Does Traversal Order Matter? A Systematic Study of Tree Traversal Methods in Transformer Grammars

Anonymous ACL submission

Abstract

Transformer Grammars (TGs) enhance language modeling by incorporating syntactic tree structures. Despite the potentially significant impact on model performance of how syntactic trees are linearized in TGs, existing studies rely solely on Depth-First Traversal (DFT) for linearization. In this paper, we expand the traversal design space by exploring Breadth-First Traversal (BFT) and a novel hybrid traversal strategy, Production-Rule Traversal (PRT), which combines the structural lookahead of BFT with the early lexical generation of DFT. We integrate these traversal methods with varying tree configurations and masking strategies, and empirically evaluate their performance on language modeling, syntactic generalization and summarization. We reveal the inherent trade-offs between nested composition and global lookahead, providing actionable recommendations for designing task-aware Transformer Grammars.

1 Introduction

The transformer architecture (Vaswani et al., 2017) has shown strong performance but lacks the inductive bias of syntactic structure, which is critical for effective generalization (Everaert et al., 2015). Syntactic language models (SLMs) (Qian et al., 2021; Sartran et al., 2022; Zhao et al., 2024; Hu et al., 2024; Ji et al., 2025) address this by introducing syntactic structural inductive biases, modeling linearized parse trees alongside surface sentences. Crucially, the chosen traversal method dictates the syntactic context, fundamentally shaping the model's syntactic bias. However, the traversal method used by existing SLMs to linearize trees into sequences remains underexplored beyond Depth-First Traversal (DFT) (Zhao et al., 2025).

We systematically explore this traversal space within Transformer Grammars (TGs). Specifically, we investigate DFT and Breadth-First Traversal (BFT). Both approaches exhibit contrasting limitations tied to tree structure. DFT favors early lexical generation but lacks structural lookahead—the ability to access global syntactic information before generating lexical tokens—forcing premature lexical predictions in flat n-ary trees. Conversely, BFT favors early syntactic generation but delays terminal generation, forcing structural predictions without concrete words in deep binary trees. To balance these extremes, we propose Production-Rule Traversal (PRT), fusing BFT's structural lookahead with DFT's early lexical generation.

We evaluate these three traversal methods across varying tree configurations and masking strategies. The experimental results reveal that the optimal traversal strategy is task-dependent. Specifically, downstream generation tasks benefit significantly from the structural lookahead of PRT, which provides a broader context for next-token prediction. In contrast, performance on syntactic generalization indicates that syntax-focused tasks favor the strictly nested composition of DFT.

2 Related Work

Compositional SLMs. Transformer Grammars (Sartran et al., 2022; Hu et al., 2024) inject structural biases by explicitly modeling and composing trees alongside text. Zhao et al. (2025) recently unified this paradigm under the Compositional SLM framework. While their study explores top-down and bottom-up linearizations, both remain confined to DFT. DFT lacks structural lookahead, leaving the balance between global syntactic information and early lexical generation unexplored in SLMs.

Traversal Strategies. Traversal strategies profoundly impact structured prediction, from exploiting hierarchical parallelism in non-autoregressive generation (Ji et al., 2025) to balancing lookahead and compositionality in discriminative parsing (Liu and Zhang, 2017). We extend these insights to

generative SLMs, systematically analyzing how traversal orders affect downstream performance.

3 Background

A Transformer Grammar (TG) jointly models a surface sentence x and its constituency tree y. For standard autoregressive processing, the tree is linearized to sequences of structural and lexical actions $a = (a_0, \ldots, a_{L-1})$ of length L, factorizing the joint probability as $p(x, y) = \prod_i p(a_i \mid a_{<i})$.

3.1 Parse Tree Binarization

While natural constituency trees are inherently non-binary, prior syntactic language models often adopt binarized structures for practical efficiency. Following the established paradigm of Zhao et al. (2025), tree binarization is typically evaluated under two configurations: (i) **Non-binary (N)** trees, which eliminate unary chains directly connected to terminals, so structures "(NP cat NP)" are simplified to "cat", and (ii) **Binary (B)** trees, constructed via standard left-binarization.

3.2 Linearization

Previous works predominantly rely on top-down Depth-First Traversal (DFT) to linearize syntactic trees (Dyer et al., 2016). Under this paradigm, the action space consists of three types: (i) opening a non-terminal, (ii) generating a terminal token, and (iii) closing a non-terminal. DFT recursively completes all actions for a subtree before executing any actions for its siblings.

3.3 Composition and Masking

A prevalent technique for composition in TGs is the internal composition mechanism (Sartran et al., 2022). In this framework, a closing non-terminal "X)" triggers a COMPOSE operation: the model's attention is restricted to the non-terminal's immediate subconstituents, encoding them into a single representation of the subtree. To resume generation, a duplicate "X)" immediately follows, attending to the broader context to allow the model to continue predicting subsequent tokens.

Following composition, models adopt one of two masking strategies for subsequent steps: (i) **Masked (M):** Blocks attention to the internal nodes of composed subconstituents (Sartran et al., 2022). (ii) **Not masked (NM):** Retains full attention to all prior tokens (Hu et al., 2024).

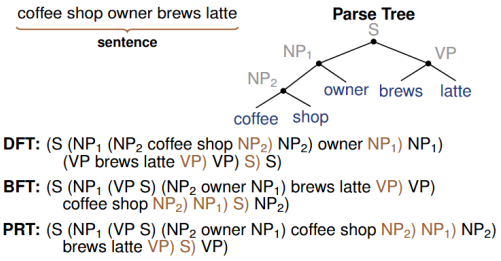

Figure 1: A comparison of tree traversal strategies to linearize an example sentence. We use brown for the closing non-terminal to trigger the COMPOSE operation.

3.4 Inference

To mitigate the probability imbalance between high-entropy terminals and low-entropy non-terminals, decoding processes often rely on word-synchronous beam search (Stern et al., 2017). Additionally, structural constraint hyperparameters (e.g., maximum non-terminal count and maximum consecutive opening parentheses) identified by Zhao et al. (2025) are typically applied to prevent the model from over-generating structural actions.

4 Design Space of Traversal Strategies

Having described the foundational mechanics of Transformer Grammars, we now outline the design space for evaluating traversal strategies. To assess the impact of explicit syntactic features, we evaluate both **labeled trees (L)**, which retain the original non-terminals (Sartran et al., 2022), and **unlabeled trees (UL)**, which use anonymized labels (Zhao et al., 2025).

4.1 Linearization

The chosen traversal order dictates the sequence of linearization actions, strictly determining the syntactic context at any given generation step. To expand the design space beyond the standard depth-first baseline, we introduce two alternative linearization strategies guided by different traversals (exemplified in Figure 1 and formalized in Appendix A):

BFT: Under BFT, the linearization operates level by level. Specifically, upon visiting a non-terminal node, the opening actions or terminal generations for all its immediate children are executed sequentially, immediately followed by the closing action for the current node.

PRT: Under PRT, linearization begins by pushing the root node onto a stack. At each step, a

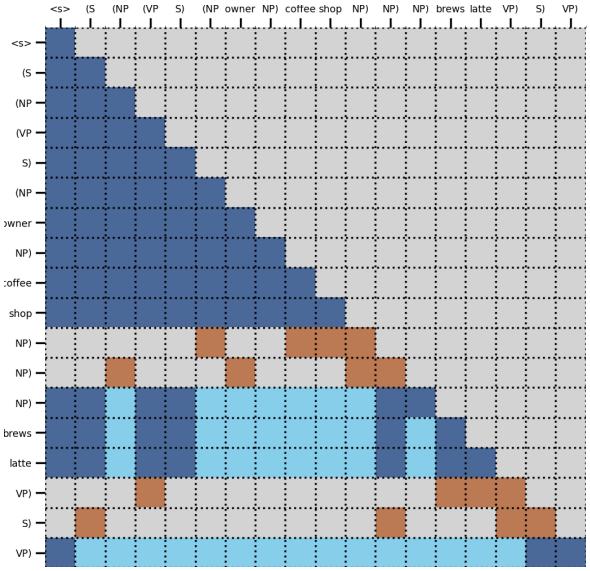

Figure 2: Illustration of the PRT compose method, which is shared with BFT. We use brown for the attention ranges of internal compositions, dark blue for ordinary attended positions, light blue for already-composed positions that are only accessible in NM.

node is popped, and the opening actions or terminal generations for all its immediate children are executed sequentially. Subsequently, the non-terminal children are pushed in reverse order, immediately followed by the closing action for the current node.

4.2 Composition

In BFT and PRT, operations for closing non-terminals are strictly dictated by the completion status of their corresponding subtrees. If a subtree remains incomplete (i.e., it contains unclosed non-terminals), the closing non-terminal is not duplicated; instead, it is used for next-token prediction, delaying composition. Conversely, once nested subtrees are fully completed, this delay is resolved by introducing a sequence of duplicated closing non-terminals corresponding to each completed subtree. These execute COMPOSE operations from bottom to top, followed by the current non-terminal resuming next-token prediction, shown in Figure 2.

4.3 Inference

Lacking a static parse tree during inference, our decoding algorithm adapts word-synchronous beam search to track subtree completion. It monitors the output sequence, detects when nested structures complete, and dynamically appends the required duplicated closing non-terminals to activate the composition mechanism described in Section 4.2.

Model Variant	B-L	B-UL	N-L	N-UL
DFT-M	34.24	23.32	24.22	23.45
DFT-NM	**30.17**	20.57	22.06	**21.74**
DFT-tree	30.32	21.40	22.40	22.20
BFT-M	36.88	24.45	24.57	24.27
BFT-NM	32.00	22.11	22.82	22.76
BFT-tree	33.18	23.00	23.90	23.01
PRT-M	35.22	22.83	24.12	23.59
PRT-NM	30.79	**20.38**	**21.83**	21.87
PRT-tree	30.96	21.47	22.93	22.38

Table 1: Perplexity (↓) results of our models, where all values represent upper bounds.

Model Variant	N-L	N-UL	B-UL
DFT-M	**71.38**	**71.25**	**72.77**
DFT-NM	71.19	69.19	70.82
DFT-tree	70.38	69.37	69.69
BFT-M	69.86	70.84	71.21
BFT-NM	69.82	69.64	69.23
BFT-tree	68.79	67.59	69.12
PRT-M	70.90	71.12	71.86
PRT-NM	70.09	70.11	70.48
PRT-tree	69.24	69.16	70.24

Table 2: BLiMP (↑) results of our models

5 Experiments

We evaluate BFT and PRT against all DFT variants, our baselines. All models are trained from scratch on the BLLIP-LG dataset of Charniak et al. (2000) with training splits from Hu et al. (2020), parsed via an off-the-shelf standard CRF constituency parser (Zhang et al., 2020), implemented in Supar[1]. We adopt a modernized GPT-2 Small architecture (∼109M parameters) for all variants, deferring comprehensive configurations to Appendix B. Across all traversal methods, the standard model trained on the linearized tree sequence is denoted as X-tree (e.g., DFT-tree). Additionally, a traditional token-level language model baseline is provided in Appendix C. Crucially, all traversal strategies operate on the exact same underlying constituency trees, differing solely in their traversal orders. We first evaluate all variants on standard language modeling, filtering out underperforming configurations before assessing the remainder on syntactic generalization and downstream summarization.

5.1 Document-Level Language Modeling

We evaluate all models on the testing split of BLLIP-LG from Hu et al. (2020).

Computing the exact string probability $p(x) = \sum_y p(x, y)$ is intractable. Following Sartran et al.

[1] https://github.com/yzhangcs/parser

	N-L				N-UL				B-UL			
Model Variant	**R1**	**R2**	**RL**	**R-AVG**	**R1**	**R2**	**RL**	**R-AVG**	**R1**	**R2**	**RL**	**R-AVG**
DFT-tree	30.77	10.04	24.10	21.64	30.26	10.10	24.02	21.46	29.31	9.52	23.16	20.66
DFT-M	26.61	7.85	21.22	18.56	27.38	8.36	21.88	19.21	25.26	7.09	19.99	17.44
DFT-NM	30.58	10.35	24.40	21.78	30.23	10.09	24.08	21.46	29.69	9.80	23.51	21.00
BFT-tree	30.47	10.06	24.07	21.53	30.63	9.97	23.96	21.52	29.88	9.42	23.15	20.02
BFT-M	27.46	7.98	21.56	19.00	27.28	7.92	21.44	18.88	26.18	7.15	20.31	17.88
BFT-NM	31.04	10.34	24.44	21.94	30.08	10.32	24.33	21.91	30.31	9.62	23.37	21.10
PRT-tree	**31.31**	**10.47**	24.64	22.14	30.45	10.02	23.95	21.47	30.03	9.56	23.35	20.98
PRT-M	27.50	8.18	21.75	19.14	27.34	7.97	21.50	18.94	26.01	7.42	20.39	17.94
PRT-NM	31.29	**10.47**	**24.71**	**22.16**	**31.33**	**10.54**	**24.65**	**22.17**	**30.65**	**10.33**	**23.96**	**21.65**

Table 3: ROUGE scores on the XSum dataset. Best results in each column are bolded.

(2022), we approximate sentence-level $p(x)$ by summing over a proposal set of 300 unlabeled constituency trees sampled without replacement via a CRF parser, establishing a strict lower bound for $p(x)$ and a valid upper bound for perplexity. For document-level modeling, marginalizing trees across historical sentences is computationally prohibitive. Thus, we approximate the context for the i-th sentence by greedily selecting the single highest-probability tree for each preceding sentence to serve as a fixed structural prefix, following Sartran et al. (2022).

As shown in Table 1, unmasked models have lower perplexity, with PRT-NM excelling in N-L and B-UL. The B-L configuration fails because forcing the model to predict labels for numerous nodes introduced by binarization consumes an excessive amount of representational capacity, which is consistent with empirical findings in Sartran et al. (2022). Therefore, we exclude B-L from downstream evaluations.

5.2 Syntactic Generalization

To measure syntactic generalization, we evaluate our models on BLiMP (Warstadt et al., 2020).

Accuracy is measured by whether a model assigns a higher marginal probability $p(x)$ to the grammatical sentence within each minimal pair, approximated using the identical 300-tree sampling method described in Section 5.1.

For syntactic generalization, masked models yield the highest accuracy, with the baseline DFT-M achieving the best overall performance in all tree structures. The results are presented in Table 2.

5.3 Summarization

We evaluate text summarization on the XSum dataset (Narayan et al., 2018).

We truncate parsed source documents to 1,800 tokens, followed by a parsed "Summary above article in one sentence." prompt. Models are finetuned on XSum for 10 epochs (batch 40). We report ROUGE (Lin and Hovy, 2003) using beam search (size 6) with linearized silver parse trees as input.

As shown in Table 3, PRT generally outperforms the BFT and DFT baselines. Across all traversals, unmasked configurations (-NM, -tree) consistently dominate masked ones (-M), culminating in our proposed PRT-NM achieving the highest overall ROUGE scores.

5.4 Overall Observations

Based on our experiments, we highlight key findings for syntactic language modeling:

(i) PRT effectively balances structured syntax with sequential generation. For document-level language modeling and summarization tasks, PRT-NM consistently outperforms traditional DFT baselines.

(ii) While BFT's structural lookahead improves summarization, its excessive delay of terminal tokens hurts language modeling and syntactic generalization. This confirms that layer-wise expansion severely disrupts the left-to-right context, highlighting why PRT's moderate approach is necessary.

(iii) While B-UL minimizes perplexity via dense non-terminals, its increased sequence length degrades long-distance attention and summarization. Compact N-L structures are optimal, balancing length with semantic guidance. Conversely, B-L fails due to the excessive classification burden of predicting labels for numerous binarized nodes.

6 Conclusion

In summary, we extend the Transformer Grammar framework by applying BFT and proposing PRT to enable structural lookahead in generative syntactic modeling. Through a systematic empirical evaluation across diverse tree configurations and attention masking mechanisms, we reveal a fundamental bifurcation in structural inductive biases: generation-focused tasks benefit significantly from the global lookahead of PRT, whereas syntax-focused tasks favor the strictly nested composition of DFT.

7 Limitations

While our study provides foundational insights, its primary limitations include the restriction to small-scale architectures (∼109M parameters), the reliance on external syntactic parsers, and an exclusive focus on internal composition mechanisms without evaluating external alternatives.

References

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

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.

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.

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. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2640–2657, Bangkok, Thailand. Association for Computational Linguistics.

Pengyu Ji, Yufei Liu, Xiang Hu, and Kewei Tu. 2025. Tree-structured non-autoregressive decoding for sequence-to-sequence text generation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 6168–6174, Suzhou, China. Association for Computational Linguistics.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 150–157.

Jiangming Liu and Yue Zhang. 2017. In-order transition-based constituent parsing. *Transactions of the Association for Computational Linguistics*, 5:413–424.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. 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.

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. *Transactions of the Association for Computational Linguistics*, 10:1423–1439.

Noam Shazeer. 2020. Glu variants improve transformer. *Preprint*, arXiv:2002.05202.

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.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

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

Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. *Preprint*, arXiv:1910.07467.

Yu Zhang, Houquan Zhou, and Zhenghua Li. 2020. Fast and accurate neural crf constituency parsing. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 4046–4053. International Joint Conferences on Artificial Intelligence Organization. Main track.

Yida Zhao, Chao Lou, and Kewei Tu. 2024. Dependency transformer grammars: Integrating dependency structures into transformer language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1543–1556, Bangkok, Thailand. Association for Computational Linguistics.

Yida Zhao, Hao Xve, Xiang Hu, and Kewei Tu. 2025. A systematic study of compositional syntactic transformer language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7070–7083, Vienna, Austria. Association for Computational Linguistics.

A Traversal Algorithm of BFT and PRT

To facilitate reproducibility and provide a rigorous formalization of our proposed traversal strategies, we detail the exact procedures for Breadth-First Traversal (BFT) and Production-Rule Traversal (PRT) in Algorithm 1 and Algorithm 2, respectively. Both algorithms take a standard constituency tree as input and deterministically produce a linearized sequence of structural and lexical tokens.

Crucially, the structural divergence between the two methods is dictated by their underlying data structures. BFT utilizes a standard First-In-First-Out (FIFO) queue to ensure strict level-by-level expansion. In contrast, PRT employs a Last-In-First-Out (LIFO) stack.

Algorithm 1 Breadth-First Traversal (BFT)

Input: T root of the constituency tree
Output: R sequence of traversal tokens

$Q \leftarrow []$ ▷ Empty queue
Q.enqueue(T)
$R \leftarrow [\text{OPEN}(T)]$ ▷ e.g., (X
while $Q \neq \emptyset$ **do**
  $N \leftarrow Q$.dequeue()
  for each child C **in** N.children **do**
    if type(C) = NonTerminal **then**
      Q.enqueue(C)
      R.append(OPEN(C))
    else
      R.append(C) ▷ Terminal token
    end if
  end for
  R.append(CLOSE(N)) ▷ e.g., X)
end while
return R

Algorithm 2 Production-Rule Traversal (PRT)

Input: T root of the constituency tree
Output: R sequence of traversal tokens

$S \leftarrow []$ ▷ Empty stack
S.push(T)
$R \leftarrow [\text{OPEN}(T)]$
while $S \neq \emptyset$ **do**
  $N \leftarrow S$.pop()
  for each child C **in** N.children (left-to-right) **do**
    if type(C) = NonTerminal **then**
      R.append(OPEN(C))
    else
      R.append(C)
    end if
  end for
  for each child C **in** N.children (right-to-left) **do**
    if type(C) = NonTerminal **then**
      S.push(C) ▷ Push for depth-first expansion
    end if
  end for
  R.append(CLOSE(N))
end while
return R

B Training and Architectural Details

Our Transformer Grammar models are implemented based on the modernized GPT-2 Small architecture. We adopt several modernized architectural choices standard in recent large language models, including SwiGLU activations (Shazeer, 2020), Rotary Position Embeddings (Su et al., 2024), and RMSNorm (Zhang and Sennrich, 2019) for pre-normalization. Furthermore, we tie the weights of the input embedding and the final output projection layer. To accelerate autoregressive generation during inference, we also implement a Key-Value (KV) cache mechanism.

The models are trained on a single NVIDIA A6000 GPU with mixed precision (`bfloat16`). We optimize the network using AdamW with a cosine learning rate decay schedule and a linear warmup phase. Detailed hyperparameters for both the architecture and the optimization process are comprehensively listed in Table 4.

Hyperparameter	Value
Model Architecture	
Number of layers	12
Hidden size (d_{model})	768
Attention heads	12
FFN intermediate size	1536
Max sequence length	2048
Vocabulary size	50,322
Activation function	SwiGLU
Normalization	RMSNorm
Position embeddings	RoPE
Bias (Linear & Norm)	True
Weight tying	True
Total Parameters	∼109M
Training & Optimization	
Optimizer	AdamW
Adam β	$(0.9, 0.95)$
Adam ϵ	1×10^{-8}
Peak learning rate	6×10^{-4}
Minimum learning rate	1×10^{-5}
Learning rate schedule	Cosine
Warmup steps	300
Weight decay	0.2
Gradient clipping norm	1.0
Global batch size	256
Microbatch size per device	16
Training epochs	15
Precision	`bfloat16`

Table 4: Architectural and optimization hyperparameters used for our TG models.

C Traditional Token Baseline Results

To contextualize the performance of the Transformer Grammars, we additionally train a standard text-only language model baseline (Token) operating purely on the unparsed text. This model shares identical architectural hyperparameters and training configurations as the main models evaluated in the text.

To facilitate a direct and comprehensive comparison, we present the full experimental results for all traversal methods and tree configurations alongside this traditional Token baseline. Specifically, Table 5 details the document-level language modeling performance measured by perplexity, while Table 6 reports the zero-shot syntactic generalization accuracy on the BLiMP benchmark. Finally, Table 7 provides the abstractive summarization ROUGE scores on the XSum dataset, where the Token baseline is replicated across all tree configuration columns for straightforward vertical comparison. Across all evaluations, the structural models clearly demonstrate superior performance over the purely sequential text baseline.

Model Variant	B-L	B-UL	N-L	N-UL
Token		21.90		
DFT-M	34.24	23.32	24.22	23.45
DFT-NM	**30.17**	20.57	22.06	**21.74**
DFT-tree	30.32	21.40	22.40	22.20
BFT-M	36.88	24.45	24.57	24.27
BFT-NM	32.00	22.11	22.82	22.76
BFT-tree	33.18	23.00	23.90	23.01
PRT-M	35.22	22.83	24.12	23.59
PRT-NM	30.79	**20.38**	**21.83**	21.87
PRT-tree	30.96	21.47	22.93	22.38

Table 5: Comprehensive perplexity (↓) results. The text-only Token baseline is provided as a universal reference.

Model Variant	N-L	N-UL	B-UL
Token		69.43	
DFT-M	**71.38**	**71.25**	**72.77**
DFT-NM	71.19	69.19	70.82
DFT-tree	70.38	69.37	69.69
BFT-M	69.86	70.84	71.21
BFT-NM	69.82	69.64	69.23
BFT-tree	68.79	67.59	69.12
PRT-M	70.90	71.12	71.86
PRT-NM	70.09	70.11	70.48
PRT-tree	69.24	69.16	70.24

Table 6: Comprehensive BLiMP accuracy (↑) results. All structural configurations are compared against the traditional Token baseline.

	N-L				N-UL				B-UL			
Model Variant	**R1**	**R2**	**RL**	**R-AVG**	**R1**	**R2**	**RL**	**R-AVG**	**R1**	**R2**	**RL**	**R-AVG**
Token	26.03	7.91	20.84	18.26	26.03	7.91	20.84	18.26	26.03	7.91	20.84	18.26
DFT-tree	30.77	10.04	24.10	21.64	30.26	10.10	24.02	21.46	29.31	9.52	23.16	20.66
DFT-M	26.61	7.85	21.22	18.56	27.38	8.36	21.88	19.21	25.26	7.09	19.99	17.44
DFT-NM	30.58	10.35	24.40	21.78	30.23	10.09	24.08	21.46	29.69	9.80	23.51	21.00
BFT-tree	30.47	10.06	24.07	21.53	30.63	9.97	23.96	21.52	29.88	9.42	23.15	20.02
BFT-M	27.46	7.98	21.56	19.00	27.28	7.92	21.44	18.88	26.18	7.15	20.31	17.88
BFT-NM	31.04	10.34	24.44	21.94	30.08	10.32	24.33	21.91	30.31	9.62	23.37	21.10
PRT-tree	**31.31**	**10.47**	24.64	22.14	30.45	10.02	23.95	21.47	30.03	9.56	23.35	20.98
PRT-M	27.50	8.18	21.75	19.14	27.34	7.97	21.50	18.94	26.01	7.42	20.39	17.94
PRT-NM	31.29	**10.47**	**24.71**	**22.16**	**31.33**	**10.54**	**24.65**	**22.17**	**30.65**	**10.33**	**23.96**	**21.65**

Table 7: Comprehensive ROUGE scores on the XSum dataset. The scores for the text-only Token baseline are duplicated across all tree configuration columns to facilitate direct vertical comparison.