more computation, or further research into efficient methods for probability estimation. Taken more broadly, our findings emphasize the ongoing importance of finding better and scalable ways to encourage language models to internalize the structural properties of language. Our implementation is available upon request.

Acknowledgments

We wish to thank the anonymous reviewers for their feedback and suggestions, as well as the action editors. We would also like to thank Jennifer Hu and Peng Qian for providing us with the BLLIP-LG reparsed data, the partial trees used for the syntactic models evaluated in Hu et al. (2020), and for their answers to our many questions. We also thank Hiroshi Noji and Yohei Oseki for providing a detailed result breakdown of their efficiently batched RNNG model. Finally, we are grateful to Mark Johnson, Kris Cao, Laura Rimell, Nando de Freitas, and our colleagues in the DeepMind Language team for their insightful thoughts and comments.

References

Ramón Fernandez Astudillo, Miguel Ballesteros, Tahira Naseem, Austin Blodgett, and Radu Florian. 2020. Transition-based parsing with stack-transformers. In *Findings of EMNLP*.

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.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1466–1477, Berlin, Germany. Association for Computational Linguistics. https://doi.org/10.18653/v1/P16-1139

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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.

Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, John Hale, and Mark Johnson. 2000. BLLIP 1987–89 WSJ Corpus Release 1, LDC2000T43. *LDC2000T43. Linguistic Data Consortium*, 36.

Ciprian Chelba and Frederick Jelinek. 2000. Structured language modeling. *Computer Speech & Language*, 14(4):283–332. https://doi.org/10.1006/csla.2000.0147

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.

Noam Chomsky. 1957. *Syntactic Structures*. Mouton, The Hague/Paris. https://doi.org/10.1515/9783112316009

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2017. Hierarchical multiscale recurrent neural networks. In *Proceedings of ICLR*.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics. https://doi.org/10.18653/v1/P18-1198

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*.

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. https://doi.org/10.18653/v1/N16-1024

Allyson Ettinger. 2020. What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics*. https://doi.org/10.1162/tacl_a_00298

Martin B. H. Everaert, Marinus A. C. Huybregts, Noam Chomsky, Robert C. Berwick, and Johan J. Bolhuis. 2015. Structures, not strings: Linguistics as part of the cognitive sciences. *Trends in Cognitive Sciences*, 19(12):729–743. https://doi.org/10.1016/j.tics.2015.09.008, PubMed: 26564247

Richard Futrell, Ethan Wilcox, Takashi Morita, Peng Qian, Miguel Ballesteros, and Roger Levy. 2019. Neural language models as

psycholinguistic subjects: Representations of syntactic state. 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 32–42, Minneapolis, Minnesota. Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1004

Michael Hahn. 2020. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171. https://doi.org/10.1162/tacl_a_00306

Adi Haviv, Ori Ram, Ofir Press, Peter Izsak, and Omer Levy. 2022. Transformer language models without positional encodings still learn positional information. (arXiv:2203.16634).

James Henderson. 2004. Discriminative training of a neural network statistical parser. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 95–102, Barcelona, Spain. https://doi.org/10.3115/1218955.1218968

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. *CoRR*, abs/2203.15556v1.

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.

Daniel Jurafsky, Chuck Wooters, Jonathan Segal, Andreas Stolcke, Eric Fosler, Gary N. Tajchman, and Nelson Morgan. 1995. Using a stochastic context-free grammar as a language model for speech recognition. In *Proceedings of ICASSP*.

Yoon Kim, Alexander Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer, and Gábor Melis. 2019. Unsupervised recurrent neural network grammars. In *Proceedings of NAACL*.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics. https://doi.org/10.18653/v1/P18-1007

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-2012

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. https://doi.org/10.18653/v1/E17-1117

Adhiguna Kuncoro, Chris Dyer, Laura Rimell, Stephen Clark, and Phil Blunsom. 2019. Scalable syntax-aware language models using knowledge distillation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3472–3484, Florence, Italy. Association for Computational Linguistics.

Adhiguna Kuncoro, Lingpeng Kong, Daniel Fried, Dani Yogatama, Laura Rimell, Chris Dyer, and Phil Blunsom. 2020. Syntactic structure distillation pretraining for bidirectional encoders. *Transactions of the Association for Computational Linguistics*, 8:776–794. https://doi.org/10.1162/tacl_a_00345

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. 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 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Christopher D. Manning, Kevin Clark, John Hewitt, Urvashi Khandelwal, and Omer Levy. 2020. Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proceedings of the National Academy of Sciences*, 117(48).

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330. https://doi.org/10.21236/ADA273556

Piotr Mirowski and Andreas Vlachos. 2015. Dependency Recurrent Neural Language Models for Sentence Completion. In *Proceeding of ACL-IJCNLP*. https://doi.org/10.3115/v1/P15-2084

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 ACL-IJCNLP*.

Hao Peng, Roy Schwartz, and Noah A. Smith. 2019. PaLM: A hybrid parser and language model. In *Proceedings of EMNLP-IJCNLP*. https://doi.org/10.18653/v1/D19-1376

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL*.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York. Association for Computational Linguistics.

Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel R. Bowman. 2020. Intermediate-task transfer learning with pretrained language models: When and why does it work? In *Proceedings of ACL*. https://doi.org/10.18653/v1/2020.acl-main.467

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. https://doi.org/10.18653/v1/2021.acl-long.289

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

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, H. 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 M. 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, 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 A. Hechtman, Laura Weidinger, Iason Gabriel, William S. Isaac, Edward Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. 2021. Scaling language models: Methods, analysis & insights from training gopher. *CoRR*, abs/2112.11446v2.

Brian Roark. 2001. Probabilistic Top-Down Parsing and Language Modeling. *Computational Linguistics*, 27(2):249–276. https://doi.org/10.1162/089120101750300526

Devendra Sachan, Yuhao Zhang, Peng Qi, and William L. Hamilton. 2021. Do syntax trees help pre-trained transformers extract information? In *Proceedings of EACL*. https://doi.org/10.18653/v1/2021.eacl-main.228

Satoshi Sekine and Michael John Collins. 1997. Evalb - bracket scoring program.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks. In *International Conference on Learning Representations*.

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.

Dhanasekar Sundararaman, Vivek Subramanian, Guoyin Wang, Shijing Si, Dinghan Shen, Dong Wang, and Lawrence Carin. 2019. Syntax-infused transformer and BERT models for machine translation and natural language understanding. *arXiv preprint arXiv:1911.06156v1*.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Elena Voita and Ivan Titov. 2020. Information-theoretic probing with minimum description length. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 183–196, Online. Association for Computational Linguistics.

Wei Wang, Bin Bi, Ming Yan, Chen Wu, Zuyi Bao, Liwei Peng, and Luo Si. 2020. StructBERT: Incorporating language structures into pre-training for deep language understanding. In *Proceedings of ICLR*.

Yaushian Wang, Hung-Yi Lee, and Yun-Nung Chen. 2019. Tree transformer: Integrating tree structures into self-attention. In *Proceedings of EMNLP-IJCNLP*. https://doi.org/10.18653/v1/D19-1098

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. *TACL*. https://doi.org/10.1162/tacl_x_00375

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. https://doi.org/10.18653/v1/N19-1334

Dani Yogatama, Yishu Miao, Gabor Melis, Wang Ling, Adhiguna Kuncoro, Chris Dyer, and Phil Blunsom. 2018. Memory architectures in recurrent neural network language models. In *Proceedings of ICLR*.

Zhuosheng Zhang, Yuwei Wu, Junru Zhou, Sufeng Duan, Hai Zhao, and Rui Wang. 2020. SG-Net: Syntax-guided machine reading comprehension. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7–12, 2020*, pages 9636–9643. AAAI Press. https://doi.org/10.1609/aaai.v34i05.6511