LanguageFold: A Bio-inspired Hierarchical Sparse Attention Mechanism for Large Language Models

Yunyi Wu, Kai Huang*
Institute of Systems and Physical Biology,
Shenzhen Bay Laboratory, Shenzhen, Guangdong, China
huangkai@szbl.ac.cn

Abstract

Large language models predominantly rely on the Transformer architecture, whose self-attention mechanism incurs a quadratic computational cost $O(N^2)$ with respect to input length, leading to significant memory and computation bottlenecks when processing ultra-long contexts. This work proposes *LanguageFold*, a hierarchical sparse attention mechanism inspired by the Self-Returning Random Walk model of genome foldingHuang et al. [2020]. LanguageFold decomposes global attention into dynamically constructed tree attention with a theoretical scaling of $O(NlogN)$. Preliminary experiments on prompt-based generation and the DROP reading comprehension benchmark indicate that this tree-structured attention enables efficient language processing while preserving accuracy and enhancing structural interpretability. These results highlight the promise of genome-inspired attention mechanisms for optimizing the scalability of large language models.

1 Introduction

The Transformer architecture Vaswani et al. [2017] has become the dominant backbone of large language models (LLMs), achieving state-of-the-art performance across machine translation, reasoning, and open-domain text generation. Its self-attention mechanism, however, incurs quadratic memory and time complexity with respect to the sequence length N:

$$\mathcal{O}(N^2 d), \tag{1}$$

*Corresponding author.

where d denotes the hidden dimensionality. This scaling poses a fundamental bottleneck for long-context modeling.

Natural language exhibits strong hierarchical and compositional structure, yet standard dense attention lacks explicit inductive biases for representing multi-level dependencies efficiently. Prior work on structured or hierarchical attention Wang et al. [2019], Strubell et al. [2018], He et al. [2024] suggests that embedding structural priors into the attention mechanism may improve interpretability and generalization.

As the language of life, the genome processes vast amounts of biological information with remarkable efficiency. How such information processing emerges from the folding of chromatin in space and time has long attracted broad research interest, including our own. Drawing inspiration from principles of genome folding, we develop **LanguageFold**, a model that introduces hierarchical structured sparsity through Tree Attention. This sparse attention mechanism is inspired by the Self-Returning Random Walk (SRRW) model of chromatin folding Huang et al. [2020]. As a minimal physical model, SRRW recursively combines expansion and self-returning dynamics, generating multiscale structures that account for a wide range of genomic features. Analogously, Tree Attention incorporates a hierarchical, tree-structured sparsity into Transformer attention, decomposing global attention into structured subgraphs that capture the hierarchical organization of token interactions.

Contributions. Our main contributions are summarized as follows:

- We propose a biologically inspired hierarchical sparse attention mechanism, **Tree Attention**, derived from principles of the Self-Returning Random Walk (SRRW) model.
- We introduce two dynamic sparsity-construction strategies—*expansion* and *returning*—which generate tree-shaped hierarchical interaction patterns without predefined structural templates.
- We develop an end-to-end framework including dynamic tree construction, sparse mask generation, and sparsity-balanced filling. We analyze its theoretical computational complexity and hierarchical sparsity characteristics, and we demonstrate that Tree Attention can be integrated into existing Transformer modules with minimal modification.
- Through proof-of-concept experiments on prompt generation and DROP reasoning tasks, we show that Tree Attention produces interpretable hierarchical structure while achieving GPU memory usage comparable to optimized

SDPA kernels. Although the current operator implementation does not yield measurable memory reduction or acceleration, the results validate the feasibility of dynamic hierarchical sparsity and provide a promising direction for future operator-level optimization.

2 Related Work

2.1 Sparse Attention Mechanisms

The self-attention mechanism in Transformers exhibits a quadratic complexity with respect to sequence length N, i.e., $\mathcal{O}(N^2)$, which limits scalability in long-text modeling. To alleviate computational and memory costs, various sparse and approximate attention mechanisms have been proposed, which can be broadly categorized as follows.

Fixed-pattern sparsity. Longformer Beltagy et al. [2020] employs a combination of sliding windows and a small number of global tokens to achieve near-linear complexity. BigBird Zaheer et al. [2020] further incorporates random connections and global nodes while maintaining theoretical guarantees of full connectivity. These models are structurally simple and stable but constrained by fixed sparsity patterns.

Block-wise and low-rank sparsity. Sparse Transformer Child et al. [2019] and Linformer Wang et al. [2020] partition the attention matrix into predefined blocks or approximate it via low-rank projections, achieving linear-time performance for long sequences by reducing redundant attention computations.

Kernel-based and linear approximations. Linear Transformer Katharopoulos et al. [2020] and Performer Choromanski et al. [2020] approximate softmax attention using kernelizable feature maps, achieving theoretical complexity of $\mathcal{O}(N)$ or $\mathcal{O}(N \log N)$. However, these methods may struggle to capture high-frequency or long-range dependencies and are prone to numerical instability.

Dynamic and content-driven sparsity. Reformer Kitaev et al. [2020] employs locality-sensitive hashing (LSH) to cluster similar tokens, while Routing Transformer Roy et al. [2020] dynamically routes attention heads based on semantic similarity, allowing the sparsity pattern to adapt to content dynamically. Instead of enforcing a fixed sparsity pattern, Sparge AttentionZhang et al. [2025] predicts token importance during the prefill stage and selectively activates only a subset

of key-value pairs. The resulting sparsity structure is therefore fully dynamic and input-dependent, enabling substantial acceleration without modifying the model architecture.

Hybrid structured–dynamic sparsity. MInference Jiang et al. [2024] falls between fixed and fully dynamic sparsity. Its sparsity patterns are predetermined (A-shape, Vertical-Slash, Block-Sparse), but the actual sparse indices are constructed dynamically for each input, allowing efficient long-context inference while preserving accuracy.

Hierarchical and structure-aware sparsity. The recent Hierarchical Document Transformer (HDT; He et al., 2024) organizes multi-level sparse attention according to document sections and paragraphs, significantly enhancing long-document discourse modeling and highlighting the potential of structural priors in sparse attention design.

Summary. Overall, existing sparse attention mechanisms explore different trade-offs between efficiency and expressivity. Fixed patterns facilitate engineering optimization, dynamic sparsity offers adaptability, and structure-aware sparsity opens new avenues for modeling long-range dependencies. However, most approaches rely on positional or semantic similarity and lack self-organizing hierarchical structure. In contrast, **LanguageFold** employs Tree Attention, inspired by the SRRW model, to dynamically construct hierarchical structure and enable structurally aware sparse attention.

2.2 Structured and Hierarchical Attention

To explicitly encode hierarchical information in Transformers, various structured attention mechanisms have been explored. Tree Transformer Wang et al. [2019] introduces a constituent-attention module that induces latent syntactic trees in an unsupervised manner. Shiv and Quirk Shiv and Quirk [2019] further propose *Tree-Structured Positional Encoding*, enabling Transformers to operate over sequence–tree and tree–tree tasks.

Beyond latent structure induction, some approaches incorporate explicit syntactic signals. LISA Strubell et al. [2018] uses multi-task learning to train an attention head to attend to syntactic parents, while Dependency Transformer Grammar (DTG; Zhao et al., 2024) modifies attention masks to emulate dependency parsing behavior.

These studies collectively show that hierarchical or dependency structures can improve Transformer modeling. However, most depend on external parsers or predefined rules. In contrast, **Tree Attention** uses hierarchical sparsity patterns end-to-end, providing structure-aware modeling without explicit syntactic supervision.

2.3 Biological Inspiration from Chromatin Folding

The Self-Returning Random Walk (SRRW) model, proposed by Huang et al. [2020], was designed to simulate the hierarchical topology of chromatin folding in three-dimensional space. Its central idea allows a spatial random walk to probabilistically "return" to previously visited positions, thereby producing multi-scale structures that balance local aggregation with global connectivity.

Drawing from this concept, we interpret the attention propagation process as a random traversal in semantic space, introducing a self-returning mechanism to form hierarchical organization in attention computation. This analogy underlies the design of **Tree Attention**, where the hierarchical contraction of information flow mimics the folding and compaction behavior of chromatin, enabling the emergence of biologically inspired multi-level sparsity.

2.4 Scaling Laws of Chromatin Contacts

High-throughput chromosome conformation capture (Hi–C) experiments Lieberman-Aiden et al. [2009] demonstrate that the contact probability $P(s)$ between two genomic loci decays as a power law of their genomic separation s, typically following $P(s) \sim s^{-1}$. Consequently, the expected number of significant contacts for a given locus scales as

$$\int_1^N s^{-1}\, ds \sim \ln N,$$

and the total number of biologically plausible interactions across a genome of size N is therefore bounded by $O(N \log N)$, far below the naive $O(N^2)$ dense interaction assumption.

The Self-Returning Random Walk (SRRW) model was introduced as a computational surrogate that reproduces this property of chromatin folding. Through recursive expansion and probabilistic self-returning dynamics, SRRW generates contact patterns exhibiting a power-law decay of the form $P(s) \sim s^{-1}$, reproducing the characteristic $O(N \log N)$ sparsity. This provides an algorithmic mechanism for producing hierarchical, distance-biased connectivity.

Tree Attention inherits this sparsity pattern indirectly: it adopts the hierarchical topology generated by SRRW. The resulting attention graph contains on the order of $O(N \log N)$ *expected* interaction edges under the SRRW generative process.

By iterating primarily over these SRRW-inspired sparse pairs—rather than the full N^2 attention matrix—Tree Attention *approximates* an effective computational cost closer to $O(N \log N)$ while preserving multiscale contextual pathways.

3 Methodology

3.1 Overview

The overall pipeline of Tree Attention consists of five stages: (1) tree node definition, (2) dynamic tree construction and update, (3) tree-to-mask mapping, (4) sparsity balancing, and (5) attention computation under hierarchical mask constraints.

Analogous to the SRRW model that describes hierarchical folding of chromatin in 3D space, Tree Attention maps this mechanism to neural attention computation, enabling structure-aware multi-level sparsity formation. Unlike full attention that computes all pairwise token interactions, Tree Attention dynamically constructs an adaptive tree structure, performing token-level expansion and returning operations to generate a hierarchical sparse attention mask. This structure preserves key information flow paths while significantly reducing matrix density, achieving a balance between computational efficiency and structural expressivity.

3.2 Tree Node Definition

We define a basic node data structure `TreeNode` as follows:

`TreeNode` = { indices, parent, children, node_rep, is_merged, visited }.

Here:

- `indices`: the set of token indices contained in the node;
- `parent`: a pointer to the parent node;
- `children`: the list of child nodes;
- `node_rep`: the aggregated node representation;
- `is_merged`: flag indicating whether the node is merged;
- `visited`: traversal status indicator.

Each attention head maintains its own independent tree structure during forward propagation to capture the tree patterns at different granularities.

3.3 Dynamic Tree Construction and Update (SRRW-Inspired)

The core of Tree Attention is to simulate the SRRW behavior of local traversal, self-returning, and hierarchical folding. For each query vector q_i, the algorithm computes similarity scores over a candidate node set $C = \{n_{\text{current}}, n_{\text{parent}}\}$:

$$\text{score}(q_i, n) = \text{mean}\left(\text{softmax}\left(\frac{q_i K_n^\top}{\sqrt{d}}\right)\right), \tag{2}$$

where K_n denotes the key representation of node n.

If $\text{score}(q_i, n_{\text{current}}) > \text{score}(q_i, n_{\text{parent}})$, an *expansion step* is executed to create a new node; otherwise, a *returning step* merges the token into its parent node. This process corresponds to the probabilistic balance between exploration and folding in SRRW, allowing dynamic formation of multi-level adaptive clusters.

3.4 Tree-to-Mask Mapping

After constructing the tree structure, we derive a sparse attention mask $M \in \{0,1\}^{N\times N}$ via a *Tree-to-Mask* function:

$$M_{i,j} = \begin{cases} 1, & j \in \text{indices}(n_i) \cup \text{indices}(\text{adjacent layer nodes}), \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

A parameter *rounding* controls the propagation depth, analogous to the contact distance in chromatin, determining the range of information diffusion. Empirically, *rounding* $= 4$–6 achieves a good trade-off between sparsity and coverage.

3.5 Sparsity Balancing

To prevent the mask from becoming excessively sparse, we introduce two balancing strategies:

1. **Stride-based filling:** Adjusts mask density according to a target sparsity ρ_{target} by inserting uniform stride-based activations.
2. **Random filling:** Randomly activates zero entries to emulate stochastic self-returning behavior in SRRW according to ρ_{target}.

These strategies balance sparsity and information coverage. Empirical results suggest that setting $\rho_{\text{target}} \approx 0.4 - 0.6$ yields optimal trade-offs between model performance and memory efficiency.

3.6 Attention Computation Under Hierarchical Mask Constraints

To accommodate the dynamic hierarchical tree structure, the proposed Tree Attention can be formulated in two logically equivalent but computationally distinct ways: (i) mask-based logical sparsification, and (ii) index-based sparse multiplication.

Method 1: Mask-based Sparsification (Logical View). This formulation keeps the standard attention pipeline while using a binary mask $M \in \{0,1\}^{N\times N}$ to prune inactive connections.

1. **Scaled dot-product attention:**

$$A = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d}} + B\right), \tag{4}$$

2. **Masking inactive entries:**

$$A' = A \odot M, \tag{5}$$

3. **Output aggregation:**

$$O = A'V. \tag{6}$$

Note: This method requires constructing the full $N \times N$ attention matrix, resulting in $O(N^2)$ memory usage.

Method 2: Index-based Sparse Multiplication (Efficient Implementation). This implementation avoids dense matrix construction and computes only the active interactions defined by the tree-derived index sets.

1. **Sparse QK computation over active sets:** For each query token i, let its active key index set be S_i:

$$A_i = \frac{Q_i K_{S_i}^\top}{\sqrt{d}} + b_i. \tag{7}$$

2. **Sparse softmax normalization:**

$$A'_i = \mathrm{softmax}(A_i). \tag{8}$$

3. **Sparse output aggregation:**

$$O_i = A'_i V_{S_i}. \tag{9}$$

Note: The complexity is reduced from $O(N^2)$ to $O(|\mathcal{E}|)$, where $\mathcal{E}$ denotes the set of active edges in the hierarchical tree.

3.7 Model Integration and Compatibility

The Tree Attention module can directly replace standard attention in existing Transformers without altering the backbone architecture. Each attention head maintains its own tree state to support autoregressive generation. For instance, in the Qwen series models:

- A distinct Tree Attention module is injected at each layer;
- Replace the original attention operator;
- Tree states are incrementally updated during inference for autoregressive decoding.

3.8 Complexity Analysis

The computational efficiency of the proposed method is grounded in its hierarchical design. Theoretically, the core Tree-only Attention mechanism exhibits an intrinsic complexity of $\mathcal{O}(N \log N \cdot d)$, derived from the *LanguageFold* paradigm that organizes token interactions into a hierarchical tree structure.

In practice, to balance computational sparsity with representation power, we introduce a balancing hyperparameter ρ to regulate the attention mask, resulting in an effective complexity of $\mathcal{O}(\rho N^2 d)$. While ρ is typically set between 40%–60% to ensure optimal performance, the tree construction process itself remains highly efficient with a complexity of $\mathcal{O}(Nd)$.

Unlike naive sparse implementations that often suffer from excessive memory overhead, our approach is designed to be memory-efficient. By employing an index-based sparse computation strategy, the mechanism achieves structural sparsity while maintaining a memory footprint comparable to highly optimized dense kernels. A detailed empirical analysis of runtime memory and operator efficiency is provided in Section 4.6.

4 Experiments

4.1 Experimental Setup

To evaluate the effectiveness of Tree Attention in terms of computational efficiency, memory optimization, and semantic preservation, we conduct systematic experiments on the Qwen2.5 model family with parameter scales ranging from 1.5B to 14B. Our objective is to assess whether the proposed mechanism can retain core language understanding and generation capabilities while keeping memory usage on par with standard baselines. We consider three categories of evaluation tasks.

Complexity Scaling and Energy Capture. This evaluation quantifies the trade-off between computational sparsity and information retention. We measure the number of active edges in the Tree-only Attention mask across varying sequence lengths N (from 2^{10} to 2^{13}) to empirically verify the $O(N \log N)$ complexity scaling against the quadratic $O(N^2)$ baseline. To assess the "quality" of this sparsity, we introduce the *Energy Capture Ratio* (ECR), defined as the proportion of cumulative attention mass from the original dense model that is preserved within the sparse Tree-only Attention pathways. This metric allows us to determine if the tree successfully concentrates its attentional budget on the most significant multiscale dependencies while filtering out redundant noise.

DROP Reading Comprehension. The DROP benchmark contains long passages requiring arithmetic reasoning and multi-hop dependencies, making it a natural stress test for sparse attention. This task allows us to examine how structured sparsification affects compositional reasoning, numerical operations, and cross-sentence consistency.

Prompt-based Text Generation. This task evaluates long-form generation quality, contextual fidelity, hierarchical organization, and instruction following. It serves as a diagnostic tool for detecting potential semantic drift, degradation of structural coherence, or hallucination introduced by sparsity.

Structural Interpretability and Operator Efficiency. For interpretability, we go beyond heatmap inspection and apply graph-theoretic analysis. Tree Attention masks at multiple depths (e.g., Layers 7, 11, 12) are clustered via the Louvain community detection algorithm to determine whether consistent semantic modules emerge. For efficiency, we benchmark VRAM usage across three implementations: a naive mask-based variant, a standard PyTorch SDPA baseline, and our sparse operator. This comparison reveals whether Tree Attention maintains structured sparsity without incurring additional memory overhead.

Unless otherwise specified, all experiments are conducted on a single compute node equipped with two NVIDIA RTX 3090 GPUs (24GB each). All benchmarks use FP32 precision. This setup simulates resource-constrained environments such as consumer-level or edge devices, highlighting the practical utility of lightweight attention mechanisms.

All comparisons are performed between the original full-attention model and its Tree Attention variant under identical hyperparameters to ensure fairness.

4.2 Empirical Validation of Biomimetic Scaling Laws

To verify whether Tree Attention inherits the structural properties predicted by the fractal–globule model, we conduct an empirical analysis of attention weight distributions using the Qwen2.5–1.5B model. We select a long-context input se-

quence ($N = 8567$) from the Qasper dataset within the LongBench benchmark, enabling us to probe attention behaviors across a wide range of genomic-like token distances.

4.2.1 Power-Law Decay of Attention Weights

A defining property of genomic organization is that the contact probability between two genomic loci decays as a power law,

$$P(s) \sim s^{-\gamma}, \quad \gamma \approx 1, \tag{10}$$

where s denotes genomic separation.

To test whether tree structure exhibits analogous scaling behavior, we extract tree-only attention matrices from multiple layers of the model and compute the mean attention weight as a function of token distance s. Figure 1 visualizes the resulting decay profile on a log–log scale.

The empirical results show a striking agreement with the theoretical s^{-1} scaling law, indicated by the dashed reference line. In the short-to-medium range ($1 < s < 10^3$), the mean attention weight across all heads closely follows a slope of $\gamma = 1$. This behavior suggests that the SRRW-inspired hierarchical topology imposes an explicit distance-biased inductive bias, prioritizing biologically (or linguistically) meaningful "contacts" while naturally sparsifying the interaction graph.

4.2.2 Complexity Scaling and Information Retention

To quantitatively validate the efficiency of the Tree-only Attention mask, we evaluate the growth of active interaction edges and the corresponding Energy Capture Ratio (ECR) for Layers 1, 7, and 12 (Figure 2).As shown in Figure 2(a), the number of active edges in the Tree-only Attention mask closely follows the $O(N \log N)$ trajectory, maintaining a significant gap below the quadratic $O(N^2)$ baseline of standard Transformers. This confirms that tree structure effectively achieves the theoretical computational sparsity.Crucially, Figure 2(b) demonstrates that this sparsity does not entail a proportional loss of information. In Layer 1, the model retains an ECR of 77.0% even at a sequence length of $N = 8192$. While the ECR naturally decreases as sequence length increases—due to the widening gap between $O(N \log N)$ and $O(N^2)$—the model consistently captures the majority of the attention mass (e.g., 54.9% in Layer 12 at $8k$ length) using only a fraction of the theoretical dense connections. This stability confirms that tree structure successfully concentrates on the most critical multiscale contextual pathways while filtering out $O(N^2)$ dense noise.

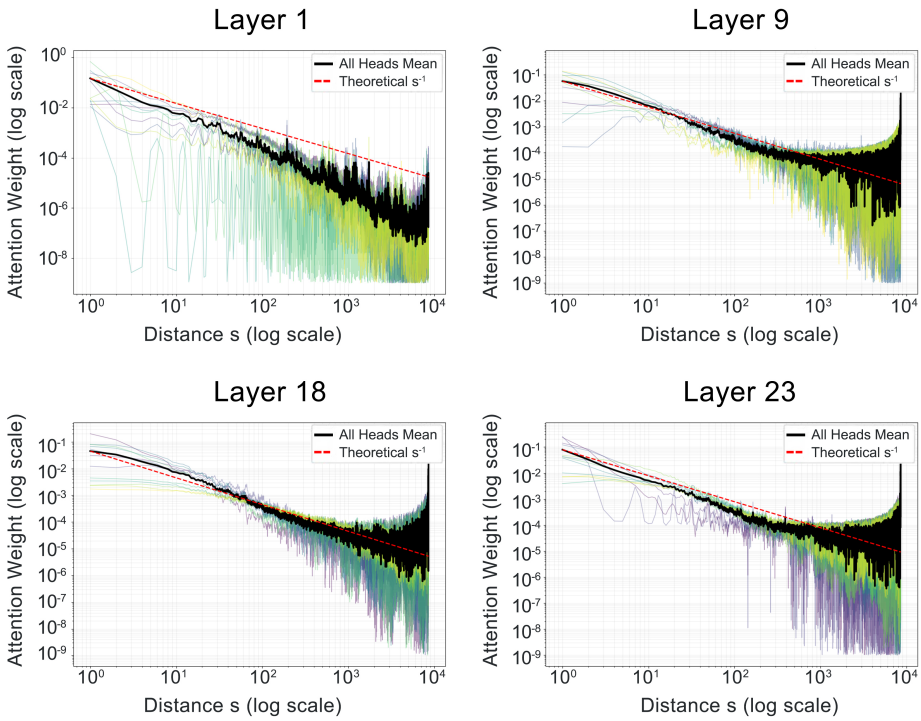

Figure 1: **Layer-wise Tree-only Attention Decay and Scaling Consistency.** Attention weights per head (colored lines) for Layers 1, 9, 18, and 23 are plotted against token distance s on a log-log scale. The solid black line denotes the all-head mean, while the red dashed line indicates the theoretical s^{-1} biomimetic scaling law.

4.3 DROP Results and Analysis

Table 1 summarizes the performance of four Qwen2.5 model scales (1.5B, 3B, 7B, 14B) on the DROP dataset.

We adopt the parameter *rounding* = 6 and the target sparsity coefficient of $\rho_{\text{target}} = 0.5$. Under this configuration, Tree Attention exhibits an average relative performance drop of approximately 49.8% with respect to the dense baseline.

This degradation reflects an inherent *structural trade-off*. As later shown in Section 4.6, the total VRAM footprint of Tree Attention is comparable to that of the highly optimized PyTorch SDPA kernel. Despite introducing hierarchical constraints and structured sparsity, the method maintains memory usage on par with industrial dense operators.

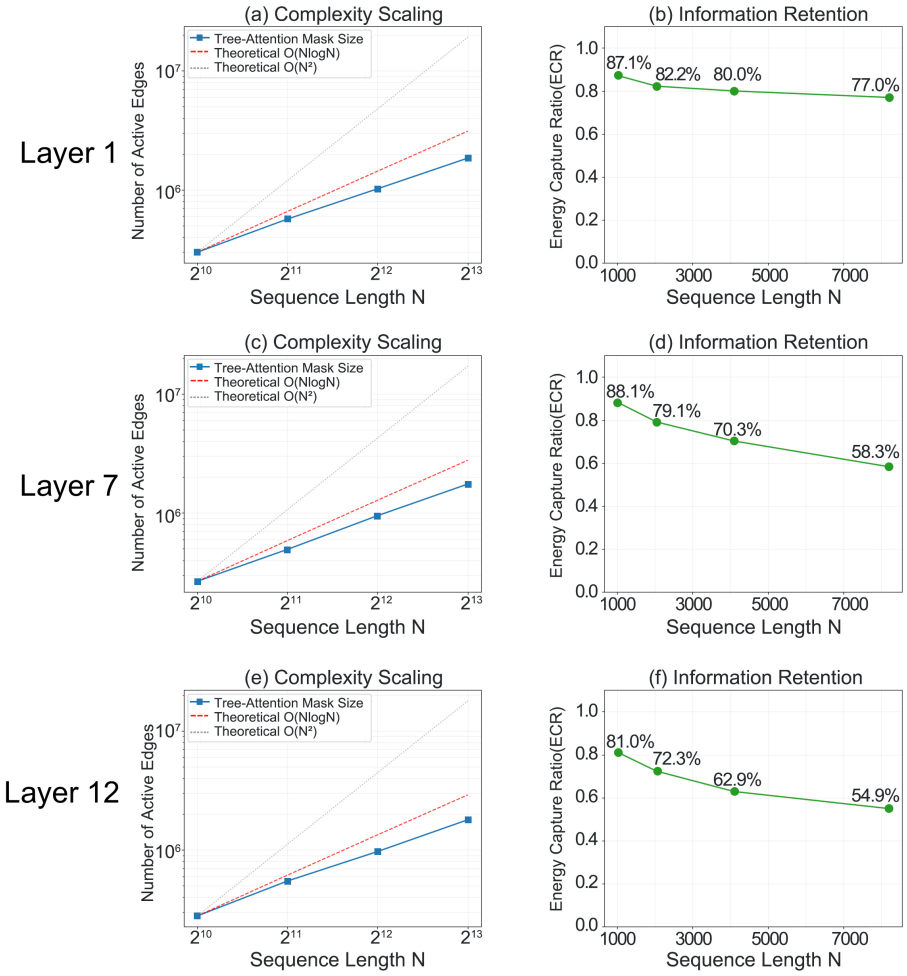

Figure 2: **Complexity Scaling and Information Retention Analysis.** Quantitative evaluation of the Tree-only Attention mask across Layers 1, 7, and 12. (a) **Complexity Scaling**: The number of active interaction edges in the Tree-only Attention mask (blue line) strictly follows the theoretical $O(N \log N)$ trajectory (red dashed line), maintaining a substantial efficiency gap compared to the $O(N^2)$ dense baseline. (b) **Information Retention**: The Energy Capture Ratio (ECR) demonstrates the proportion of attention mass preserved within the sparse mask. Even at $N = 8192$, the model retains a significant portion of the global attention energy (e.g., 77.0% in Layer 1 and 54.9% in Layer 12), confirming that the hierarchical mask accurately captures the most critical multi-scale contextual pathways.

Model	Variant	Total	Correct	Score
Qwen2.5–1.5B	Original	5000	1705	0.3410
	Tree-Attn	5000	831	0.1662
Qwen2.5–3B	Original	5000	1941	0.3882
	Tree-Attn	5000	1004	0.2008
Qwen2.5–7B	Original	5000	2248	0.4496
	Tree-Attn	5000	1030	0.2060
Qwen2.5–14B	Original	5000	2803	0.5606
	Tree-Attn	3126	954	0.3052

Table 1: DROP performance across Qwen2.5 model scales.

To better understand where performance diverges and whether core linguistic competencies are preserved, we conduct a detailed qualitative analysis on 5000 predictions from Qwen2.5–7B:

Numerical Reasoning. DROP places strong emphasis on arithmetic and multi-hop reasoning. Most Tree-Attention-specific errors occur in cases requiring long-range aggregation of scattered operands. *Example.* For a subtraction problem involving "53 yards" and "24 yards," the dense model correctly outputs 29, whereas the sparse variant fails due to missing cross-sentence connections, illustrating that numerical operations are sensitive to sparsity-induced dependency breaks.

Entity Extraction. In contrast, Tree Attention preserves robustness in entity-centric reasoning. *Example.* For the question "Who scored the longest rushing touchdown?", both variants identify *Felix Jones*. This suggests that Tree Attention maintains strong semantic salience tracking even under high sparsity.

Conservativeness and Reduced Hallucination. A notable pattern is that Tree Attention tends to produce more conservative outputs. When critical context is masked out, the model often responds with *"unanswerable"* or *"not enough information"*. This indicates reduced hallucination and a bias toward safer predictions under uncertainty.

Sparsity Configuration. All DROP evaluations employ Tree Attention with stride-based balancing, ensuring deterministic local connectivity. Such stability is important for low-variance linking between adjacent tokens in reasoning-heavy tasks.

4.4 Prompt Generation: Impact of Structured Sparsity

To understand how structured sparsity influences long-form generation—particularly contextual fidelity and global coherence—we conduct stress tests on Qwen2.5–

1.5B under high sparsity levels ($\geq 50\%$). Using a news article on U.S. tariff policy as input, we compare four configurations:

- Full Attention (baseline)
- Random-only sparsity
- Tree + Stride
- Tree + Random

Failure of Unstructured Sparsity. When sparsity is applied through random dropping alone (50%), the model collapses and produces nonsensical text. This highlights the necessity of Tree Attention's hierarchical scaffold for maintaining stability under severe sparsity.

Local Bias in Tree + Stride. Although Tree+Stride performs well on structurally grounded tasks such as DROP, its behavior degrades in abstractive summarization. *Observed behavior:* It repeatedly hallucinates a nonexistent "25% tariff on steel" phrase. *Cause:* The deterministic locality of Tree+Stride amplifies parameterized priors when global context is inaccessible, producing structured but incorrect repetitions.

Tree + Random: Improved Abstraction. Introducing randomness atop the tree structure relaxes local bias while preserving global anchors, yielding summaries with higher abstraction potential but greater output variance. *Example (best case):*

> *"President Trump's Tariff Hike: A Game-Changing Moment for the Global Economy."*

This suggests that the combination of hierarchical structure and stochastic connectivity facilitates flexible global reasoning.

Method	**Sparsity**	**Sample Output**	**Interpretation**
Full	0%	Faithful summary	Baseline
Random-only	50%	Gibberish	Model collapse
Tree+Stride	50%	Output repetition	Local-bias failure
Tree+Random	50–60%	Accurate headline	High-variance success

Table 2: Comparison of sparsity strategies on summarization.

4.5 Interpretability: Semantic Topology and Functional Modularity

To investigate the structural organization introduced by the Tree Attention mechanism, we interpret each attention mask as a *semantic connectivity graph* and apply Louvain community detection to analyze emergent modularity. Figure 3 visualizes the attention matrices for Layers 3, 11, and 28, along with their corresponding community clusters.

Emergence of Strong Diagonal Modularity. As shown in the heatmaps, the attention patterns exhibit a pronounced block-diagonal structure across all sampled layers. The Louvain community clusters (visualized as colored bars at the bottom of each plot) align precisely with these dense square blocks along the diagonal. This alignment confirms that the model is attending to information within discrete, localized "semantic neighborhoods" rather than maintaining a diffuse global context.

Layer-wise Evolution of Semantic Clusters. All three layers display a similar block-diagonal structure, and Louvain clustering consistently identifies separable communities. Although cluster boundaries and densities vary with depth, the global modular pattern persists, indicating that the Tree Attention mask provides a stable yet flexibly refined semantic partition of the input.

Case Study: Semantic Coherence at Layer 11. To further illustrate the interpretability benefits introduced by Tree Attention, we analyze the community structure at **Layer 11, Head 0** (Figure 1). The Louvain algorithm partitions the input news into several highly cohesive semantic communities. The color bar at the bottom clearly delineates these boundaries, which, when aligned with the source text, correspond precisely to the four major logical segments of the news article. Table 3 summarizes the mapping between community clusters, textual segmentation, and their semantic roles.

Cluster	Textual Segment	Semantic Function
Community Brown-1	Trump's tariff remarks, Rose Garden announcement, end of globalization	Thesis Establishment
Community Light Blue	Missile-like tariffs, 20% rate, reciprocal mechanisms, VAT discussion	Mechanism Specification
Community Dark Blue	G7 negotiations, UK economic contraction, Aston University study	Quantitative Consequences
Community Brown-2	Smoot–Hawley Act, Vance's remarks, historical synthesis	Philosophical Synthesis

Table 3: Semantic alignment of Louvain clusters at Layer 11 (Head 0).

Structural Basis for Hallucination Suppression. The visualization confirms

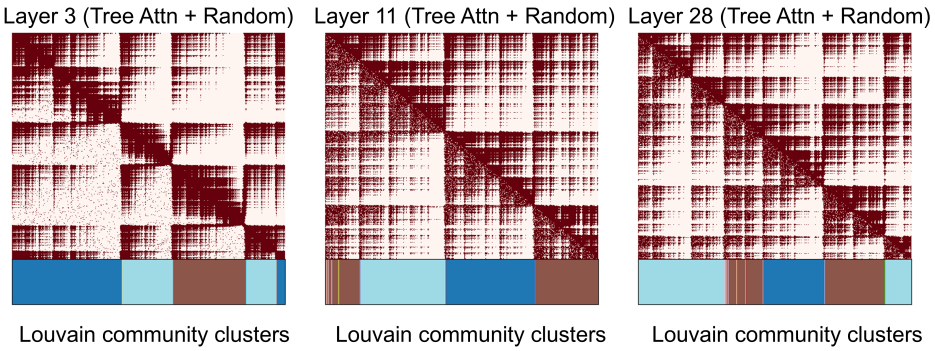

Figure 3: **Attention Modularity across Layers.** Layers 3, 11, and 28 exhibit similar block-diagonal attention patterns. Louvain clustering identifies comparable but not identical communities across depth, indicating that the Tree Attention mask induces stable yet layer-dependent modular structure.

that Tree Attention induces a rigid, interpretable topology. By enforcing this block-wise locality, the architecture:

1. **Restricts Attention Bleed:** Tokens are physically prevented from attending to semantically irrelevant blocks, reducing the probability of hallucinating unrelated facts.

2. **Mimics Hierarchical Reading:** The progression from broad blocks to finer cluster definitions mirrors a human-like decomposition of text into chapters and paragraphs.

4.6 Efficiency and Memory Overhead of Sparse Operators

We evaluate the runtime memory characteristics of different implementation strategies for Tree Attention under `Float32` precision. All measurements are performed on two GPUs using the Qwen2.5–1.5B model, with prompt lengths averaging ≈ 613 tokens.

To quantify the benefit of our "activation-index-based sparse multiplication" in Methodology 3.6, we compare:

1. **Standard Qwen Baseline.** The dense PyTorch SDPA implementation representing the lower-bound memory cost of standard attention.

2. **Naive Mask (Method 1).** A Python-level implementation that explicitly constructs the $N \times N$ attention mask.

3. **Tree Attention (Method 2).** Our fused sparse operator that performs attention via dynamic index selection without materializing dense intermediates.

Table 4: Peak memory usage under `Float32` precision. Static weights occupy ∼5889 MiB; dynamic memory is computed as Peak minus Static.

Method	Operator Type	Peak Memory	Dynamic Memory
Standard Qwen Baseline	PyTorch SDPA	6876 MiB	987 MiB
Naive Mask (Method 1)	Python Explicit	13556 MiB	7667 MiB
Tree Attention (Method 2)	Sparse Operator	6967 MiB	1078 MiB

Naive Mask Implementation. The explicit mask construction leads to large intermediate tensors, including the full attention matrix and multiple broadcasted masks. These artifacts inflate dynamic memory to 7.6 GiB, illustrating that unfused sparsity mechanisms are unsuitable for realistic deployment.

Efficiency of the Sparse Operator. Our fused implementation achieves the intended sparsity without incurring notable overhead:

- **Substantial reduction over naive mask.** Dynamic memory drops by more than 85%, and peak usage is reduced by nearly half.
- **Near-dense memory footprint.** Tree Attention requires only ∼90 MiB more than the optimized SDPA kernel—a 1.3% difference—despite its dynamic indexing and sparse data flow.

Although sparse attention designs commonly introduce additional buffers or irregular access patterns, our operator demonstrates that Tree Attention can be implemented with *negligible* memory overhead. Its memory profile remains comparable to dense SDPA, enabling practical integration into large-scale LLMs while retaining structural sparsity benefits.

5 Discussion

5.1 The Trade-off Between Structural Inductive Bias and Dense Numeracy

Our experiments reveal a distinct trade-off introduced by Tree Attention. While the mechanism successfully reduces the computational complexity to $\mathcal{O}(\rho N^2 d)$

and maintains memory footprints comparable to optimized dense kernels, this efficiency comes at a cost to granular numerical reasoning. The performance drop on the DROP benchmark ($\approx 49.8\%$) suggests that tasks requiring the aggregation of scattered operands (e.g., arithmetic over long contexts) rely heavily on the diffuse, fully connected nature of standard attention. However, the model's retained ability to extract entities and facts indicates that Tree Attention successfully captures the "semantic skeleton" of the text, even if it loses some of the "numerical flesh."

5.2 Biological Fidelity and the Role of Stochasticity

The comparison between "Tree + Stride" and "Tree + Random" configurations highlights the importance of the biological inspiration behind this work. The SRRW model in chromatin dynamics Huang et al. [2020] balances local folding with global accessibility. Similarly, we found that strictly deterministic sparsity ("Tree + Stride") led to local bias and hallucination in generation tasks. Reintroducing stochasticity ("Tree + Random")—analogous to the probabilistic dynamics that establish long-range, non-local contacts in chromatin structure—restored global coherence and allowed for better abstraction. This suggests that effective sparse attention must balance rigid hierarchical structures with flexible, stochastic connections to model the full complexity of natural language.

5.3 Emergent Modularity and Interpretability

A significant advantage of Tree Attention is the unsupervised emergence of interpretable structures. Without external syntactic parsers, the model self-organized into block-diagonal communities that align with human-perceivable document segments (e.g., thesis establishment, mechanism specification). This confirms that the SRRW dynamics successfully induce a hierarchical prior that mirrors the compositional structure of language. Furthermore, this physical restriction of attention flow appears to function as a regularizer, reducing hallucinations by preventing the model from attending to semantically irrelevant blocks during uncertain generation.

6 Conclusion

In this work, we presented Tree Attention, a hierarchical sparse attention mechanism inspired by the Self-Returning Random Walk (SRRW) model of chromatin folding. By dynamically constructing tree-constrained attention masks, our approach introduces controllable structural sparsity into Large Language Models without requiring pre-trained structural templates.

Our empirical evaluation on the Qwen2.5 model family demonstrates that Tree Attention achieves a pragmatic balance between memory efficiency and semantic preservation. While there is a notable degradation in complex numerical reasoning tasks, the method maintains precise entity capture and reduces hallucination through conservative "unanswerable" predictions when context is ambiguous. Crucially, we developed a fused sparse operator that matches the memory footprint of highly optimized dense SDPA kernels (within $\approx 1.3\%$), proving the practical feasibility of dynamic, index-based sparse attention.

Future research will focus on five key areas:

1. **Towards Asymptotically Optimal Complexity:** While Tree-only Attention is inherently hierarchical and supports $\mathcal{O}(N \log N)$ scaling, our current "Tree + Random" and "Tree + Stride" configurations still rely on a simplified masking approach that does not fully exploit this theoretical bound. Future work will focus on refining the balancing algorithms to ensure the complexity is strictly bounded by $\mathcal{O}(N \log N)$.

2. **From Inference-time Priors to Trainable Dynamics:** Currently, Tree Attention is applied as a dynamic prior during inference based on pre-defined SRRW rules. To further enhance performance, we plan to make this hierarchical structure end-to-end trainable. By parameterizing the formation of tree, the model can learn to optimize its own attention topology. This would allow it to adaptively balance between sparse paths and the dense connections needed for complex tasks, effectively learning the most efficient attention pattern during training.

3. **Kernel Optimization:** While memory usage is optimized, the current operator does not yet yield wall-clock acceleration. Developing specialized CUDA kernels for the tree-based sparse matrix multiplication is a priority to realize the theoretical time complexity benefits.

4. **Stochasticity and Robustness Optimization:** Given that the "Tree + Random" configuration outperforms the deterministic "Tree + Stride" configuration in maintaining global coherence and abstraction capability, future work will also focus on integrating adaptive stochastic controllers or learnable probabilistic parameters into the attention construction to enhance the model's accuracy and generalization ability in complex reasoning tasks.

5. **Long-Context Scaling:** We intend to apply Tree Attention to ultra-long contexts ($> 100k$ tokens), where the quadratic bottleneck of standard Transformers is most acute and the hierarchical priors of Tree Attention may prove most beneficial.

Acknowledgements

We acknowledge computational support from the Shenzhen Bay Laboratory High Performance Computing and Informatics Core. This work was supported by The Major Program of Shenzhen Bay Laboratory (S241101001), the National Natural Science Foundation of China (32571445 and 32521002) to K.H.

Author contributions

K.H. conceived the study, designed the LanguageFold model, and supervised the project. Y.W. implemented the model, performed the analyses, and wrote the manuscript.

Declaration of interests

The authors declare no competing interests.

References

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The Long-Document Transformer. *arXiv e-prints*, art. arXiv:2004.05150, April 2020. doi: 10.48550/arXiv.2004.05150.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating Long Sequences with Sparse Transformers. *arXiv e-prints*, art. arXiv:1904.10509, April 2019. doi: 10.48550/arXiv.1904.10509.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking Attention with Performers. *arXiv e-prints*, art. arXiv:2009.14794, September 2020. doi: 10.48550/arXiv.2009.14794.

Haoyu He, Markus Flicke, Jan Buchmann, Iryna Gurevych, and Andreas Geiger. HDT: Hierarchical Document Transformer. *arXiv e-prints*, art. arXiv:2407.08330, July 2024. doi: 10.48550/arXiv.2407.08330.

Kai Huang, Yue Li, Anne R. Shim, Ranya K. A. Virk, Vasundhara Agrawal, Adam Eshein, Rikkert J. Nap, Luay M. Almassalha, Vadim Backman, and Igal Szleifer. Physical and data structure of 3d genome. *Science Advances*, 6(2):eaay4055,

2020. doi: 10.1126/sciadv.aay4055. URL https://www.science.org/doi/abs/10.1126/sciadv.aay4055.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. MInference 1.0: Accelerating Pre-filling for Long-Context LLMs via Dynamic Sparse Attention. *arXiv e-prints*, art. arXiv:2407.02490, July 2024. doi: 10.48550/arXiv.2407.02490.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. *arXiv e-prints*, art. arXiv:2006.16236, June 2020. doi: 10.48550/arXiv.2006.16236.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The Efficient Transformer. *arXiv e-prints*, art. arXiv:2001.04451, January 2020. doi: 10.48550/arXiv.2001.04451.

Erez Lieberman-Aiden, Nynke L. van Berkum, Louise Williams, Maxim Imakaev, Tobias Ragoczy, Agnes Telling, Ido Amit, Bryan R. Lajoie, Peter J. Sabo, Michael O. Dorschner, Richard Sandstrom, Bradley Bernstein, M. A. Bender, Mark Groudine, Andreas Gnirke, John Stamatoyannopoulos, Leonid A. Mirny, Eric S. Lander, and Job Dekker. Comprehensive mapping of long-range interactions reveals folding principles of the human genome. *Science*, 326(5950):289–293, 2009. doi: 10.1126/science.1181369. URL https://www.science.org/doi/abs/10.1126/science.1181369.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient Content-Based Sparse Attention with Routing Transformers. *arXiv e-prints*, art. arXiv:2003.05997, March 2020. doi: 10.48550/arXiv.2003.05997.

Vighnesh Shiv and Chris Quirk. Novel positional encodings to enable tree-based transformers. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/6e0917469214d8fbd8c517dcdc6b8dcf-Paper.pdf.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. Linguistically-Informed Self-Attention for Semantic Role Labeling. *arXiv e-prints*, art. arXiv:1804.08199, April 2018. doi: 10.48550/arXiv.1804.08199.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *arXiv e-prints*, art. arXiv:1706.03762, June 2017. doi: 10.48550/arXiv.1706.03762.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-Attention with Linear Complexity. *arXiv e-prints*, art. arXiv:2006.04768, June 2020. doi: 10.48550/arXiv.2006.04768.

Yau-Shian Wang, Hung-Yi Lee, and Yun-Nung Chen. Tree Transformer: Integrating Tree Structures into Self-Attention. *arXiv e-prints*, art. arXiv:1909.06639, September 2019. doi: 10.48550/arXiv.1909.06639.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big Bird: Transformers for Longer Sequences. *arXiv e-prints*, art. arXiv:2007.14062, July 2020. doi: 10.48550/arXiv.2007.14062.

Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. SpargeAttention: Accurate and Training-free Sparse Attention Accelerating Any Model Inference. *arXiv e-prints*, art. arXiv:2502.18137, February 2025. doi: 10.48550/arXiv.2502.18137.

Yida Zhao, Chao Lou, and Kewei Tu. Dependency Transformer Grammars: Integrating Dependency Structures into Transformer Language Models. *arXiv e-prints*, art. arXiv:2407.17406, July 2024. doi: 10.48550/arXiv.2407.17406.