Gated Tree Cross-Attention for Checkpoint-Compatible Syntax Injection in Decoder-Only LLMs

Xinyu Gao[1], Shaonan Wang[2], Nai Ding[1*]
[1]College of Biomedical Engineering & Instrument Science, Zhejiang University
[2]Department of Language Science and Technology, The Hong Kong Polytechnic University
{xinyu_gao, ding_nai}@zju.edu.cn
shaonan.wang@polyu.edu.hk

Abstract

Decoder-only large language models achieve strong broad performance but are brittle to minor grammatical perturbations, undermining reliability for downstream reasoning. However, directly injecting explicit syntactic structure into an existing checkpoint can interfere with its pretrained competence. We introduce a checkpoint-compatible gated tree cross-attention (GTCA) branch that reads precomputed constituency chunk memory while leaving backbone architecture unchanged. Our design uses a token update mask and staged training to control the scope and timing of structural updates. Across benchmarks and Transformer backbones, GTCA strengthens syntactic robustness beyond continued training baselines without compromising Multiple-Choice QA performance or commonsense reasoning, providing a practical checkpoint-compatible route to more syntax-robust decoder-only LLMs. Our code is available at https://github.com/Pineandgrass/GatedTreeCrossAttention.

1 Introduction

Transformer-based pre-trained language models (PLMs) now underpin most Natural Language Understanding (NLU) and Natural Language Generation (NLG) systems because scaled self-attention transfers broadly across tasks (Vaswani et al., 2017; Devlin et al., 2019; Brown et al., 2020; Liu et al., 2019; Raffel et al., 2020; Touvron et al., 2023). However, for reliable deployment, it is not enough to score well on aggregate benchmarks: a model should keep its decision stable under small grammatical perturbations and syntactic alternations that preserve meaning. In practice, decoder-only LLMs can flip preferences on minimally different inputs (e.g., agreement or licensing minimal pairs), a failure mode that users experience as "same meaning, different answer" and that can cascade into downstream reasoning errors (Ribeiro et al., 2020).

This mismatch is visible on the targeted stress tests. Natural Language Inference (NLI) models trained in Multi-Genre Natural Language Inference (MNLI) can appear strong while relying on shallow heuristics, and fail in Heuristic Analysis for NLI Systems (HANS) where correct labels require syntactic generalization (Gururangan et al., 2018; McCoy et al., 2019). Similarly, the Benchmark of Linguistic Minimal Pairs (BLiMP) reveals persistent gaps in structure-sensitive constraints (e.g., islands), with GPT2 far below human performance (Warstadt et al., 2020).

Why does this brittleness persist even as the models seem linguistically informed? The probing work shows that the non-trivial syntactic structure is recoverable from hidden states (Hewitt and Manning, 2019) and is reflected in layers and attention patterns (Clark et al., 2019; Eisape et al., 2022; Müller-Eberstein et al., 2022). However, encoding is not usage. Recoverability does not ensure that a model uses structure in its predictions, and probes can be misleading under redundancy or artifacts (Diego-Simón et al., 2025; Agarwal et al., 2025; Tucker et al., 2022). From an adaptation standpoint, naive structure injection can also interfere with pretrained representations, triggering instability and catastrophic forgetting during continued training (Iwamoto et al., 2023). This motivates controllable syntax injection, we want to expose structure in a way that the model can selectively exploit, without rewriting the checkpoint.

Our solution is a minimally invasive side path: a checkpoint-compatible gated cross-attention (GTCA) branch that lets token states read cached constituency chunk memory while leaving the backbone architecture untouched. The gate learns when to trust the structural signal, turning the structure into a regulated update rather than a hard constraint. We further restrict interference with token update masking and a three-stage training schedule.

Evaluation covers multiple-choice QA (MCQA)

*Corresponding author: Nai Ding

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 35274–35288

benchmarks, including CLOTH (Xie et al., 2018) and MMLU (Hendrycks et al., 2021), syntactic benchmarks, including BLiMP (Warstadt et al., 2020) and CoLA (Warstadt et al., 2019) from GLUE (Wang et al., 2018), and commonsense reasoning tasks, including HellaSwag (Zellers et al., 2019) and WinoGrande (Sakaguchi et al., 2021). GTCA consistently improves syntax-focused results while maintaining MCQA accuracy and overall capability across different backbones. Layer-wise probe of Unlabeled Undirected Attachment Score (UUAS) further connects these gains to a more syntax-consistent internal structure (Hewitt and Manning, 2019; Müller-Eberstein et al., 2022; Limisiewicz and Mareček, 2021). Specifically, GTCA increases the accuracy of BLiMP from 78.58 to 83.12 in Qwen-2.5-7B and from 79.95 to 84.61 in Llama-3-8B, despite leaving the backbone architecture unchanged.

Our major contributions are as follows.

- We propose GTCA, a checkpoint-compatible gated cross-attention side branch that injects hierarchical constituency information via cached chunk memory without modifying backbone architecture.
- We introduce two stabilization mechanisms, token update masking and staged training, to control interference during continued training.
- Across two decoder-only backbones with comparable capacities, GTCA yields consistent syntax improvements while preserving or improving broad competence.

2 Related Work

2.1 Injecting Explicit Syntactic Structure into Pre-Trained Transformers

A long line of work has incorporated syntax as an explicit inductive bias in neural NLP, ranging from tree-structured composition to attention-based structure integration. Early architectures encoded hierarchical structure through tree-structured recurrence (Tai et al., 2015). Later approaches injected syntactic information into attention-based models by biasing attention towards syntactic relations, such as biasing self-attention using syntactic relations (Strubell et al., 2018) or designing dependency-aware attention mechanisms for sequence-to-sequence modeling (Bugliarello and Okazaki, 2020). With the rise of PLMs, researchers have increasingly asked whether explicit trees still help when models already capture syntax implicitly, and how to leverage trees without expensive retraining. Representative efforts have included plug-in-style syntax integration into pre-trained Transformers (Bai et al., 2021) and systematic studies on when syntactic structure is beneficial under different injection strategies and tasks (Sachan et al., 2021). Taken together, previous work spans architectural biases, objective-level supervision, and structure-native pretraining.

A related line of work injected syntax through training objectives rather than architectural rewrites, e.g., syntax-guided contrastive learning (Zhang et al., 2022), or continuing training pipelines that explicitly considered optimization stability and catastrophic forgetting when incorporating syntactic knowledge (Iwamoto et al., 2023). In parallel, modeling native language structure on a scale aims to induce or enforce hierarchical structure during pretraining (Hu et al., 2024; Zhao et al., 2024), and graph-based formulations offer a broader perspective of structured modeling (Plenz and Frank, 2024). At the same time, cautionary evidence suggests that tree bias alone is not sufficient. Its effectiveness depends on how the structure is represented and integrated (Ginn, 2024). These findings motivate checkpoint-compatible, minimally invasive structure injection for decoder-only LLMs, where hierarchical signals are exposed through an auxiliary pathway and integrated in a way that the pre-trained backbone can reliably exploit.

2.2 Probing Syntactic Representations and Training-Efficient Adaptation

Probing has been central to characterizing syntactic information in neural language models and to clarifying the mechanisms-outcomes tension. Structural probing showed that dependency-like geometry could be recovered from contextual embeddings (Hewitt and Manning, 2019), while attention and layer-wise analyses have revealed systematic associations between model components and linguistic relations or processing stages (Jawahar et al., 2019; Tenney et al., 2019; Clark et al., 2019). Subsequent work improved probe formulations and constraints (Limisiewicz and Mareček, 2021; Müller-Eberstein et al., 2022), extended probing to incremental parse states in autoregressive settings (Eisape et al., 2022), and proposed richer representational codes such as polar probes (Diego-Simón et al., 2024).

However, recent evidence has emphasized that representational recoverability does not guaran-

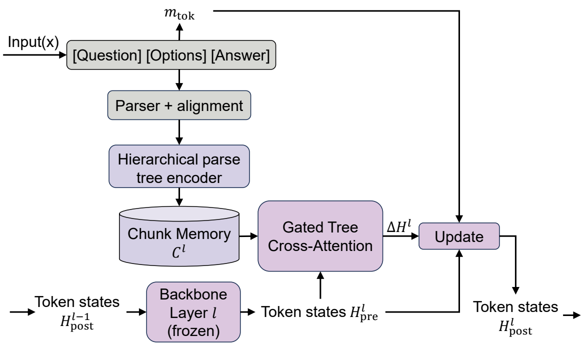

Figure 1: Overview of GTCA (Gated Tree Cross-Attention) structural injection.Given an input (question, options, and answer field), we run an offline parser with span alignment and encode the resulting hierarchy with a parse-tree encoder to build per-layer chunk memory C^ℓ. At Transformer layer ℓ, the pre-update token states H^ℓ_{pre} query C^ℓ through gated tree cross-attention to produce a structural residual update ΔH^ℓ. This update is then added to obtain the post-update token states H^ℓ_{post}, which are passed to the next layer.

tee behavioral usage. Redundancy and probe artifacts can produce misleading conclusions, motivating intervention-style and dropout-based analyses (Tucker et al., 2022; Agarwal et al., 2025; Diego-Simón et al., 2025), while other work argued for a richer generative-style structure than previous probes captured (Kennedy, 2025). In parallel, practical checkpoint adaptation has been shaped by parameter-efficient fine-tuning methods such as LoRA (Hu et al., 2021) and QLoRA (Dettmers et al., 2023), prompt-based alternatives including prefix and prompt tuning (Li and Liang, 2021; Lester et al., 2021; Liu et al., 2022), and stability-oriented components such as gated attention variants that improve optimization dynamics (Qiu et al., 2025). Together, this literature motivates approaches that control interference during continued training and that evaluate syntax gains through both targeted behavioral tests and layer-wise structural probes. These principles are reflected in our staged training pipeline and our joint evaluation.

3 Method

We augment a decoder-only Transformer with an auxiliary, checkpoint-compatible pathway that injects hierarchical syntactic structure through gated cross-attention. The backbone architecture remains unchanged, and the structural pathway is attached as a checkpoint-compatible forward wrapper and can be enabled or disabled without re-training from scratch (Figure 1).

3.1 Precomputed Parse Tree Structure Aligned to Token Spans

Parse tree memory. As shown in Figure 2 (a), given a tokenized input sequence $x_{1:n}$, we precompute a constituency parse tree offline, using Berkeley Neural Parser (Kitaev and Klein, 2018). Each tree node (chunk) u is aligned to a token span $S(u) \subseteq \{1, \ldots, n\}$ and stores its chunk type and parent or child relations. When a parser word is split into multiple Byte Pair Encoding (BPE) tokens, we treat its subword block as a subnode (Tang et al., 2021). For efficiency and determinism, we cache the parsed structure and span alignments, indexed by a hash of the input token IDs. This design eliminates parsing overhead during training and ensures that the same input always retrieves the same tree.

MCQA-specific structure and token update mask. Our training uses a MCQA format, where the input contains a question region and a set of candidate options (Figure 2 (a)). To prevent the structural branch from altering the hidden states of candidate options (which would change their relative likelihoods under likelihood-based scoring), we treat option tokens as read-only memory under the structural pathway.

Concretely, we define a binary token update mask $m_{\text{tok}} \in \{0, 1\}^n$ that enables structural updates only for question tokens and the answer field, while forcing $m_{\text{tok}} = 0$ for option tokens. This prevents option preference drift during continued training and confines structural intervention to the parts of the input where syntactic reasoning is intended to operate.

3.2 Hierarchical Parse Tree Encoder for Chunk Memory Construction

For the token i, we denote the token embedding by $E^i \in \mathbb{R}^{1\times d}$, where d is the hidden size. The precomputed constituency tree contains a set U of chunks, and each chunk $u \in U$ is aligned to a token span $S(u) \subseteq \{1, \ldots, n\}$. And we define the height of the chunk u as $h(u) = D - depth(u)$, where $depth(u)$ is the number of edges from the root to u and D is the maximum depth over the leaf tokens in the constituency tree for the current input. Thus, the deepest leaf tokens have height 0 and the root has the maximum height (Figure 2 (b)). For each chunk u at height $h(u)$, we first aggregate token

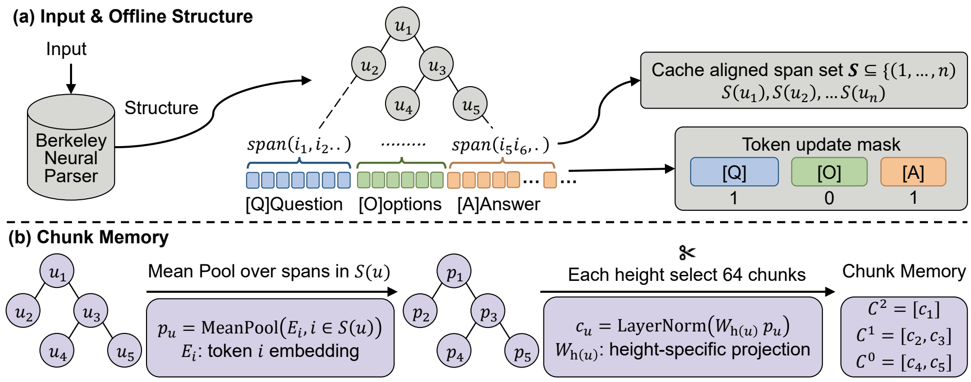

Figure 2: A constituency parser produces cached span-aligned nodes $S(u)$ and update mask. Chunk memory is computed by mean-pooling over $S(u)$ followed by a height-specific projection and LayerNorm.

states over its span through mean pooling:

$$p_u = \text{MeanPool}\left(\overline{E_i}, i \in S(u)\right) \tag{1}$$

We then apply a height specific linear projection based on tree height and normalize the resulting chunk embeddings for stability:

$$c_u = \text{LayerNorm}\left(W_{h(u)}\, p_u\right) \tag{2}$$

where $W_{h(u)} \in \mathbb{R}^{d\times d}$ is shared by chunks of height $h(u)$. To couple tree height with Transformer layers, we use the layer index ℓ to select a height level for chunk memory, then the per-chunk embeddings are stacked to form the chunk memory matrix at layer ℓ:

$$\tilde{\ell} = \min(\ell, D), \quad C^{\ell} = [c_u]_{u\in U,\, h(u)=\tilde{\ell}} \tag{3}$$

Because the maximum chunk height D varies by input, when $D < \ell$ we reuse the top-level chunk memory for higher Transformer layers. To keep computation and memory manageable, we retain at most $K = 64$ chunks per height. Concretely, we traverse the constituency tree in a left-to-right breadth-first order, and for each height h we keep the first K chunks encountered.

This layer-aligned design, where the chunk C^{ℓ} is consumed by the Transformer layer ℓ, is intended to encourage an intuitive hierarchy of structural utilization, such that the higher layers draw on the higher-level parse chunks while the lower layers primarily rely on more local structure. We empirically probe this effect via UUAS (Section 4.4). Controlled ablations on Qwen-2.5-7B for the projection scheme, span pooling, and memory size are reported in Appendix A.

3.3 Gated Parse Tree Cross-Attention Injected via Wrappers

We inject parse tree structure through an auxiliary cross-attention branch implemented as a forward wrapper, using head-wise gated attention (Qiu et al., 2025). The headwise gate is adopted because previous work showed that it introduces far fewer parameters than the elementwise gate while achieving comparable perplexity and downstream accuracy, thus reducing memory overhead and stabilizing training.

At Transformer layer ℓ, let C^{ℓ} denote the chunk memory at height ℓ (defined in Section 3.2), and $\overline{H^{\ell}_{\text{pre}}}$ denote the pre-normalization token hidden states before the GTCA update. We use projection matrices $W^{\ell}_Q, W^{\ell}_K, W^{\ell}_V$ and W^{ℓ}_G, then compute queries, keys, values, gate score, and attention output with head dimension d_h:

$$\begin{aligned} Q^{\ell} &= \overline{H^{\ell}_{\text{pre}}} W^{\ell}_Q, \quad K^{\ell} = C^{\ell} W^{\ell}_K, \\ V^{\ell} &= C^{\ell} W^{\ell}_V, \quad G^{\ell} = \overline{H^{\ell}_{\text{pre}}} W^{\ell}_G, \end{aligned} \tag{4}$$

$$\text{Attn}^{\ell} = \text{softmax}\left(\frac{Q^{\ell}(K^{\ell})^T}{\sqrt{d_h}} + M_{\text{causal}}\right) V^{\ell} \tag{5}$$

where M_{causal} masks out chunks whose right boundary exceeds the current token position, ensuring that the cross-attention never accesses future information. Following prior work on head-wise gated attention, G^{ℓ} serves as the gating logit after the attention output:

$$\text{Gated\_Attn}^{\ell} = \text{Attn}^{\ell} \odot \sigma(G^{\ell}) \tag{6}$$

where σ is the logistic sigmoid and $\odot$ denotes elementwise multiplication. Here, G^{ℓ} provides one logit per attention head, broadcast over d_h. The gated heads are then merged and projected, and we use them to update token hidden states:

$$\Delta H^{\ell} = \text{Merge}(\text{Gated\_Attn}^{\ell}) W_O^{ca,\ell} \tag{7}$$

where $W_O^{ca,\ell}$ is the cross-attention output projection that maps the merged multi-head output back to the model hidden size, and token hidden states are updated as:

$$H_{\text{post}}^{\ell} \leftarrow H_{\text{pre}}^{\ell} + \alpha_{\text{struct}} \big(m_{\text{tok}}[:, \text{None}] \odot \Delta H^{\ell} \big) \tag{8}$$

where the token update mask $m_{\text{tok}} \in \{0,1\}^n$ is defined in Section 3.1 and the structural coefficient α_{struct} follows the schedule in Section 3.4. Here $m_{\text{tok}}[:, \text{None}]$ denotes broadcasting along the hidden dimension, so that each row of ΔH^{ℓ} is multiplied by the corresponding mask entry before the masked update is added to H_{pre}^{ℓ} to obtain H_{post}^{ℓ}.

3.4 Three-Stage Training to Balance Syntax Gains and Retention

We use a staged training scheme to balance syntax specialization and the retention of broad general ability.

Stage 1: Adaptation of the task. We apply LoRA adapters to the backbone projection matrices to fit the MCQA-style training objective and the corresponding instruction format with limited trainable parameters. During this stage, the structural pathway is disabled by setting α_{struct}=0, so the adaptation does not rely on the injected structure.

Stage 2: Integration of the structure. We freeze the backbone and LoRA adapters, train only the parse tree encoder, cross-attention, and gating modules. To avoid abrupt distribution shifts, α_{struct} is warmed up from 0 to $\alpha^{*}_{\text{struct}}$ over the first 10% of Stage 2 steps.

Stage 3: Joint refinement. We jointly train the structural modules and LoRA adapters while keeping the rest of the backbone frozen. This stage reconciles structure-sensitive updates with the task-adapted representations, and aims to preserve the syntax gains from Stage 2 without incurring expensive full fine-tuning.

Overall, GTCA's staged design enables efficient structure-aware adaptation by introducing only an $O(nK)$ cross-attention overhead, which remains cheaper than the $O(n^2)$ self-attention in the backbone, while keeping the pre-trained model largely intact and the process inspectable.

	#Train	#Dev	#Test	#Class
CLOTH	76850	11067	11516	4
MMLU	99842	285	14042	4
CoLA	8551	527	516	2
BLiMP	/	/	67000	/
HellaSwag	39905	10042	10003	4
WinoGrande	40398	1267	1767	2

Table 1: The statistics and segmentation usage of the dataset, only the first two datasets were used for training.

4 Experiments

4.1 Experimental Setup

Model variants. We evaluated four checkpoint-based variants instantiated from a common pre-trained backbone: (i) Backbone, the original pre-trained model without continued training; (ii) LoRA-only, low-rank adapter task adaptation while disabling the structural pathway ($\alpha_{\text{struct}} = 0$). This baseline is trained for the full number of optimization steps and therefore is not the intermediate Stage1 checkpoint; (iii) Direct-Joint, single-stage training of structural modules and LoRA adapters in Stage 3 setup, skipping Stages 1 to 2; and (iv) GTCA, our proposed gated tree cross-attention model trained with the three-stage pipeline (Section 3.4).

Unless otherwise stated, all experiments in the main manuscript are reported in both Qwen-2.5-7B (Qwen et al., 2025) and Llama-3-8B (Grattafiori et al., 2024), using the same maximum sequence length and prompts, computing budgets, and the number of training steps, while keeping the default tokenizer of each backbone. Additionally, Appendix B provides parameter counts for the 8 variants (4 variants × 2 backbones), along with dataset details, the overall compute budget and computing infrastructure.

Datasets and preprocessing. Our continued training used CLOTH and MMLU in a unified MCQA format (Hendrycks et al., 2021; Xie et al., 2018). We evaluated on CLOTH, MMLU, BLiMP, CoLA, HellaSwag, and WinoGrande. MMLU and CLOTH are included to assess MCQA performance. BLiMP and CoLA primarily test targeted syntactic competence. HellaSwag and WinoGrande primarily measure more general-purpose capabilities, such as commonsense and broader reasoning. Dataset split statistics and usage are summarized in Table 1.

For structure-aware variants, we precomputed constituency parse trees offline, cached the result-

Stage	Learning rate	LoRA rank/α_{lora}	Max_length	α_{struct}
Stage1	5e-5	16/32	1024	0
Stage2	3e-5	0/0	1024	0 to 0.15
Stage3	5e-5	16/32	1024	0.15

Table 2: Hyperparameters at different stages.

ing parse tree memory. To ensure that all examples could be successfully parsed, we applied a minimal text normalization to the raw data by collapsing consecutive whitespace into a single space, and modeled each sentence independently in all the experiments. For MCQA prompts, we parsed each field independently (question, each option, and the answer prefix), rather than parsing the full concatenated prompt. We then concatenated the resulting span-aligned chunks in the original prompt order. In addition, Appendix C provides the prompts used for each benchmark along with the corresponding dataset, checkpoint licensing and terms of use.

Training details. We followed the three-stage training scheme in Section 3.4. The key hyperparameters are reported in Table 2. We selected the structural coefficient $\alpha^*_{\text{struct}} = 0.15$ through grid search (Appendix D).

Due to the checkpoint-compatible design, the absolute differences between the backbone parameters before and after training are 0.0 for both backbones, indicating that the original backbone checkpoint parameters remain unchanged.

Evaluation details. For MCQA-style benchmarks, given a prompt $prompt$ and r candidate options $\{option^r\}$, each option is scored by its conditional log-likelihood under the model. More precisely, it is computed as the sum of token-level log-probabilities when generating the option conditioned on the prompt and previously generated tokens in the option:

$$\text{score}\big(option^j \mid prompt\big) = \sum_{i=1}^{|option^j|} \log p_\theta\Big(option^j_i \mid prompt,\ option^j_{<i}\Big) \tag{9}$$

where $j \in \{1, \ldots, r\}$ indexes candidate options and i indexes tokens within $option^j$. And our token update mask can keep the option log-likelihoods in Equation 9 comparable across candidates, thereby cleanly reflecting the influence of syntactic structure in the tree. The prediction was the option with the highest score:

$$\hat{j} = \arg\max_{j \in \{1,\ldots,r\}} \text{score}\big(option^j \mid prompt\big) \tag{10}$$

When option lengths vary strongly, length normalized option scoring is preferable, but in our setting it leaves the method ranking unchanged.

Under k-shot in-context evaluation, we prepended k demonstration examples to the prompt at inference time, and applied the same scoring rule. For BLiMP, we instead used the pairwise likelihood preference, considering an item as correct if $\log p(s^+) > \log p(s^-)$, where s^+ is the good sentence and s^- is the bad sentence. Following standard practice, the Matthews correlation coefficient (MCC, %) is used for CoLA and accuracy (%) for all other benchmarks. And for all our experiments in this paper, we reported mean scores over 5 runs.

Model	MCQA		Syntax		Commonsense	
	CLOTH (0-shot)	MMLU (0-shot)	BLiMP	CoLA (0-shot)	HellaSwag (10-shot)	WinoGrande (5-shot)
Backbone: Qwen-2.5-7B						
Backbone	74.60	69.98	78.58	**59.71**†	61.79	73.01
LoRA-only	83.71	70.12	80.87	53.31	62.74	**75.10**
Direct-Joint	**84.80**†	69.55	80.27	52.75	62.04	74.23
GTCA	83.98	**71.02**†	**83.12**†	56.59	**63.23**†	74.95
Backbone: Llama-3-8B						
Backbone	41.79	54.14	79.95	53.57	61.76	76.72
LoRA-only	81.38	53.78	80.70	53.68	63.24	76.01
Direct-Joint	80.46	52.36	80.36	52.46	62.86	76.98
GTCA	**82.74**†	**54.97**†	**84.61**†	**56.69**†	**64.85**†	**77.89**†

Table 3: Total scores on CLOTH, BLiMP, MMLU, CoLA, HellaSwag, and WinoGrande for Backbone, LoRA-only, Direct-Joint, and GTCA. Boldface with † indicates the best result when the improvement is significant under paired tests with Holm–Bonferroni correction ($p < 0.05$ after correction).

4.2 Multiple-Choice QA Performance on CLOTH

The CLOTH and MMLU results in both Qwen-2.5-7B and Llama-3-8B are shown in Table 3. Notably, the Llama-3-8B Backbone starts much lower on CLOTH, suggesting it is less well aligned with our instruction-style MCQA prompt format before adaptation. LoRA-only substantially improves in-format accuracy over the Backbone method on both backbone models, confirming that parameter-efficient continued training effectively adapted the model to the MCQA format.

Importantly, adding parse-tree memory via gated cross-attention preserves in-format performance across backbone models. Our method GTCA achieves 83.98/71.02 (CLOTH/MMLU) on Qwen-2.5-7B and 82.74/54.97 on Llama-3-8B, outperforming both the Backbone and the LoRA-only baseline, indicating that the token update mask and gated structural pathway do not impede optimization on training-format tasks. Direct-Joint attains slightly higher CLOTH accuracy on Qwen-2.5-7B, but, as shown next, exhibits a less favorable trade-off on retention and targeted syntactic generalization.

4.3 Retention, Robustness, and Syntactic Generalization

The overall performance is reported in Table 3 for both Qwen-2.5-7B and Llama-3-8B. On the syntax focused benchmarks BLiMP and CoLA, GTCA improves BLiMP accuracy over both Backbone and LoRA-only and partially recovers the CoLA degradation introduced by continued training across both backbones. In particular, the LoRA-only stage also improves BLiMP. This gain is attributed to the fact that some special task adaptation alone can already sharpen local grammatical preferences to some extent, even without explicit structural augmentation.

In the general capability benchmarks HellaSwag and WinoGrande, GTCA delivers strong general capacity retention across both backbones. In both Qwen-2.5-7B and Llama-3-8B, GTCA achieves the best HellaSwag performance among the continued training baselines. For WinoGrande, GTCA matches LoRA-only in Qwen-2.5-7B and is strongest in Llama-3-8B. Although GTCA is not uniformly best on every general benchmark and backbone, it achieves top performance in most settings and remains competitive with the strongest baselines elsewhere. Together, these results indicate that staged training and gated integration produce measurable syntactic gains while maintaining broad general capability largely stable.

4.4 Layer-Wise Syntactic Probing

To connect behavioral improvements to representational changes, we performed layer-wise syntactic probing and evaluated it with the UUAS metric. For each model variant, we extracted hidden states from each Transformer layer and trained a lightweight structural probe to recover syntactic structure under controlled conditions. Probing was conducted on a random subset of 3,000 parse trees from the Penn Treebank (Marcus et al., 1999). Following Hewitt and Manning (2019), we evaluated UUAS against dependency trees derived from Penn Treebank annotations, keeping the probing dataset, tree formalism, and probe configuration fixed across model variants to isolate representational differences.

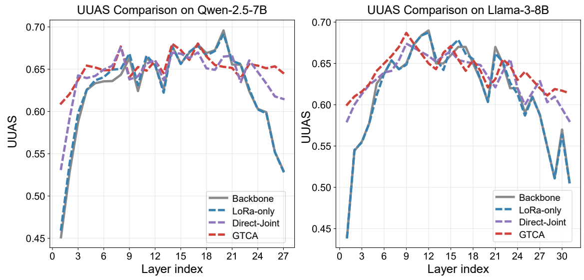

Figure 3: UUAS of the Backbone, LoRA-only, Direct-Joint, and GTCA. UUAS is computed on intermediate layers, excluding the embedding layer.

Figure 3 plots UUAS as a function of layer for Backbone, LoRA-only, Direct-Joint and GTCA in both Qwen-2.5-7B and Llama-3-8B. Across both backbones, structure injection increases syntactic recoverability primarily in the lower and upper layers. This layer-localized shift aligns with the BLiMP and CoLA improvements, providing evidence that GTCA changes internal geometry in a syntax-consistent direction rather than merely overfitting to the training objective.

5 Analysis

We analyzed which components drove the BLiMP and CoLA gains of syntax injection, and how they interacted with retention on general benchmarks. We first ablated gating, token update masking, and staged training, then tested whether improvements depended on coherent parse structure rather than generic auxiliary capacity. To isolate component effects without introducing confounding differences from backbone and tokenizer-specific differences, we conducted these follow-up analyses on Qwen-2.5-7B, while the main results already demonstrate consistent trends across both backbones.

5.1 Ablations and Stage-Wise Effects

Table 4 presents controlled ablations of the entire model. Overall, GTCA consistently delivers the best trade-off between syntactic generalization and retention, suggesting that each design choice contributes meaningfully rather than acting as redundant extra capacity. In particular, removing the gate degrades both the targeted syntax in BLiMP and the broad competence in MMLU, highlighting the role of the gate in controlling interference from the structural pathway. In contrast, disabling the token update mask leaves syntactic performance essentially unchanged but slightly weakens MCQA

Setting	Multiple-Choice QA		Syntactic Robustness	
	CLOTH (0-shot)	MMLU (0-shot)	BLiMP	CoLA (0-shot)
GTCA	**83.98**†	**71.02**†	**83.12**†	**56.59**†
No Gate	82.84	69.80	81.68	55.04
No Mask	83.71	70.50	83.12	55.68

Table 4: Ablation results on key model components. No Mask sets $m_{\text{tok}} = 1$ for all tokens. Boldface with † indicates the best result when the improvement is significant under paired tests with Holm–Bonferroni correction ($p < 0.05$ after correction).

Stage	Multiple-Choice QA		Syntactic Robustness	
	CLOTH (0-shot)	MMLU (0-shot)	BLiMP	CoLA (0-shot)
Stage1	83.71	70.12	80.86	53.31
Stage2	83.70	68.07	**83.50**†	54.13
Stage 1 then Stage 3	83.62	70.66	82.12	55.71
Stage3	**83.98**†	**71.02**†	83.12	**56.59**†

Table 5: Performance dynamics across specialization stages. Boldface with † indicates the best result when the improvement is significant under paired tests with Holm–Bonferroni correction ($p < 0.05$ after correction).

format stability and general retention. On a held-out set in Qwen-2.5-7B, the mean absolute change in pairwise option score gaps is 0.013 per token with the mask and 0.021 without it, consistent with the No Mask ablation in Table 4. This suggests that token update masking mainly stabilizes continued training under the MCQA interface rather than directly driving syntactic gains.

Table 5 makes the stage-wise dynamics explicit and also includes a two stage pipeline that skips Stage 2. Stage 2 delivers the greatest improvement in BLiMP (about +2.6 points), showing that freezing the backbone and specializing structural modules is where syntactic generalization is most strongly acquired. This gain comes with a temporary drop in MMLU, but Stage 3 largely restores and even improves broad competence while preserving most of the syntactic improvement. This pattern suggests that Stage 2 and Stage 3 play complementary roles and that skipping either one yields a weaker tradeoff.

We further conducted a qualitative error analysis on BLiMP in Appendix E, where the remaining errors are concentrated in filler gap dependencies and island effects, suggesting that the model can still over-rely on local surface compatibility and underweight global constraints that require tracking long-distance dependencies or respecting extraction islands.

Tree	Multiple-Choice QA		Syntactic Robustness	
	CLOTH (0-shot)	MMLU (0-shot)	BLiMP	CoLA (0-shot)
Weak parser	80.66	67.33	78.25	54.21
Strong parser*	**83.98**†	**71.02**†	**83.12**†	**56.59**†
Random trees	70.52	64.58	74.36	50.36
Permuted trees	52.57	65.55	60.55	52.66

Table 6: Tree quality and structure-faithfulness controls. Strong parser* corresponds to GTCA (full parser). Boldface with † indicates the best result when the improvement is significant under paired tests with Holm–Bonferroni correction ($p < 0.05$ after correction).

5.2 Tree Quality and Structure-Faithfulness Controls

Table 6 tests whether performance depends on a linguistically meaningful structure and the Strong Parser column corresponds to GTCA. For the Weak Parser setting, we used the Berkeley parser variant with the smallest parameterization, while the remaining settings are obtained by degrading the strong parses by corrupting the tree structure (e.g., randomizing or permuting the resulting trees). As shown in the table, models benefit only when the parse trees are high quality and coherent. Replacing a strong parser with weaker ones consistently lowers both syntactic generalization and broad benchmark performance, indicating that low quality trees act more like noise than guidance. The corrupt structure further harms performance, with random trees causing a clear drop and permuted trees producing the most severe drop. These results support a structure-faithfulness interpretation that gains arise from consistent hierarchical signals in the parse tree memory rather than from an unconstrained auxiliary attention pathway. This interpretation is further supported by additional robustness analyses in Appendix F, where GTCA remains more stable under controlled grammatical perturbations and mild surface noise than the continued training baselines.

5.3 Interpreting the Structural Gate

To summarize, a key design goal of GTCA is to expose syntactic information without forcing the backbone to always use it. The headwise gate provides an interpretable control signal for this purpose. For the token position i in the layer ℓ and the attention head $head$, the effective structural contribution is modulated by

$$g^{\ell}_{head}(i) = \sigma(G^{\ell}_{head}(i)) \tag{11}$$

where $g^{\ell}_{head}(i) \in (0, 1)$ can be interpreted as the confidence of the model in applying a structure-based update to this head. This makes GTCA behaviorally inspectable: when $g^{\ell}_{head}(i)$ is close to zero, the model effectively ignores parse-derived memory for that head; when it is close to one, the model performs a strong structure-conditioned update.

Importantly, this gating mechanism also clarifies why GTCA can improve syntactic benchmarks without consistently harming broad capability. Rather than injecting structure as a hard constraint, GTCA implements structure as a regulated residual. This matches the empirical role of the gate observed in ablations (Table 4), removing the gate degrades both targeted syntactic generalization and retention, suggesting that the gate is not merely an extra capacity, but a stability mechanism that limits interference from potentially noisy or mismatched chunks.

6 Conclusion

This paper presents a checkpoint-compatible framework for strengthening syntactic competence in decoder-only LLMs by attaching a checkpoint-compatible gated tree cross-attention pathway (GTCA). The approach exposes hierarchical constituency structure through precomputed parse tree memory aligned to token spans, and modulates the cross-attention update with a learned headwise gate, while keeping the backbone architecture unchanged. To limit interference during continued training, the method combines token update masking in MCQA inputs with a three-stage training pipeline that separates task adaptation, structure specialization, and joint refinement. Empirically, GTCA improves syntactic generalization while preserving broad competence across multiple evaluations, and this trend holds across different backbones, including Qwen-2.5-7B and Llama-3-8B. Analysis further indicates that gating, token update masking and staged training are important for balancing syntactic gains and retention, and that performance is sensitive to tree quality and coherence, supporting a structure-faithfulness interpretation rather than a pure extra capacity effect.

Limitations

The method relies on external constituency parsers and cached tree memories. As shown by the weak-parser and tree-corruption controls, low-quality or inconsistent parses can act as harmful noise, which makes the approach sensitive to parser accuracy and domain shift. This sensitivity can lead to brittle behavior and degraded outputs when the parse quality drops in real use. In addition, offline parsing and the auxiliary cross-attention branch introduce preprocessing, storage, and runtime overhead, which may become non-trivial for long-context settings or large-scale deployment and can increase latency and compute cost.

Our evaluation focuses on English benchmarks and primarily measures broad competence using MCQA benchmarks. Moreover, extending GTCA to NLI-style training and stress-test setups (e.g., HANS-style controlled evaluations) is left for future work. And these results do not fully characterize downstream impacts on open-ended generation, multi-turn interaction, or long-context reasoning, where failure modes and reliability concerns may differ.

Finally, the current stability mechanisms are tailored to MCQA-style continued training, and broader validation across training paradigms, domains, and model families is required to establish robustness and generality, and to better understand how the method behaves under shifts in data, objectives, and model scale.

Ethics Statement

We use only publicly available pretrained checkpoints and benchmark datasets and follow their licenses and terms of use. No new human participants or manual annotators were involved.

Acknowledgment

The authors would like to thank the anonymous reviewers for their helpful suggestions and comments. This work was supported by the Brain Science and Brain-like Intelligence Technology - National Science and Technology Major Project 2021ZD0204100 (2021ZD0204105 to N. D.).

References

Ananth Agarwal, Jasper Jian, Christopher D Manning, and Shikhar Murty. 2025. Mechanisms vs. outcomes: Probing for syntax fails to explain performance on targeted syntactic evaluations. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 33737–33757, Suzhou, China. Association for Computational Linguistics.

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

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Emanuele Bugliarello and Naoaki Okazaki. 2020. Enhancing machine translation with dependency-aware self-attention. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1618–1627, Online. Association for Computational Linguistics.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: efficient finetuning of quantized llms. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Pablo Diego-Simón, Stéphane D'Ascoli, Emmanuel Chemla, Yair Lakretz, and Jean-Rémi King. 2024. A polar coordinate system represents syntax in large language models. *Preprint*, arXiv:2412.05571.

Pablo J. Diego-Simón, Emmanuel Chemla, Jean-Rémi King, and Yair Lakretz. 2025. Probing syntax in large language models: Successes and remaining challenges. *Preprint*, arXiv:2508.03211.

Tiwalayo Eisape, Vineet Gangireddy, Roger Levy, and Yoon Kim. 2022. Probing for incremental parse states in autoregressive language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2801–2813, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Michael Ginn. 2024. Tree transformers are an ineffective model of syntactic constituency. *Preprint*, arXiv:2411.16993.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. 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. *Preprint*, arXiv:2009.03300.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

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.

Ran Iwamoto, Issei Yoshida, Hiroshi Kanayama, Takuya Ohko, and Masayasu Muraoka. 2023. Incorporating syntactic knowledge into pre-trained language model using optimization for overcoming catastrophic forgetting. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10981–10993, Singapore. Association for Computational Linguistics.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Mary Kennedy. 2025. Evidence of generative syntax in LLMs. In *Proceedings of the 29th Conference on Computational Natural Language Learning*, pages 377–396, Vienna, Austria. 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.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Tomasz Limisiewicz and David Mareček. 2021. Introducing orthogonal constraint in structural probes. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 428–442, Online. Association for Computational Linguistics.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, Dublin, Ireland. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *Preprint*, arXiv:1907.11692.

Mitchell P. Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor. 1999. Treebank-3. LDC Catalog No.: LDC99T42, Web Download.

R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Max Müller-Eberstein, Rob van der Goot, and Barbara Plank. 2022. Spectral probing. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7730–7741, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Moritz Plenz and Anette Frank. 2024. Graph language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4477–4494, Bangkok, Thailand. Association for Computational Linguistics.

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. *Preprint*, arXiv:2505.06708.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Devendra Sachan, Yuhao Zhang, Peng Qi, and William L. Hamilton. 2021. Do syntax trees help pre-trained transformers extract information? In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2647–2661, Online. Association for Computational Linguistics.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038, Brussels, Belgium. Association for Computational Linguistics.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China. Association for Computational Linguistics.

Anfu Tang, Louise Deléger, Robert Bossy, Pierre Zweigenbaum, and Claire Nédellec. 2021. Does constituency analysis enhance domain-specific pretrained bert models for relation extraction? *Preprint*, arXiv:2112.02955.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline. *Preprint*, arXiv:1905.05950.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Mycal Tucker, Tiwalayo Eisape, Peng Qian, Roger Levy, and Julie Shah. 2022. When does syntax mediate neural language model performance? evidence from dropout probes. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5393–5408, Seattle, United States. 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 *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

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.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Qizhe Xie, Guokun Lai, Zihang Dai, and Eduard Hovy. 2018. Large-scale cloze test dataset created by teachers. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2344–2356, Brussels, Belgium. Association for Computational Linguistics.

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.

Shuai Zhang, Wang Lijie, Xinyan Xiao, and Hua Wu. 2022. Syntax-guided contrastive learning for pretrained language model. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2430–2440, Dublin, Ireland. Association for Computational Linguistics.

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.

A Additional Ablations

We ran a targeted ablation set on the tree encoder and memory construction. Then varied one factor at a time, and keep the rest identical to GTCA. The metrics are in Appendix Table 1. All results are on Qwen-2.5-7B, and we set temperature=0 and disabled sampling (do_sample=False).

The results show that a shared projection across heights reduces all metrics, this supports the idea that different granularities need different parameterization. Learned attention pooling gives a small gain on CoLA, K equals 128 chunks per height gives a small gain on BLiMP, but it adds parameters or introduces another learned attention module. Considering the joint trade-off between resource consumption and throughput we use GTCA default as in the main text.

Setting	CLOTH	MMLU	BLiMP	CoLA
GTCA default	**83.98**	**71.02**	83.12	56.59
Single shared projection across heights	81.70	70.76	82.08	**56.90**
Span pooling learned attention pooling	82.04	70.88	80.41	56.04
32 chunks per height	81.25	70.90	82.45	56.20
128 chunks per height	80.36	70.70	**83.20**	55.05

Appendix Table 1: Ablation results on tree encoder and memory construction. All results are on Qwen-2.5-7B with temperature=0 and do_sample=False.

Variant	Backbone	Trainable Params
Backbone	Qwen-2.5-7B	/
LoRA-only	Qwen-2.5-7B	0.01B
Direct-Joint	Qwen-2.5-7B	1.55B
GTCA	Qwen-2.5-7B	1.55B
Backbone	LLaMA-3-8B	/
LoRA-only	LLaMA-3-8B	0.01B
Direct-Joint	LLaMA-3-8B	2.43B
GTCA	LLaMA-3-8B	2.43B

Appendix Table 2: Model variants and sizes.

B Model Size and Compute

B.1 Model Variants and Parameter Counts

For computational efficiency, the reported results for HellaSwag and WinoGrande are evaluated on the validation split. And Appendix Table 2 reports the parameter counts and key configuration differences for each variant.

B.2 Overall Compute Budget and Computing Infrastructure

All continued training and evaluation runs were conducted with a fixed compute budget. Each run was trained with sequence length 1024. Unless otherwise noted, we reported mean results over five runs with 5 different random seeds.

Training was implemented in PyTorch using DistributedDataParallel (DDP) across 8 GPUs per run. The total compute budget was approximately 200–300 GPU-hours. We used bf16 mixed precision and the AdamW optimizer. Relevant software versions are as follows: CUDA 12.8, cuDNN 9.10.2, PyTorch 2.8, NCCL 2.27.3, nltk 3.9.1, and spacy 3.8.7.

C Prompts Used for Benchmark

We used only publicly available benchmark datasets and pretrained checkpoints and followed their original licenses and terms of use. The pretrained backbones are used under Apache License 2.0 for Qwen2.5 and the Meta LLaMA 3 Community License for LLaMA 3, and the benchmark datasets are used under their respective licenses such as MIT, Apache License or Creative Commons, with CLOTH restricted to non-commercial research use. We did not redistribute any dataset content, or model weights, and any derived models are intended solely as research prototypes for controlled evaluation consistent with the access conditions of the underlying resources.

Below are the prompts for each benchmark. For k-shot settings, k demonstration examples were prepended verbatim before the test instance.

C.1 CLOTH (0-shot) and MMLU (0-shot)

Instruction: Choose the correct option based on the question. Output the full text of the chosen option exactly as it appears under Options.
Question: {Question}
Options:
A. {option_A}
B. {option_B}
C. {option_C}
D. {option_D}
Answer:

C.2 BLiMP (minimal pairs, pairwise preference)

Prompt template (no answer token):
{sentence_good}
{sentence_bad}
This evaluation uses two separate forward passes, scoring {sentence_good} and {sentence_bad} independently.

C.3 CoLA (0-shot, binary classification as MCQA)

Instruction: You are a linguist. Decide if the following English sentence is grammatically acceptable. Output 1 for acceptable, 0 for unacceptable. Output only a single character: 0 or 1.
Sentence: {sentence}
Answer:

C.4 HellaSwag (10-shot)

Instruction: Choose the correct option based on the question. Output the full text of the chosen option exactly as it appears under Options.
Demonstration format (repeated 10 times)
Context: Read the context and choose the most plausible continuation.
{context}
Options:
A. {ending0}
B. {ending1}
C. {ending2}
D. {ending3}
Answer: {gold_answer}
Context: Read the context and choose the most plausible continuation.
{context}
Options:

A. {ending0}
B. {ending1}
C. {ending2}
D. {ending3}
Answer:

C.5 WinoGrande (5-shot)

Instruction: Choose the correct option based on the question. Output the full text of the chosen option exactly as it appears under Options.
Demonstration format (repeated 5 times)
Question: {Question}
Options:
A. {option_A}
B. {option_B}
Answer: {gold_answer}
Question: {Question}
Options:
A. {option_A}
B. {option_B}
Answer:

D Grid Search for the Structural Coefficient α_{struct}

GTCA injects parse tree information through a gated tree cross-attention branch and integrates the resulting structural update via a token-update-masked residual weighted by a structural coefficient α_{struct}. In practice, α_{struct} controls the strength of structure injection and therefore mediates a trade-off between (i) syntactic specialization and (ii) retention of general-purpose capability. We select the target α_{struct} via grid search, as stated in the main experiments section.

We grid search over a discrete set of target values, a recommended starting grid is: $\alpha^*_{\text{struct}} \in \{0.00, 0.05, 0.10, 0.15, 0.20\}$. For each candidate α^*_{struct}, we use the same staged training pipeline as Appendix Table 3 in Section3.4, with:

- Stage 1: $\alpha_{\text{struct}} = 0$ (structure path disabled).
- Stage 2: linearly warm up α from 0 to α^*_{struct} over the warm-up steps.
- Stage 3: keep α fixed at α^*_{struct}.

An explicit form of the warm-up can be written as:

$$\alpha(t) = \min\left(\alpha^*_{\text{struct}}, \alpha^*_{\text{struct}} \cdot \frac{t}{T_{\text{warm}}}\right)$$

where t is the current optimization step within Stage 2 and T_{warm} is the warm-up length.

α_{struct}	Multiple-Choice QA		Syntactic Robustness	
	CLOTH (0-shot)	MMLU (0-shot)	BLiMP	CoLA (0-shot)
0.05	81.35	67.78	74.65	53.65
0.10	**82.51**	67.12	75.32	52.14
0.15	81.46	**69.36**	**79.63**	**55.20**
0.20	80.36	65.35	80.12	55.14

Appendix Table 3: Grid search over α on the validation split. All results are on Qwen-2.5-7B, and we set temperature=0 and disabled sampling (do_sample=False).

We select α^*_{struct} using held-out validation performance under a retention-aware criterion. Appendix Table 3 reports the validation results across candidate α_{struct} values.

Across the grid, small α_{struct} values tend to under-utilize structural memory, while overly large α_{struct} can introduce interference that degrades retention. The selected α_{struct} operates in a regime where structure injection is strong enough to improve targeted syntactic judgments yet remains compatible with preserving general-purpose capability. We finally select $\alpha^*_{\text{struct}} = 0.15$, because it achieves the best trade-off between syntactic gains (BLiMP) and retention (MMLU) under the criterion above.

E Error Analysis

Across the BLiMP phenomena that directly probe word order and movement, the residual errors concentrate in filler-gap dependencies and island effects. These cases are informative because the lexical content is largely preserved while the syntactic configuration changes. The examples below show that the model can still over-rely on local surface compatibility and can underweight global constraints that require tracking long-distance dependencies or respecting extraction islands. This points to clear directions for improvement, such as training objectives that better enforce global constraint satisfaction and evaluation protocols that stress paired stability under controlled syntactic transformations.

Examples of residual errors on filler-gap dependencies and island effects. For each phenomenon, we show a grammatical sentence, an ungrammatical sentence with a minimal structural change, and the model's preference. The model consistently prefers the ungrammatical version, indicating overreliance on local surface compatibility and underweighting of global syntactic constraints.

1. **Filler-gap dependency**

   Grammatical: Kendra saw some ice cream that Maria isn't hiding.

   Ungrammatical: Kendra saw what Maria isn't hiding some ice cream.

   Model preference: prefers ungrammatical.

2. **Filler-gap dependency**

   Grammatical: Gina had known that person that Dan hadn't forgotten that had criticized many dancers.

   Ungrammatical: Gina had known who that person that Dan hadn't forgotten had criticized many dancers.

   Model preference: prefers ungrammatical.

3. **Filler-gap dependency**

   Grammatical: Timothy can conceal who those guys that come here weren't scaring.

   Ungrammatical: Timothy can conceal that those guys that come here weren't scaring.

   Model preference: prefers ungrammatical.

4. **Island effects**

   Grammatical: Who wouldn't Mitchell investigate he is kissing?

   Ungrammatical: Who wouldn't Mitchell investigate who is kissing?

   Model preference: prefers ungrammatical.

5. **Island effects**

   Grammatical: Who would actresses' hiding most guests stun.

   Ungrammatical: Who would actresses' hiding stun most guests.

   Model preference: prefers ungrammatical.

F Robustness under Controlled Perturbations and Surface Noise

We construct a controlled grammatical perturbation suite on 1000 randomly sampled CLOTH questions. We apply meaning preserving transformations to the question region, including active passive alternation, relative clause insertion, and adjunct reordering, while keeping the gold answer unchanged. As shown in Appendix Table 4, GTCA achieves the highest perturbed accuracy and the highest consistency. This indicates that GTCA reduces answer flips under syntactic variation and better preserves decision stability when surface form changes but meaning remains constant.

Model	Controlled Grammatical Perturbation on CLOTH		
	Original accuracy	Perturbed accuracy	Consistency
Backbone	76.21	72.34	91.90
LoRA-only	78.47	74.76	91.05
Direct-Joint	80.11	79.02	94.18
GTCA	**81.68**	**80.42**	**98.27**

Appendix Table 4: Performance under controlled grammatical perturbations on 1000 randomly sampled CLOTH questions. Perturbations preserve the gold answer while altering surface syntax. Consistency denotes the fraction of items whose predicted option remains unchanged across the original and perturbed questions. All results are on Qwen-2.5-7B, and we set temperature=0 and disabled sampling (do_sample=False).

Model	Controlled Surface Noise on MMLU		
	Clean accuracy	Noisy accuracy	Consistency
Backbone	67.50	64.80	82.67
LoRA-only	69.10	66.00	82.60
Direct-Joint	70.20	68.60	84.43
GTCA	**72.00**	**71.10**	**88.95**

Appendix Table 5: Performance under controlled surface noise on 1000 randomly sampled MMLU questions. Noise is applied only to the question region and preserves the intended meaning. Consistency denotes the fraction of items whose predicted option remains unchanged across the clean and noisy questions. All results are on Qwen-2.5-7B, and we set temperature=0 and disabled sampling (do_sample=False).

We further test robustness to meaning preserving surface noise, including punctuation deletion, mild typos, and whitespace corruption, on 1000 randomly sampled MMLU questions. Appendix Table 5 shows that GTCA also yields the highest noisy accuracy and the highest consistency under this setting. These results suggest that the structural pathway remains beneficial even when parser inputs are mildly degraded, although the method still relies on parser quality as discussed in the limitations.