

A Scaled-Up Empirical Study of Syntactic Language Models

Pengcheng Wang^{1,2} Zhexuan Zhang^{1,2} Chao Lou³ Kewei Tu^{1,2*}

¹School of Information Science and Technology, ShanghaiTech University

²Shanghai Engineering Research Center of Intelligent Vision and Imaging

³Independent researcher

{wangpch2024, zhangzhx2023, tukw}@shanghaitech.edu.cn

louchaobrn@outlook.com

Abstract

Syntactic language models (SLMs) that generate sentences along with their syntactic tree structures have shown promise in both syntactic generalization and downstream tasks. Prior studies, however, have mostly evaluated smaller models trained on less pretraining data and tested on narrower task suites. In addition, the design space of tree linearization and attention masks remains underexplored. We conduct a large-scale empirical study of 12 controlled configurations—one terminal baseline and 11 SLM variants spanning different linearizations and masks—at scales up to 1B parameters and 100B training tokens, evaluated across 11 downstream tasks and two syntactic benchmarks. The experimental results show that SLMs with standard causal attention match or exceed structural-mask variants across downstream and syntactic benchmarks. They require no architectural modifications and remain compatible with FlashAttention. We also observe that when scaling up to the 1B-parameter scale, the top-performing SLM variant no longer shows a clear advantage over a compute-matched non-structural baseline, suggesting that SLMs may become less competitive with scale. Our code is available at <https://github.com/qlwpc/TG-Interpolation>.

1 Introduction

Standard causal language models process text as flat token sequences. Although they can learn distributional proxies for syntax (Hu et al., 2020), hierarchical constituency structures—phrases composing into sentences—are left implicit. Syntactic language models (SLMs) inject syntactic biases into Transformers (Vaswani et al., 2017). One approach jointly models surface sentences and linearized versions of their syntactic parse trees (Sartran et al., 2022; Zhao et al., 2024). These models typically employ *structural* attention masks that allow a non-terminal to compose its child tokens once they are

generated by focusing its attention on them. Zhao et al. (2025) conduct a systematic empirical comparison of various design choices for SLMs, but they do not fully explore the complete design space of tree linearization and attention masks. Moreover, prior SLM studies have mostly evaluated models up to 300M parameters, trained on up to 9B tokens, and tested on only two or three tasks.

In this paper, we conduct a larger-scale empirical study of SLMs, evaluating selected models with up to 1B parameters and approximately 100B terminal training tokens on 11 downstream tasks and two syntactic benchmarks. We investigate a design space of 12 configurations: one terminal baseline and 11 SLM variants with different tree linearizations and attention masks. Our BBC experiments cover the full family at 100M and eight configurations at 500M on deduplicated data, with pause-token controls in both settings. We then study causal tree models and pause-token baselines at 1B on FineWeb-Edu.

Our findings are as follows: (i) Tree linearization with standard causal attention provides strong generation and syntactic performance. In both BBC settings, Tree and TGTREE outperform structural-mask variants on XSum; TGTREE also achieves the highest BLIMP score at 500M. Hybrid attention improves document perplexity, showing that the preferred attention mechanism depends on the objective. (ii) At 1B on FineWeb-Edu, TGTREE and Pause-2 have similar average downstream accuracy, while tree models perform better on syntactic evaluation. Explicit structure thus remains useful, but its benefits must be assessed against the additional computation introduced by structural tokens.

Our contributions are:

- We scale SLM evaluation to 1B parameters and approximately 100B terminal training tokens, across 11 downstream tasks and two syntactic benchmarks.

*Corresponding author.

- We systematically examine tree linearization and attention design on BBC News at 100M and deduplicated BBC News at 500M.
- We compare SLMs with pause-token baselines to assess the benefits of explicit structure alongside additional token positions.

2 Related Work

Prior work on syntactic language models (SLMs) has explored a broad range of ways to incorporate linguistic structure into neural sequence modeling. Transformer Grammars (TG) (Sartran et al., 2022) encode constituency trees as a sequence of actions and enforce hierarchical attention patterns. A subsequent study extends the idea to dependency trees (Zhao et al., 2024). Building on this line of work, Zhao et al. (2025) provide the most systematic comparison of SLM design choices including those of tree linearization and composition mechanisms. These studies show that syntactic inductive bias can improve language modeling and syntactic evaluation, but they typically couple modeling of linearized trees with specialized attention mechanisms.

Other studies incorporate syntactic inductive bias through alternative architectural mechanisms rather than explicit representations. These approaches include latent structure induction during pretraining (Hu et al., 2024), stack-based recursive state tracking (Murty et al., 2023), and dependency-graph-informed attention mechanisms (Huang et al., 2026).

3 Variants of Syntactic Language Models

3.1 Syntactic Language Models

Let $\mathbf{x} = (x_1, \dots, x_N)$ be a sentence of N terminal tokens, and let $\mathbf{y}$ be a labeled constituency parse tree for $\mathbf{x}$, where each node carries a phrase category (e.g., S, NP, or VP) and spans a contiguous subsequence of $\mathbf{x}$. A syntactic language model predicts the sentence and its structural representation through a linearized token sequence $\mathbf{z} = (z_1, \dots, z_T)$, factorized autoregressively as

$$p(\mathbf{z} := \text{LINEARIZE}(\mathbf{x}, \mathbf{y})) = \prod_{t=1}^T p(z_t \mid z_{<t}) \quad (1)$$

where LINEARIZE maps the sentence–tree pair into a flat sequence. When the mapping is one-to-one, $p(\mathbf{z})$ is the joint probability $p(\mathbf{x}, \mathbf{y})$; lossy linearizations can assign the same sequence to multiple trees.

Prior SLMs vary in the **linearization** that serializes the tree $\mathbf{y}$ into $\mathbf{z}$, and the **architecture** that parameterizes the conditional probabilities $p(z_t \mid z_{<t})$. In Transformer-based SLMs, this architectural choice is largely realized through the **attention mechanism**, which governs how syntactic relationships among tokens in $\mathbf{z}$ are processed.

3.2 Tree Linearization

The function $\text{LINEARIZE}(\mathbf{x}, \mathbf{y})$ embeds sentence $\mathbf{x}$ and tree $\mathbf{y}$ into token sequence $\mathbf{z}$. We explore several variants of linearization, all of which share a simple recursive form:

$$\begin{aligned} \text{LIN}(v) &:= \text{LEFT}(v) \\ &\quad \circ [\text{LIN}(c) \text{ for } c \text{ IN CHILDREN}(v)] \\ &\quad \circ \text{RIGHT}(v) \end{aligned} \quad (2)$$

$$\text{LIN}(t) := t \quad (3)$$

where $v \in \mathbf{y}$ is a node, $t \in \mathbf{x}$ is a terminal token, and $\circ$ denotes concatenation. $\text{LEFT}(v)$ and $\text{RIGHT}(v)$ produce *structural tokens* to enclose the linearized children. $\text{CHILDREN}(v)$ returns the ordered children of v in the tree.

Table 1 summarizes all linearization variants that we study. LIN^1 , LIN^2 and LIN^3 are three variants of linearization, differing only in the number of structural tokens emitted by $\text{RIGHT}(v)$. Specifically, LIN^1 generates sequences resembling the widely used bracket notation for constituency trees, but with additionally labeled closing brackets. LIN^2 was proposed by Sartran et al. (2022) to define their structural attention mask (referred to as STACK/COMPOSE in our paper). LIN^3 is a variant designed to test whether providing more closure tokens helps. We also include two additional variants, $\text{LIN}_{\text{ont}}^1$ and $\text{LIN}_{\text{merge}}^1$, to enable a more systematic comparison. Other variations (e.g., different traversal orders) are not included because they would introduce confounds that distract from our central goal: disentangling tree linearization from the attention mask design.

Different linearizations introduce varying numbers of structural tokens, inherently altering the total sequence length for training and inference (Table 2). To assess the contribution of structural information alongside additional computation, we include three non-structural controls. First, $\text{LIN}_{\text{shuf}}^1$ shuffles the structural tokens while preserving terminal order. The other two, Pause-1 and Pause-2, insert one or two copies of a dedicated learned

Name	Left(v)	Right(v)	Post-process	Example
Linearization				
LIN^1	$ONT(v)$	$CNT(v)$	-	(S Cats (VP chase mice VP) S)
LIN^2	$ONT(v)$	$2 \times CNT(v)$	-	(S Cats (VP chase mice VP) VP) S) S)
LIN^3	$ONT(v)$	$3 \times CNT(v)$	-	(S Cats (VP chase mice VP) VP) VP) S) S) S)
LIN^1_{-ont}	$\emptyset$	$CNT(v)$	-	Cats chase mice VP) S)
LIN^1_{-merge}	$ONT(v)$	$CNT(v)$	Merge CNTs	(S Cats (VP chase mice)
Non-structural				
LIN^0	$\emptyset$	$\emptyset$	-	Cats chase mice
LIN^1_{shuf}	$ONT(v)$	$CNT(v)$	Shuffle	(S VP) Cats (VP chase S) mice
Pause-1	$\emptyset$	$\emptyset$	Insert 1 <PAUSE>	Cats <PAUSE> chase <PAUSE> mice
Pause-2	$\emptyset$	$\emptyset$	Insert 2 <PAUSE>	Cats <PAUSE> <PAUSE> chase <PAUSE> <PAUSE> mice

Table 1: Overview of linearizations and non-structural token sequences used in this paper. “Post-process” refers to operations applied after linearization. ONT and CNT denote opening (e.g., (S and (VP) and closing non-terminal tokens (e.g., S) and VP)), respectively. v indicates that the emitted token carries a node label. “Merged CNT” signifies that consecutive CNTs (ignoring labels) are collapsed into a single) token. Finally, LIN^1_{shuf} is generated by shuffling the LIN^1 token sequence while preserving the original terminal order.

PAUSE token (Goyal et al., 2024) after each terminal, providing approximately comparable token budgets.

Dataset	LIN^0	LIN^1	LIN^2
BBC News	0.801B	1.963B	2.545B
FineWeb-Edu	~98B	~233B	~301B

Table 2: Corpus tokens by linearization, including special tokens. BBC counts use the deduplicated reference corpus. Historical training exposure, including repeated articles, is reported in Appendix D.

3.3 Structural Attention

A Transformer-based SLM can also incorporate structural information directly into the model itself via specially-designed attention mechanisms. We study three attention mechanisms as listed below.

Causal Attention The joint $p(\mathbf{x}, \mathbf{y})$ is modeled through standard autoregressive factorization over $\mathbf{z} = \text{LINEARIZE}(\mathbf{x}, \mathbf{y})$ with *no* modification to the attention mechanism. Constituency structures are encoded entirely in the token sequence; the model learns to route information through Opening Non-Terminal (ONT) and Closing Non-Terminal (CNT) using standard causal attention.

Structural Attention Structural attention factorizes $p(\mathbf{x}, \mathbf{y})$ autoregressively over the linearized tree sequence $\mathbf{z} = \text{LIN}^2(\mathbf{x}, \mathbf{y})$ and uses structure-dependent masks. In the original Transformer Grammar (TG) formulation (Sartran et al., 2022),

each CNT is duplicated into two actions. CNT_1 performs COMPOSE attention over only the current constituent’s children, producing a composed representation. CNT_2 performs STACK attention over only the representations on the active parse stack. TG then hides the child representations from later positions, forcing them to access the constituent through CNT_1 . Zhao et al. (2025) isolate this post-composition masking and show that removing it yields consistently better performance. We call the resulting variant **Nomask**: it retains the COMPOSE mask at CNT_1 but gives all other positions causal access to earlier terminals, ONT s, and CNT_1 representations. Thus, “no mask” refers specifically to removing TG’s post-composition mask, not to unrestricted bidirectional attention. Our default Nomask still hides CNT_2 states from later positions; Nomask-Aug exposes them as well.

Hybrid Attention The hybrid attention mechanism factorizes $p(\mathbf{x}, \mathbf{y})$ autoregressively over $\mathbf{z} = \text{LIN}^2(\mathbf{x}, \mathbf{y})$ and replaces standard causal attention with **head-level mask mixing**: rather than applying a single mask type across all attention heads, different heads receive different masks. Hybrid attention enables the model to combine multiple structural attention patterns within a single Transformer layer. Since enumerating all possible head-level mask assignments is computationally infeasible, we restrict ourselves to a straightforward setting: half of the attention heads receive one type of structural mask, and the remaining heads use another (either

#	Variant	Linearization	Attention
1	Terminal	LIN ⁰	Causal
2	Tree	LIN ¹	Causal
3	TGTree	LIN ²	Causal
4	TG	LIN ²	STACK/COMPOSE
5	TGNomask	LIN ²	Nomask
6	TGNomask-Aug*	LIN ²	Nomask
7	Tree-NoONT	LIN ¹ _{ont}	Causal
8	Tree-Compress	LIN ¹ _{merge}	Causal
9	Tree-TripleCNT	LIN ³	Causal
10	Tree-Shuffle	LIN ¹ _{shuf}	Causal
11	TGNomask-Mix-TG	LIN ²	Mixing heads (Nomask + STACK/COMPOSE)
12	TGTree-Mix-TG	LIN ²	Mixing heads (Causal + STACK/COMPOSE)

Table 3: The 12 configurations in our controlled design space: one terminal baseline and 11 SLM variants. ONT = opening non-terminal, CNT = closing non-terminal. “Hybrid” denotes mixing-head models. *TGNomask-Aug differs from TGNomask: CNT₂ remains visible to subsequent attention in Aug, while TGNomask prevents subsequent tokens from attending to it.

causal or structural). This configuration provides a tractable way to study hybrid attention.

Table 3 summarizes the 12 controlled configurations across the tree-linearization and attention-mask designs studied in our framework.

4 Terminal Format Evaluation

To evaluate SLMs on multiple-choice (MC) tasks, we follow the OLMES completion/cloze formulation (Gu et al., 2025) with multi-shot in-context demonstrations. Each answer is treated as a candidate completion, rather than as a label predicted from a displayed option list. We parse both the context and continuation with Benepar (Kitaev and Klein, 2018) and use their respective 1-best constituency parses. Appendix B.5 provides the complete input construction.

SLM continuations interleave terminal content with structural ONT and CNT tokens, creating a choice between scoring the complete sequence and scoring only terminal tokens. Let $\mathbf{p}$ denote the evaluation context and $\mathbf{c}_j$ the linearized continuation for candidate j . Let $\mathcal{T}(\mathbf{c}_j)$ and $\mathcal{N}(\mathbf{c}_j)$ denote its terminal and non-terminal indices, respectively. The full score includes both token types, whereas the terminal score includes only answer-content tokens:

$$S_{\text{full}}(\mathbf{c}_j | \mathbf{p}) = \sum_{t \in \mathcal{T} \cup \mathcal{N}} \log p(c_{j,t} | \mathbf{p}, \mathbf{c}_{j,<t}) \quad (4)$$

$$S_{\text{term}}(\mathbf{c}_j | \mathbf{p}) = \sum_{t \in \mathcal{T}} \log p(c_{j,t} | \mathbf{p}, \mathbf{c}_{j,<t}) \quad (5)$$

We use terminal scoring to focus the comparison on answer content while retaining the parse as context. For prediction, we apply the task-specific normalization to S_{term} : tasks using length normalization divide by the terminal continuation’s character count, and the remaining tasks use the raw sum. The candidate with the highest resulting score is selected. Appendix F compares the two raw scores on an eight-task suite and describes its overlap with the main evaluation.

5 Experiments

5.1 Setup

We train models from scratch for one epoch over each run’s training stream using the same OLMo-based (Groeneveld et al., 2024) decoder-only Transformer family. Learning rates and batch sizes follow the Step Law (Li et al., 2025) using each representation’s actual training token budget. Causal models use FlashAttention-2 (Dao, 2024), while structural and hybrid models use specialized attention kernels. Appendix A provides the architectures and training hyperparameters.

Table 3 defines the design space for our linearization and attention-mask ablations. We compare SLMs with pause-token baselines (Goyal et al., 2024): PAUSE- $K = (x_1, \underbrace{p, \dots, p}_K, x_2, \underbrace{p, \dots, p}_K, x_3, \dots)$, where p is a dedicated learned PAUSE token. Pause-1 doubles the terminal sequence length,

compared with approximately $2.4\times$ for LIN¹; Pause-2 triples it, compared with approximately $3.2\times$ for LIN².

We also include Pushdown (Murty et al., 2023) as an architectural reference in the 100M experiment. It jointly trains a language model and an attachment parser, using the predicted stack depth to bias attention. Our reimplementation shares the OLMo backbone with the other models.

5.2 Evaluation Methods

Document-Level Language Modeling. Computing sentence probabilities for an SLM requires marginalizing over syntactic representations. Following Sartran et al. (2022) and Zhao et al. (2025), we obtain up to 300 valid proposals from a CRF constituency parser (Kitaev and Klein, 2018). Let $\mathcal{Z}'_i$ be the set of distinct token sequences obtained by linearizing these proposals for sentence $\mathbf{x}_i$. We approximate its probability by

$$\tilde{p}(\mathbf{x}_i | \mathbf{h}_i) = \sum_{\mathbf{z} \in \mathcal{Z}'_i} p(\mathbf{z} | \mathbf{h}_i). \quad (6)$$

Here, $\mathbf{h}_i$ is the preceding document context. Each term scores all tokens in the linearized sequence (Eq. 1); the sum excludes padding and uses no parser weights. Deduplication follows each model’s linearization, so proposals that collapse under NoONT or Compress contribute only once.

To avoid marginalizing over all preceding histories, we retain the highest-probability sequence for each completed sentence:

$$\hat{\mathbf{z}}_j = \arg \max_{\mathbf{z} \in \mathcal{Z}'_j} p(\mathbf{z} | \mathbf{h}_j). \quad (7)$$

The selected sequences form the prefix $\mathbf{h}_i = \hat{\mathbf{z}}_1 \circ \dots \circ \hat{\mathbf{z}}_{i-1}$. We marginalize only the current sentence. For this fixed history, truncation gives a lower bound on sentence probability, rather than on the full document marginal.

For a corpus of D documents, let $\mathbf{X}_d = (\mathbf{x}_{d,1}, \dots, \mathbf{x}_{d,N_d})$ denote document d , where $\mathbf{x}_{d,i}$ is its i -th sentence. Document-level perplexity is then

$$\mathcal{L}_K = \sum_{d=1}^D \sum_{i=1}^{N_d} \log \tilde{p}(\mathbf{x}_{d,i} | \mathbf{h}_{d,i}),$$

$$\text{PPL}_{\text{syn}} = \exp(-\mathcal{L}_K/M). \quad (8)$$

Here, M counts terminal and EOS tokens across the corpus. Terminal and pause-token models use

their terminal-token likelihoods with the same denominator. Appendix B.4 gives model-specific details and the candidate-uniqueness audit.

Evaluations. We measure downstream performance with task-specific fine-tuning on XSum summarization (Narayan et al., 2018), reporting ROUGE-1/2/L (Lin, 2004), and on BoolQ (Clark et al., 2019) for boolean question answering. For each fixed BBC pretrained checkpoint, we run five independent fine-tuning seeds and report mean $\pm$ sample standard deviation for both tasks. For FineWeb-Edu models, we adopt the OLMES task formats and completion/cloze formulation (Gu et al., 2025). Our 11-task suite contains WinoGrande (Sakaguchi et al., 2020), HellaSwag (Zellers et al., 2019), BoolQ, MMLU (Hendrycks et al., 2021), MMLU-Redux (Gema et al., 2025), OpenBookQA (Mihaylov et al., 2018), CommonsenseQA (Talmor et al., 2019), SocialIQA (Sap et al., 2019), ARC-Easy and ARC-Challenge (Clark et al., 2018), and PIQA (Bisk et al., 2020). We additionally evaluate syntactic competence on SG (Hu et al., 2020) and BLiMP (Warstadt et al., 2020).

5.3 Experiment 1: Disentangling Tree Linearization and Attention

We study two BBC News training settings, using articles from FineWeb (Penedo et al., 2024) and constituency parses from Benepar (Kitaev and Klein, 2018). The deduplicated reference corpus contains 1.47M articles and 0.801B terminal tokens (Table 2). At 100M, we evaluate all 12 controlled configurations, two pause-token controls, and Pushdown on the original stream, including repeated and near-duplicate articles. At 500M, we evaluate eight configurations and two pause-token controls on the deduplicated version. Table 4 reports both experiments. We compare variants within each setting; differences in article versions, cumulative training exposure, and optimization prevent attributing cross-setting changes to model scale alone.

The dedup runs use reserved-clean development and test sets of 1,000 and 5,000 documents. We evaluate Doc-PPL for both settings on this same test set, replacing the earlier PPL split after identifying substantial training overlap. Appendix D describes the data audit and additional 500M experiments on the original corpus.

Model	XSum $\uparrow$	BoolQ $\uparrow$	SG $\uparrow$	BLiMP $\uparrow$	Doc-PPL $\downarrow$
BBC 100M: original training corpus					
Terminal-100M	22.02 $\pm$ 0.06	66.70 $\pm$ 0.36	69.80	70.77	15.369
Tree-100M	22.84 $\pm$ 0.03	68.07 $\pm$ 0.35	81.40	80.95	14.736
TGTree-100M	23.03 $\pm$ 0.02	68.12 $\pm$ 0.62	79.47	81.46	14.709
TG-100M	20.15 $\pm$ 0.04	64.50 $\pm$ 0.40	79.19	83.28	16.405
TGNomask-100M	21.91 $\pm$ 0.04	66.66 $\pm$ 0.45	78.62	81.78	14.658
TGNomask-Aug-100M	22.12 $\pm$ 0.02	67.01 $\pm$ 0.28	78.68	83.63	14.700
Tree-NoONT-100M	22.41 $\pm$ 0.04	68.67 $\pm$ 0.32	85.38	81.00 ‡	15.079
Tree-Compress-100M	22.70 $\pm$ 0.05	68.48 $\pm$ 0.49	78.27	81.40 ‡	14.748
Tree-TripleCNT-100M	22.43 $\pm$ 0.04	68.09 $\pm$ 0.12	81.57	83.42	14.801
Tree-Shuffle-100M †	21.56 $\pm$ 0.09	68.08 $\pm$ 0.33	76.56	67.77	17.852
TGNomask-Mix-TG-100M	21.87 $\pm$ 0.04	66.13 $\pm$ 0.54	75.77	83.14	14.573
TGTree-Mix-TG-100M	22.16 $\pm$ 0.07	67.91 $\pm$ 0.36	78.84	82.75	14.482
Pause-1-100M	22.38 $\pm$ 0.05	62.04 $\pm$ 0.19	75.11	70.85	14.916
Pause-2-100M	22.25 $\pm$ 0.06	65.70 $\pm$ 3.45	76.97	72.59	15.116
Pushdown-100M †	21.06 $\pm$ 0.05	65.54 $\pm$ 0.29	74.86	74.97	19.316
BBC 500M: deduplicated training corpus					
Terminal-500M	23.11 $\pm$ 0.07	70.54 $\pm$ 0.51	80.88	72.31	14.029
Tree-500M	23.57 $\pm$ 0.06	70.05 $\pm$ 0.58	83.43	82.76	13.864
TGTree-500M	23.59 $\pm$ 0.05	68.43 $\pm$ 0.26	80.68	83.72	13.947
Pause-1-500M	23.42 $\pm$ 0.06	70.26 $\pm$ 0.41	79.00	70.85	14.019
Pause-2-500M	23.38 $\pm$ 0.02	68.86 $\pm$ 0.32	78.18	72.69	14.030
Tree-NoONT-500M	23.30 $\pm$ 0.03	70.24 $\pm$ 0.26	83.67	81.10 ‡	13.955
TG-500M	20.67 $\pm$ 0.07	65.55 $\pm$ 0.30	79.55	83.32	15.465
TGNomask-500M	22.14 $\pm$ 0.07	69.54 $\pm$ 0.61	81.58	83.21	13.995
TGTree-Mix-TG-500M	23.31 $\pm$ 0.04	69.41 $\pm$ 0.33	79.76	83.37	13.785
TGNomask-Mix-TG-500M	22.37 $\pm$ 0.06	69.06 $\pm$ 0.29	80.11	83.22	13.973

Table 4: BBC results at 100M and dedup 500M. XSum reports R-AVG; XSum and BoolQ show mean $\pm$ sample SD over five fine-tuning seeds. Other metrics are single evaluations. Best values within each training regime are in **bold**. All Doc-PPL values use the same reserved-clean test; NoONT/Compress candidates are deduplicated after linearization. The shaded Pushdown row is an architectural reference. † marks model-specific PPL protocols; ‡ marks BLiMP scores whose projected candidate uniqueness remains unchecked (Appendix B.4).

Structural Attention vs. Causal Attention.

Tree and TGTree achieve the highest XSum means in both BBC settings. At 100M, they exceed TGNomask-Aug and TG, whereas TGNomask-Aug has the highest BLiMP score. At dedup 500M, Tree and TGTree exceed TGNomask by more than 1.4 XSum points and TG by more than 2.8 points; TGTree also leads on BLiMP. Removing TG’s post-composition mask improves XSum, BoolQ, and perplexity in both settings, consistent with Zhao et al. (2025). Causal attention does not lead on every task: TGNomask exceeds TGTree on SG and BoolQ at 500M, and Terminal has the highest BoolQ mean in that setting.

Hybrid Attention. Mixing causal and structural heads improves TGTree’s document perplexity from 14.709 to 14.482 at 100M and from 13.947 to 13.785 at dedup 500M, giving the lowest value in each setting. At 500M, it also improves BoolQ by 0.98 points, but lowers XSum, SG, and BLiMP. Mixing TGNomask and TG heads likewise low-

ers perplexity in both settings. Hybrid attention is therefore useful for language modeling, even though it does not improve on causal Tree models for summarization.

Pause-token Controls. Pause tokens improve XSum over Terminal in both settings, indicating that additional token positions can help generation. Tree and TGTree nevertheless exceed their corresponding pause controls on XSum, SG, BLiMP, and document perplexity. At 500M, both tree models gain more than 11 BLiMP points. BoolQ favors the tree models at 100M but the pause controls at 500M, by 0.21 and 0.43 points. Thus, explicit structure offers benefits beyond those observed with pause tokens, particularly on syntactic evaluation.

Document-Level Perplexity. Tree and TGTree improve on Terminal at 100M (14.736 and 14.709 vs. 15.369) and dedup 500M (13.864 and 13.947 vs. 14.029). The full STACK/COMPOSE mask performs worse, while TGNomask recovers much of the loss. After deduplicating candidate sequences,

NoONT obtains 15.079 at 100M and 13.955 at 500M; Compress obtains 14.748 at 100M. Both linearizations have higher perplexity than Tree in their respective settings. The lowest perplexity comes from hybrid attention, whereas the highest XSum mean comes from TGTre.

Tree Linearization Ablations. Removing opening non-terminals improves SG but lowers XSum in both settings, suggesting that opening tokens are more useful for generation than for the distinctions tested by SG. Doubling closing tokens gives the highest XSum mean in both experiments. At 100M, tripling them further improves SG and BLiMP over TGTre but lowers XSum; merging consecutive closing tokens lowers SG. Shuffling structural tokens causes a substantial BLiMP decline. The SG subtask results for dedup 500M appear in Appendix E.

Scaling Up. We study Tree and TGTre at 1B because of their generation performance and compatibility with efficient causal attention. They use the same attention architecture as Terminal and the pause-token baselines, enabling a direct comparison of input representations. Appendix A.2 reports throughput measurements for the earlier implementations.

5.4 Experiment 2: Scaling to 1B on FineWeb-Edu

FineWeb-Edu-100BT (Penedo et al., 2024) is a ~ 100 B-token sampled subset of the FineWeb-Edu corpus, filtered for educational quality and spanning diverse domains. We train five models at 1B parameters on this data: Terminal, Tree, TGTre, Pause-1 and Pause-2. Evaluation uses the 11-task OLMES suite (Gu et al., 2025), with the content-focused terminal score as the primary MC metric (§4), together with the SG and BLiMP syntactic benchmarks. Table 5 reports per-task downstream accuracy, and Table 6 reports the syntactic evaluation results. We quantify evaluation-set uncertainty using 10,000 paired bootstrap resamples; full confidence intervals and tests appear in Appendix C, and Appendix F reports sensitivity to full versus terminal scoring.

Downstream Performance. TGTre achieves the highest average accuracy across the 11 tasks (55.88), exceeding Terminal by 2.68 points. Its gains over Terminal are significant on five tasks under paired bootstrap tests. Tree improves the av-

erage by only 0.36 points and performs particularly poorly on HellaSwag, suggesting that the choice of linearization remains important at this scale.

Explicit Structure vs. Dummy Tokens. TGTre and Pause-2 have nearly identical average accuracy (55.88 vs. 55.80), with complementary strengths. TGTre is significantly better on BoolQ, while Pause-2 is significantly better on HellaSwag; differences on the other nine tasks are not significant at $\alpha = .05$. Excluding BoolQ puts Pause-2 ahead (55.04 vs. 54.71). These results show that a strong non-structural baseline can achieve similar aggregate downstream accuracy to an SLM under approximately comparable token budgets. Appendix C reports the paired comparisons, and Appendix F.1 examines an alternative task suite.

Syntactic Evaluation. Tree achieves the highest scores on both syntactic benchmarks, exceeding Terminal by 2.14 points on BLiMP and 1.94 points on SG (Table 6). TGTre also improves both scores. Pause tokens improve BLiMP but lower SG relative to Terminal, leaving a larger gap to the tree models on this benchmark. Thus, the similar downstream averages of TGTre and Pause-2 coexist with a syntactic advantage for explicit tree representations.

6 Discussion

6.1 Why Does Tree Linearization Help Without Structural Attention?

We propose three complementary hypotheses:

Implicit multi-task learning hypothesis. The model jointly predicts terminal tokens and syntactic structure, making constituency boundaries explicit through non-terminals instead of inferring them only from co-occurrence statistics. Sartran et al. (2022) attribute the perplexity gains of such joint modeling to the “additional syntactic supervision” that non-terminals provide by constraining admissible word classes at each position.

Compositional bottleneck hypothesis. CNT positions may act as “summary points” that encourage compression of constituent information. Under this hypothesis, causal attention allows the model to learn which information to route through CNT positions, unlike TG’s hard-coded COMPOSE attention. It can also preserve direct access to earlier words when a task requires lexical detail. We do not directly test this mechanism.

Model	WG↑	HS↑	BQ↑	MM↑	MMR↑	OBQA↑	CSQA↑	SIQA↑	ARCE↑	ARCC↑	PIQA↑	AVG↑
Terminal	58.17	61.53	58.65	36.79	37.61	43.60	53.24	48.98	69.30	43.48	73.88	53.20
Tree	61.64	54.10	63.85	36.88	37.53	44.00	56.02	48.46	68.42	45.15	73.07	53.56
TGTree	63.38	63.51	67.55	37.76	38.42	45.80	58.15	50.31	70.00	44.82	74.97	55.88
Pause-1	58.96	63.05	65.41	37.70	38.81	42.80	56.18	51.28	74.04	43.48	75.14	55.17
Pause-2	62.59	64.45	63.39	38.38	39.46	44.00	56.76	48.67	72.98	48.16	74.97	55.80

Table 5: Primary content-focused terminal-score accuracy at 1B on FineWeb-Edu, based on the scoring definition in Section 4. WG = WinoGrande, HS = HellaSwag, BQ = BoolQ, MM = MMLU, MMR = MMLU-Redux, OBQA = OpenBookQA, CSQA = CommonsenseQA, SIQA = SocialQA, ARCE = ARC-Easy, ARCC = ARC-Challenge. Best per task in **bold**; Table 10 reports 95% bootstrap intervals.

Model	BLiMP↑	SG↑
Terminal	79.65	88.01
Tree	81.79	89.95
TGTree	80.98	89.00
Pause-1	80.83	86.54
Pause-2	80.36	86.59

Table 6: Syntactic evaluation at 1B on FineWeb-Edu. For Tree and TGTree, SG uses word-synchronous beam search with a maximum linearized decoding length of six times the terminal sequence length.

Adaptive computation hypothesis. Pause- K allocates a fixed number of learned latent tokens after each terminal. Tree models instead use parse-dependent structural tokens: richer constituency structures introduce more ONT/CNT positions at phrase boundaries. This placement may allocate additional computation near constituent composition. The gains of both tree and pause-token models over Terminal motivate studying how the number and placement of these positions affect learning.

6.2 Syntactic Generalization and Downstream Performance

The best linearization differs between syntactic evaluation and downstream tasks. In both BBC settings, Tree-NoONT has the highest SG score, whereas TGTree has the highest XSum mean. On FineWeb-Edu, Tree leads on both syntactic benchmarks, but TGTree has the higher downstream average. Predicting additional structural tokens therefore does not improve all forms of syntactic generalization equally.

The pause-token comparisons further distinguish these outcomes. Pause-2 approaches TGTree’s average downstream accuracy at 1B, yet remains below it on SG and BLiMP. Additional token posi-

tions can support downstream performance without producing the same syntactic behavior as explicit tree tokens. This distinction motivates evaluating SLMs on both targeted syntactic tests and a broad downstream suite.

7 Conclusion

We conduct a scaled-up empirical study of syntactic language models, examining tree linearization and attention design and evaluating selected models up to 1B parameters. Across BBC 100M and dedup 500M, causal tree models achieve strong generation and syntactic performance while remaining compatible with FlashAttention; hybrid attention provides the best document perplexity. At 1B on FineWeb-Edu, TGTree and Pause-2 have similar aggregate downstream performance, while tree models retain an advantage on syntactic evaluation. These findings support tree linearization as a practical way to introduce syntactic bias and highlight the need to compare its benefits with those of additional non-structural computation.

Limitations

Constructing linearized tree inputs requires a parser for training and evaluation. This adds preprocessing cost, and parsing errors may propagate into the model. We do not quantify the effect of parse quality and study only English. The compositional bottleneck and adaptive computation accounts remain hypotheses; testing them would require interventions on the models’ internal representations or attention patterns.

Each pretraining configuration is represented by one run. The five BBC seeds capture fine-tuning variation, while the FineWeb-Edu bootstrap intervals capture evaluation uncertainty. Neither measures variation across pretraining runs. BBC rankings are descriptive, and the FineWeb-Edu per-task

tests are exploratory without multiple-comparison correction.

The historical BBC PPL test contains substantial training overlap. Our replacement splits were filtered against the historical corpus. A full token-level audit finds no exact document matches against the deduplicated training arrays; normalized-text, near-duplicate, and local-fragment checks against those arrays remain incomplete. The BBC and FineWeb-Edu experiments differ in data, token budgets, and optimization settings, so they do not isolate the effects of deduplication or scale. Pause-token controls approximately match sequence lengths rather than exact FLOPs. We do not evaluate structural-mask models at 1B on FineWeb-Edu.

Document PPL approximates the parse marginal using a finite candidate set and one selected history. The multiple-choice evaluation conditions on silver parses and omits structural-token likelihoods; its scoring analysis reuses several main-evaluation splits. Public code, configurations, and selected result evidence are available, while full data and checkpoint downloads remain pending.

Acknowledgments

This work was supported by the HPC Platform of ShanghaiTech University, the SIST Computing Platform and the Core Facility Platform of Computer Science and Communication, SIST, ShanghaiTech University.

References

- Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. [PIQA: Reasoning about physical commonsense in natural language](#). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7432–7439.
- Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. [BoolQ: Exploring the surprising difficulty of natural yes/no questions](#). 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 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.
- Tri Dao. 2024. [FlashAttention-2: Faster attention with better parallelism and work partitioning](#). In *International Conference on Learning Representations*, volume 2024, pages 35549–35562.
- Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. 2025. [FlexAttention: A programming model for generating fused attention variants](#). In *Proceedings of Machine Learning and Systems*, volume 7. MLSys.
- Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, Claire Barale, Robert McHardy, Joshua Harris, Jean Kaddour, Emile Van Krieken, and Pasquale Minervini. 2025. [Are we done with MMLU?](#) In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5069–5096, Albuquerque, New Mexico. Association for Computational Linguistics.
- Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. 2024. [Think before you speak: Training language models with pause tokens](#). In *International Conference on Learning Representations*, volume 2024, pages 27896–27923.
- Dirk Groeneveld, Iz Beltagy, Evan Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, and 24 others. 2024. [OLMo: Accelerating the science of language models](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15789–15809, Bangkok, Thailand. Association for Computational Linguistics.
- Yuling Gu, Oyvind Tafjord, Bailey Kuehl, Dany Hadad, Jesse Dodge, and Hannaneh Hajishirzi. 2025. [OLMES: A standard for language model evaluations](#). In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5020–5048, Albuquerque, New Mexico. Association for Computational Linguistics.
- John Hale. 2001. [A probabilistic Earley parser as a psycholinguistic model](#). In *Second Meeting of the North American Chapter of the Association for Computational Linguistics*.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. [Measuring massive multitask language understanding](#). In *International Conference on Learning Representations*.
- 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.
- Tianyu Huang, Yida Zhao, Chuyan Zhou, and Kewei Tu. 2026. [GiLT: Augmenting transformer language models with dependency graphs](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 31226–31237, San Diego, California, United States. Association for Computational Linguistics.
- Nikita Kitaev and Dan Klein. 2018. [Constituency parsing with a self-attentive encoder](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.
- Houyi Li, Wenzhen Zheng, Qiufeng Wang, Hanshan Zhang, Zili Wang, Shijie Xuyang, Yuantao Fan, Zhenyu Ding, Haoying Wang, Ning Ding, Shuigeng Zhou, Xiangyu Zhang, and Daxin Jiang. 2025. Predictable scale: Part i, step law–optimal hyperparameter scaling law in large language model pretraining. *arXiv preprint arXiv:2503.04715*.
- Chin-Yew Lin. 2004. [ROUGE: A package for automatic evaluation of summaries](#). In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.
- Ilya Loshchilov and Frank Hutter. 2019. [Decoupled weight decay regularization](#). In *International Conference on Learning Representations*.
- Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. [Can a suit of armor conduct electricity? a new dataset for open book question answering](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.
- Shikhar Murty, Pratyusha Sharma, Jacob Andreas, and Christopher Manning. 2023. [Pushdown layers: Encoding recursive structure in transformer language models](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3233–3247, Singapore. Association for Computational Linguistics.
- 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.
- Guilherme Penedo, Hynek Kydlíček, Loubna Ben al-lal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. [The fineweb datasets: Decanting the web for the finest text data at scale](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 30811–30849. Curran Associates, Inc.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.
- Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. [WinoGrande: An adversarial Winograd schema challenge at scale](#). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740.
- Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. [Social IQa: Commonsense reasoning about social interactions](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473, Hong Kong, China. 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. *arXiv 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.
- Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. [CommonsenseQA: A question answering challenge targeting commonsense knowledge](#). 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 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. [Qwen3 technical report](#). *arXiv preprint arXiv:2505.09388*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. [HellaSwag: Can a machine really finish your sentence?](#) In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Biao Zhang and Rico Sennrich. 2019. [Root mean square layer normalization](#). In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

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 Xue, 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 Training Details

A.1 Implementation and Hyperparameters

We use an OLMo-based (Groeneveld et al., 2024) decoder-only Transformer with SwiGLU (Shazeer, 2020) activation, RoPE (Su et al., 2024) positional encoding, RMSNorm (Zhang and Sennrich, 2019) for pre-normalization, and tied input and output embeddings. Table 7 lists the three scale configurations. Scale names refer to non-embedding parameters; the 500M models contain 508M non-embedding parameters and 579M total parameters with tied embeddings.

Scale	d_{model}	MLP ratio	Layers	Heads	Params
100M	768	4	12	12	~100M
500M	1408	8	16	16	~500M
1B	2048	8	16	16	~1B

Table 7: Model configurations at each scale.

We train all models from scratch for one epoch over each run’s training stream. Training uses the AdamW optimizer (Loshchilov and Hutter, 2019) with $\beta_1=0.9$, $\beta_2=0.95$, $\epsilon=10^{-8}$, weight decay of 0.1, and gradient clipping at norm of 1.0. We use cosine learning-rate decay with 2,000 warmup steps and a minimum learning rate of 10^{-5} . The maximum sequence length is 2,048 tokens except for dedup Pause-2, whose 683 terminal positions expand to 2,049 positions. The learning rate η and batch size B are set according to the Step Law (Li et al., 2025):

$$\begin{aligned}\eta(N, D) &= 1.79N^{-0.713}D^{0.307} \\ B(D) &= 0.58D^{0.571}\end{aligned}\tag{9}$$

where N is the number of non-embedding parameters and D is the actual training token budget, including repeated articles and structural or pause positions. This budget is distinct from the deduplicated BBC corpus size in Table 2. We convert $B(D)$ from tokens to sequences and round for distributed training. Table 8 gives the BBC 500M settings. All ten runs use seed 6198, BF16, and two DDP replicas, with a per-device microbatch of four and gradient accumulation to the listed global batch. Training processes all complete batches for one epoch. Tree uses 1.226 times as many input positions as Pause-1, and TGTREE uses 1.059 times as many as Pause-2; their learning rates and batch sizes also differ according to Eq. 9.

Data Preparation. The deduplicated BBC reference corpus contains 1,466,132 articles from 89 crawl shards spanning 2013–2022. We remove exact terminal-sequence, normalized-text, and near-duplicate articles from the historical parsed crawl pool, then encode retained articles with PTB mapping and explicit BOS/EOS tokens. The terminal representation contains 800,848,405 tokens (0.801B); tree representations add the structural tokens listed in Table 8. We evaluate the final checkpoint from each run. Training loss includes structural or pause targets, whereas document PPL is normalized by terminal and EOS tokens. Data manifests, tokenizer definitions, and checkpoint

Model	Training tokens D	Peak LR ($\times 10^{-4}$)	Global batch	Length	Final step
Terminal	800,848,405	6.012086	34	2048	11,501
Tree	1,963,388,153	7.917501	58	2048	16,529
TGTree	2,544,658,027	8.573605	66	2048	18,825
Pause-1	1,601,696,810	7.437747	52	2048	15,039
Pause-2	2,402,545,215	8.423672	64	2049	18,321
Tree-NoONT	1,382,118,279	7.108576	46	2048	14,670
TG	2,544,658,027	8.573605	66	2048	18,825
TGNomask	2,544,658,027	8.573605	66	2048	18,825
TGTree-Mix-TG	2,544,658,027	8.573605	66	2048	18,825
TGNomask-Mix-TG	2,544,658,027	8.573605	66	2048	18,825

Table 8: Training settings for the ten BBC dedup 500M models. Batch counts sequences, length counts input positions, and steps identify the final checkpoints used for evaluation. Pause budgets include inserted pause tokens.

Scale	Causal	Structural	Ratio
100M	62,175	37,975	$1.6\times$
500M	12,550	8,316	$1.5\times$

Table 9: Historical H800 training throughput (token/s), using FlashAttention-2 for causal and FlexAttention for structural masks. The historical 100M mixing-head measurement was 3,375 token/s (approximately $18\times$ slower than causal). These measurements do not characterize the new RTX 5090 typed-kernel runs.

identifiers are provided with the code.

A.2 Computational Costs

The deduplicated BBC models each train on two RTX 5090 GPUs. Causal models use FlashAttention-2, and TG, TGNomask, and hybrid models use typed structural-attention kernels. Table 9 reports throughput from the earlier H800 implementation, where structural models used FlexAttention (Dong et al., 2025). These measurements describe that implementation; throughput was not remeasured under a common setup for the RTX 5090 runs. Earlier campaign summaries report 133 hours on four H800 GPUs for the 100M variants, 125 hours on four H800 GPUs for the historical 500M campaign, and 385 hours on 64 A800 GPUs for the 1B variants.

A.3 Tokenizer and Structural Tokens

We use tokenizers with reserved non-terminal tokens (e.g., NP), (S, and VP)). The tokenizer is based on the GPT-2 tokenizer (Radford et al., 2019) or the Qwen3 tokenizer (Yang et al., 2025). For tree linearization models, opening non-terminals (ONT) and closing non-terminals (CNT) are represented as reserved tokens so that linearized constituency structure is preserved during pretraining and evaluation.

B Evaluation Setup Details

B.1 Downstream Fine-Tuning on BBC News: XSum and BoolQ

On BBC News, we evaluate XSum and BoolQ through task-specific fine-tuning from each pre-trained checkpoint, with the optimizer state reset. We use the full XSum test set (11,333 examples) and BoolQ validation set (3,270 questions). All models are fine-tuned with AdamW ($\beta_1=0.9$, $\beta_2=0.95$, weight decay 0.1) and a cosine learning rate schedule with 100 warmup steps decaying to 10^{-6} .

XSum. We fine-tune all models for 3 epochs with a peak learning rate of 6×10^{-5} and a global batch size of 40. During evaluation, we generate summaries with beam search using a beam size of 6 and a maximum length of 150 tokens. For tree linearization models, beam search proceeds at the word level (word-sync) with a maximum of 75 word steps and a grammar constraint of at most 4 consecutive opening non-terminals. ROUGE-1, ROUGE-2, and ROUGE-L are computed with the standard evaluate library; R-AVG denotes their arithmetic mean.

BoolQ. We fine-tune all models for 5 epochs with peak LR 3×10^{-4} , global batch size 40. We measure accuracy on the validation split.

The XSum and BoolQ entries in Tables 4 and 12 are means $\pm$ sample standard deviations over five fine-tuning seeds (42, 2026, 6198, 13171, and 31723). Pushdown uses right-binorized, unary-collapsed silver spans. Its XSum decoder conditions on the source parse and jointly searches summary tokens and attachment actions with beam size 6 and no limit on consecutive reductions. BoolQ uses teacher-forced candidate scoring with supplied parser spans.

B.2 Syntactic Generalization (SG)

Following Hu et al. (2020), we evaluate the six SG test suites without fine-tuning by comparing target-region surprisal (Hale, 2001). For a tagged span $\mathbf{w} = \mathbf{x}_{s:e}$ and its left context $\mathbf{C} = \mathbf{x}_{<s}$, we define

$$\begin{aligned} S(\mathbf{w} \mid \mathbf{C}) &= -\log p(\mathbf{w} \mid \mathbf{C}) \\ &= -\log p(\mathbf{x}_{\leq e}) + \log p(\mathbf{x}_{<s}). \end{aligned}$$

The logarithm base does not affect the SG predictions because the task-specific formulas only compare surprisals.

For terminal and pause-token models, we compute this span surprisal by summing token-level cross-entropy losses over the tagged span in one forward pass. For tree-linearization models other than Tree-Shuffle, we approximate prefix probabilities with DFS word-synchronous beam search (Stern et al., 2017). At each synchronized terminal boundary t , the beam $\mathcal{B}_t$ contains the b highest-scoring latent structural prefixes $\mathbf{y}_t^{(h)}$. We form the beam-truncated marginal and the corresponding span score as

$$\begin{aligned} \tilde{p}(\mathbf{x}_{\leq t}) &= \sum_{h \in \mathcal{B}_t} p(\mathbf{x}_{\leq t}, \mathbf{y}_t^{(h)}), \\ \tilde{S}(\mathbf{w} \mid \mathbf{C}) &= -\log \tilde{p}(\mathbf{x}_{\leq e}) \\ &\quad + \log \tilde{p}(\mathbf{x}_{<s}). \end{aligned}$$

These span surprisals are the scores supplied to the task-specific formulas of Hu et al. (2020).

At each terminal step, the decoder must emit the observed next terminal, but may first expand non-terminal hypotheses through depth-first search. Hypotheses synchronize after emitting the terminal, and the b highest-scoring hypotheses are retained. Following Murty et al. (2023) and Sartran et al. (2022), we set $b=300$; the within-step sub-beam is 300 and the terminal fast-track size is 30. All such SG evaluations cap the linearized decoding length at $\max(6L, 10)$, where L is the terminal-token sequence length. No separate limit is imposed on the total or consecutive number of non-terminal tokens.

Tree-Shuffle uses terminal-format teacher-forced scoring at test time. Pushdown instead incrementally marginalizes over attachment decisions with a beam size of 300.

B.3 BLiMP

BLiMP (Warstadt et al., 2020) evaluates whether models assign higher likelihood to grammatical

sentences in minimal pairs across 12 grammatical phenomena. For tree-linearization models, we obtain 300 parse proposals ($K = 300$) from the Benepar parser (Kitaev and Klein, 2018) and score each sentence by summing the joint probabilities of its candidate sequences. Tree-NoONT and Tree-Compress BLiMP scores have not been reevaluated with sequence deduplication; their reported values are marked accordingly. A pair is scored correct if the good sentence receives higher marginalized probability than the bad sentence. Each of the 67 subtasks contains 1,000 pairs. For terminal and pause-token models, $K = 1$ (no tree marginalization). Tree-Shuffle likewise uses terminal-format scoring with $K = 1$. Pushdown’s supplied-tree scorer sums token likelihoods conditioned on candidate attachment sequences, without attachment probabilities. Its historical BLiMP result is retained as a reference; the exact scoring snapshot for that run is unavailable, so we do not interpret it as joint token–attachment marginalization.

B.4 Model-Specific Document-PPL Protocols

For reserved-clean-v1, all 5,000 documents, 129,085 sentences, and 3,068,713 terminal/EOS tokens are scored. The base terminal array contains 3,073,713 positions; one BOS per document is context only. For Tree/TG candidates, 122,338 sentences have 300 unique valid candidates and 6,747 have 122. Repeated padding slots are excluded from Eq. 6. Candidate joint likelihoods follow each model’s label mask, with no parser weighting or division by candidate count.

After scoring the candidates for each sentence, we retain the highest-probability sequence as context for the next sentence, breaking ties by candidate order. Histories reset at document boundaries, and padding positions are excluded from the retained cache. Evaluation uses FP32 with SDPA and disables TF32. Candidate microbatches share the sentence’s padding length.

Tree-NoONT and Tree-Compress can map different proposal trees to the same token sequence. NoONT removes opening tokens, while Compress merges consecutive closing tokens and discards their number and labels. We deduplicate the complete, unpadded sequences within each sentence after applying these transformations, retaining the first occurrence. Each distinct sequence contributes once to Eq. 6, irrespective of how many proposal trees map to it. We reevaluate NoONT at 100M and dedup 500M and Compress at 100M under this

rule.

The full test contains 37,524,534 valid original proposals. NoONT has repeated projected sequences in 117,847 sentences, leaving 34,053,187 distinct candidates; Compress has repeats in 57,733 sentences, leaving 37,177,859. All candidate terminal projections and document boundaries are checked against the same test corpus. Their BLiMP candidates still need the corresponding check; we retain the recorded scores with markers and exclude them from claims about marginalized likelihood.

Tree-Shuffle uses the masked historical checkpoint with terminal-only scoring. Pushdown uses native n-ary attachment candidates, the v1 stack-legal action space, joint token-attachment sums, and model-best histories. These two models use distinct probability/candidate spaces and are contextual comparisons, not entries in the controlled Tree/TG PPL ranking. Neither is included in the new ten-model experiment.

The BBC 100M models in Table 4 and historical 500M checkpoints in Appendix D use the same reserved-clean PPL protocol. TripleCNT is evaluated with its corresponding grammar. The earlier PPL values, which used the overlapping test and parser-first histories, are excluded.

B.5 OLMES Multiple-Choice Evaluation on FineWeb-Edu

All 11 downstream tasks use the completion/cloze formulation standardized in OLMES (Gu et al., 2025). This formulation evaluates each answer string separately as a natural-language completion instead of predicting an answer label. All evaluations are multi-shot. We use 5-shot prompts for HellaSwag, WinoGrande, MMLU, MMLU-Redux, and OpenBookQA, and 3-shot prompts for BoolQ, ARC-Easy, ARC-Challenge, PIQA, SocialQA, and CommonsenseQA.

We preprocess the natural-language content of every demonstration, target prompt, and answer option with Benepar (Kitaev and Klein, 2018). For each parsed component x , we retain only its 1-best constituency parse $\hat{y}(x)$ as a silver tree. Depending on the task template, fields are parsed separately or the completed candidate sequence is parsed and split at the cloze boundary. Task-specific template text is represented with the same constituency-tree notation. The resulting trees are converted to the linearization required by each model. Thus, the syntactic inputs differ in representation but share the same underlying terminal prompt and answer

text.

For each demonstration, the parsed gold answer suffix is appended to its parsed prompt. These demonstrations precede the target instance in every candidate prefix $\mathbf{p}_j$. We then append the candidate-specific parsed suffix $\mathbf{c}_j$ and score only that suffix conditional on $\mathbf{p}_j$. The prediction is $\arg \max_j S(\mathbf{c}_j \mid \mathbf{p}_j)$, using the task-specific option-score normalization. The primary results use S_{term} in Eq. 5; Appendix F also reports S_{full} . For WinoGrande, the answer fills the blank in a candidate-specific prefix, while the common post-blank text is the scored suffix, following OLMES. Unlike BLiMP, this MC protocol uses one silver tree per textual component and uses neither multiple parse proposals nor tree marginalization.

C Bootstrap Confidence Intervals and Paired Tests

We estimate uncertainty for the 1B FineWeb-Edu downstream evaluation using $B = 10,000$ bootstrap resamples with seed 2027. For each task, examples are resampled with replacement and the same sampled indices are used for every model. Table 10 reports percentile 95% confidence intervals for accuracy. The macro-average intervals instead resample tasks to measure sensitivity to task-suite composition. With shared task indices across models, TGTree ranks first in 52.18% of resamples, Pause-2 in 42.27%, and Pause-1 in 5.55%. Excluding BoolQ changes these proportions to 27.00%, 67.30%, and 5.70%, respectively. Terminal and Tree never rank first in either comparison.

Table 11 reports the paired contrasts most relevant to our claims: TGTree versus the Terminal baseline and each tree model versus its approximately budget-matched pause-token baseline. We compute two-sided bootstrap p -values from the paired difference distribution. The tests are exploratory and uncorrected for multiple comparisons; $p < .05$ is used only as a descriptive threshold.

D Historical BBC Experiments and Clean Reevaluation

Historical BBC training comprises 15,187,326 article instances, including repeated and near-duplicate versions, with 10,061,025,584 terminal tokens of cumulative exposure (10.061B). The LIN^1 and LIN^2 budgets are approximately 24.71B and 32.03B input tokens. These training budgets

Task	Terminal	Tree	TGTree	Pause-1	Pause-2
WG	58.17 [55.49, 60.93]	61.64 [58.96, 64.33]	63.38 [60.69, 66.06]	58.96 [56.20, 61.64]	62.59 [59.91, 65.27]
HS	61.53 [60.57, 62.48]	54.10 [53.14, 55.09]	63.51 [62.57, 64.45]	63.05 [62.09, 63.99]	64.45 [63.51, 65.39]
BQ	58.65 [56.97, 60.34]	63.85 [62.23, 65.50]	67.55 [65.93, 69.11]	65.41 [63.82, 67.06]	63.39 [61.71, 65.08]
MM	36.79 [35.99, 37.59]	36.88 [36.06, 37.69]	37.76 [36.96, 38.56]	37.70 [36.88, 38.51]	38.38 [37.57, 39.18]
MMR	37.61 [36.37, 38.89]	37.53 [36.28, 38.77]	38.42 [37.18, 39.70]	38.81 [37.56, 40.07]	39.46 [38.23, 40.72]
OBQA	43.60 [39.20, 48.00]	44.00 [39.60, 48.40]	45.80 [41.40, 50.20]	42.80 [38.40, 47.00]	44.00 [39.60, 48.40]
CSQA	53.24 [50.45, 56.02]	56.02 [53.32, 58.80]	58.15 [55.36, 60.93]	56.18 [53.40, 58.97]	56.76 [53.97, 59.54]
SIQA	48.98 [46.83, 51.23]	48.46 [46.26, 50.67]	50.31 [48.16, 52.56]	51.28 [49.03, 53.58]	48.67 [46.47, 50.92]
ARCE	69.30 [65.44, 72.98]	68.42 [64.56, 72.11]	70.00 [66.14, 73.68]	74.04 [70.35, 77.54]	72.98 [69.30, 76.49]
ARCC	43.48 [38.13, 49.16]	45.15 [39.46, 50.84]	44.82 [39.13, 50.50]	43.48 [38.13, 49.16]	48.16 [42.47, 53.85]
PIQA	73.88 [71.87, 75.90]	73.07 [71.06, 75.08]	74.97 [72.96, 76.93]	75.14 [73.18, 77.09]	74.97 [73.01, 76.93]
AVG [†]	53.20 [46.38, 60.30]	53.56 [46.79, 60.42]	55.88 [48.39, 63.08]	55.17 [47.86, 62.66]	55.80 [48.68, 63.15]
AVG w/o BQ [†]	52.66 [45.29, 60.26]	52.53 [45.38, 60.00]	54.71 [47.18, 62.60]	54.14 [46.27, 62.17]	55.04 [47.46, 62.91]

Table 10: Accuracy point estimates and 95% bootstrap confidence intervals for the terminal-format 1B evaluation. Per-task rows use paired resampling of evaluation examples. [†]Macro-average intervals resample tasks and therefore measure task-suite sensitivity rather than uncertainty within a fixed suite. Task abbreviations follow Table 5.

Task	TGTree – Terminal	Tree – Pause-1	TGTree – Pause-2
WG	+5.21 [2.29, 8.13]; $p < .001$	+2.68 [−0.32, 5.68]; $p = .082$	+0.79 [−2.05, 3.63]; $p = .604$
HS	+1.98 [1.22, 2.74]; $p < .001$	−8.94 [−9.96, −7.89]; $p < .001$	−0.94 [−1.68, −0.19]; $p = .014$
BQ	+8.90 [6.48, 11.31]; $p < .001$	−1.56 [−3.27, 0.12]; $p = .074$	+4.16 [2.02, 6.30]; $p < .001$
MM	+0.97 [0.26, 1.67]; $p = .009$	−0.82 [−1.53, −0.09]; $p = .027$	−0.62 [−1.30, 0.08]; $p = .078$
MMR	+0.81 [−0.32, 1.96]; $p = .172$	−1.28 [−2.40, −0.18]; $p = .024$	−1.04 [−2.14, 0.05]; $p = .069$
OBQA	+2.20 [−1.40, 5.80]; $p = .262$	+1.20 [−2.80, 5.20]; $p = .592$	+1.80 [−1.80, 5.40]; $p = .367$
CSQA	+4.91 [2.46, 7.45]; $p < .001$	−0.16 [−2.46, 2.21]; $p = .934$	+1.39 [−0.90, 3.69]; $p = .248$
SIQA	+1.33 [−0.67, 3.38]; $p = .210$	−2.81 [−4.86, −0.77]; $p = .007$	+1.64 [−0.36, 3.68]; $p = .120$
ARCE	+0.70 [−3.16, 4.56]; $p = .761$	−5.61 [−9.30, −1.93]; $p = .002$	−2.98 [−6.67, 0.70]; $p = .113$
ARCC	+1.34 [−4.01, 6.69]; $p = .666$	+1.67 [−3.68, 6.69]; $p = .570$	−3.34 [−9.03, 2.34]; $p = .261$
PIQA	+1.09 [−0.65, 2.83]; $p = .247$	−2.07 [−3.92, −0.27]; $p = .024$	+0.00 [−1.74, 1.69]; $p = 1.000$

Table 11: Paired bootstrap differences in percentage points with 95% confidence intervals and uncorrected two-sided p -values. Positive values favor the model named first. Tree–Pause-1 and TGTree–Pause-2 are approximate budget comparisons.

differ from the deduplicated corpus size in Table 2. Tables 4 and 12 report the 100M and supplementary historical 500M models. TGNomask-Aug exposes CNT₂ representations to later tokens, unlike TGNomask in the deduplicated experiment.

A full token audit finds 2,071,964 distinct historical document sequences totaling 1,188,938,040 tokens (1.189B), each occurring 1–64 times. The 0.801B reference additionally merges normalized and near-duplicate versions and is separately encoded; it does not equal the exact unique content of the historical array. Both settings were trained for one epoch over their respective streams. Variable multiplicity does not correspond to uniform epochs over identical deduplicated examples.

An exact token comparison found complete training copies of 4,752 of the 4,966 original PPL test documents (95.69%). We replace those perplexities with evaluation on reserved-clean-v1, retaining the original XSum, BoolQ, SG, and BLiMP mea-

surements. The replacement dev/test splits were filtered against the historical corpus using exact, near-duplicate, and local-reuse checks. A separate full scan of all 1,466,132 deduplicated training documents finds no complete token-identical matches to the 1,000 development or 5,000 test documents. The NoONT training arrays have the same terminal projection as this scanned corpus. This exact-token result does not rule out normalized-text matches, near duplicates, or shared fragments. The deduplicated corpus was separately parsed and encoded, so it is not assumed to be a byte-identical subset of the historical data.

The historical Tree checkpoint has worse SG and clean PPL at 500M than at 100M (63.13 vs. 81.40 and 16.001 vs. 14.736). The deduplicated 500M Tree reaches 83.43 SG and 13.864 PPL. Because training data, budgets, and optimization settings changed together, this comparison does not identify the cause of the difference.

Historical model	XSum $\uparrow$	BoolQ $\uparrow$	SG $\uparrow$	BLiMP $\uparrow$	Clean Doc-PPL $\downarrow$
Terminal-500M	20.75 $\pm$ 0.06	69.69 $\pm$ 0.60	71.11	64.74	17.303
Tree-500M	22.25 $\pm$ 0.05	70.72 $\pm$ 0.35	63.13	76.09	16.001
TGTree-500M	22.01 $\pm$ 0.08	70.63 $\pm$ 0.32	65.95	77.65	15.895
TGNomask-Aug-500M	21.77 $\pm$ 0.06	70.78 $\pm$ 0.74	57.58	75.97	15.484

Table 12: Supplementary 500M models trained on the original BBC corpus. Doc-PPL is reevaluated on reserved-clean; the other metrics retain their original evaluations. XSum and BoolQ show means $\pm$ sample SD over five fine-tuning seeds. Best values are in **bold**. These models use a different training corpus from the dedup 500M models in Table 4.

E SG Subtask Breakdown

Table 13 breaks down SG performance for the ten deduplicated BBC models. Tree and TGTree both improve Agreement by 10.53 points over Terminal. Their differences emerge elsewhere: TGTree performs better on Garden Path Effects, while Tree performs better on Center Embedding and Long-Distance Dependencies. Tree-NoONT leads on Garden Path Effects and Licensing, which contribute to its highest overall SG score. TG and TGTree-Mix-TG have the highest Agreement scores but perform less well on Gross Syntactic Expectation and Long-Distance Dependencies.

Table 14 reports the corresponding breakdown for the 1B FineWeb-Edu models. Tree and TGTree use the maximum linearized decoding length of six times the terminal sequence length described in Appendix B.2.

F Full-versus-Terminal Score Sensitivity

We compare the raw full and terminal scores in Eq. 5 on an eight-task suite (Validation-8; 17,691 examples): WinoGrande, HellaSwag, OpenBookQA, CommonsenseQA, SocialIQA, ARC-Easy, ARC-Challenge, and PIQA. Except for OpenBookQA validation, these are the same splits used in the main evaluation. Fixed in-context demonstrations come from training data and do not overlap with the scored examples. Table 15 reports predictions based on raw log-probability sums for both models, without the task-specific length normalization used in Table 5.

Omitting structural-token scores raises the macro average by 2.16 points for Tree and 0.62 points for TGTree; the latter interval includes zero. For Tree, 7.46% of predictions change from wrong to correct and 5.30% change in the opposite direction. The corresponding TGTree rates are 4.92% and 4.30%. The benefit varies by task, with two TGTree tasks favoring full scoring. These results

characterize sensitivity to structural-token likelihoods under raw scoring. Because the analysis shares evaluation examples and uses different normalization, it does not independently validate the main scoring rule.

The two scores address related but distinct questions. S_{full} scores the candidate together with one parser-selected silver tree. S_{term} accumulates only terminal-token factors while retaining that same tree as context. Neither is the exact marginal sentence likelihood $\sum_y p(\mathbf{x}, y)$.

F.1 Alternative Ten-Task Comparison

Validation-10 adds BoolQ and MMLU validation to the eight tasks above, yielding 22,207 scored examples, and applies the task-specific terminal-score normalization. Eight tasks share 20,461 examples with the main evaluation (92.1% of this suite); OpenBookQA and MMLU use different splits. This is a sensitivity comparison rather than an independent evaluation set. For MMLU, five fixed demonstrations per subject are removed from scoring, leaving 1,246 examples across 57 subjects. MMLU accuracy is macro-averaged across subjects; other task accuracies are micro-averaged. The headline score averages the ten tasks, with 10,000 paired bootstrap resamples and seed 2026.

Model	Agr $\uparrow$	CtrEmb $\uparrow$	GrdnPath $\uparrow$	GrsSyn $\uparrow$	Lic $\uparrow$	LDD $\uparrow$	SG $\uparrow$
Terminal	64.91	91.07	78.29	94.57	78.95	77.52	80.88
Tree	75.44	92.86	80.26	91.30	80.45	80.28	83.43
TGTree	75.44	85.71	85.53	85.87	78.57	72.94	80.68
Pause-1	66.67	82.14	80.92	96.74	73.68	73.85	79.00
Pause-2	64.91	91.07	79.61	85.87	73.31	74.31	78.18
Tree-NoONT	68.42	91.07	88.16	92.39	87.22	74.77	83.67
TG	84.21	91.07	78.95	73.91	78.95	70.18	79.55
TGNomask	70.18	94.64	82.89	79.35	83.08	79.36	81.58
TGTree-Mix-TG	84.21	89.29	81.58	66.30	86.09	71.10	79.76
TGNomask-Mix-TG	78.95	91.07	77.63	75.00	82.33	75.69	80.11

Table 13: SG subtask breakdown at 500M on deduplicated BBC News. Agr = Agreement, CtrEmb = Center Embedding, GrdnPath = Garden Path Effects, GrsSyn = Gross Syntactic Expectation, Lic = Licensing, LDD = Long-Distance Dependencies. Best per metric in **bold**.

Model	Agr $\uparrow$	CtrEmb $\uparrow$	GrdnPath $\uparrow$	GrsSyn $\uparrow$	Lic $\uparrow$	LDD $\uparrow$	SG $\uparrow$
Terminal	87.72	94.74	93.42	98.91	74.81	78.44	88.01
Tree	85.96	100.00	98.03	100.00	78.20	77.52	89.95
TGTree	80.70	100.00	94.08	100.00	78.95	80.28	89.00
Pause-1	75.44	100.00	94.08	98.91	73.31	77.52	86.54
Pause-2	82.46	94.74	95.39	100.00	72.18	74.77	86.59

Table 14: SG subtask breakdown at 1B on FineWeb-Edu. Best per metric in **bold**; abbreviations follow Table 13.

Task	Tree			TGTree		
	Full	Term	Δ	Full	Term	Δ
WinoGrande	61.17	61.64	+0.47	64.64	63.38	-1.26
HellaSwag	37.83	43.92	+6.08	47.11	49.45	+2.34
OpenBookQA	33.60	33.60	+0.00	34.20	34.00	-0.20
CommonsenseQA	54.79	55.86	+1.06	57.17	57.49	+0.33
SocialIQA	43.65	45.60	+1.94	44.42	45.80	+1.38
ARC-Easy	71.05	74.74	+3.68	73.68	74.39	+0.70
ARC-Challenge	39.80	41.81	+2.01	38.80	39.80	+1.00
PIQA	70.24	72.25	+2.01	72.63	73.29	+0.65
Macro average	51.52	53.68	+2.16	54.08	54.70	+0.62

Table 15: Accuracy (%) using raw full and terminal scores on the eight-task sensitivity suite; Δ = Term - Full. Macro averages are unweighted. Paired 95% bootstrap intervals for the macro Δ are [1.35, 2.99] for Tree and [-0.09, 1.35] for TGTree (10,000 resamples; seed 2026).

Model	Macro accuracy	Δ vs. Terminal (95% CI)
Terminal	54.43	reference
Tree	54.75	+0.32 [-0.67, 1.30]
TGTree	57.35	+2.92 [1.95, 3.84]
Pause-1	56.47	+2.05 [1.15, 2.93]
Pause-2	57.09	+2.66 [1.82, 3.49]

Table 16: Terminal-score accuracy (%) on Validation-10. The paired TGTree-Pause-2 difference is +0.26 points (95% CI [-0.68, 1.20]).