A Scaled-Up Empirical Study of Syntactic Language Models

Pengcheng Wang[1,2] **Zhexuan Zhang**[1,2] **Chao Lou**[3] **Kewei Tu**[1,2]*
[1]School of Information Science and Technology, ShanghaiTech University
[2]Shanghai Engineering Research Center of Intelligent Vision and Imaging
[3]Independent 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) (Sartran et al., 2022; Hu et al., 2024; Murty et al., 2023; Zhao et al., 2024; Huang et al., 2026) inject syntactic biases into Transformers (Vaswani et al., 2017) using a simple technique: jointly modeling surface sentences and linearized versions of their syntactic parse trees. 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 SLMs with up to 1B parameters, up to 100B training tokens, a contemporary suite of 11 downstream tasks and two syntactic benchmarks. We also investigate new designs of tree linearization and attention masks in a controlled design space of 12 configurations: one terminal baseline and 11 SLM variants. Our findings are as follows: (i) SLMs with standard causal attention match or surpass structural-mask variants across downstream and syntactic benchmarks. They require no architectural modifications and remain compatible with FlashAttention. (ii) At 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.

In summary, our contributions include:

- We scale SLM evaluation to 1B parameters and 100B training tokens, across 11 downstream tasks and two syntactic benchmarks.
- We systematically examine a controlled family of 12 configurations (one terminal baseline and 11 SLM variants) with varying tree linearization and attention mask strategies, disentangling their impact on model performance.
- We include compute-matched comparisons between SLMs and non-structural baselines.

*Corresponding author.

- Our experiments produce interesting observations regarding SLM design and the competitiveness of SLMs at scale.

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 $\mathbf{x} = (x_1, \ldots, x_N)$ 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 defines a joint distribution $p(\mathbf{x}, \mathbf{y})$, factorized autoregressively over a linearized token sequence $\mathbf{z} = (z_1, \ldots, z_T)$:

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

where LINEARIZE maps the sentence–tree pair into a flat sequence. Under this formulation, prior SLMs can be understood as different ways of instantiating the same autoregressive modeling problem, varying 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 $\textsc{Linearize}(\mathbf{x}, \mathbf{y})$ 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:

$$\begin{aligned} \textsc{Lin}(v) := \; & \textsc{Left}(v) \\ & \circ \, [\, \textsc{Lin}(c) \text{ for } c \text{ in } \textsc{children}(v) \,] \\ & \circ \, \textsc{Right}(v) \end{aligned} \quad (2)$$

$$\textsc{Lin}(t) := t \quad (3)$$

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

Table 1 summarizes all linearization variants that we study. $\textsc{Lin}^1$, $\textsc{Lin}^2$ and $\textsc{Lin}^3$ are three variants of linearization, differing only in the number of structural tokens emitted by $\textsc{Right}(v)$. Specifically, $\textsc{Lin}^1$ generates sequences resembling the widely used bracket notation for constituency trees, but with additionally labeled closing brackets. $\textsc{Lin}^2$ was proposed by Sartran et al. (2022) to define their structural attention mask (referred to as STACK/COMPOSE in our paper). $\textsc{Lin}^3$ is a variant designed to test whether providing more closure tokens helps. We also include two additional variants, $\textsc{Lin}^1_{\text{–ont}}$ and $\textsc{Lin}^1_{\text{–merge}}$, 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 distinguish performance gains due to syntactic structure from gains due simply to increased computation, we include three token-budget-matched baselines alongside the naive terminal sequence. First, $\textsc{Lin}^1_{\text{shuf}}$ shuffles the structural tokens to serve as a control for the necessity of valid structural information. The other two, Pause-1 and Pause-2, insert one or two copies of a dedicated learned PAUSE token (Goyal et al., 2024) after each terminal.

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×CNT(v)	-	`(S Cats (VP chase mice VP) VP) S) S)`
LIN^3	ONT(v)	3×CNT(v)	-	`(S Cats (VP chase mice VP) VP) VP) S) S) S)`
$\text{LIN}^1_{\text{–ont}}$	$\emptyset$	CNT(v)	-	`Cats chase mice VP) S)`
$\text{LIN}^1_{\text{–merge}}$	ONT(v)	CNT(v)	Merge CNTs	`(S Cats (VP chase mice )`
Non-structural				
LIN^0	$\emptyset$	$\emptyset$	-	`Cats chase mice`
$\text{LIN}^1_{\text{shuf}}$	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, $\text{LIN}^1_{\text{shuf}}$ is generated by shuffling the LIN^1 token sequence while preserving the original terminal order.

Dataset	LIN^0	LIN^1	LIN^2
BBC News	∼10B	∼24.7B	∼32B
FineWeb-Edu 100BT	∼98B	∼233B	∼301B

Table 2: Training token counts for each linearization. Structural tokens inflate the token budget 2.4–3.2× per raw document.

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 $z = \text{LIN}^2(\mathbf{x}, \mathbf{y})$ 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, ONTs, 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 $z = \text{LIN}^2(\mathbf{x}, \mathbf{y})$ 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 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-

#	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	$\text{LIN}^1_{-\text{ont}}$	Causal
8	Tree-Compress	$\text{LIN}^1_{-\text{merge}}$	Causal
9	Tree-TripleCNT	LIN^3	Causal
10	Tree-Shuffle	$\text{LIN}^1_{\text{shuf}}$	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: CNT_2 remains visible to subsequent attention in Aug, while TGNomask prevents subsequent tokens from attending to it.

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:

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

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

We compare these scoring rules on a held-out eight-task validation suite (Validation-8; 17,691 examples). Terminal scoring increased macro-average accuracy by 2.16 points for Tree (95% CI [1.35, 2.99]) and by 0.62 points for TGTree (95% CI [−0.09, 1.35]). Based on this result, we use terminal scoring as the primary metric and predict $\arg\max_j S_{\text{term}}(\mathbf{c}_j \mid \mathbf{p})$ for MC tasks. Appendix E defines the validation suite and reports the corresponding full-score sensitivity analysis.

5 Experiments

5.1 Setup

We train all models from scratch for one epoch using the same OLMo-based (Groeneveld et al., 2024) decoder-only Transformer family across scales. Learning rates and batch sizes follow the Step Law (Li et al., 2025). Causal attention models use FlashAttention-2 (Dao, 2024), and structural-mask models require FlexAttention (Dong et al., 2025). Further details of model architectures and hyperparameter selection are provided in Appendix A.

Table 3 defines the controlled design space used for our linearization and attention-mask ablations. To address the varying training token budgets induced by different linearizations, we compare SLMs with pause-token baselines (Goyal et al., 2024): $\text{PAUSE-}K = (x_1, \underbrace{p, \ldots, p}_{K}, x_2, \underbrace{p, \ldots, p}_{K}, x_3, \ldots)$, where p is a dedicated learned PAUSE token. For $K = 1$, the computational load is roughly comparable to LIN^1 (2.4×); for $K = 2$, it is roughly comparable to LIN^2 (3.2×).

We additionally include a 100M-parameter *ar-*

chitectural reference baseline outside this design space. Pushdown (Murty et al., 2023) jointly trains the language model and an attachment parser, whose predictions induce the stack-depth attention biases used at inference. We reimplement Pushdown in our OLMo-based Transformer architecture and pretrain it for one epoch on silver parse trees from the same BBC News corpus. This reproduction ensures that Pushdown uses the same Transformer backbone and pretraining data as the other models. We include it for reference only: it is not counted among the 12 controlled configurations or used in the ablations and scaling selection.

5.2 Evaluation Methods

Document-Level Language Modeling. Our SLMs define a joint distribution $p(\mathbf{x}, \mathbf{y})$ over a sentence and its constituency tree (Eq. 1). For each sentence–tree pair, we form $\mathbf{z} = \textsc{Linearize}(\mathbf{x}, \mathbf{y})$ and compute its joint probability autoregressively as $p(\mathbf{x}, \mathbf{y}) = \prod_{t=1}^{|\mathbf{z}|} p(z_t \mid \mathbf{z}_{<t})$. The sentence probability $p(\mathbf{x}) = \sum_{\mathbf{y} \in \mathcal{Y}_\mathbf{x}} p(\mathbf{x}, \mathbf{y})$ requires marginalizing over all valid parses and is therefore intractable. Following Sartran et al. (2022) and Zhao et al. (2025), we approximate each conditional marginal using 300 labeled proposal trees from a CRF constituency parser (Kitaev and Klein, 2018):

$$\widetilde{p}(\mathbf{x}_i \mid \mathbf{h}_i) = \sum_{\mathbf{y} \in \mathcal{Y}'_i} p(\mathbf{x}_i, \mathbf{y} \mid \mathbf{h}_i). \tag{6}$$

Here, $\mathcal{Y}'_i \subset \mathcal{Y}_{\mathbf{x}_i}$ is the proposal set, and $K = |\mathcal{Y}'_i| = 300$ is its size. The truncated sum is a lower bound on the full marginal and therefore yields an upper bound on perplexity.

For document-level evaluation, marginalization over the parse histories of all preceding sentences would incur exponential cost. We therefore follow the greedy single-path approximation of prior work. For each preceding sentence j, we retain the proposal with the highest joint probability under the evaluated model,

$$\hat{\mathbf{y}}_j = \arg\max_{\mathbf{y} \in \mathcal{Y}'_j} p(\mathbf{x}_j, \mathbf{y} \mid \mathbf{h}_j). \tag{7}$$

The selected parses form the prefix $\mathbf{h}_i = \textsc{Linearize}(\mathbf{x}_{<i}, \hat{\mathbf{y}}_{<i})$, and only the current sentence is marginalized over all 300 proposals.

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

$$\mathcal{L}_K = \sum_{d=1}^{D} \sum_{i=1}^{N_d} \log \widetilde{p}(\mathbf{x}_{d,i} \mid \mathbf{h}_{d,i}),$$
$$\text{PPL}_{\text{syn}} = \exp(-\mathcal{L}_K / M). \tag{8}$$

Here, M is the total number of scored terminal tokens. Except for the model-specific procedures in Appendix B.4, tree models use Eq. 6; flat and pause-token models use terminal-token autoregressive PPL.

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

For training corpus construction, we use the Benepar parser (Kitaev and Klein, 2018), which is trained on news-domain data, to obtain constituency parses for BBC News articles filtered from each subset of the FineWeb dataset (Penedo et al., 2024). The resulting corpus covers approximately 15.2M articles from BBC News domains and contains $\sim$10B terminal tokens. We then train all 100M controlled configurations and comparison baselines on this corpus for one epoch and evaluate them on XSum ROUGE, BoolQ, SG, and BLiMP. Our goal is to identify which tree linearization and which attention mechanisms drive the syntactic benefit, and to screen members of the controlled design space for scaling. Table 4 separates the shaded architectural reference baseline from the controlled configurations and compute controls.

Model	Linearization	Attention	XSum↑	BoolQ↑	SG↑	BLiMP↑	Doc-PPL‡↓
Terminal-100M	LIN^0	Causal	22.02 ± 0.06	66.70 ± 0.36	69.80	70.77	9.89
Tree-100M	LIN^1	Causal	22.84 ± 0.03	68.07 ± 0.35	81.40	80.95	10.01
TGTree-100M	LIN^2	Causal	$\mathbf{23.03 \pm 0.02}$	68.12 ± 0.62	79.47	81.46	10.21
TG-100M	LIN^2	S/C	20.15 ± 0.04	64.50 ± 0.40	79.19	83.28	11.02
TGNomask-100M	LIN^2	Nomask	21.91 ± 0.04	66.66 ± 0.45	78.62	81.78	9.97
TGNomask-Aug-100M	LIN^2	Nomask	22.12 ± 0.02	67.01 ± 0.28	78.68	**83.63**	10.08
Tree-NoONT-100M	$\text{LIN}^1_{\text{-ont}}$	Causal	22.41 ± 0.04	$\mathbf{68.67 \pm 0.32}$	**85.38**	81.00	9.95
Tree-Compress-100M	$\text{LIN}^1_{\text{-merge}}$	Causal	22.70 ± 0.05	68.48 ± 0.49	78.27	81.40	9.91
Tree-TripleCNT-100M	LIN^3	Causal	22.43 ± 0.04	68.09 ± 0.12	81.57	83.42	10.26
Tree-Shuffle-100M	$\text{LIN}^1_{\text{shuf}}$	Causal	21.56 ± 0.09	68.08 ± 0.33	76.56	67.77	13.41
TGNomask-Mix-TG-100M	LIN^2	N+S/C	21.87 ± 0.04	66.13 ± 0.54	75.77	83.14	10.01
TGTree-Mix-TG-100M	LIN^2	C+S/C	22.16 ± 0.07	67.91 ± 0.36	78.84	82.75	10.00
Pause-1-100M	Pause-1	Causal	22.38 ± 0.05	62.04 ± 0.19	75.11	70.85	**9.77**
Pause-2-100M	Pause-2	Causal	22.25 ± 0.06	65.70 ± 3.45	76.97	72.59	10.32
Pushdown-100M	LIN^0	Stack bias	21.06 ± 0.05	65.54 ± 0.29	74.86	74.97	13.29
Terminal-500M	LIN^0	Causal	20.75 ± 0.06	69.69 ± 0.60	**71.11**	64.74	**2.83**
Tree-500M	LIN^1	Causal	$\mathbf{22.25 \pm 0.05}$	70.72 ± 0.35	63.13	76.09	3.18
TGTree-500M	LIN^2	Causal	22.01 ± 0.08	70.63 ± 0.32	65.95	**77.65**	3.28
TGNomask-Aug-500M	LIN^2	Nomask	21.77 ± 0.06	$\mathbf{70.78 \pm 0.74}$	57.58	75.97	3.19

Table 4: Key 100M and 500M results on BBC News. The shaded row is an architectural reference baseline for reference only, not a member of the controlled variant family. XSum reports R-AVG; XSum and BoolQ show mean ± sample standard deviation over five fine-tuning seeds. Other metrics are single evaluations. S/C = STACK/COMPOSE; N+S/C and C+S/C mix it at head level with Nomask and causal attention, respectively. Stack bias is Pushdown's stack-depth bias, derived from its attachment parser. Highest means per scale and best available single-evaluation values are in **bold**. ‡Doc-PPL protocols are detailed in Appendix B.4.

Structural Attention vs. Causal Attention. SLMs with causal attention outperform structural attention models on downstream tasks. On XSum, TGTree and Tree have the highest means (23.03 and 22.84), exceeding TGNomask-Aug (22.12), Terminal (22.02), and TG (20.15); BoolQ shows the same ordering of means among these models. The full STACK/COMPOSE mask (TG) underperforms even the terminal baseline on both tasks, while a softer variant (TGNomask-Aug) yields a partial recovery. Structural attention models exhibit a clear syntax–utility tradeoff: TG and its variants lead only on BLiMP (up to 83.63), where the hard constraints of STACK/COMPOSE attention enforce grammatical consistency, but the same rigidity penalizes general language modeling. Causal attention models suffer no such tradeoff—TGTree has the highest 100M XSum mean, while Tree combines a 22.84 XSum mean with 81.40 SG; TGNomask-Aug manages only 78.68 SG. The mask is neither necessary nor beneficial for most objectives.

Architectural Reference Baseline. Pushdown does not improve XSum or BoolQ over Terminal but improves both syntactic scores (74.86 and 74.97) under its model-specific inference procedure.

Hybrid Attention Fails to Improve. Hybrid attention variants (TGNomask-Mix-TG, TGTree-Mix-TG) yield no benefit over causal Tree SLM, adding complexity without improving performance.

Pause-token Controls. The pause-token baselines improve over Terminal on XSum (22.38 and 22.25 vs. 22.02), indicating that additional latent-computation positions can help generation. Nevertheless, the compute-matched tree models remain stronger across downstream and syntactic evaluations: Tree exceeds Pause-1 on XSum, BoolQ, SG, and BLiMP, and TGTree likewise exceeds Pause-2 on all four metrics. Thus, the gains from tree linearization cannot be explained solely by the larger token budget.

Document-Level Perplexity. The PPL of Tree remains close to that of Terminal (10.01 vs. 9.89), with a gap of 0.12 PPL points, indicating that linearizing trees into the input sequence incurs only a small LM cost. The STACK/COMPOSE mask is the exception: TG degrades PPL to 11.02, consistent with its downstream underperformance. TGNomask is competitive with causal tree models,

supporting Zhao et al.'s (2025) finding that removing sub-constituent masks improves LM quality. Among the tree-linearization variants, Tree-Compress achieves the lowest PPL at 9.91. Tree-Shuffle obtains 13.41 under its terminal-only protocol (Appendix B.4), so we exclude it from direct comparisons with CRF-marginalized PPL values. Among all 100M models, Pause-1 achieves the lowest PPL at 9.77, slightly outperforming Terminal at 9.89, whereas Pause-2 reaches 10.32. Except for Tree-Shuffle, the reproduced PPL values cluster near Terminal even when downstream scores differ, indicating that PPL and downstream performance provide complementary signals.

Tree Linearization Ablations. The experiments on all SLMs with causal attention form an ablation study of tree linearization. Tree-NoONT improves syntactic discrimination but impairs generation, suggesting ONTs primarily help the model anticipate upcoming constituent structures rather than enforce syntactic constraints. Tree-Shuffle performs substantially worse on BLiMP. Doubling CNTs (TGTree) achieves the best trade-off between syntax and generation; tripling CNTs (Tree-TripleCNT) raises SG relative to TGTree (81.57 vs. 79.47) at the cost of generation quality and reaches 83.42 on BLiMP. Merging consecutive identical closing non-terminals, as in Tree-Compress, reduces SG, suggesting that fine-grained closure information helps targeted syntactic generalization.

Scaling Up. After the 100M experiments, we select four variants and train them at 500M parameters on BBC News to verify that the advantage of tree linearization with causal attention persists before committing to larger-scale experiments. Table 4 shows the results, confirming that Tree and TGTree maintain their downstream lead over the terminal baseline and the structural attention variant. SG scores decline for all three scaled tree-input models, while Terminal changes little; we discuss this pattern in §6.2.

There is a practical note on throughput: SLMs with causal attention benefit from FlashAttention compatibility, whereas SLMs with structural attention require slower custom attention implementations. Table 5 reports measured training throughput.

The throughput gap persists at scale, and mixing-head mask models are 18× slower. Combined with the performance evidence above, the causal tree approach is the clear practical choice for scaling.

Scale	Causal Mask	Structural Mask	Ratio
100M	62,175 token/s	37,975 token/s	1.6×
500M	12,550 token/s	8,316 token/s	1.5×

Table 5: Training throughput on H800 GPUs. Causal models use FlashAttention-2, whereas structural-mask models use FlexAttention. The mixing-head mask achieves 3,375 token/s at the 100M scale, which is 18× slower than causal attention.

Based on these results, we select tree linearization Tree (LIN^1) and TGTree (LIN^2), both with causal attention, as the primary variants for scaling, dropping all mixing-head and structural attention approaches.

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

FineWeb-Edu-100BT (Penedo et al., 2024) is a ∼100B-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, TGTree, 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 6 reports per-task downstream accuracy, and Table 7 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 E reports sensitivity to full versus terminal scoring.

Main result. TGTree has the highest 11-task point estimate (55.88, +2.68 over Terminal), but the bootstrap ranking is not decisive: its probability of ranking first is 0.29, compared with 0.28 for Pause-2. Excluding BoolQ reverses their point-estimate order (Pause-2: 55.04; TGTree: 54.71). Relative to Terminal, TGTree shows significant improvements on five of the 11 tasks; the full paired comparisons are reported in Appendix C.

Sensitivity to scoring format. As defined in Section 4, terminal scoring is our primary content-focused metric; full scoring is a complementary sensitivity analysis. On Validation-8, terminal scoring changes 17.58% of the Tree rankings and 12.35% of the TGTree rankings relative to full scoring. It raises macro-average accuracy by 2.16

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 6: 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 = SocialIQA, 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 7: 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.

points for Tree (95% CI [1.35, 2.99]) and 0.62 points for TGTree (95% CI [−0.09, 1.35]). This supports the primary-metric choice: Tree improves significantly, while TGTree's point estimate rises without a statistically distinguishable change. Appendix E reports the task-level results and limitations.

Explicit Structure vs. Dummy Tokens. TGTree and the compute-matched Pause-2 baseline have nearly identical 11-task point estimates (55.88 vs. 55.80). In paired tests, TGTree is significantly better only on BoolQ (+4.16 [2.02, 6.30], $p < .001$), while Pause-2 is significantly better on HellaSwag (+0.94 [0.19, 1.68], $p = .014$); the other nine task differences are not significant at $\alpha = .05$. A separate ten-task validation evaluation yields the same aggregate conclusion (57.35 vs. 57.09; difference +0.26, 95% CI [−0.68, 1.20]; Table 15). These models therefore show compute-matched competitive performance with complementary task-level strengths, rather than dominance. A combination of explicit structure and learned latent computation is a natural direction for future work.

Syntactic evaluation at 1B. Tree has the highest point estimate on both syntactic benchmarks, exceeding Terminal by 2.14 points on BLiMP and 1.94 points on SG. TGTree also exceeds Terminal on both benchmarks, whereas the compute-matched pause-token baselines improve BLiMP but not SG. Without pretraining replicates, we treat these results as task-level evidence rather than a general model ordering.

5.5 Scaling Trend Summary

Table 8 shows positive point-estimate differences over Terminal across all scales and datasets. On BBC, the mean gap between Terminal and the best evaluated SLM is non-monotonic (+1.01 → +1.50 → +0.94; the 1B BBC run was supplementary), possibly reflecting saturation on the narrow 10B-token corpus. On diverse FineWeb-Edu data, the point-estimate gap is +2.68 on the 11-task average. As discussed in §5.4, however, the identity of the top model is not stable under bootstrap resampling; the stronger evidence is the set of significant task-level improvements over Terminal.

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

Scale	Data	Metric	Terminal	Best SLM (Δ)
100M	BBC	XSum R-AVG	22.02 ± 0.06	TGTree: 23.03 ± 0.02 (+1.01)
500M	BBC	XSum R-AVG	20.75 ± 0.06	Tree: 22.25 ± 0.05 (+1.50)
1B	BBC	XSum R-AVG	20.47 ± 0.06	Tree: 21.42 ± 0.08 (+0.94)
1B	FW-Edu	11-task AVG	53.20	TGTree: 55.88 (+2.68)

Table 8: Scaling trend across model sizes and datasets. BBC XSum values are mean $\pm$ sample standard deviation over five fine-tuning seeds; the FineWeb-Edu row is a single evaluation. Δ is the difference between means for BBC and the point-estimate difference for FineWeb-Edu.

this hypothesis, causal attention allows the model to learn which information to route through CNT positions, unlike TG's hard-coded COMPOSE attention. This flexibility may be more useful for diverse data and may contribute to the larger gains on FineWeb-Edu. We do not directly test this mechanism.

Adaptive computation hypothesis. Pause-K controls for extra tokens by allocating 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. TGTree and Pause-2 are statistically indistinguishable on nine of the 11 downstream tasks, so our results motivate this hypothesis but do not establish it.

6.2 Syntactic Degradation on BBC News

On BBC News, scaling Tree from 100M to 500M lowers SG from 81.40 to 63.13 and BLiMP from 80.95 to 76.09, whereas Terminal SG remains stable (69.80 to 71.11). This decline is not accompanied by worse in-domain modeling: Tree's document PPL improves from 10.01 to 3.18. For parse ranking, we score 300 Benepar n-best parses for each of 100 BBC sentences. At $K = 300$, the 500M model selects Benepar's top parse more often than the 100M model (0.81 vs. 0.72). Thus, the larger model better fits both BBC News text and the analyses preferred by the news-domain parser, despite weaker generalization to SG and BLiMP.

The degradation does not recur on the more diverse FineWeb-Edu corpus. At 1B, Tree and TGTree score 89.95 and 89.00 on SG, both above Terminal's 88.01. On BLiMP, they score 81.79 and 80.98, again above Terminal's 79.65 (Table 7). Together, the improved in-domain fit, stronger parse ranking, and cross-corpus contrast are consistent with overfitting to the narrow BBC News distribution. Because corpus diversity, tokenizer, and training scale are not independently controlled, this evidence supports but does not establish a causal domain-overfitting explanation.

7 Conclusion

We conduct a scaled-up empirical study of syntactic language models, evaluating one terminal baseline and 11 SLM variants at scales up to 1B parameters and 100B training tokens. Our evaluation spans 11 downstream tasks and two syntactic benchmarks. Our controlled comparisons disentangle tree linearization from attention design: SLMs with standard causal attention match or surpass structural-mask variants across downstream and syntactic benchmarks. They also require no architectural modifications and remain compatible with FlashAttention, making causal tree models the more practical SLM design. At the 1B scale, however, the top-performing SLM and a compute-matched non-structural baseline show competitive aggregate performance with complementary task-level strengths, rather than clear SLM dominance. This finding suggests that SLMs may become less competitive with scale and that their advantages should be assessed against equally compute-intensive non-structural alternatives.

Limitations

Constructing linearized tree inputs requires a parser for both training data and inputs. This adds preprocessing cost, and parsing errors may propagate into the training signal. We do not quantify the impact of parse quality. In addition, we investigate only English, leaving syntactic structures in other languages unexplored.

The compositional bottleneck and adaptive computation accounts are hypotheses motivated by the observed comparisons, not established mech-

anisms. The core conclusions are supported by controlled comparisons across linearizations, attention masks, pause-token baselines, and scales, but we do not directly intervene on internal representations or attention dynamics to prove how tree tokens are used. Such mechanistic probes are left to future work.

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 Haddad, 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 allal, 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 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 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. Under this architecture, we train different SLMs with 100M, 500M, and 1B parameters. Table 9 lists the three scale configurations.

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 9: Model configurations at each scale.

We train all models from scratch for one epoch. Training uses the AdamW optimizer (Loshchilov and Hutter, 2019) with β_1=0.9, β_2=0.95, ϵ=10^{-8}, weight decay of 0.1, and gradient clipping at norm of 1.0. We use a cosine learning rate scheduler with 2,000 warmup steps; after the learning rate decays to 10^{-5}, it is held constant for the rest of training. The maximum sequence length is 2,048 tokens. The learning rate η and batch size B are set according to the Step Law (Li et al., 2025):

$$\eta(N, D) = 1.79N^{-0.713}D^{0.307}$$
$$B(D) = 0.58D^{0.571} \quad (9)$$

where N is the number of non-embedding model parameters and D is the size of training dataset.

A.2 Computational Costs

At each parameter scale, we report the aggregate training time across all model variants evaluated at that scale. The 100M-parameter variants required 133 hours on 4 H800 GPUs. The 500M-parameter variants required 125 hours on 4 H800 GPUs. The 1B-parameter variants required 385 hours (approximately 16.0 days) on 64 A800 GPUs.

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 the pretrained checkpoint, with the optimizer state reset. All models are fine-tuned with AdamW (β_1=0.9, β_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 Table 4 are means ± sample standard deviations over five fine-

tuning seeds (42, 2026, 6198, 13171, and 31723). Pushdown fine-tuning uses right-binarized, unary-collapsed gold spans. Its XSum decoder conditions on the gold parse of the source and jointly searches summary tokens and attachment actions (beam size 6, at most 4 consecutive reductions); BoolQ uses tree-aware candidate scoring with beam size 20.

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

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

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 each BLiMP minimal pair evaluated with CRF-based tree marginalization, we obtain the 300 highest-scoring parse proposals ($K = 300$) from the Benepar parser (Kitaev and Klein, 2018) and compute the tree-marginalized probability of each sentence via $p(\mathbf{x}) \approx \sum_{k=1}^{K} p(\mathbf{x}, \mathbf{y}_k)$, following the same marginalization procedure as document-level PPL (Eq. 6). 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 $K = 1$. For Pushdown, each candidate tree supplies a fixed sequence of attachment actions, and the score marginalizes over the 300 candidate trees.

B.4 Model-Specific Document-PPL Protocols

CRF-parsed tree-linearization models other than Tree-Shuffle use the 300-proposal truncated marginal in Eq. 6. Under the same greedy parser-selected history, truncation lower-bounds the exact current-sentence probability and therefore upper-bounds its PPL. Tree-Shuffle uses reproduced terminal-only scoring. Pushdown sums joint token–attachment probabilities over up to 300 native attachment candidates for the current sentence and uses candidate 0 for each preceding sentence. Because the Tree-Shuffle and Pushdown procedures differ from CRF-based marginalization, their PPL values are contextual comparisons.

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, SocialIQA, 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 $\hat{\mathbf{y}}(x)$ 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 E 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. For the macro-average rows, tasks rather than examples are resampled; these wider intervals characterize sensitivity to task-suite composition. The probability that each model ranks first under this task resampling is 0.09 (Terminal), 0.11 (Tree), 0.29 (TGTree), 0.23 (Pause-1), and 0.28 (Pause-2). Excluding BoolQ changes these probabilities to 0.11, 0.10, 0.26, 0.22, and 0.29, respectively.

Table 11 reports the paired contrasts most relevant to our claims: TGTree versus the Terminal baseline and each tree model versus its compute-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 SG Subtask Breakdown

Table 12 breaks down SG performance by subtask for key 100M variants. Agreement shows the largest improvement (Tree +41.4 vs. Terminal). Center Embedding shows ceiling effects across all models (87.5–94.6). For Long-Distance Dependencies, Tree and Tree-NoONT improve by 3.21 points and TGTree by 5.96 points, whereas TG decreases by 0.46 points relative to Terminal.

Table 13 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.

E Full-versus-Terminal Score Sensitivity

We evaluate the scoring choice on the held-out Validation-8 data by decomposing the MC score into terminal and non-terminal components (Eq. 5). Validation-8 comprises the validation splits of WinoGrande, HellaSwag, OpenBookQA, CommonsenseQA, SocialIQA, ARC-Easy, ARC-Challenge, and PIQA, totaling 17,691 examples. We draw fixed in-context demonstrations from the training data and verify that their normalized text does not overlap with the scored examples. Table 14 reports both scores for the 1B Tree and TGTree models.

Non-terminal tokens have higher mean log-probability than terminal tokens by 2.68 nats for Tree and 2.86 nats for TGTree, but this scale difference does not by itself determine option rankings. With terminal scoring, 7.46% of Tree examples flip to correct and 5.30% to wrong. The corresponding TGTree rates are 4.92% and 4.30%. Other ranking changes switch between two incorrect options and therefore do not affect accuracy. At the macro level, terminal scoring improves Tree by 2.16 points (95% CI [1.35, 2.99]). TGTree rises by 0.62 points, but its interval includes zero (95% CI $[-0.09, 1.35]$). Two TGTree tasks favor full scoring. These results support terminal scoring as the primary content-focused metric while motivating full scoring as a complementary sensitivity analysis.

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 6.

Task	TGTree − Terminal	Tree − Pause-1	TGTree − Pause-2
WG	+5.21 [2.29, 8.13]; $p < .001$	+2.68 [−0.32, 5.68]; $p = .082$	+0.78 [−2.05, 3.63]; $p = .604$
HS	+1.98 [1.22, 2.74]; $p < .001$	−8.95 [−9.96, −7.89]; $p < .001$	−0.94 [−1.68, −0.19]; $p = .014$
BQ	+8.90 [6.48, 11.31]; $p < .001$	−1.57 [−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]; $p = .172$	−1.27 [−2.40, −0.18]; $p = .024$	−1.04 [−2.14, 0.05]; $p = .069$
OBQA	+2.16 [−1.40, 5.80]; $p = .262$	+1.21 [−2.80, 5.20]; $p = .592$	+1.76 [−1.80, 5.40]; $p = .367$
CSQA	+4.92 [2.46, 7.45]; $p < .001$	−0.15 [−2.46, 2.21]; $p = .934$	+1.39 [−0.90, 3.69]; $p = .248$
SIQA	+1.34 [−0.67, 3.38]; $p = .210$	−2.81 [−4.86, −0.77]; $p = .007$	+1.65 [−0.36, 3.68]; $p = .120$
ARCE	+0.67 [−3.16, 4.56]; $p = .761$	−5.63 [−9.30, −1.93]; $p = .002$	−3.00 [−6.67, 0.70]; $p = .113$
ARCC	+1.36 [−4.01, 6.69]; $p = .666$	+1.64 [−3.68, 6.69]; $p = .570$	−3.32 [−9.03, 2.34]; $p = .261$
PIQA	+1.07 [−0.65, 2.83]; $p = .247$	−2.09 [−3.92, −0.27]; $p = .024$	−0.02 [−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 compute-matched comparisons.

Model	Agr↑	CtrEmb↑	GrdnPath↑	GrsSyn↑	Lic↑	LDD↑
Terminal	39.66	87.50	78.29	84.78	61.28	68.35
Tree	81.03	89.29	84.21	83.70	80.08	71.56
TGTree	67.24	94.64	82.24	88.04	69.92	74.31
TG	79.31	92.86	81.58	79.35	71.80	67.89
Tree-NoONT	87.93	89.29	85.53	97.83	80.08	71.56

Table 12: SG subtask breakdown at 100M. Agr = Agreement, CtrEmb = Center Embedding, GrdnPath = Garden Path Effects, GrsSyn = Gross Syntactic Expectation, Lic = Licensing, LDD = Long Distance Dependencies. ↑ = higher is better.

Model	Agr↑	CtrEmb↑	GrdnPath↑	GrsSyn↑	Lic↑	LDD↑	SG↑
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 13: SG subtask breakdown at 1B on FineWeb-Edu. Best per metric in **bold**; abbreviations follow Table 12.

	Tree			TGTree		
Task	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 14: Full and terminal accuracy (%) on Validation-8; $\Delta = \mathrm{Term} - \mathrm{Full}$. 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).

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 score is the exact marginal sentence likelihood $\sum_{\mathbf{y}} p(\mathbf{x}, \mathbf{y})$. Accordingly, S_{term} is primary, while S_{full} measures sensitivity to including the parser-selected structural continuation.

E.1 Separate Validation-10 Comparison

Validation-10 adds BoolQ and a held-out MMLU validation set to the eight tasks above, yielding 22,207 scored examples. For MMLU, five fixed demonstrations are selected for each of 57 subjects and removed from scoring, leaving 1,246 examples; MMLU accuracy is macro-averaged across subjects. All other task accuracies are micro-averaged, and the headline score is the unweighted macro-average over ten tasks. We use 10,000 paired bootstrap resamples with 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 15: Terminal-score accuracy (%) on Validation-10. The paired TGTree–Pause-2 difference is +0.26 points (95% CI [−0.68, 1.20]).