

Explanation Document

November 21, 2025

1 The Goal: Topological Masking

Paper Reference: Equation (4) and Section 3.1 in `compute_graph_mask`.

The paper argues that nodes in a graph shouldn't attend to everyone equally. They should attend based on Topological Distance.

$$M_a(G) := \sum_{k=0}^{\infty} \alpha_k W^k$$

where

- W is the normalized adjacency matrix normalized.
- α_k are learnable coefficients so if α_1 is high, immediate neighbors matter most. If α_{10} is high, distant relatives matter.
- W^k is a matrix where entry (i, j) represents the number (or weight) of walks of length k between node i and node j .

The Problem: Calculating this sum explicitly creates a dense $N \times N$ matrix.

If $N = 30,000$, your GPU runs out of memory.

2 The Conflict: Masking vs. Linear Attention

Paper Reference: Equation (2) vs Equation (3)

Standard Linear Attention works by decomposing Q and K into feature maps $\phi(\cdot)$ so we never calculate the $N \times N$ attention matrix:

$$Att = \phi(Q)(\phi(K)^T V)$$

But we want to inject the mask M into the middle:

$$Att_{masked} = (\phi(Q)\phi(K)^T \odot M)V$$

Mathematical block: You cannot distribute the Hadamard product (\odot) easily.

$$(A \times B) \odot C \neq A \times (B \odot C)$$

Because linear attention relies on the associativity of matrix multiplication $(AB)C = A(BC)$, introducing the element-wise mask M breaks the "linear" trick. You are forced to materialize the $N \times N$ matrix again.

3 The Solution: The "Dot Product of Outer Products"

Paper Reference: Lemma 3.1 and Equation (5) & (6) This is the most important math concept in the paper.

The authors realize that if they can decompose the Mask M into its own feature vectors, they can merge them with the Q/K features.

Step A: Decompose the Mask:

They prove that the topological mask can be written as a dot product of "Graph Features":

$$M_{i,j} = \phi_G(v_i)^T \phi_G(v_j)$$

where $\phi_G(v_i)$ is a vector representing the "topological view" from node i.

Step B: The Tensor Trick (Equation 5)

Now we have two dot products happening:

1. **Content:** $\phi(q_i)^T \phi(k_j)$ (Are these tokens semantically related?)
2. **Topology:** $\phi_G(v_i)^T \phi_G(v_j)$ (Are these nodes topologically close in the graph?)

We want to multiply them:

$$Score_{i,j} = (\phi(q_i)^T \phi(k_j)) \cdot (\phi_G(v_i)^T \phi_G(v_j))$$

Using the property "The product of dot products is the dot product of outer products", we get:

$$Score_{i,j} = \text{vec}(\phi(q_i) \otimes \phi_G(v_i))^T \cdot \text{vec}(\phi(k_j) \otimes \phi_G(v_j))$$

- \otimes (Outer Product) creates a matrix combining semantic features and graph features.
- vec flattens the matrix into a long vector.

Why this solves the problem: We have now created a single new "super-feature" Φ_{new} . We can now go back to standard linear attention:

$$Att_{masked} = \Phi_{new}(Q)(\Phi_{new}(K)^T V)$$

We regained the associativity! We don't need the $N \times N$ matrix anymore.

4 Graph Random Features (GRF)

Paper Reference: Section 3.3, Equations (9) and (10) and `sample_random_walks`.

The paper defines the graph feature $\hat{\phi}_G(v_i)$ stochastically. Instead of calculating all paths, we let a "walker" randomly explore the graph starting from node i.

$$\hat{\phi}_G(v_i) = \frac{1}{n} \sum_{walks} (\dots)$$

- If a random walker starting at node i lands on node q, then the q-th entry of the vector $\hat{\phi}_G(v_i)$ gets a value.

- If the walker never visits node z, then the z-th entry is 0.

The Mathematical Interpretation: The dot product $\hat{\phi}_G(v_i)^T \hat{\phi}_G(v_j)$ measures the collision probability of random walks. Do walkers starting at i and walkers starting at j tend to visit the same nodes?

- If yes, dot product is high \rightarrow high Mask value \rightarrow strong attention.

5 Sparsity and Complexity (Theorem 3.2)

Paper Reference: Theorem 3.2 and Corollary 3.4

You might ask: "Wait, isn't the feature vector $\hat{\phi}_G$ size N ? If I have to carry around a vector of size N for every node, isn't that still $O(N^2)$ memory?"

The Math:

The authors prove (Theorem 3.2) that you don't need to fill the whole vector. Because random walks die out (with prob p_{halt}), the walker only visits a small, constant number of unique nodes (C).

Therefore, the vector $\hat{\phi}_G(v_i)$ is Sparse. It has N entries, but only C are non-zero (where $C \ll N$).

6 Presentation summary

The Wish: We want attention:

$$\text{Attention} = (QK^\top \odot \text{GraphTopology}) V.$$

The Barrier: We can't compute:

$$QK^\top \in \mathbb{R}^{N \times N} \text{ is too large, } \text{GraphTopology} \in \mathbb{R}^{N \times N} \text{ is also too large.}$$

The Trick (Factorization): Decompose standart attention:

$$QK^\top \rightarrow \phi(Q) \phi(K)^\top.$$

Decompose topology:

$$\text{GraphTopology} \rightarrow \phi_G \phi_G^\top \quad (\text{via Random Walks}).$$

The Merger: Combine them into one super feature:

Define the "Super Feature"

$$\Psi = \phi(Q) \otimes \phi_G.$$

Then attention becomes

$$\text{Attention} = \Psi(Q) \Psi(K)^\top V.$$

The Speedup:

Since random walks are short,

ϕ_G is sparse (mostly zeros).

Therefore the entire computation becomes

$O(N)$ instead of $O(N^2)$.