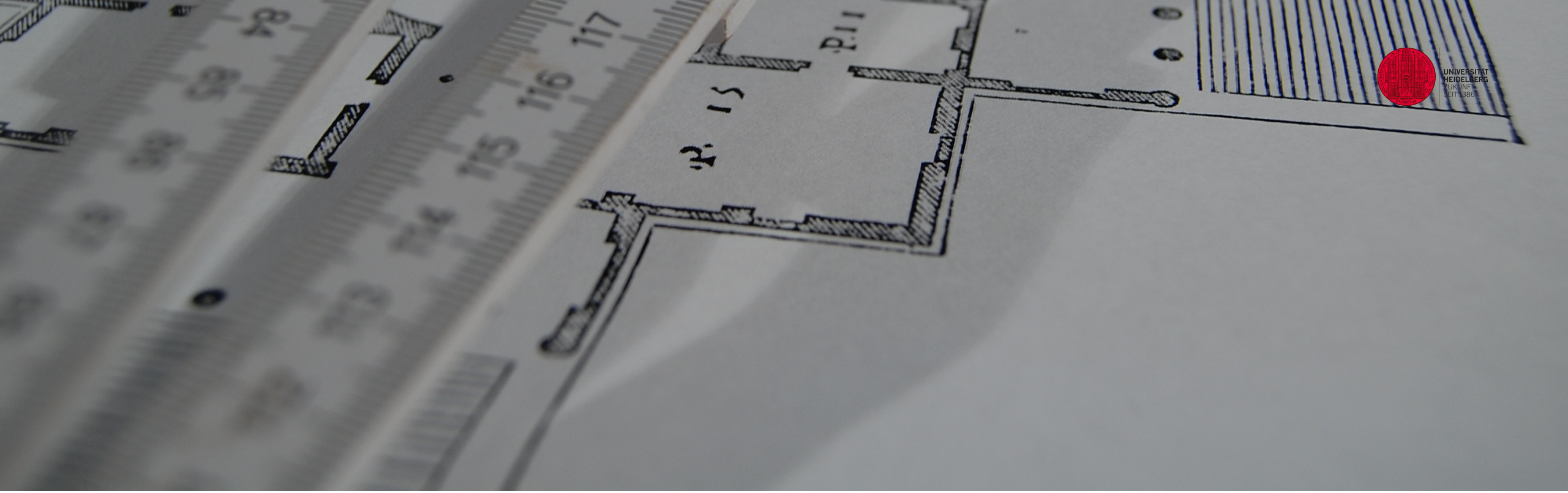


GOFA: A Generative One-For-All Model for Joint Graph Language Modeling

Presenter: Zhaokun Wang | July 7, 2025

Content

The Age of Foundation Models

- GPT-4, LLaMA, Claude... We've all heard the buzz.
- These models understand and generate natural language impressively.
- But what about non-text data? Specifically, **graphs**?

Example

How does ChatGPT handle a social network or a molecular structure?

Why Graphs Are Special

Key Difference

Text is linear. Graphs are not.

Graphs represent diverse structures:

- Social networks (e.g., who follows whom)
- Knowledge bases (e.g., "Paris is capital of France")
- Molecules, traffic networks, file systems...

Structure Matters

Who connects to whom? How? Think: Facebook's friend graph vs. Wikipedia articles.

Why It's Hard for LLMs

- LLMs need linear input (sequences of text).
- Graphs are *permutation invariant* (reordering nodes doesn't change the graph's meaning).
- Flattening graphs to text inevitably loses structural information.

Analogy

It's like trying to explain a complex subway map by just listing every station's name in one long sentence. The crucial connections are lost.

Motivation for a Graph Foundation Model (GFM)

To be useful, it must:

- Learn from vast amounts of **unlabeled** graphs (i.e., be self-supervised).
- **Generalize** across diverse downstream tasks (classification, question answering, generation...).
- Understand both node/edge **content** and network **structure**.

The Goal

Not just "what is this node?" but "what is its role in the network?"

Existing Approaches (and Their Limitations)

1. LLM as Predictor

- Flatten graph $\rightarrow$ feed to LLM.
- \+ Easy to generate text.
- \- Structure is weakly modeled.

2. LLM as Enhancer

- Use LLM to add features to GNNs.
- \+ Good structure modeling.
- \- Cannot handle diverse tasks.

Conclusion

Neither approach can fully serve as a true Graph Foundation Model.

Enter GOFA

Generative One-For-All

A New Architecture

A novel model that **interleaves** Graph Neural Networks (GNNs) directly into a Large Language Model (LLM).

- Learns to model graphs in a generative, self-supervised way.
- Maintains the LLM's flexibility and zero-shot capabilities.
- Adds the GNN's strong sense of structure and topology.

Analogy

It's like inserting a specialized subway map reader directly into a text-generating machine.

GOFA = GNN + LLM (at token level!)

Architectural Overview

- **Input**: Text-Attributed Graph (TAG)
- **GOFA Encoder**: LLM + interleaved GNN layers
- **GOFA Decoder**: LLM-based text generator

Key Idea

GNN layers inject structure awareness into the LLM's representations.

A Unified Framework

Core Idea

Treat all graph tasks as **graph completion**.

- Just as LLMs complete a sentence:
  The cat sat on the ___
- GOFA completes a graph:
  Given this graph of papers, a promising research direction is ___

TAG = Text-Attributed Graph

Each Node / Edge has text:

- **"Node A"**: Paper title, abstract
- **"Edge A-B"**: Co-citation relationship

$$\text{TAG} = \{\text{Nodes, Edges, Text at Nodes, Text at Edges}\}$$

Analogy

Like an annotated map with a narrative at every stop.

Text-Attributed Graph (TAG)

The **Text-Attributed Graph (TAG)** converts everything into natural language.

BEFORE (Chaos)

- Node Q76
- Node with feature $[0.9, 0.2]$
- Node user_123

AFTER (Clarity with TAG)

- “This is the entity for Barack Obama...”
- “This paper is about AI and has a citation count of 500.”
- “This is User 123. Their interests include hiking.”

Now, the model only needs to do one thing: **read**.

Node of Generation (NOG)

The **Node of Generation (NOG)** is a temporary node we add to the graph to guide the model.

1. Its text is the **user's prompt** ("Find the shortest path...").
2. It **connects** to the relevant nodes (A and D).
3. The model's job is to generate the answer **at the NOG**.

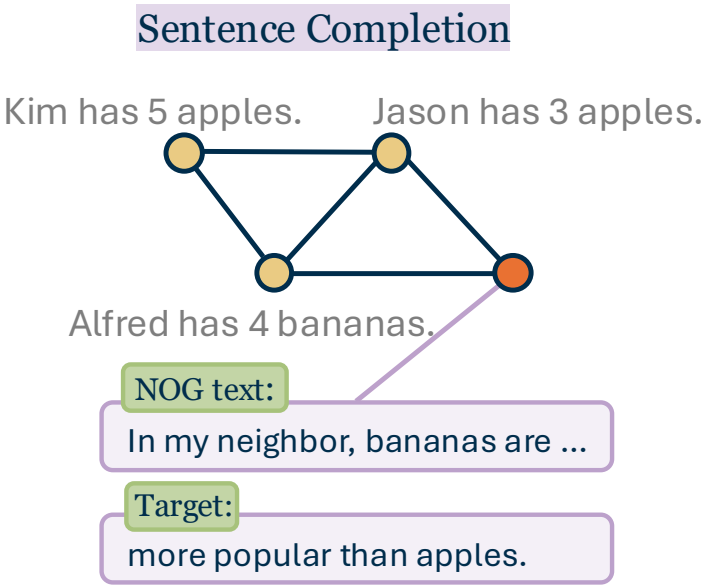


Figure 1: NOG Architecture.

An Example

1. **The Raw Graph**
   Nodes for “Nolan”, “Inception”, “Zimmer”.
2. **The TAG Version**
   Nodes now have text: “Director: Christopher Nolan”, “Composer: Hans Zimmer”, etc.
3. **The User’s Query**

   “What is the connection?”
4. **The NOG is Created & Connected**
   A new NOG node with the query text appears and links to “Nolan” and “Zimmer” nodes.
5. **The Model Generates the Answer at the NOG**

“Christopher Nolan worked with composer Hans Zimmer on the film Inception.”

In-Context Autoencoder (ICAE)

- **What It Is**: An autoencoder built with two LLMs (a compressor and a decoder) that compresses sentences into a fixed number of "memory tokens".
- **How It Works**: The decoder reconstructs the original text by only attending to these compressed memory tokens.
- **Purpose in GOFA**: To create dense, fixed-size text representations for graph nodes, which allows the GNN to perform message passing with minimal information loss.

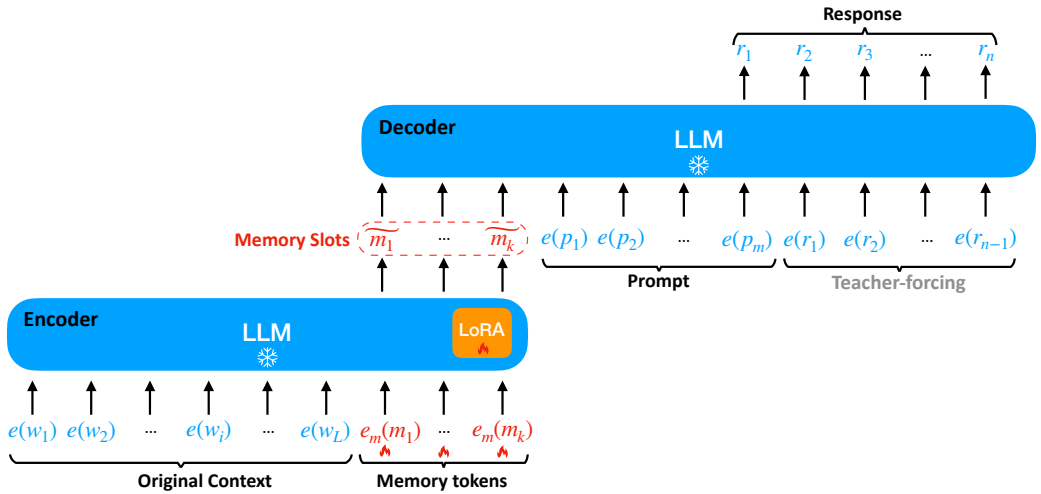


Figure 2: ICAE Architecture.

Transformer Convolutional GNN (TransConv)

- **Core Design**: GOFA uses a custom Transformer Convolutional GNN (TransConv), with its layers interleaved between the LLM's transformer layers.
- **Mechanism**: It functions like a standard Transformer attention layer, where a node's representation is updated from an attention-weighted sum of its neighbors' features.
- **Key Details**: The implementation is stabilized with pre-layer normalization and residual connections.

The GOFA Architecture

Key Design

A hybrid design that **interleaves** GNN and frozen LLM layers.

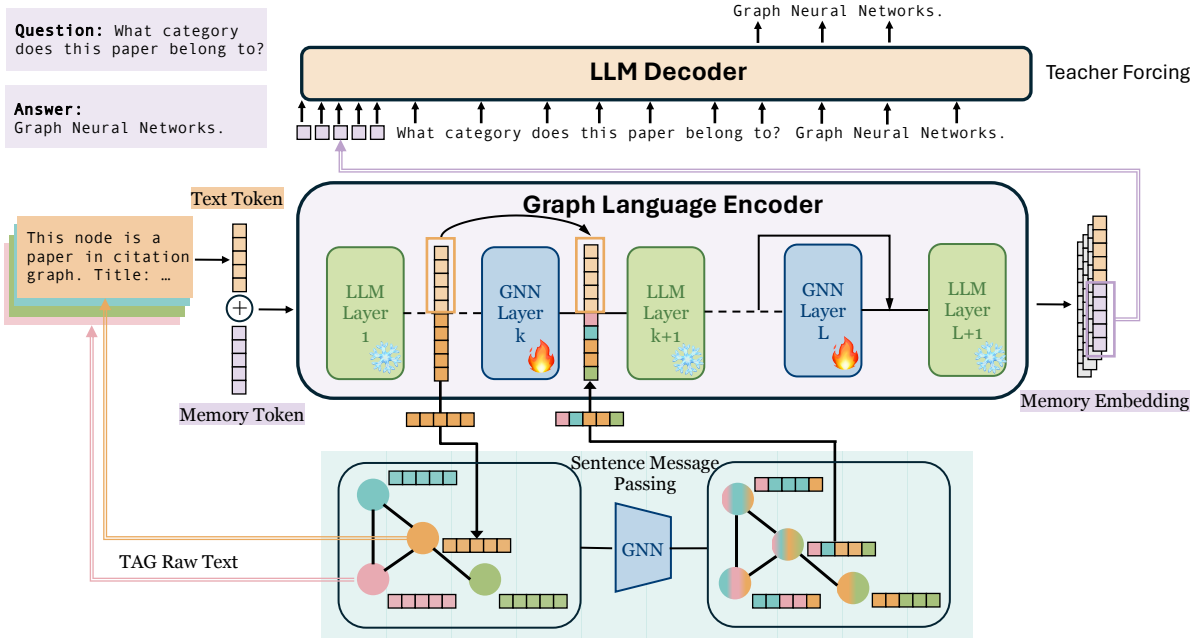


Figure 3: GOFA Architecture.

GOFA Encoder = GNN + LLM (Compressor)

How to encode a TAG?

- **Sentence compressor (ICAE)**: converts node text $\rightarrow$ token-level embeddings.
- **GNN layers**: exchange messages between token embeddings, not just whole nodes.
- **LLM layers**: perform attention across tokens within a single node.

Information Flow

- Tokens within a node attend to each other
- Tokens from neighboring nodes message-pass via GNN

Why Token-Level Message Passing?

Naive GNN: Treats node as one big vector

Pooling loses sentence detail.

GOFA's Token-GNN: Passes messages between token positions

- Preserves word-level meaning.
- Enables finer-grained structure reasoning.

Like passing phrases, not just labels, between neighbors.

Step 1: Smart Compression with Memory Tokens

Solution: Distill, Don't Discard

GOFA uses a pre-trained sentence compressor (ICAE) to distill a sentence into a fixed set of *K* **Memory Tokens**.

- These are not a single summary; they are *rich, multi-faceted "bullet points"* that preserve the original information.

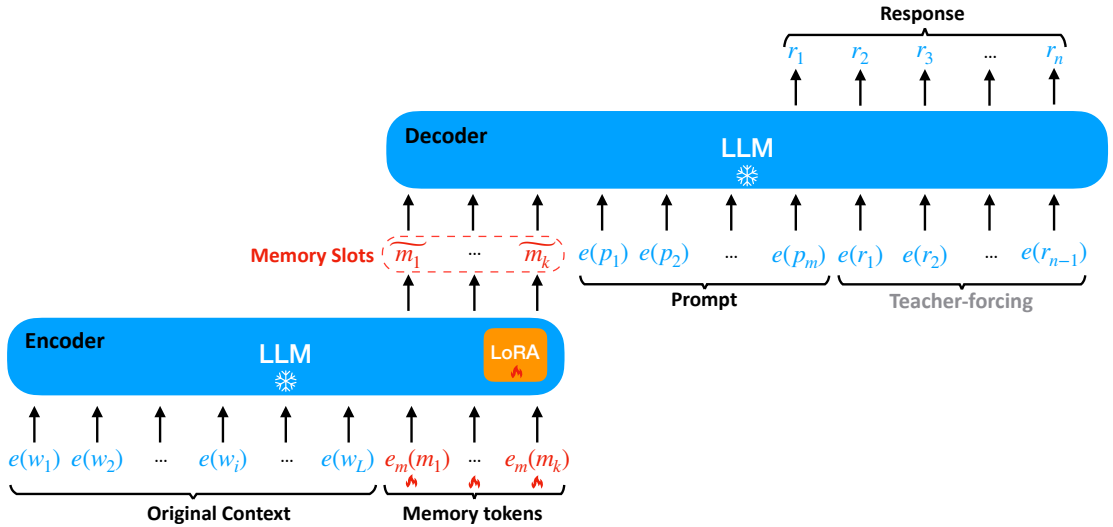


Figure 4: ICAE Architecture.

Step 2: Semantic Processing

The Memory Tokens for each node first pass through a standard **LLM (Transformer) layer**.

- At this point, the model processes each node's text **in isolation**.
- The self-attention mechanism deepens semantic understanding of the node's own content.

Goal

Understand what the node is about *before* considering its connections.

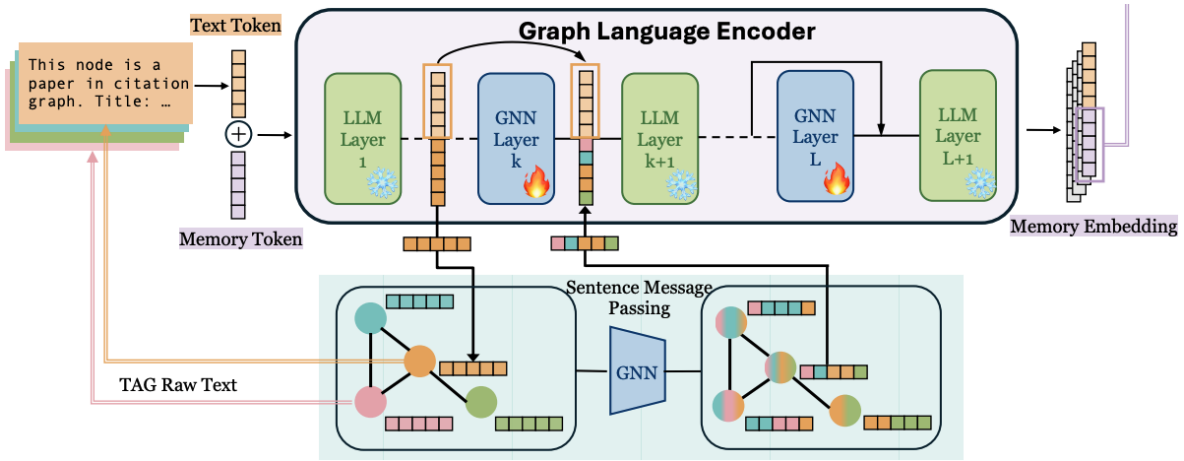


Figure 5: The Encoder.

Step 3: The GNN Layer

The output immediately flows into a **GNN Layer** which performs message passing at the **token level**.

- The k-th memory token of a node only receives messages from the k-th memory token of its neighbors.

Analogy: Parallel Phone Calls

Line 1 of Node A only talks to Line 1 of its neighbors. Line 2 only talks to Line 2, and so on. No crosstalk... yet.

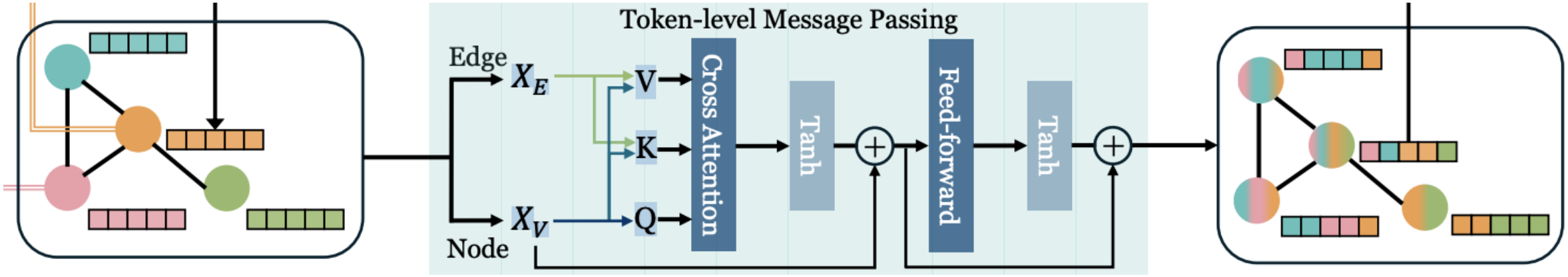


Figure 6: The GNN Layer.

Step 4: The Fusion

The now structurally-aware tokens flow into the **next LLM layer**.

- The LLM's powerful self-attention mechanism now fuses everything.
- The 1st token (which heard from its neighbors) can now interact with the 2nd, 3rd, and all other tokens of its own node.

Analogy: The Conference Room

After the separate phone calls (GNN), everyone gets into a conference room (the LLM layer) to share notes and build a complete, unified picture.

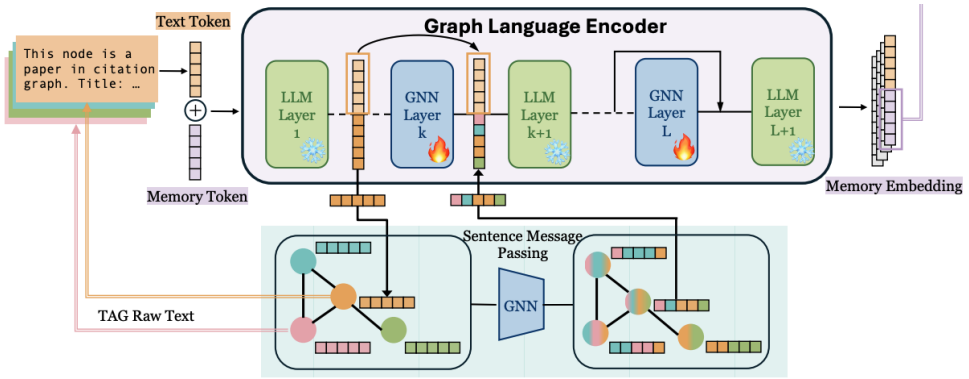


Figure 7: The Encoder.

Decoding the Answer

1. The LLM $\rightarrow$ GNN cycle repeats, progressively deepening the fused representation.
2. A tanh gate on the GNN output acts as a safety switch, allowing the model to ignore graph structure if it's not useful.
3. Finally, the enriched memory tokens from the **NOG** (the query node) are fed to the LLM Decoder.
4. The Decoder, having received a perfectly briefed and context-rich summary, generates the final answer.

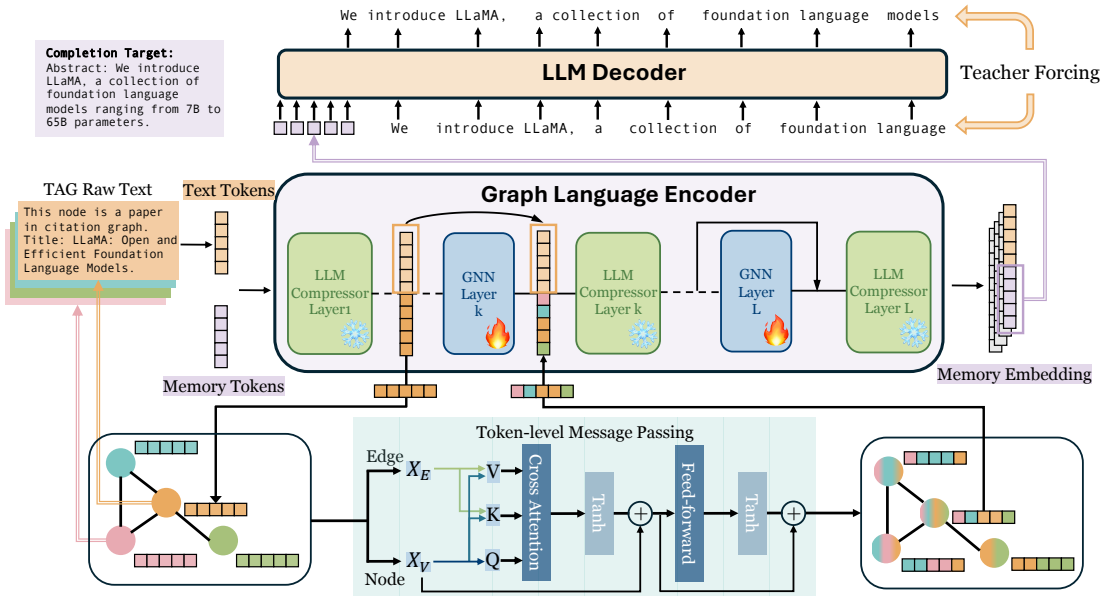


Figure 8: GOFA Architecture.

Summary of the Architecture

Key Takeaways

- GOFA is modular: can plug into any decoder-only LLM.
- GNN layers are interleaved for structural awareness.
- Handles any graph task via a generative interface.
- A truly graph-aware, instruction-following LLM.

Experimental Overview

We will answer four fundamental questions:

- **Q1: Effectiveness** – Does the training work?
- **Q2: Generalization** – Can it handle unseen tasks?
- **Q3: Efficiency** – Is it better than just using an LLM?
- **Q4: Fluency** – Is it a truly flexible reasoner?

The Setup

Core Method

- All tasks are converted into subgraph problems, using k-hop subgraphs around target nodes as model input.

Pre-training

- **Objective**: GOFA is pre-trained using four self-supervised tasks.
- **Data**: Training utilizes large-scale, diverse graph datasets like MAG240M, Arxiv, Pubmed, and Wikikg90m.
- **Setup**: Initially, only the GNN layers are tuned. Training was performed on 8 NVIDIA A100 GPUs.

Fine-tuning

- **Zero-Shot**: The model is instruction-tuned on a small set of tasks (e.g., from Arxiv/Pubmed) to learn task formats. Prompts for this stage include detailed instructions and options.
- **Supervised**: The model is simultaneously fine-tuned on the training sets of all evaluation datasets, with prompts designed for direct answer generation.

The Setup

Datasets

- **Citation networks**: Cora, Arxiv
- **Knowledge graphs**: FB15K237
- **Product graph**: Amazon
- **Commonsense QA**: ExplaGraphs

Baselines

- **LLMs**: LLaMA2, Mistral
- **Graph LLMs**: GraphGPT, UniGraph, OFA

4 Types of Pretraining Tasks

A powerful architecture needs powerful training. GOFA is pre-trained on four novel tasks designed to build the three key properties of a GFM.

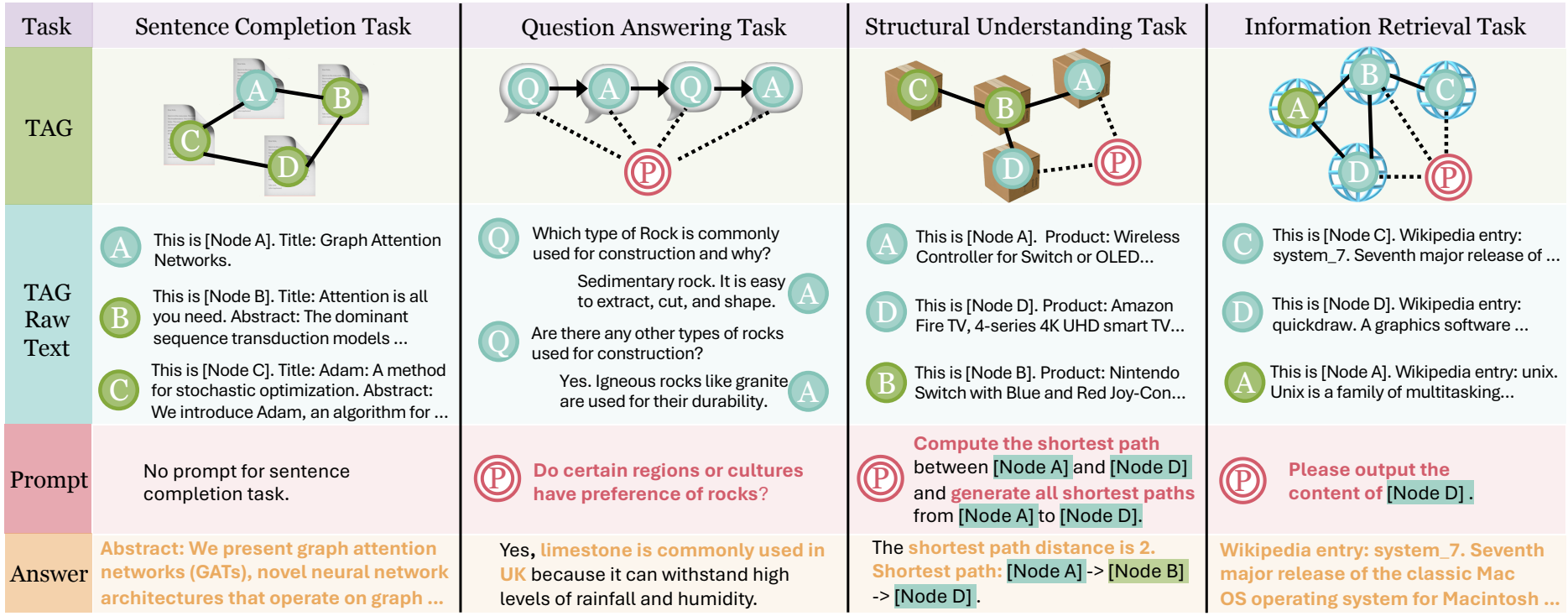


Figure 9: Examples of our pre-training tasks.

Q1: Does the training work?

Setup

- **Perplexity** ↓: How "surprised" the model is by new text. Lower is better.
- **SPD & CN RMSE** ↓: Error in predicting graph structure. Lower is better.
- **Test**: Evaluate on unseen datasets vs. base LLM.

Results

- **Language**: GOFA perplexity is much lower (21.3 vs 30.1).
- **Structure**: GOFA error (RMSE) is significantly lower.

Table 1: Evaluation for pre-trained GOFA.

	Perplexity ↓	SPD ↓	CN ↓
Mistral-7B	30.12	1.254	1.035
GOFA-SN	26.20	-	-
GOFA	21.34	0.634	0.326

Q2: Can it handle unseen tasks?

Zero-Shot Classification Results

Setup: Zero-Shot Evaluation

- **Process**: Instruction-tuned on a small set of tasks.
- **Test**: Evaluated on completely new, unseen datasets.
- **Baselines**: Compared against SOTA Graph-LLMs (UniGraph, LLaGA).

Table 2: Node Classification Task Results (Accuracy).

Task	Cora-Node		WikiCS		Products			ExplaGraphs	Cora-Link
Way	7	2	10	5	47	10	5	2	2
LLama2-7B	47.92	73.45	40.10	58.77	27.65	58.71	64.33	57.76	48.15
Mistral-7B	60.54	88.39	63.63	71.90	43.99	70.16	74.94	68.77	49.43
OFA-Llama2	28.65	56.92	21.20	35.15	19.37	30.43	39.31	51.36	52.22
GraphGPT	44.65	-	-	-	18.84	-	-	-	50.74
UniGraph	69.53	**89.74**	43.45	60.23	38.45	66.07	75.73	-	-
ZeroG	64.21	87.83	31.26	48.25	31.24	51.24	71.29	-	-
LLaGA	51.85	62.73	-	-	23.10	34.15	39.72	-	**88.09**
GOFA-T	**70.81**	85.73	**71.17**	**80.93**	54.60	79.33	87.13	**79.49**	85.10
GOFA-F	69.41	87.52	68.84	80.62	**56.13**	**80.03**	**88.34**	71.34	86.31

Q2: Can it handle unseen tasks?

Zero-Shot Classification Results

Setup: Zero-Shot Evaluation

- **Process**: Instruction-tuned on a small set of tasks.
- **Test**: Evaluated on completely new, unseen datasets.
- **Baselines**: Compared against SOTA Graph-LLMs (UniGraph, LLaGA).

Table 3: Link Classification Task Results (Accuracy).

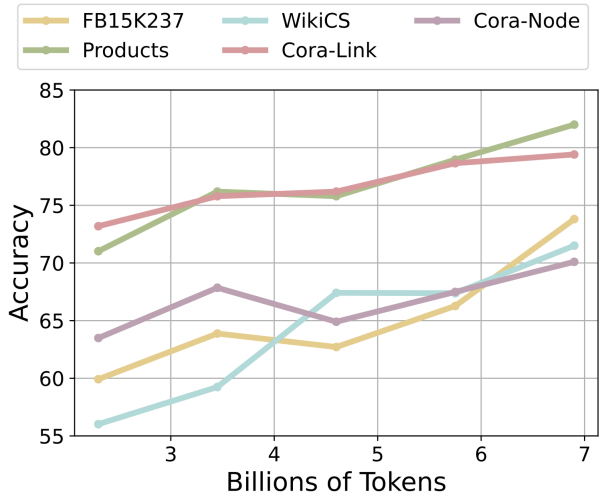
Task Format	FB15K237 10-Way	SceneGraphs QA
Llama2-7B	48.32	38.62
Mistral-7B	62.48	**45.95**
GOFA-T	73.59	34.06
GOFA-F	**80.69**	31.36

Intuition Behind Zero-Shot Ability

1. Scaling Works

More pre-training data leads to better performance. This is a hallmark of a successful foundation model.

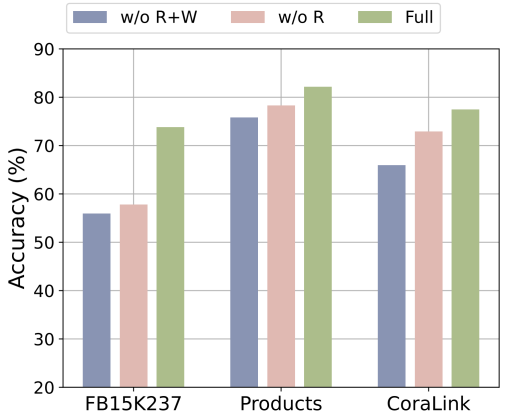
Figure 10: Performance vs. pre-training sample size.



2. Every Pre-training Task Matters

An ablation study shows that removing specific pre-training tasks hurts downstream performance. Removing the **Information Retrieval** task is particularly harmful for KGs.

Figure 11: Ablation study results.



Q3: Is it better than just using an LLM?

The Experiment

- **GOFA:** Gets the graph as input.
- **LLM-N:** Gets all node texts *plus* textual descriptions of every connection (e.g., "Node A connects to Node B").

Table 4: Comparison between GOFA and LLM with the same input.

Task Metric	ExplaGraphs Acc ↑	Time sec/sample ↓	WikiCS Acc ↑	Time sec/sample ↓	Cora-Link Acc ↑	Time sec/sample ↓	FB15k237 Acc ↑	Time sec/sample ↓
LLM-N	74.13	1.50	OOM	OOM	50.36	3.84	51.25	3.92
GOFA-F	79.49	0.48	71.17	2.43	85.10	1.67	73.49	3.37
Improvement	7.23%	68.00%	NA	NA	68.98%	56.51%	43.40%	14.03%

Accuracy

GOFA is far more accurate on all tasks (e.g., 69% improvement).

Efficiency

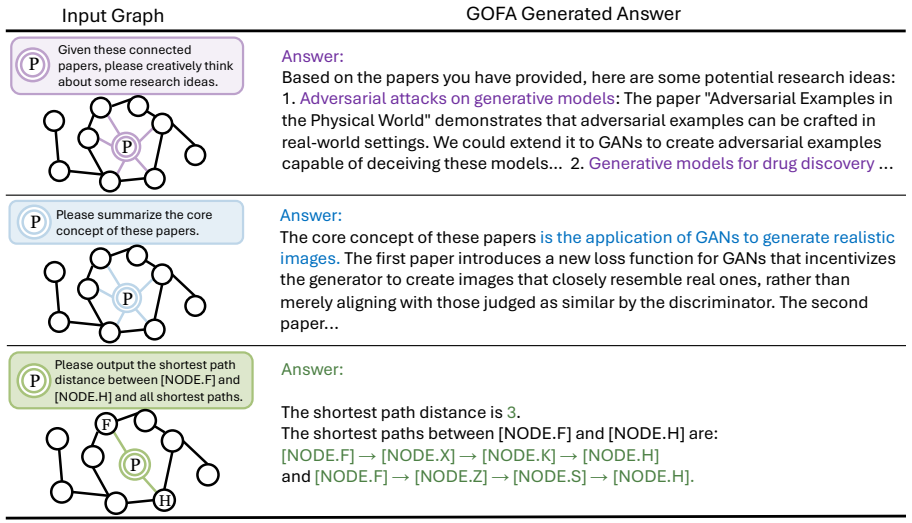
GOFA is much faster and avoids "Out of Memory" errors on large graphs.

Q4: Can GOFA "Think" Flexibly?

Setup

The same graph is given to GOFA with three different open-ended prompts.

Figure 12: Comparison between GOFA and LLM with the same input.



Summary of Findings

- GOFA matches LLMs in text tasks
- Outperforms on graph-related tasks
- Generalizes to unseen datasets and instructions

One model. All tasks. Structure + Language.

The Bigger Picture: Towards Unified AI Reasoning

What GOFA enables:

- **Complex structural QA**: “List all paths from X to Y"
- **Conceptual synthesis**: “Summarize the main idea of a citation cluster"
- **Graph querying via prompts**

The Pieces Coming Together:

Text + Graph + Instruction = Universal Reasoning Interface

Limitations

- **Frozen LLM Compressor**
  The compressor can't adapt to the specific semantics of the graph data.
- **Format Dependence**
  Instruction tuning assumes known task templates; completely new formats are still a challenge.
- **No Truly End-to-End Training**
  The encoder and decoder components are still trained or utilized in separate stages.

Looking Ahead: What Comes Next?

Unified Training

Train the compressor, GNN, and decoder components end-to-end for deeper integration.

Universal Graph Agents

Combine GOFA with retrieval systems or external tools to act on real-world knowledge graphs dynamically.

Cross-Modal Graph Reasoning

Extend the architecture to reason over graphs where nodes are images, audio, or other complex data types (e.g., visual scene graphs).

A Thought to Leave With...

“Graphs are how the world connects. Language is how we explain it. What if a model could do both?”

GOFA is a bold step. But it’s only the beginning.

References

[1] L. Kong, J. Feng, H. Liu, C. Huang, J. Huang, Y. Chen, M. Zhang (2024).
Gofa: A Generative One-for-All Model for Joint Graph Language Modeling.
arXiv:2407.09709.

[2] Z. Wang, Z. Liu, T. Ma, J. Li, Z. Zhang, X. Fu, Y. Li, Z. Yuan, W. Song, Y. Ma, et al. (2025).
Graph Foundation Models: A Comprehensive Survey.
arXiv:2505.15116.

[3] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, Y. Sun (2021).
Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification.
arXiv:2009.03509.

[4] T. Ge, J. Hu, L. Wang, X. Wang, S.-Q. Chen, F. Wei (2024).
In-context Autoencoder for Context Compression in a Large Language Model.
arXiv:2307.06945.

Q&A

Thank You! Questions?