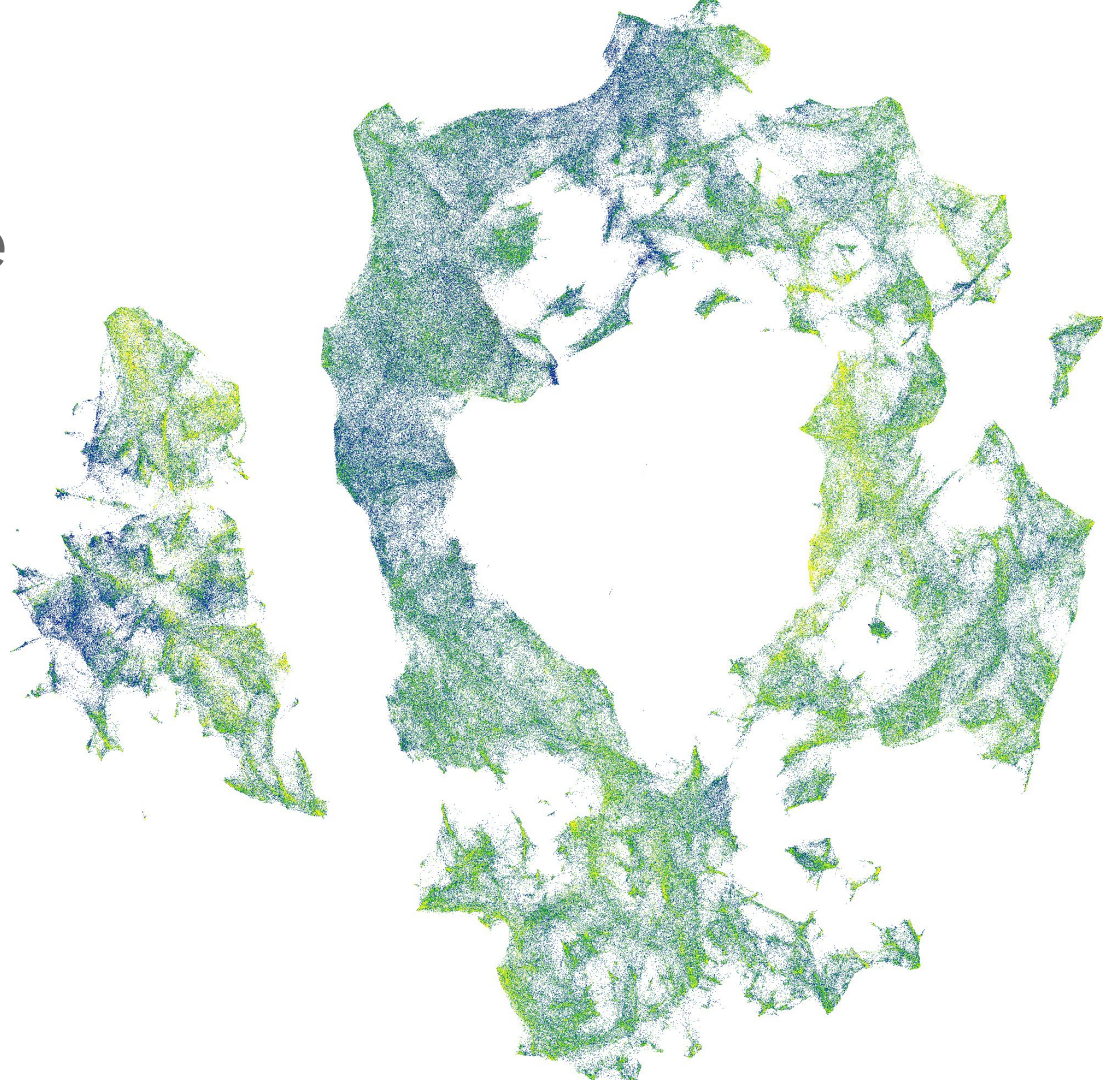
Graph Reasoning in Large Language Models



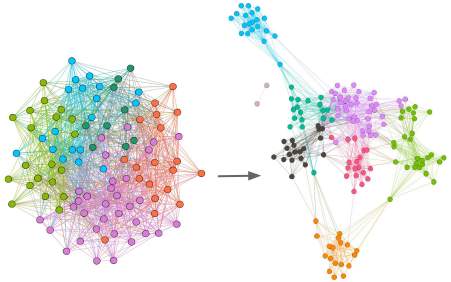
Bryan Perozzi, Clayton Sanford, Jonathan Halcrow
Google Research

NeurIPS'24
Vancouver, CA
4 p.m. - 5 p.m. PST
West Meeting Room 220-222
12/10/24

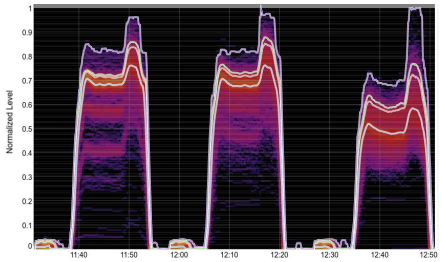
Graph Mining @ Google

Our group focuses on using the structure of large scale data. We are often dealing with billions of nodes and many more edges. To work with data at this scale, we have to combine algorithmic ideas with the right systems and ML models.

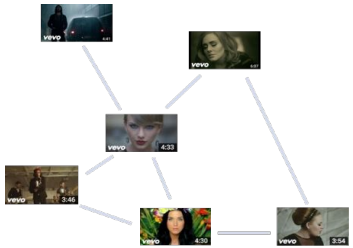
These tools power hundreds of projects at Google in Search, Ads, Youtube, Play, Cloud, Maps, Payments, and more.



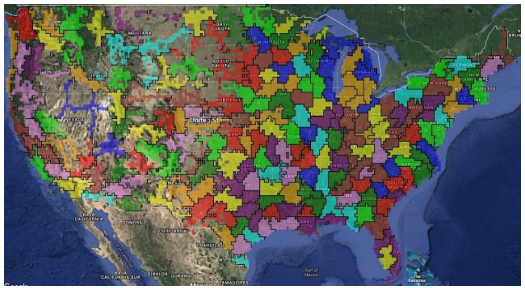
Same-meaning queries for Keyword matching systems



Better Caching for saving 32% Flash I/O for Search Infra(VLDB'19).



Collaborative Filtering for YouTube Recommendations



Finding micro-markets in designing A/B experiments [KDD'19, NeurIPS'19]

Motivation

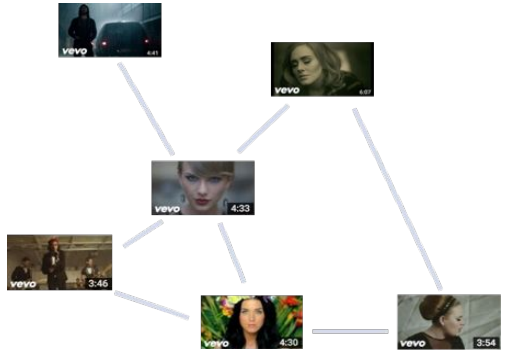
We have a powerful new tool (GenAI models), that allows us incredible new capabilities in language, vision, video, and more. However, there are weaknesses stopping us from applying this tool everywhere.

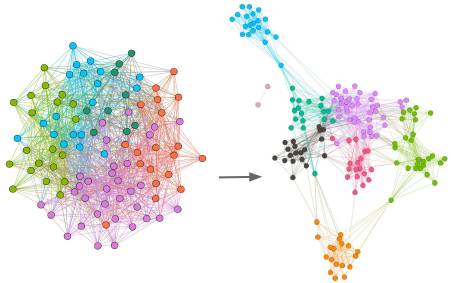
- LLMs make stuff up (***hallucinations***)
- LLMs have stale pretraining (***freshness***)
- There's a desire for *private* LLMs that can use personal data without leaking it (***privacy***)
- Training and inference is expensive (***cost***)

How can Graphs help?

Graphs are tools for modeling the fundamental structure of data. They allow us to

- ***Represent knowledge*** in human-accessible formats (knowledge graphs, relational databases, etc)
  - Grounding
  - Structured Data in prompt
  - Private information in prompt
- Use sparsity for efficient algorithms and hierarchical decompositions (***efficiency***)
  - Graph models vs Transformers
    - Graphs: more parameter efficient
    - Graphs: higher sample efficiency





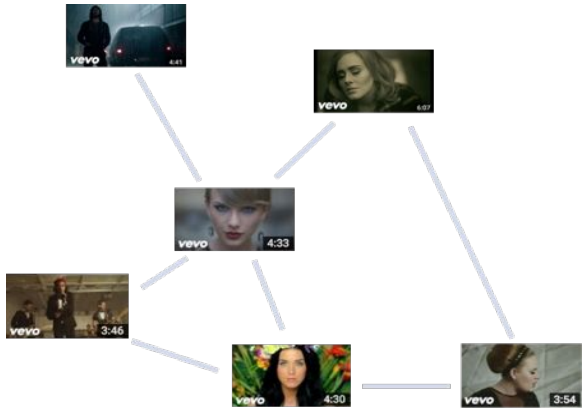
Part I

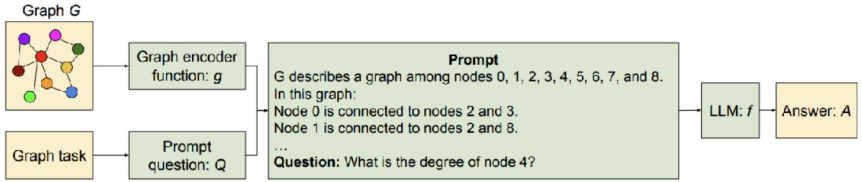
Foundations of LLM Reasoning
via Graph Algorithms

Graph-powered LLM insights

1. Graphs as LLM benchmarks:
Graph algorithmic tasks (e.g. connectivity, shortest path) are **abstractions** of core LLM functionality.

GraphQA benchmark:
Dataset with 10+ graph reasoning tasks applied to randomly generated graphs.





Graph-powered LLM insights

1. Graphs as LLM benchmarks

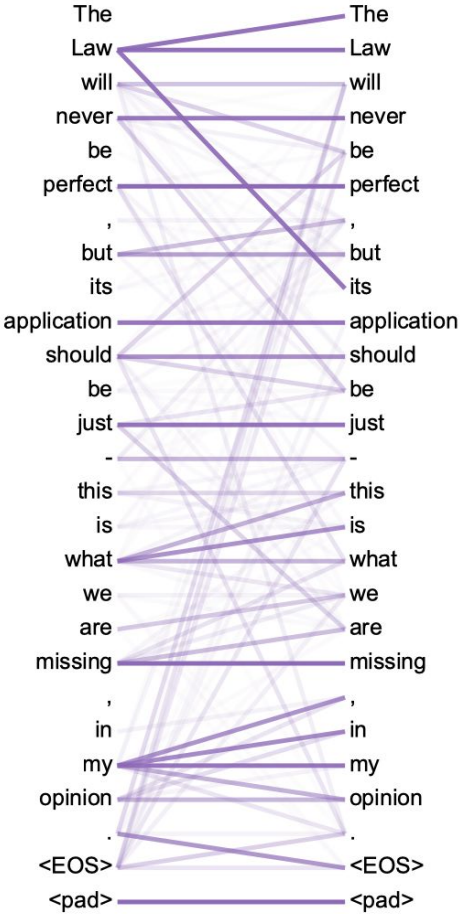
2. Graphs as interpretability for transformers:

Core primitive of transformers
= associative **self-attention unit**.

Each self-attention output depends on **attention matrix** $A = \text{softmax}(XQK^TX^T)$, an N x N matrix.

- $A_{i,j}$ is large → value Vx_j "shared" with token i.

Self-attention matrices are often interpretable!



Vaswani, et al, '17

Graph-powered LLM insights

1. **Graphs as LLM benchmarks**
2. **Graphs as interpretability for transformers:**
3. **Graphs for architectural and embedding tradeoffs:**

What's the right way to input a graph instance to a neural network?

- Use a GNN (input determines NN topology/connections)?
- Augmentation of transformer and GNN structures?
- Pure transformers with abstract graph representations?
- Pretrained LLM with textual encoding of the graph?

This work

1. **Graphs as LLM benchmarks**
2. **Graphs as interpretability for transformers:**
3. **Graphs for architectural and embedding tradeoffs:**

Sharp trade-offs between transformers and GNNs on graph reasoning benchmarks.

Empirics: wide-ranging experiments on GraphQA benchmark.

- GNNs win on “local” problems, transformers win on “global.”

Theory: complexity hierarchy of graph reasoning tasks for transformers.

- Follows analysis of transformers as message-passers.

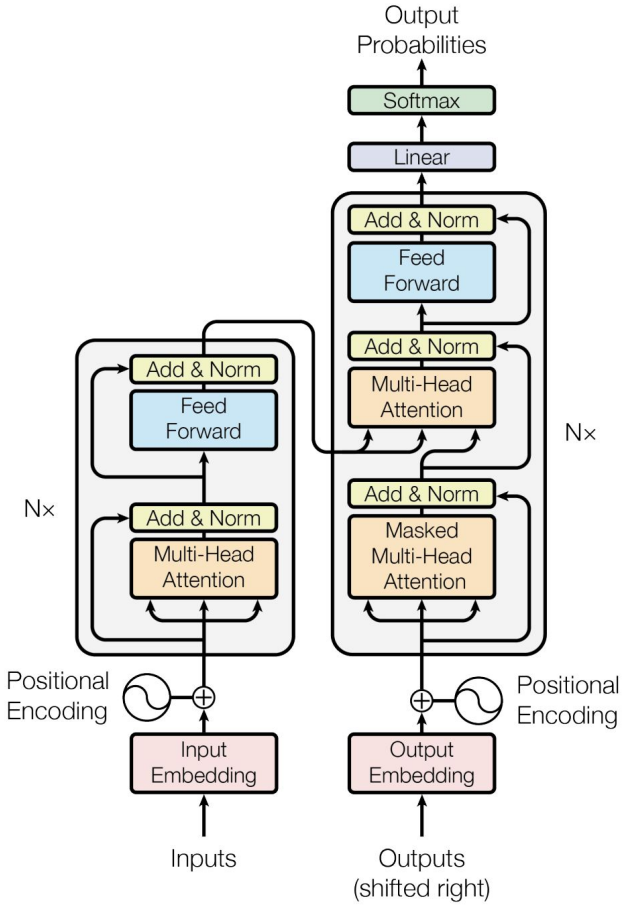
Transformers

Attention head:
$f(X) = \text{softmax}(XQK^TX^T)XV$
$= A\,XV, \qquad \text{for } 0 \le A_{i,j} \le 1.$

A is often sparse $\rightarrow$
$f(X)_i \approx \text{agg}(\{Vx_j : A_{i,j} >> 0\})$.

Transformer model:
Interleaved multi-headed attention and element-wide multi-layer perceptrons.

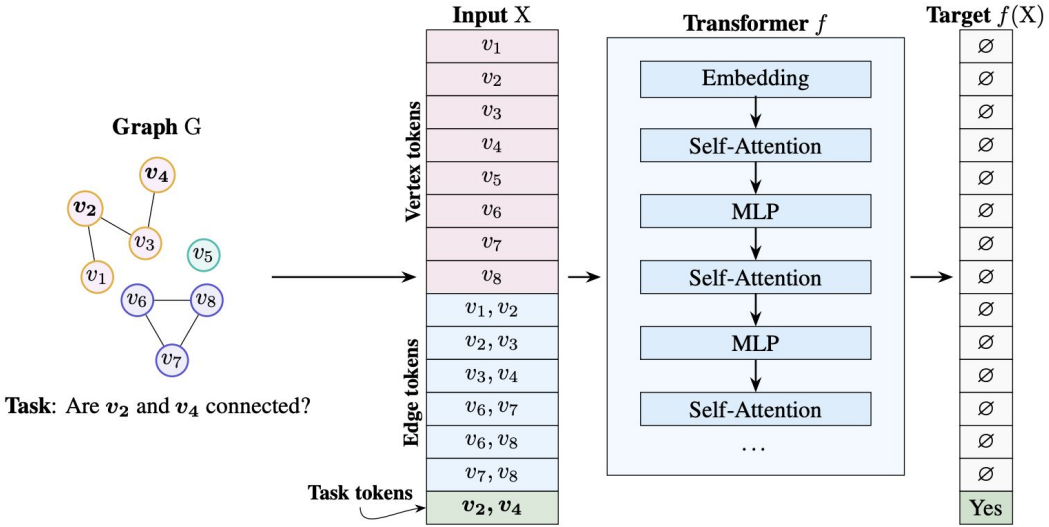
Adaptive communication:
"Everyone talks to everyone"



Vaswani, et al, '17

“Vanilla graph encoding”
Importing graph algorithm problems to transformers

- **GraphQA problems**: connectivity, shortest path, node count, edge count, edge existence, node degree, triangle count, cycle check.
- **“Pure transformer”** encoding of [Kim, et al, ‘22]: one token per node/edge.
- No further model augmentation (no graph embeddings, attention masking, etc.)



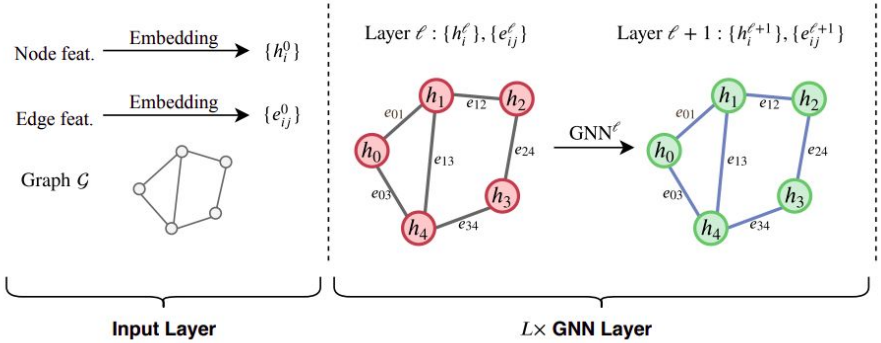
Message-passing neural networks (MPNNs)

Input graph: *G = (V, E)*

Message-passing:
$f(v_i) = \text{agg}(\{\Phi(v_j): (v_i, v_j) \in E\}$

Full model:
Multiple layers of message-passing, with MLPs Φ.

Fixed communication:
“Everyone talks to their neighbors”



Dwivedi, et al, ‘20

Transformers

Attention head:
$f(X) = \text{softmax}(XQK^TX^T)XV$
$= A\,XV, \qquad \text{for } 0 \le A_{i,j} \le 1.$

A is often sparse $\longrightarrow$
$f(X)_i \approx \text{agg}(\{Vx_j : A_{i,j} >> 0\})$.

Transformer model:

Interleaved multi-headed attention and element-wide multi-layer perceptrons.

Adaptive communication:
"Everyone talks to everyone"

MPNNs

Input graph: $G = (V, E)$

Message-passing:

$f(v_i) = \text{agg}(\{\Phi(v_j) : (v_i, v_j) \in E\})$

Full model:

Multiple layers of message-passing, with MLPs Φ.

Fixed communication:
"Everyone talks to their neighbors"

Similar high-level message-passing idea!

Architectural trade-offs: the big idea

Transformer message-passing:

$f(X) = \text{softmax}(XQK^TX^T)XV$

$= A\,XV, \qquad \text{for } 0 \le A_{i,j} \le 1.$

A is often sparse →

$f(X)_i \approx \text{agg}(\{Vx_j : A_{i,j} >> 0\})$.

MPNN message-passing:

$f(v_i) = \text{agg}(\{\Phi(v_j) : (v_i, v_j) \in E\})$

Key modeling difference:
GNNs are restricted to communication between neighbors.

Advantage: Positive **inductive biases** for learning relationships between neighbors → more sample-efficient!

Disadvantage: Cannot infer long-range affinities or perform complex aggregation.

Result 1: GNNs succeed at “intrinsically local” tasks

Experimental setup:
Trained transformers and GNNs on GraphQA tasks with 1K and 100K samples.

Benefits of inductive bias:
For both sample complexities, MPNNs output 60M-parameter transformers are **node degree** and **cycle check** problems.

	Node Degree		Cycle Check	
Model	**1K**	**100K**	**1K**	**100K**
GCN [42]	9.8	9.4	83.2	83.2
MPNN [26]	99.4	**99.8**	99.0	**100.0**
GIN [82]	36.2	37.8	98.8	83.2
60M transformer	31.6	91.7	97.1	98.0

GNN winning on local tasks

Result 2: Transformers succeed at “global” tasks

Experimental setup:
Trained transformers and GNNs on GraphQA tasks with 1K and 100K samples.

Sample-efficiency of GNNs:
In 1K sample regime, MPNNs outperform transformers.

Fundamental GNN limitations:
But, GNN capacity restrictions prevent it from doing as well with many samples.

	# of training samples	
Model	**1K**	**100K**
GCN [42]	50.2	55.0
MPNN [26]	66.8	72.6
GIN [82]	54.0	58.6
60M transformer	57.4	**97.1**

Connectivity results

Result 3: Trained transformers vs LLM prompting

Comparison of training modes:

- Small transformers (60M parameters) specifically trained to solve GraphQA tasks.
- Pretrained PaLM models (xxB parameters) with zero-shot, few-shot, or chain-of-thought prompting.

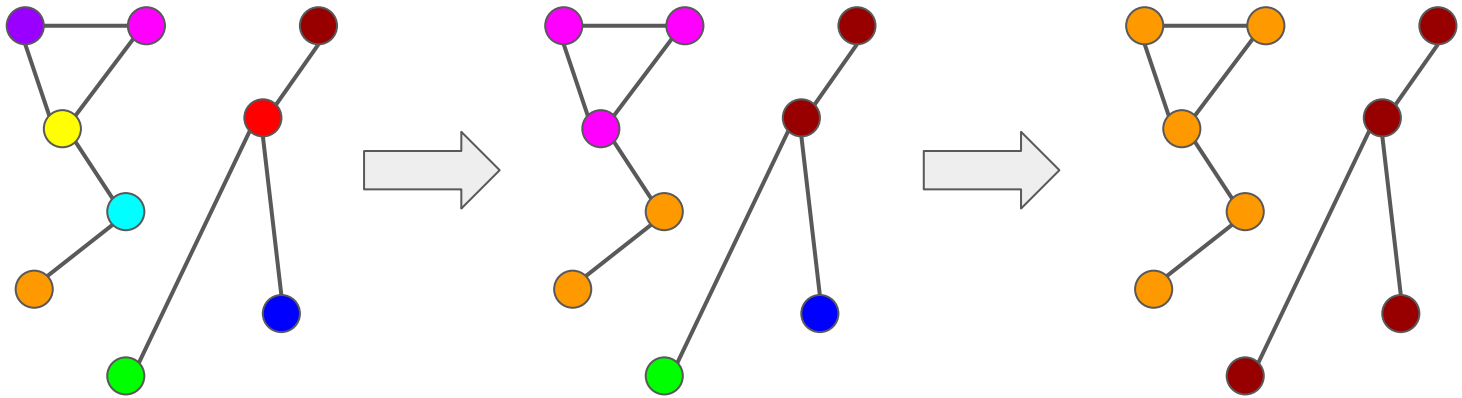
Widespread dominance of small models:

		Retrieval tasks				Parallelizable Tasks		Search Tasks	Subgraph Counting
	Method	**Node count**	**Edge count**	**Edge existence**	**Node degree**	**Connectivity**	**Cycle check**	**Shortest path**	**Triangle counting**
Prompting	ZERO-SHOT [22]	21.7	12.4	44.5	14.0	84.9	76.0	11.5	1.5
	ZERO-COT [22]	14.6	9.4	33.5	10.4	73.5	32.3	33.6	12.7
	FEW-SHOT [22]	25.3	12.0	36.8	17.4	79.4	37.4	22.7	3.0
	COT [22]	27.6	12.8	42.8	29.2	45.2	58.0	38.6	8.1
	COT-BAG [22]	26.9	12.5	37.3	28.0	45.2	52.1	40.4	8.1
Ours	60M transformer-1K	100.0	100.0	67.6	31.5	92.9	97.1	57.4	33.4
	60M transformer-100K	**100.0**	**100.0**	96.1	**91.7**	98.0	**98.0**	97.2	**40.5**
	11B transformer (FT)-1K	**100.0**	45.0	**100.0**	68.8	**98.4**	**98.0**	**92.8**	26.0

Intuition: Parallel algorithms for graph connectivity

Why are transformers better at connectivity?

Long-range communication makes it possible to simulate parallel algorithms.



log(*N*) rounds of parallel computation are enough…
but requires long-range communication between cluster “leaders”

Graph connectivity theory

Positive transformer result [Sanford, Hsu, Telgarsky ‘24]:
A transformer of depth $D = O(\log N)$ and width $m = O(N^{0.01})$ solves graph connectivity on N-node graphs.

Negative GNN result 1 [Xu, et al ‘18]:
Featureless MPNNs **cannot** distinguish between connected and disconnected graphs. (Weisfeiler-Lehman graph isomorphism test.)

Negative GNN result 2 [Loukas ‘19]:
Any MPNN **cannot** distinguish between connected and disconnected graphs, unless $D\, m^{1/2} > N^{1/2}$. (CONGEST distributed computing reduction.)

Theory beyond connectivity

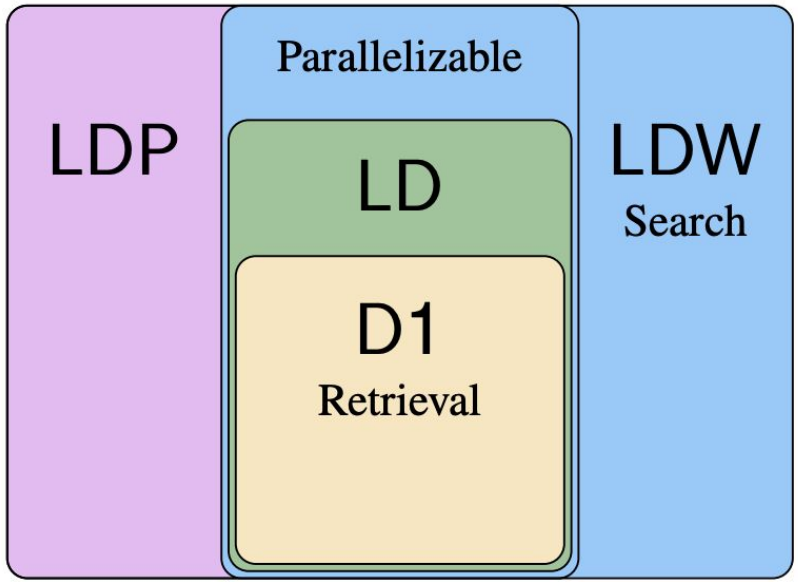
Which tasks are advantageous for transformers (e.g. connectivity), and which are not (e.g. node degree)?

Novel task complexity hierarchy by transformer size:

- **Retrieval tasks**: simple look-up or aggregations, only require one pass over the graph.
  - Node count, edge count, node degree, node existence.
- **Parallelizable tasks**: benefit greatly from parallel algorithms.
  - Connectivity, cycle check, minimum spanning forest, bipartiteness, planarity…
- **Search tasks**: more difficult to parallelize with few rounds.
  - Shortest path, diameter, (directed) reachability.

Theory beyond connectivity

Task class	Example tasks	Complexity
Retrieval (§3.3) $L = 1$ $m = O(\log N)$	Node count Edge count Edge existence Node degree	D1 D1 D1 D1
Parallelizable (§3.1) $L = O(\log N)$ $m = O(N^{\epsilon})$	Connectivity Cycle check Bipartiteness	LD LDP∩ LDW LDP∩ LDW
Search (§3.2) $L = O(\log N)$ $m = O(N^{1/2+\epsilon})$	Shortest path Diameter	LDW LDW

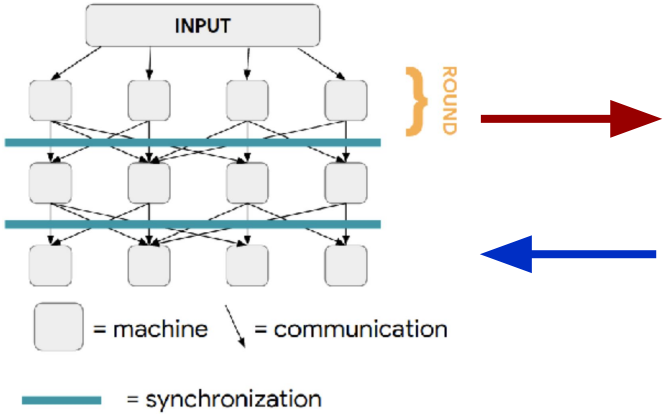


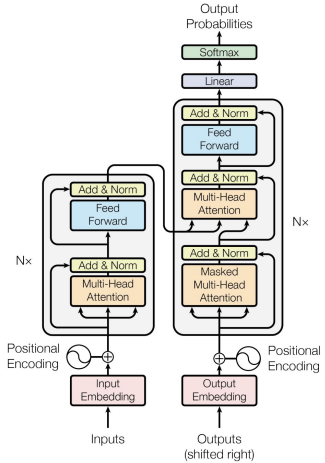
Main proof ingredients

1. Transformers + massively parallel computation (MPC) equivalence.

Any *R*-round MPC protocol with *s* local memory can be simulated by a transformer with depth *O(R)* and width $s^{1.01}$.

Any transformer depth *D* and width *m* can be simulated by an MPC protocol with *O(D)* rounds and local memory *O(m)*.





Vaswani, et al, ‘17

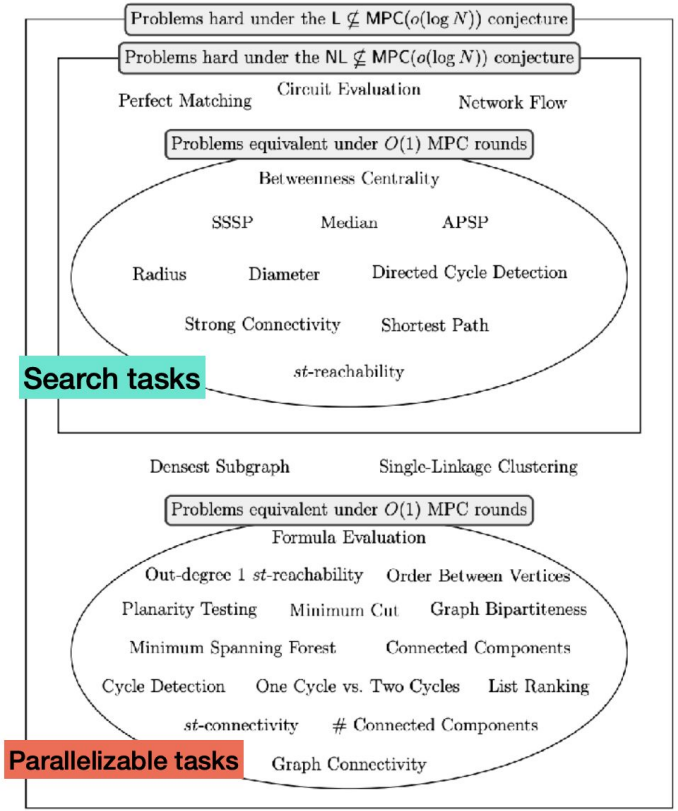
Main proof ingredients

1. Transformers + MPC equivalence.

2. Parallel complexity classes for graph algorithms. [Nanongkai, Scquizzato '22]

If some parallelizable task can be solved in *R* rounds of parallel computation, then any parallelizable task can be solved in *R + O(1)* rounds.

If some search task can be solved in *R* rounds of parallel computation, then any search task can be solved in *R + O(1)* rounds.



Recap

- Transformers and GNNs are both message-passing models.
- Neighborhood restrictions for GNNs yield better sample complexity…
- But they also provide sharp capacity limitations.
- Novel transformer size complexity hierarchy for graph reasoning tasks.

Understanding Transformer Reasoning Capabilities via Graph Algorithms

Clayton Sanford, Bahare Fatemi, Ethan Hall, Anton Tsitsulin, Mehran Kazemi, Jonathan Halcrow, Bryan Perozzi, Vahab Mirrokni

Which transformer scaling regimes are able to perfectly solve different classes of algorithmic problems? While tremendous empirical advances have been attained by transformer-based neural networks, a theoretical understanding of their algorithmic reasoning capabilities in realistic parameter regimes is lacking. We investigate this question in terms of the network's depth, width, and number of extra tokens for algorithm execution. Our novel representational hierarchy separates 9 algorithmic reasoning problems into classes solvable by transformers in different realistic parameter scaling regimes. We prove that logarithmic depth is necessary and sufficient for tasks like graph connectivity, while single-layer transformers with small embedding dimensions can solve contextual retrieval tasks. We also support our theoretical analysis with ample empirical evidence using the GraphQA benchmark. These results show that transformers excel at many graph reasoning tasks, even outperforming specialized graph neural networks.

NeurIPS 2024!

Augmented architectures

Motivation: Develop an architecture that utilizes the advantages of transformers, state-space models, and GNNs.

Graph sequence models:

- Sequential input order from hierarchical clustering
- Local encoding with GNN-powered embeddings
- Global encoding created by few-pass state-space models

Best of Both Worlds: Advantages of Hybrid Graph Sequence Models

Ali Behrouz, Ali Parviz, Mahdi Karami, Clayton Sanford, Bryan Perozzi, Vahab Mirrokni

Modern sequence models (e.g., Transformers, linear RNNs, etc.) emerged as dominant backbones of recent deep learning frameworks, mainly due to their efficiency, representational power, and/or ability to capture long-range dependencies. Adopting these sequence models for graph-structured data has recently gained popularity as the alternative to Message Passing Neural Networks (MPNNs). There is, however, a lack of a common foundation about what constitutes a good graph sequence model, and a mathematical description of the benefits and deficiencies in adopting different sequence models for learning on graphs. To this end, we first present Graph Sequence Model (GSM), a unifying framework for adopting sequence models for graphs, consisting of three main steps: (1) Tokenization, which translates the graph into a set of sequences; (2) Local Encoding, which encodes local neighborhoods around each node; and (3) Global Encoding, which employs a scalable sequence model to capture long-range dependencies within the sequences. This framework allows us to understand, evaluate, and compare the power of different sequence model backbones in graph tasks. Our theoretical evaluations of the representation power of Transformers and modern recurrent models through the lens of global and local graph tasks show that there are both negative and positive sides for both types of models. Building on this observation, we present GSM++, a fast hybrid model that uses the Hierarchical Affinity Clustering (HAC) algorithm to tokenize the graph into hierarchical sequences, and then employs a hybrid architecture of Transformer to encode these sequences. Our theoretical and experimental results support the design of GSM++, showing that GSM++ outperforms baselines in most benchmark evaluations.

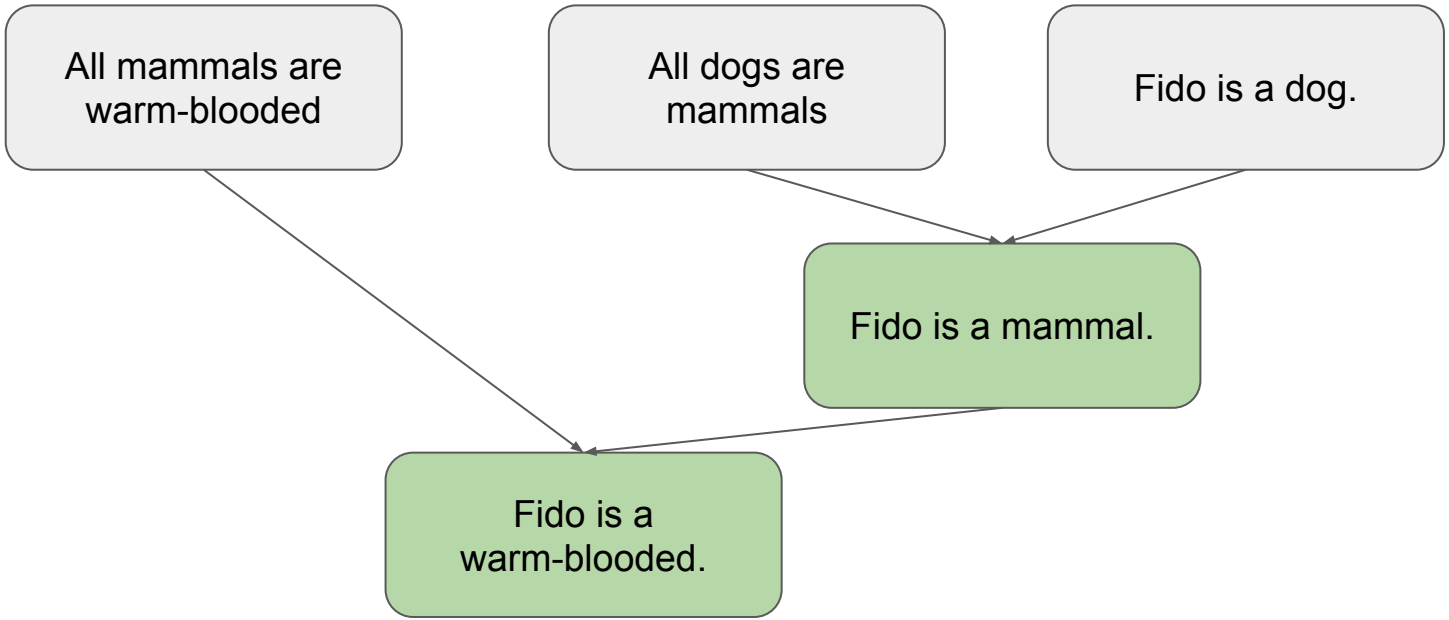
New result!

Part II

Creating Synthetic Problem Instances with Graphs

Reasoning is fundamentally a process over Graphs.

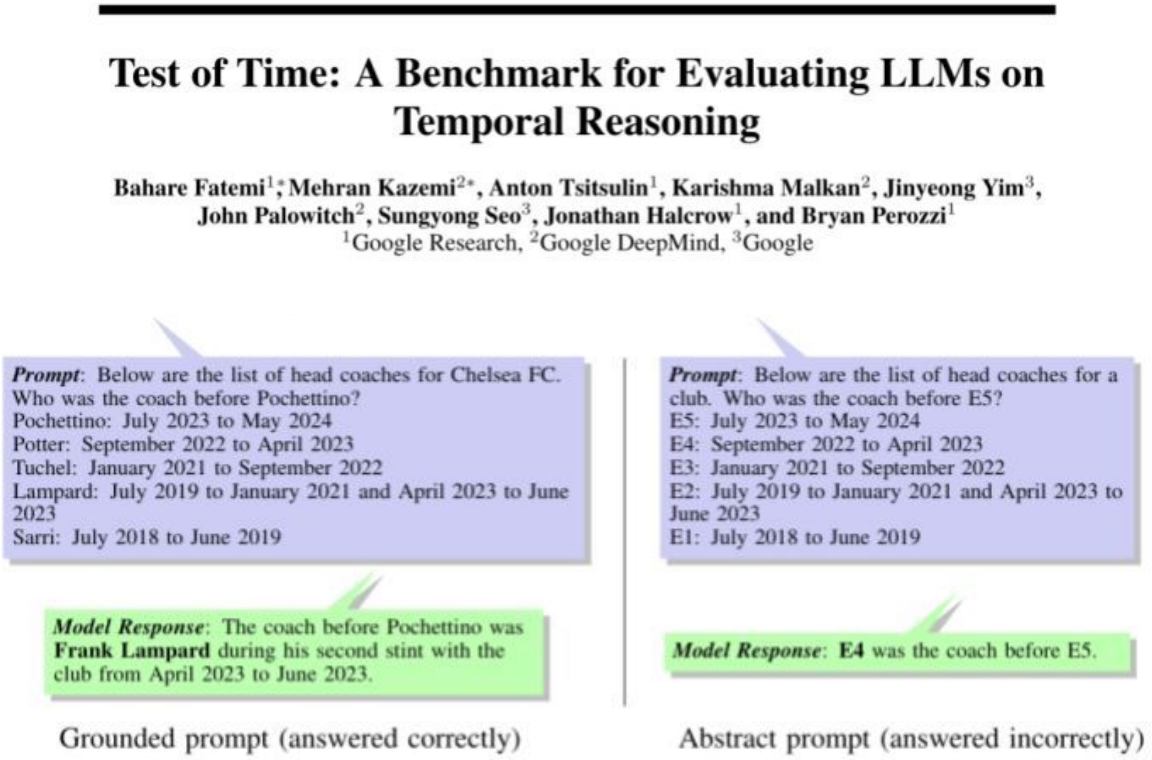
- When we ***reason***, we assemble a ***chain of inferences***, where each claim directly follows from another.
- **Knowledge Graphs** give us a powerful tool for encoding arbitrarily structured facts about the world.
- So, the ability to reason over graph structured data is ***critical*** to the project of developing systems that can ***reason*** in novel settings.



Graphs for synthetic data generation

Synthetic data generation is becoming more popular for evaluating LLMs.

- Why?
  - Generating new data gives us a source which we know was not in our training set.
  - Synthetic data can disentangle “reasoning” from memorized facts (“parametric knowledge”)
- Graphs can help here!



Test of Time: A Benchmark for Evaluating LLMs on Temporal Reasoning

Bahare Fatemi[1]*, Mehran Kazemi[2]*, Anton Tsitsulin[1], Karishma Malkan[2], Jinyeong Yim[3], John Palowitch[2], Sungyong Seo[3], Jonathan Halcrow[1], and Bryan Perozzi[1]
[1]Google Research, [2]Google DeepMind, [3]Google

Prompt: Below are the list of head coaches for Chelsea FC. Who was the coach before Pochettino?
Pochettino: July 2023 to May 2024
Potter: September 2022 to April 2023
Tuchel: January 2021 to September 2022
Lampard: July 2019 to January 2021 and April 2023 to June 2023
Sarri: July 2018 to June 2019

Model Response: The coach before Pochettino was **Frank Lampard** during his second stint with the club from April 2023 to June 2023.

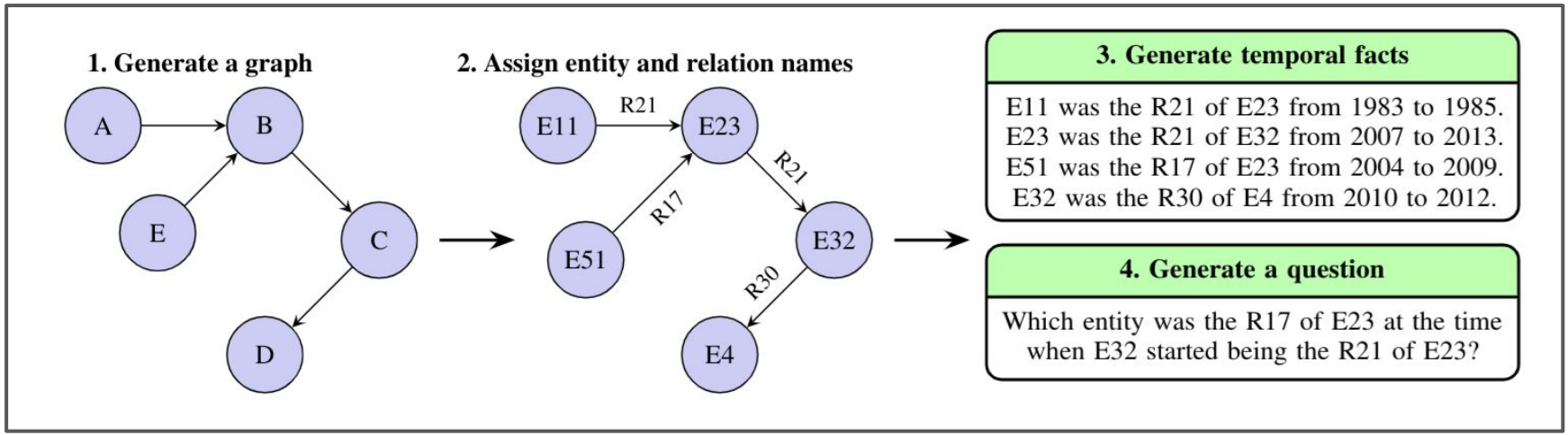
Grounded prompt (answered correctly)

Prompt: Below are the list of head coaches for a club. Who was the coach before E5?
E5: July 2023 to May 2024
E4: September 2022 to April 2023
E3: January 2021 to September 2022
E2: July 2019 to January 2021 and April 2023 to June 2023
E1: July 2018 to June 2019

Model Response: **E4** was the coach before E5.

Abstract prompt (answered incorrectly)

Turning Random Graphs into Random KGs



Illustrating different synthetic temporal datasets

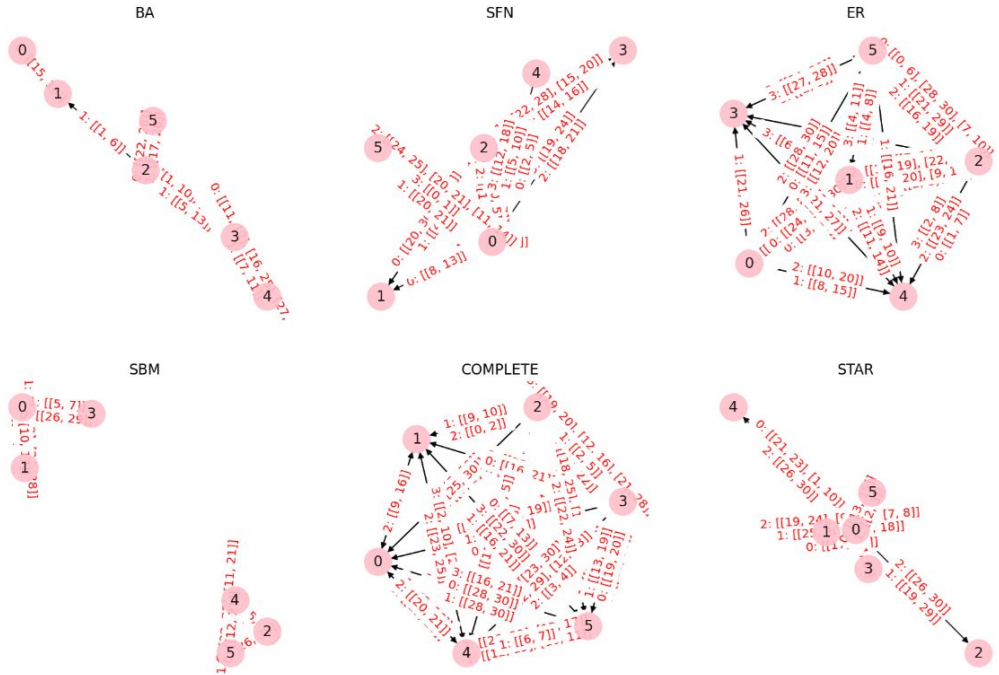


Figure 5: A visualization of a representative graph from each graph generator: Erdős-Rényi (ER), Scale-Free Networks (SFN), Barabási–Albert (BA), Stochastic Block Model (SBM), star-graph, and complete-graph.

Types of Questions Generated Through Random Graphs

Table 1: Example for each question type in the `ToT-Semantic` dataset.

Question Type	**Example**
EventAtTimeT	Find the entity that evidently was the R17 of E69 in year 1932.
EventAtWhatTime	At what time did E69 start being the R90 of E22?
NumberOfEventsInTimeInterval	Find the number of unique entities that were the R82 of E27 between 1952 to 1957. Relations that ended in 1952 or started in 1957 must be counted as well.
BeforeAfter	Immediately before E59, which entity was the R20 of E6?
EventAtTimeOfAnotherEvent	E94 was the R82 of which entity at the time when E83 started being the R20 of E59?
FirstLast	Which entity was the first that was the R35 of E91?
RelationDuration	When E24 was the R53 of E11 for the 2nd time, for how many years did the relation last? The duration can be computed by subtracting the start time from the end time.
Timeline	Which entities were the R17 of E69?

Differences in Performance

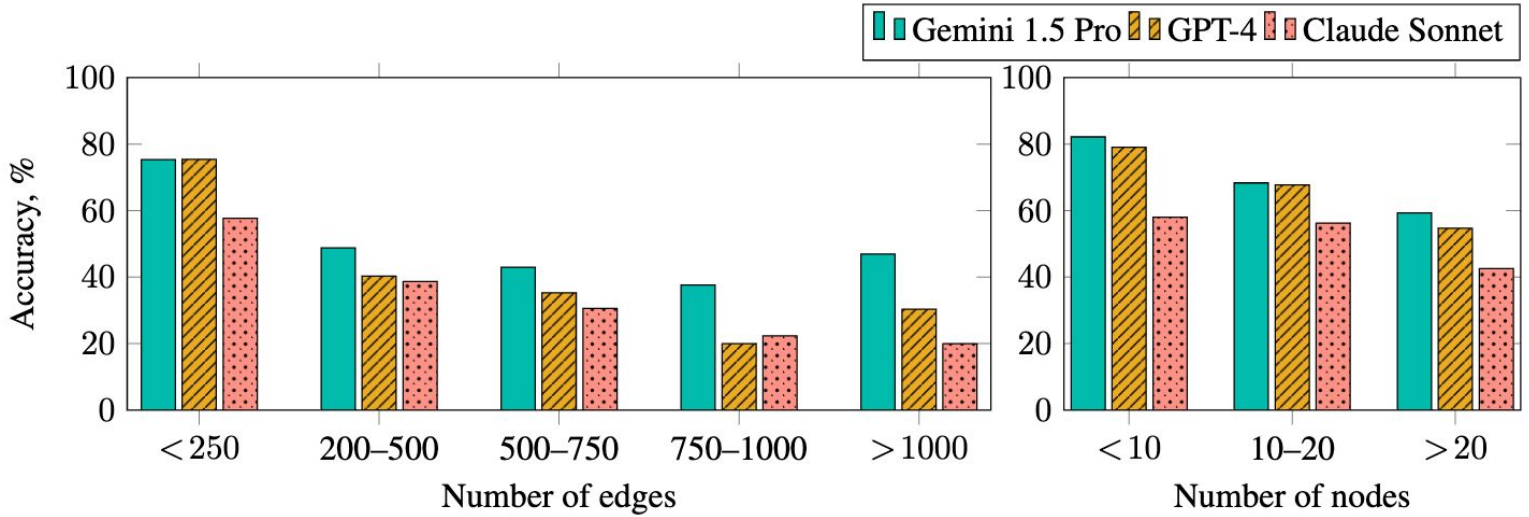
Table 5: LLM accuracy on temporal reasoning by question category.

Temporal Question Type	Claude-3-Sonnet	GPT-4	Gemini 1.5 Pro
EventAtTimeT	47.14	65.43	**72.29**
EventAtWhatTime	90.29	89.43	**93.14**
NumberOfEventsInTimeInterval	29.71	**61.43**	59.14
BeforeAfter	53.14	**55.43**	52.86
EventAtTimeOfAnotherEvent	50.00	67.14	**71.43**
FirstLast	**68.57**	67.71	**68.57**
RelationDuration	41.43	80.00	**84.57**
Timeline	24.00	28.29	**36.29**
Average Rank	2.56	2.00	**1.44**

The influence of graph structure on accuracy.

Graph	Claude-3-Sonnet	GPT-4	Gemini 1.5 Pro
BA	48.50	**63.25**	62.75
Complete	34.00	40.25	**52.50**
ER	42.25	48.75	**60.50**
SBM	42.00	50.75	**57.75**
SFN	58.75	75.25	**75.75**
Star	59.50	**80.25**	74.25
AWE	68.75	**92.00**	87.50
Average Rank	3.00	1.57	**1.43**

The influence of problem size on accuracy.



Performance is sensitive to the ordering of facts.

Order of facts	Claude-3-Sonnet	GPT-4	Gemini 1.5 Pro
Shuffle	45.71	60.71	**63.04**
RelationAndStartTime	54.29	65.36	**68.57**
StartTimeAndRelation	47.68	60.54	**64.64**
StartTimeAndTarget	49.11	61.61	**65.18**
TargetAndStartTime	73.57	62.60	**75.00**
Average Rank	2.80	2.20	**1.00**

ToT-Arithmetic

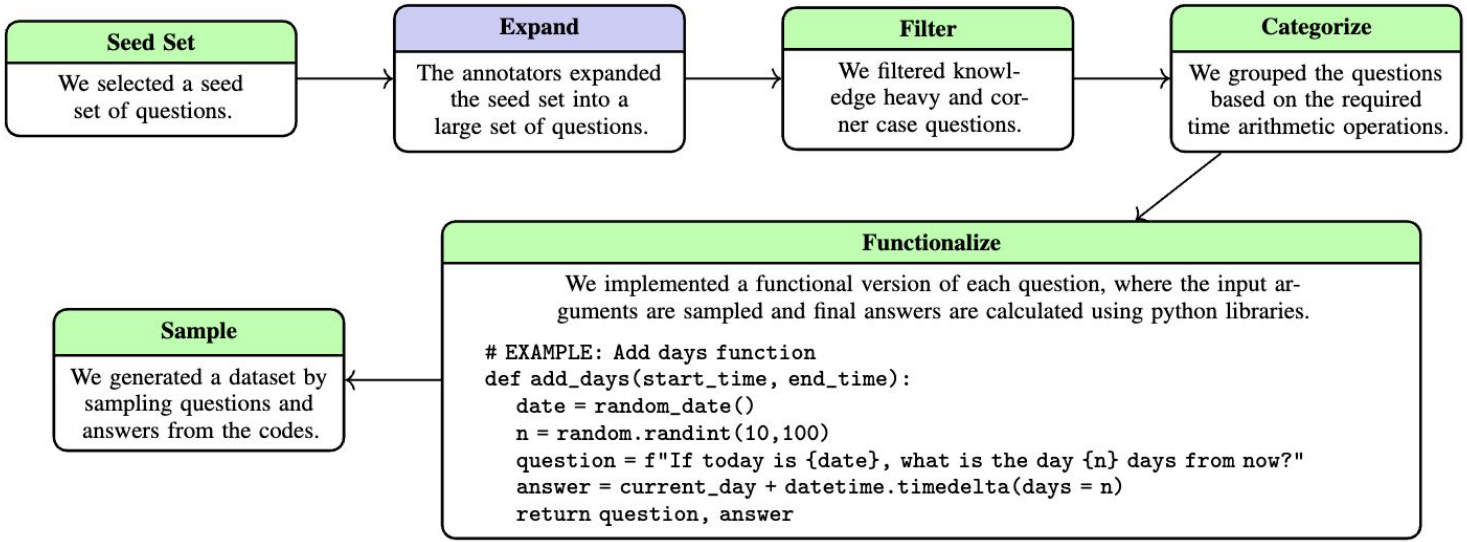


Figure 3: Steps for creating the `ToT-Arithmetic` dataset. The green and blue colors represent the operations done by the authors and the annotators respectively.

Temporal Arithmetic Problems

Table 2: Examples for each question type in the `ToT-Arithmetic` dataset.

Category	**Example**
AddSubtract	Your driver's license expires on 18 May, 2017. You receive a renewal notice saying it can be renewed 117 days in advance. What's the earliest date you can renew your license?
Compare	E42 was discovered in 14 April, 52 BC and E11 was discovered in 05 October, 530 BC. Which was discovered earlier?
Duration	Stella and William were born on 1999-Dec-16 and 2000-Oct-03 respectively. When William was 400 days old, how old was Stella in days?
Schedule	Lucas is available from 11 to noon and also from 3:30 to 5. Asher is available from 11 to 12:30 and also from 4 to 5. They want to have a 30 minute meeting. The meeting has to start on the hour or half hour. How many possibilities are there for the meeting time?
Timezone	Flight departs location A at 11:08 (24hr) UTC(+0000). It reaches location B at 07:23:20 PM IST(+0530). What is the total time duration taken to fly?
Trick	If the date for the day before tomorrow in yyyy-mm-dd format is 2016-01-20, what is the date 27 days from now in the same format?
MultiOp	Alex solves 2 puzzles in 4 hours, 50 minutes, and 22 seconds. What is the time taken by them to solve 6 puzzles, at the same pace.

Temporal Arithmetic is different from Temporal Reasoning.

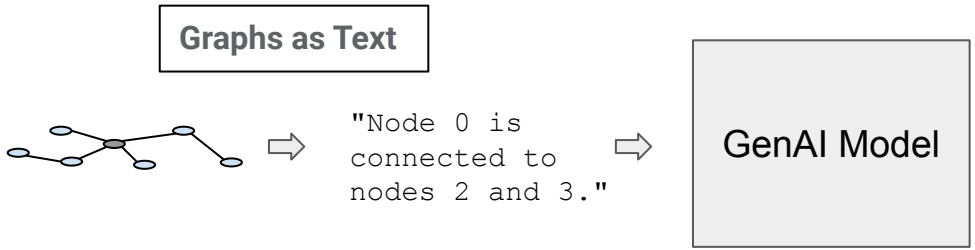
Category	Claude-3-Sonnet	GPT-4	Gemini 1.5 Pro
AddSubtract	58.57	**76.28**	71.14
Compare	39.14	**63.14**	55.43
Duration	15.00	**16.00**	13.50
Schedule	29.60	**43.60**	40.00
Timezone	74.00	88.00	**90.00**
Trick	40.40	**45.60**	41.20
MultiOp	26.57	46.86	**62.57**
Average Rank	2.86	**1.29**	1.86

Part III

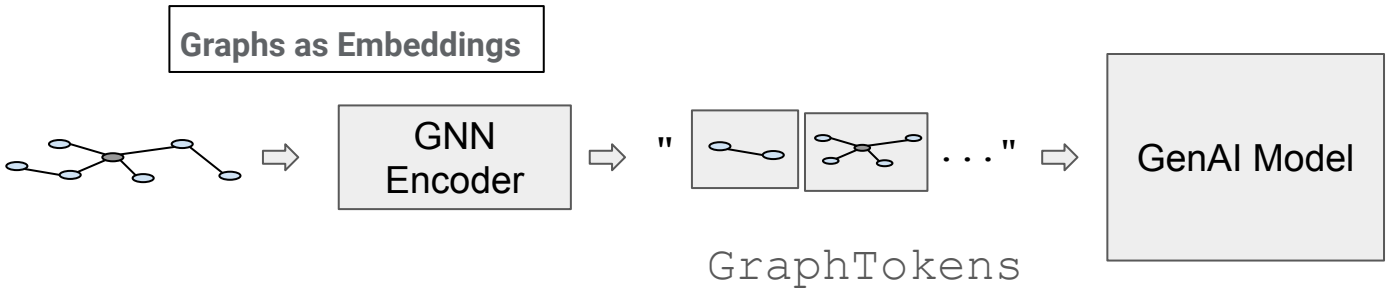
Representing Graph Structured Data for In-Context Learning

How can we best get graph structure into GenAI models?

Idea 1: Graphs as Text



Idea 2: Graphs as Embedding



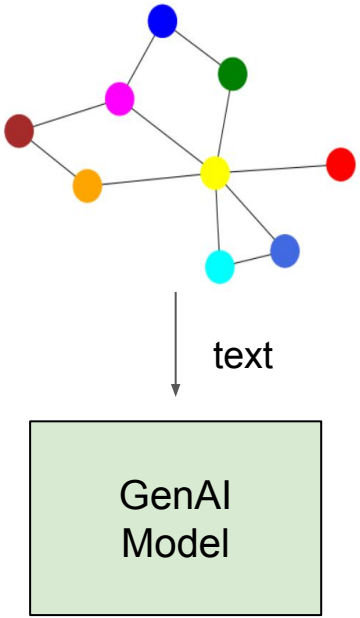
Question 1: How to best encode graphs as Text?

In this work, we seek to find the best way to encode graphs as text for “black box” use in LLMs.

Why ***text***? → Dominant LLM interface

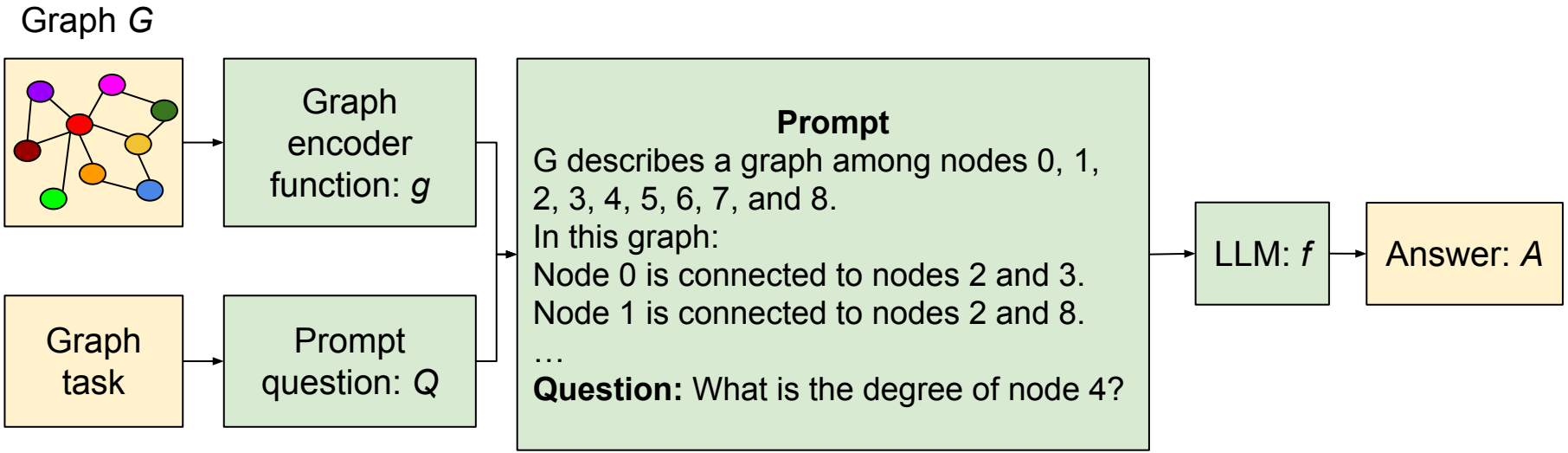
What’s novel here?

- ❖ Exploration of graph prompting
- ❖ Evaluation of graph encoding
- ❖ Analysis of the effect of graph structure



TALK LIKE A GRAPH: ENCODING GRAPHS FOR LARGE LANGUAGE MODELS (ICLR’24)
https://arxiv.org/pdf/2310.04560.pdf

Overview of Framework

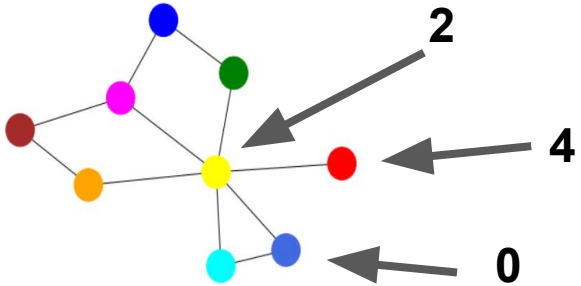


Goal: We seek to find a *graph encoder* and *prompt question* that can maximize the score of the correct answers over the training dataset.

TALK LIKE A GRAPH: ENCODING GRAPHS FOR LARGE LANGUAGE MODELS
https://arxiv.org/pdf/2310.04560.pdf

Encoding Graphs to Text

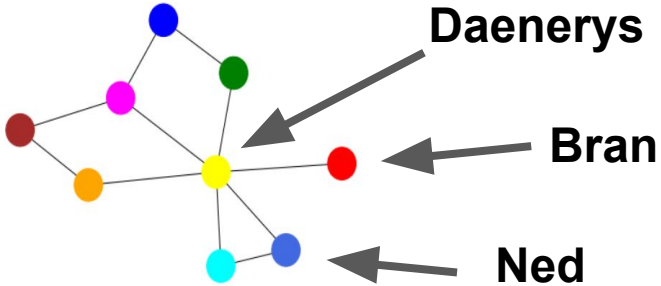
As computer scientists, we'd normally think about encoding graphs like this:



Incident: G describes a graph among 0, 1, 2, 3, 4, 5, 6, 7, and 8.
In this graph:
Node 0 is connected to nodes 1, 2.
Node 1 is connected to nodes 0, 2.
Node 2 is connected to nodes 0, 1, 3, 4, 5, 7.
Node 3 is connected to nodes 2, 8.
Node 4 is connected to node 2.

What's in a node id?

Q: **What if we used Fantasy Characters?**



GOT: G describes a friendship graph among Ned, Cat, Daenerys, Jon, Bran, Sansa, Arya, Cersei, and Jaime.
In this friendship graph: Ned and Cat are friends, Ned and Daenerys are friends, Cat and Daenerys are friends, Daenerys and Jon are friends, Daenerys and Bran are friends, Daenerys and Sansa are friends, Daenerys and Cersei are friends, Jon and Jaime are friends, Sansa and Arya are friends, Arya and Cersei are friends, Cersei and Jaime are friends.

(More encoder descriptions in paper)

Graph Encoding Choice Matters

graph encoding	ZERO-SHOT
Adjacency	4.83
Incident	6.16
Co-authorship	6.08
Friendship	5.16
SP	5.16
GOT	4.33
Social Network	4.58
Politician	**3.50**
Expert	5.16

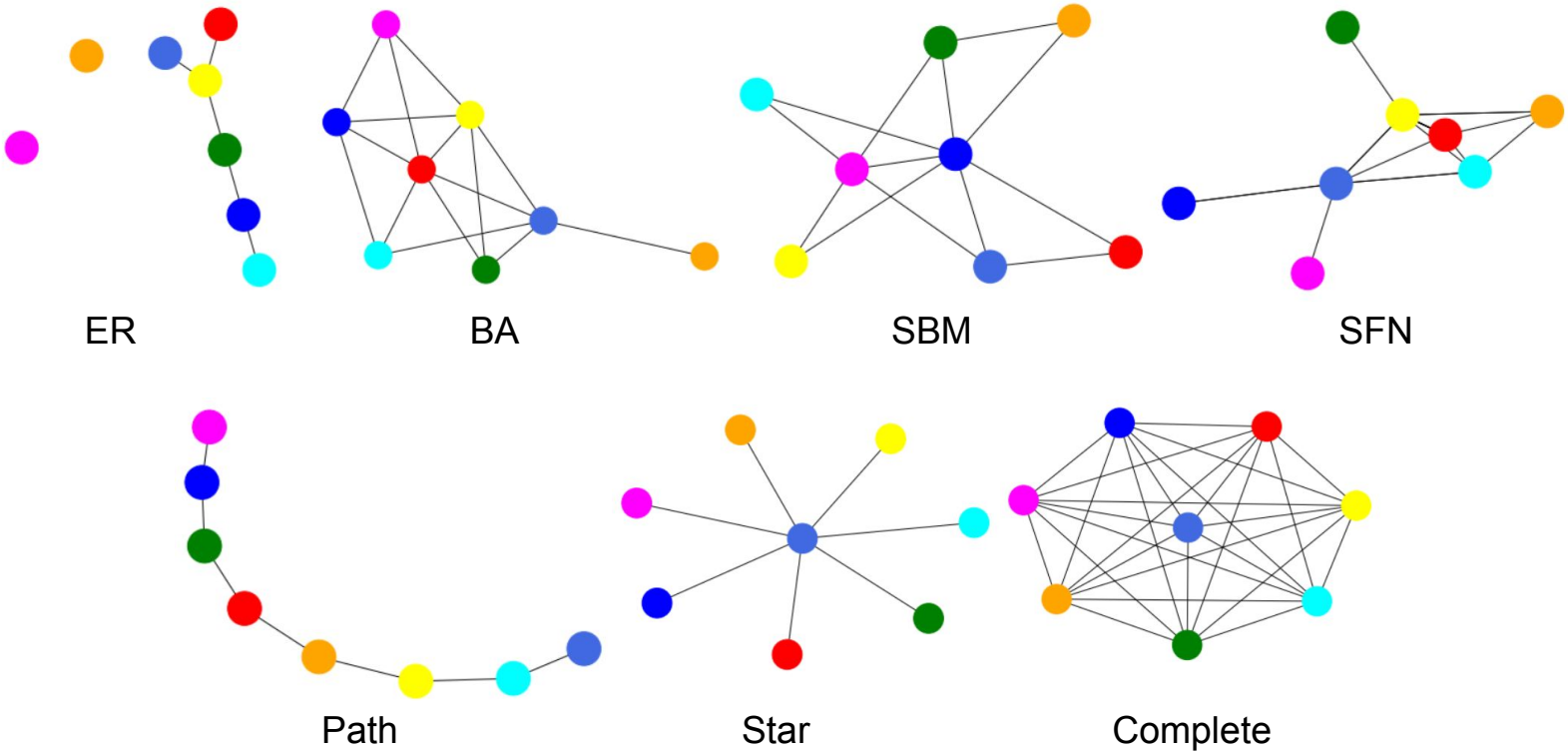
Table 5: Ranking of graph encodings from experiment in Section 3.1 (lower better).

Graph Encoding Choice Matters

graph encoding	ZERO-SHOT	ZERO-COT	FEW-SHOT	COT	COT-BAG
Adjacency	4.83	3.25	2.16	3.00	1.83
Incident	6.16	**2.58**	**2.00**	**2.33**	**1.33**
Co-authorship	6.08	6.33	5.58	6.75	8.83
Friendship	5.16	6.41	6.25	4.66	6.00
SP	5.16	4.50	5.25	5.75	4.66
GOT	4.33	4.08	5.83	5.00	6.25
Social Network	4.58	6.50	5.83	6.16	6.41
Politician	**3.50**	6.33	6.25	5.58	4.00
Expert	5.16	5.00	5.83	5.75	5.66

Table 5: Ranking of graph encodings from experiment in Section 3.1 (lower better).

Does the structure of the graph matter?



Sample graphs in GraphQA generated with different graph generators

Graph structure has a significant impact on the LLM's performance.

Method	Graph generator	Edge Existence	Node degree	Node count	Edge count	Connected nodes	Cycle check
ZERO-SHOT	Overall	49.1	17.6	23.0	12.1	23.3	75.2
	ER	45.1	13.6	22.1	11.7	14.9	76.3
	BA	50.2	18.0	24.9	13.6	20.1	72.0
	SBM	45.0	13.8	21.9	9.2	13.8	86.5
	Star	58.0	34.0	32.8	31.7	61.7	8.1
	SFN	57.6	23.1	19.9	8.0	38.1	90.0
	Path	**60.9**	14.8	31.9	28.8	26.6	5.9
	Complete	19.8	12.6	20.7	6.2	13.3	**91.7**
COT	Overall	40.4	29.6	31.7	12.2	24.3	59.5
	ER	41.2	28.4	28.8	12.6	12.8	61.2
	BA	40.0	30.0	35.0	14.3	20.8	58.5
	SBM	40.3	26.5	30.2	8.7	13.0	65.8
	Star	40.3	**38.0**	**41.8**	**31.6**	**68.6**	21.3
	SFN	40.2	32.2	30.8	7.1	43.2	66.0
	Path	42.0	35.1	35.3	31.1	27.6	19.7
	Complete	39.6	21.9	28.9	3.9	14.6	69.3

Table 4: Comparing different graph generators on different graph tasks on PaLM 62B. The most effective prompting heuristic is highlighted with an underline, and the top-performing graph generator algorithm for the respective heuristic is highlighted in bold.

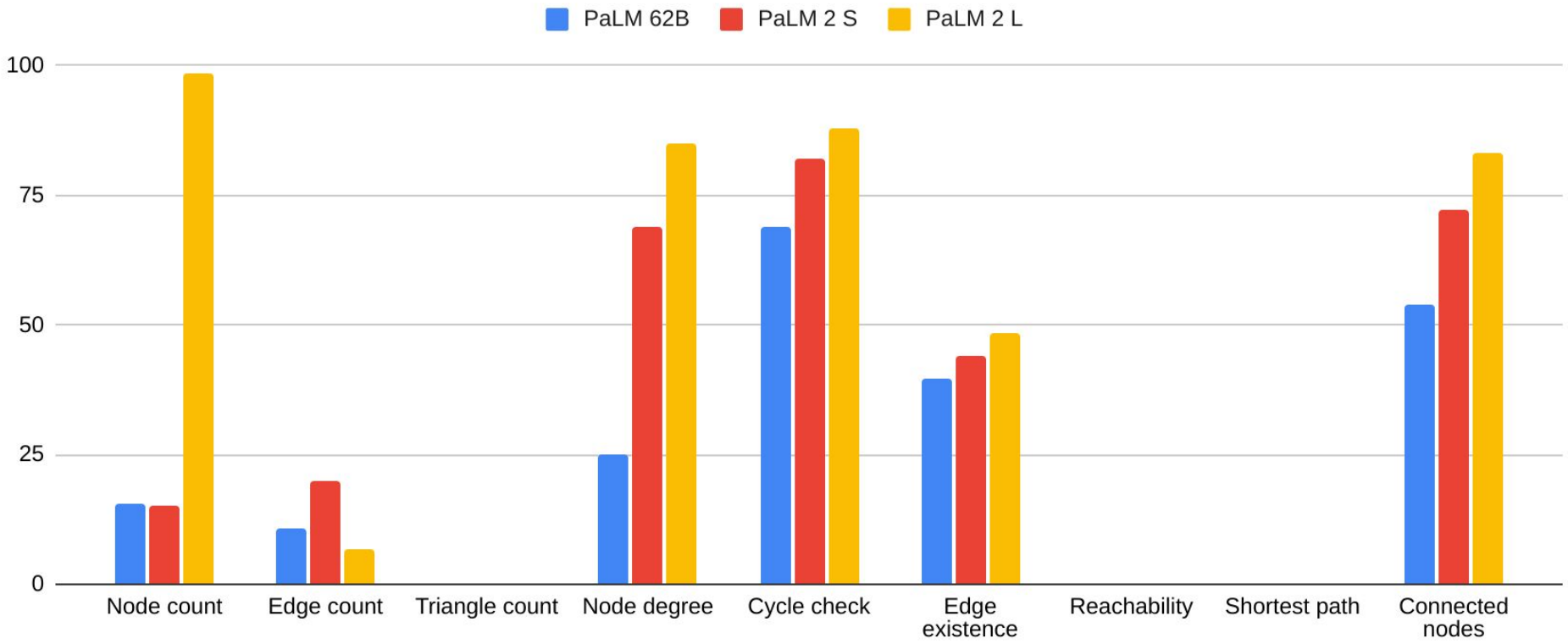
Extra edges may distract LLMs

Method	Graph generator	Edge Existence	Node degree	Node count	Edge count	Connected nodes	Cycle check
ZERO-SHOT	Overall	49.1	17.6	23.0	12.1	23.3	75.2
	ER	45.1	13.6	22.1	11.7	14.9	76.3
	BA	50.2	18.0	24.9	13.6	20.1	72.0
	SBM	45.0	13.8	21.9	9.2	13.8	86.5
	Star	58.0	34.0	32.8	31.7	61.7	8.1
	SFN	57.6	23.1	19.9	8.0	38.1	90.0
	Path	**60.9**	14.8	31.9	28.8	26.6	5.9
	Complete	19.8	12.6	20.7	6.2	13.3	**91.7**
COT	Overall	40.4	29.6	31.7	12.2	24.3	59.5
	ER	41.2	28.4	28.8	12.6	12.8	61.2
	BA	40.0	30.0	35.0	14.3	20.8	58.5
	SBM	40.3	26.5	30.2	8.7	13.0	65.8
	Star	40.3	**38.0**	**41.8**	**31.6**	**68.6**	21.3
	SFN	40.2	32.2	30.8	7.1	43.2	66.0
	Path	42.0	35.1	35.3	31.1	27.6	19.7
	Complete	39.6	21.9	28.9	3.9	14.6	69.3

Table 4: Comparing different graph generators on different graph tasks on PaLM 62B. The most effective prompting heuristic is highlighted with an underline, and the top-performing graph generator algorithm for the respective heuristic is highlighted in bold.

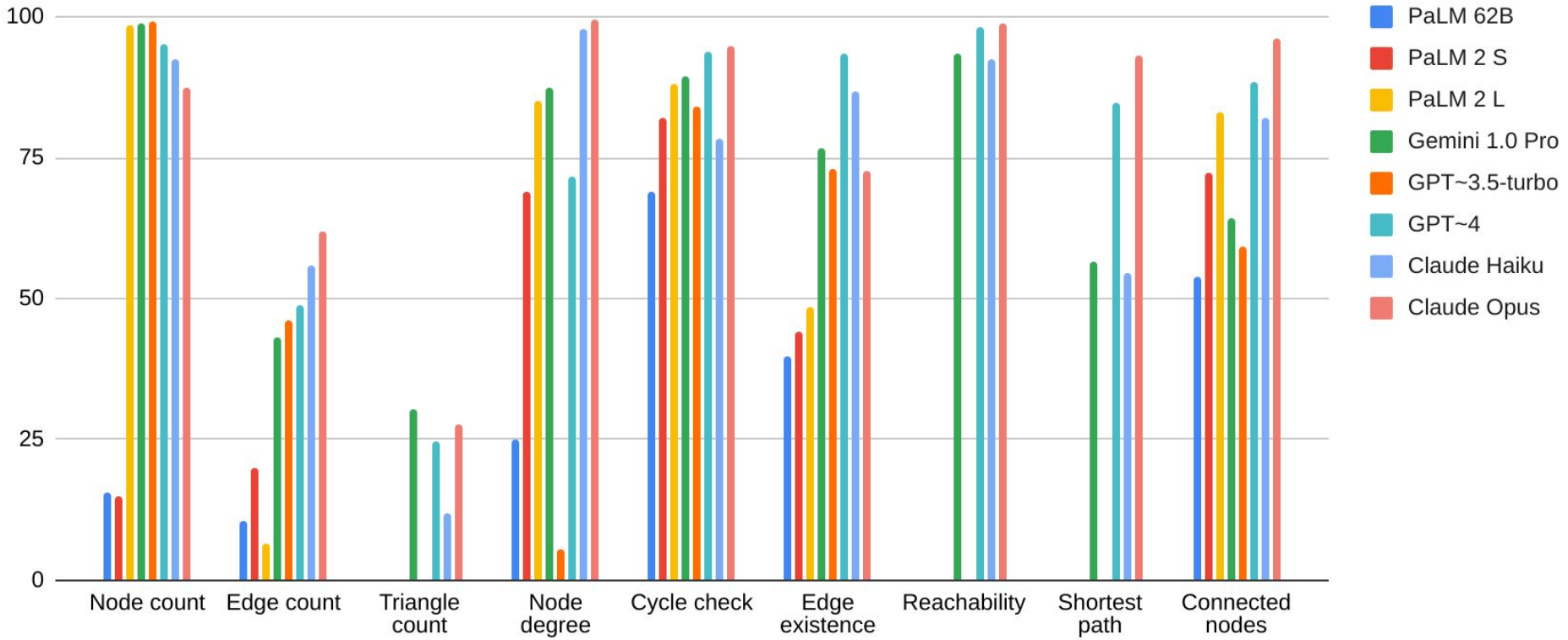
“Talk like a Graph” Leaderboard

Sept 20th, 2023

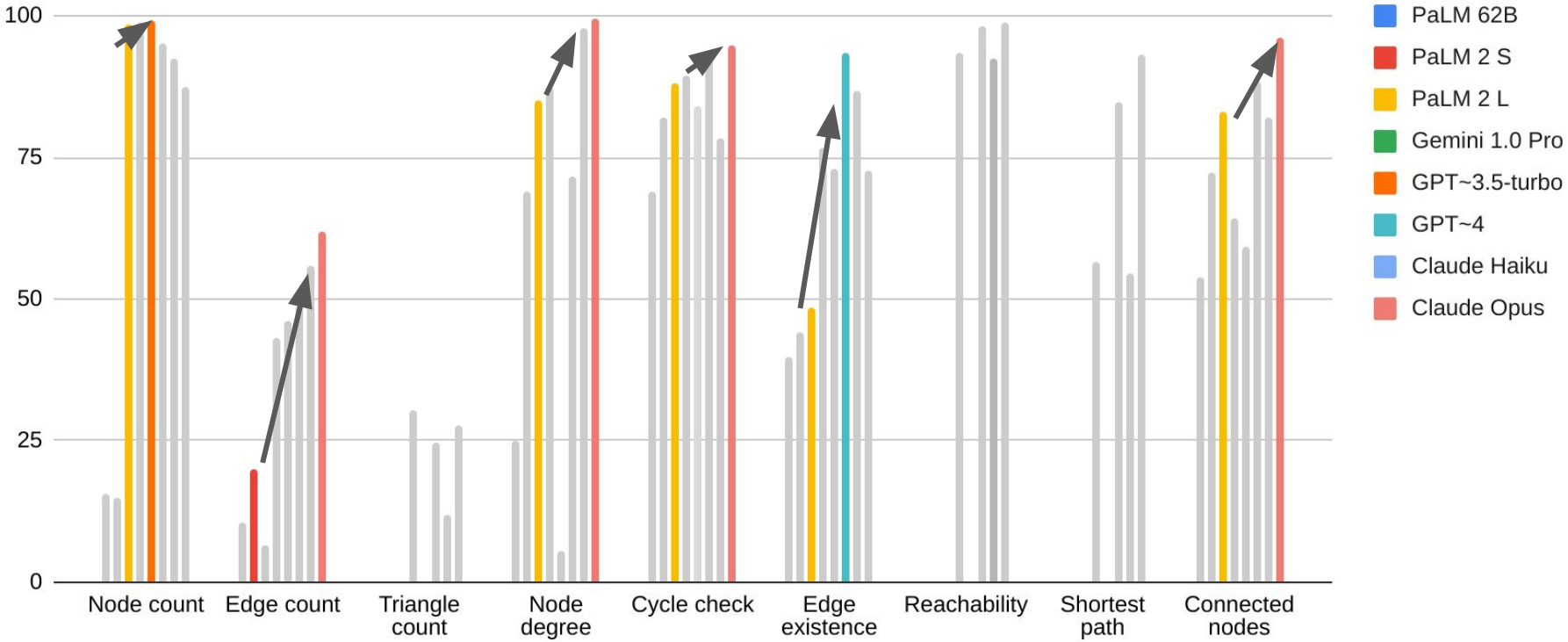


“Talk like a Graph” Leaderboard

May 8th, 2024



Takeaway: Models are getting much better at these tasks!



Question 2: How to Best Encode Data for LLMs?

In this work, we seek to find the best way to encode data for LLMs.

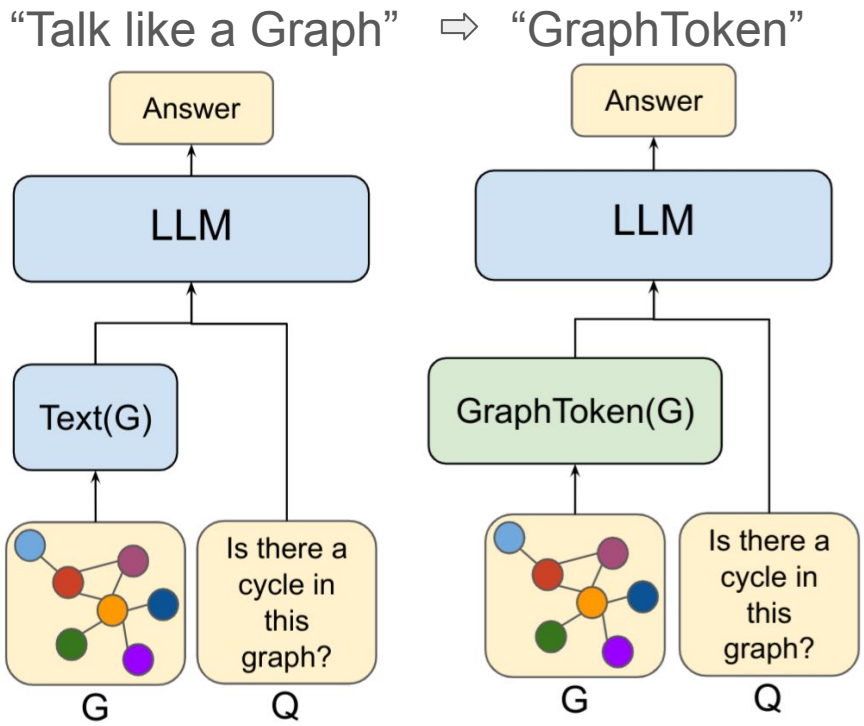
Why ***graphs***? → Can encode anything, e.g.

- complex interactions
- multi-scale effects
- multi-modal data

Why ***not text***? →*Talk like a Graph* showed some limitations of text encoding.

What's novel here?

- ❖ GraphToken → Fundamentally new way of learning graph embeddings
- ❖ GNN Soft Prompting Functions
- ❖ Generalization experiments of GraphToken

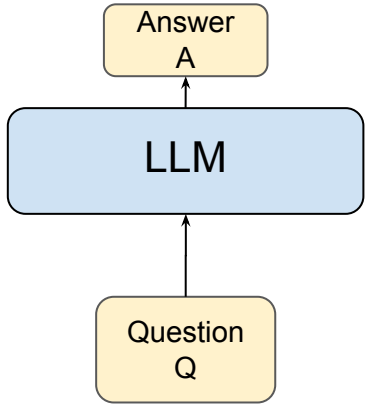


LET YOUR GRAPH DO THE TALKING: ENCODING STRUCTURED DATA FOR LLMS
https://arxiv.org/pdf/2402.05862.pdf

Review: Prompt Engineering

Prompting seeks to find a question Q such that the model *f()* will return the answer A.

$$A = f(Q)$$



Prompt engineering has yielded a number of techniques to improve model performance, such as:

- Few Shot Prompting
- Chain of Thought Prompting
- Expert Prompting
- …

Expert: You are a graph analyst and you have been given a graph G among A, B, C, D, E, F, G, H, and I. G has the following undirected edges: A -> B, A -> C, ..., H -> I.

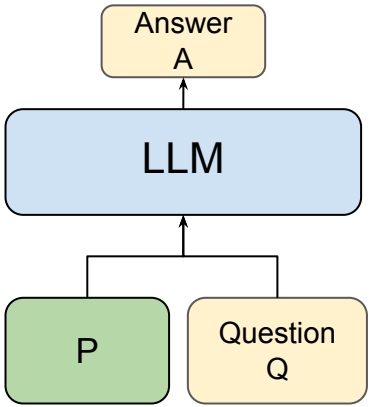
Expert prompting from ‘Talk Like a Graph’.

Review: Soft Prompting

Soft Prompting methods (e.g. Lester et al, EMNLP'21) operates on the assumption that the question Q that best captures your intent is hard to find (and may not exist).

Big Idea: Instead of learning of optimizing a discrete prompt, why not optimize it in continuous space?

They learn a prefix "P" using learnable tokens such that f(P + Q) = A.

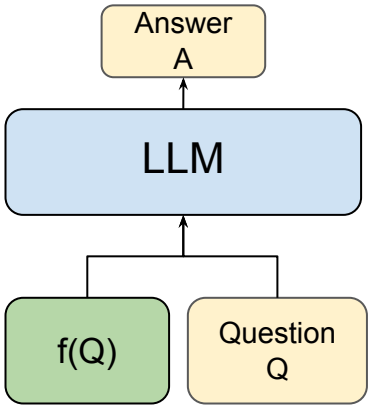


Review: Soft Prompt Functions

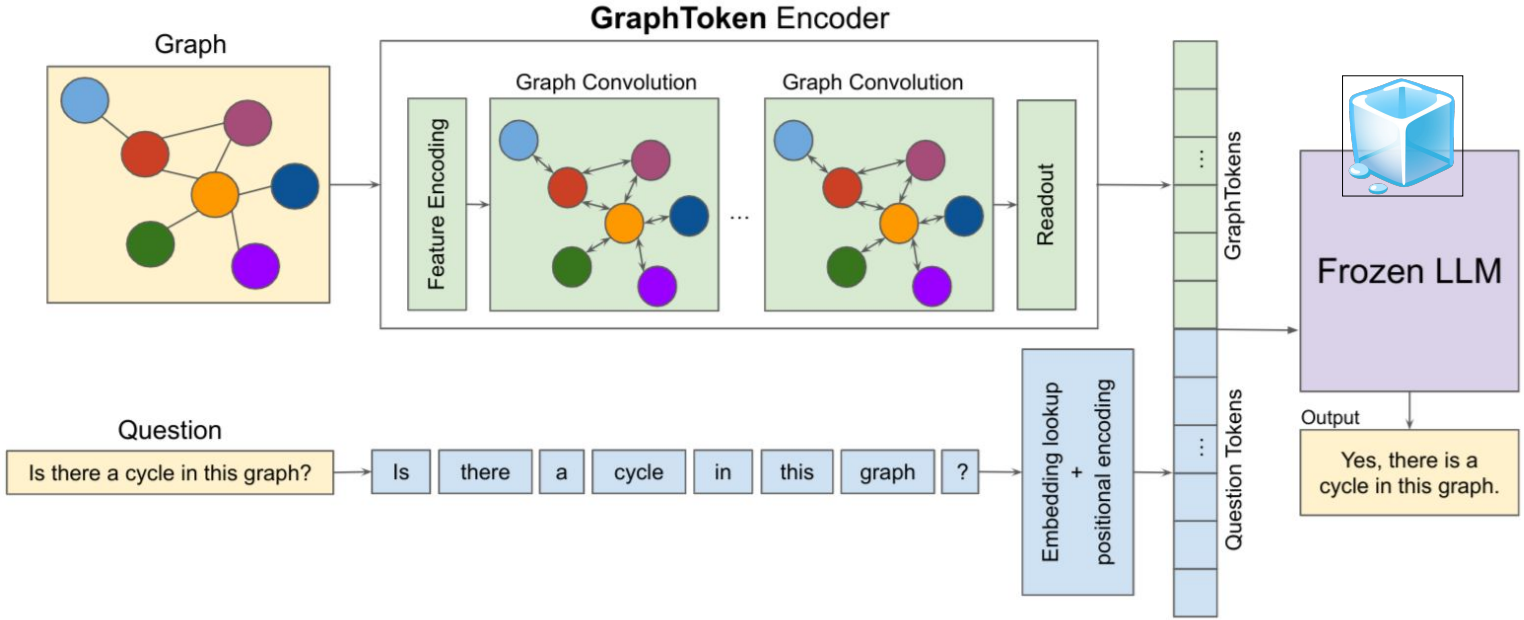
Why limit ourselves to one (or a few prompts) when we could learn them?

(Levine et al'22) examine *soft prompt functions*, which learn a function to create soft prompts. (similar: HyperPrompt (He et al'22), which do multitask learning with prompt tuning).

These open the door for injecting data directly into the model's token space.



GraphToken: *Graphs that Talk*



Goal: We seek a GNN encoding function that can project structured data into the token space of a frozen model.

LET YOUR GRAPH DO THE TALKING: ENCODING STRUCTURED DATA FOR LLMS
https://arxiv.org/pdf/2402.05862.pdf

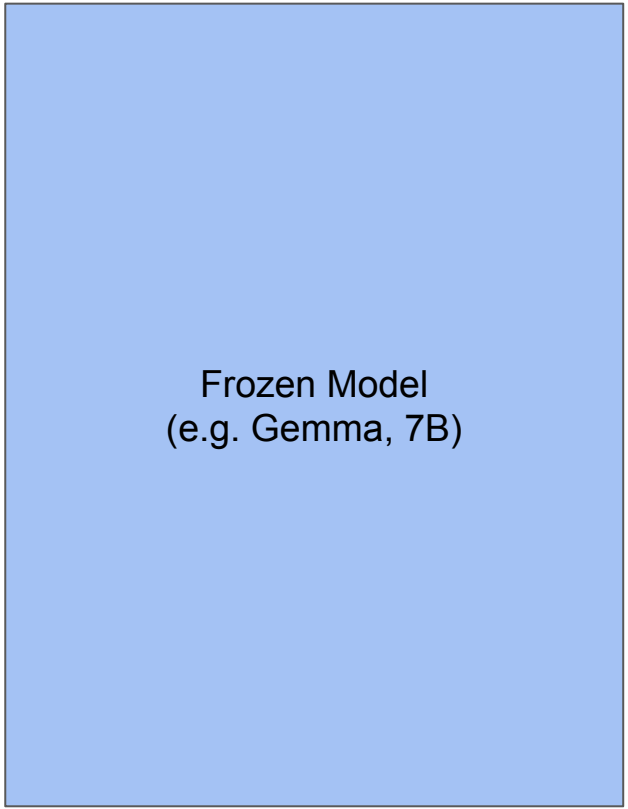
Aside: Parameter Efficiency

GraphToken is a ***parameter-efficient*** method by design.

Updating the graph encoder requires many less parameters than updating an LLM.

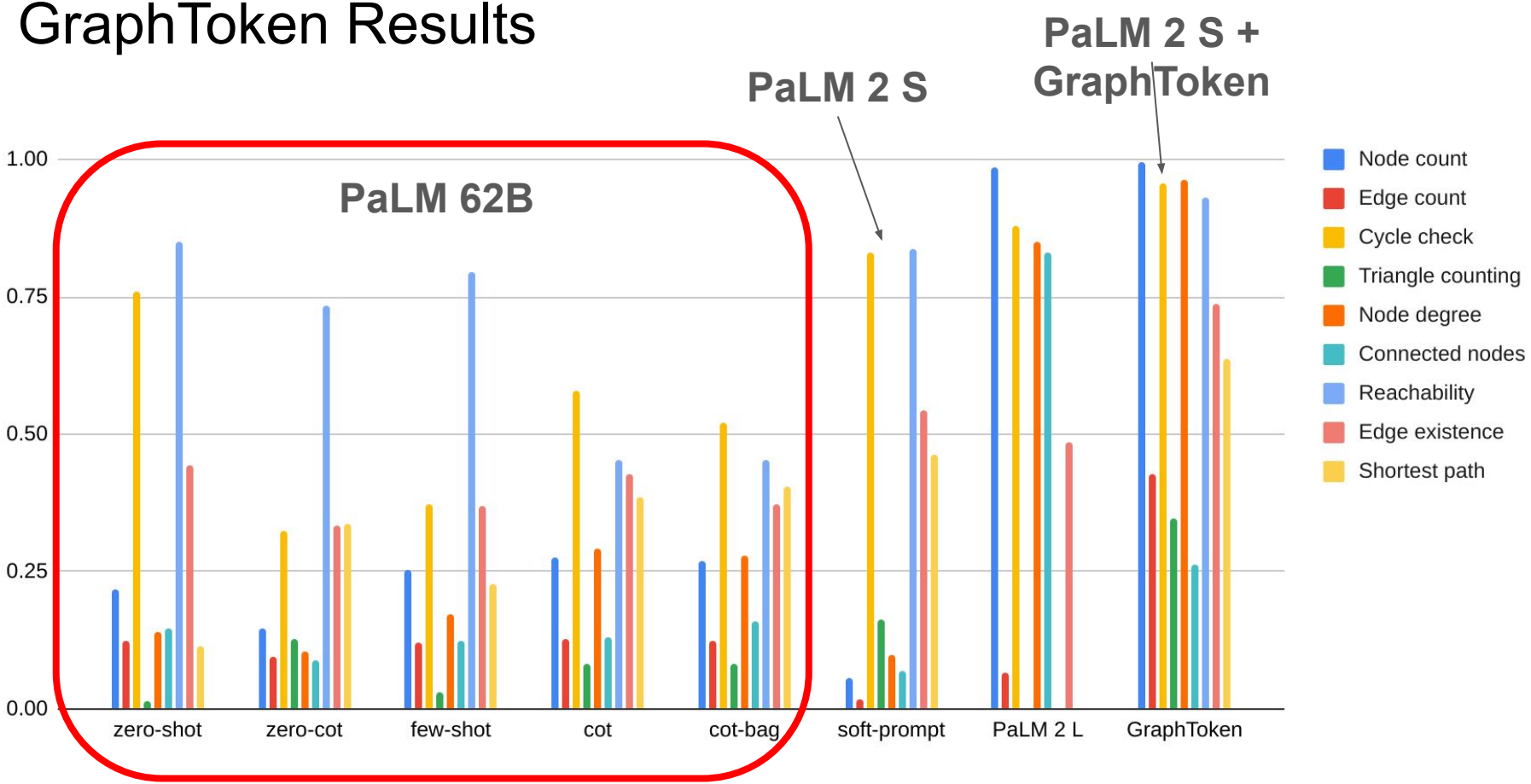
For context, even the smallest LLMs (e.g. Gemma 7B) have billions of parameters → GraphToken requires O(10M)

In other words, the results shown here require adding *less than* ~0.14% (10M / 7B) additional model parameters to a small LLM, and this ratio obviously drops as the LLM parameter count increases.

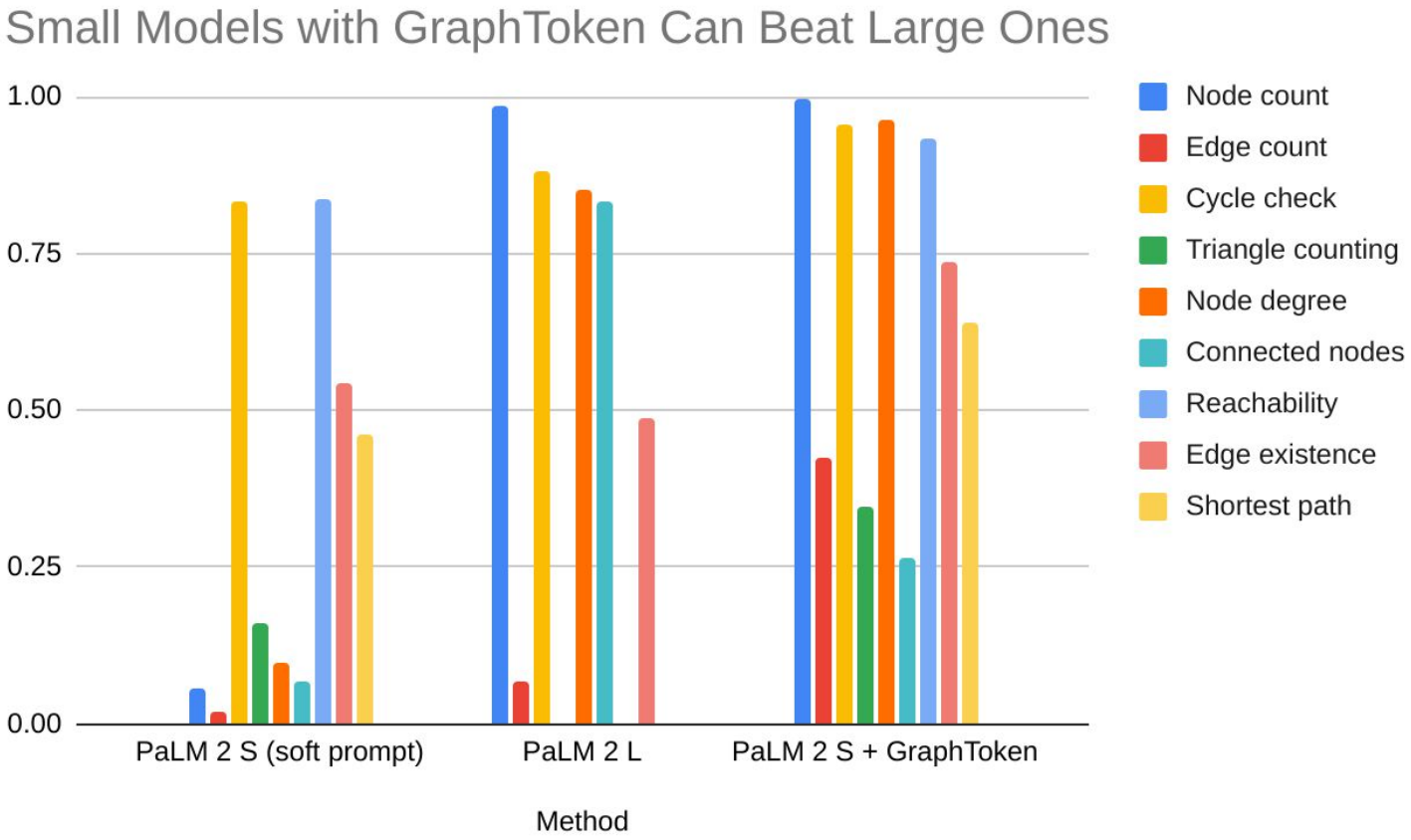


*Not to scale.

GraphToken Results



Key Takeaway



Understanding Transformer Reasoning Capabilities via Graph Algorithms (NeurIPS'24)
Sanford, Fatemi, Hall, Tsitsulin, Kazemi, Halcrow, Perozzi, Mirrokni
https://arxiv.org/pdf/2405.18512

Test of Time: A Benchmark for Evaluating LLMs on Temporal Reasoning
Fatemi, Kazemi, Tsitsulin, Malkan, Yim, Palowitch, Seo, Halcrow, Perozzi
https://arxiv.org/abs/2406.09170

Thanks!

Best of Both Worlds: Advantages of Hybrid Graph Sequence Models
Behrouz, Parviz, Karami, Sanford, Perozzi, Mirrokni
https://arxiv.org/pdf/2411.15671

Talk Like a Graph: Encoding Graphs for Large Language Models (ICLR'24)
Fatemi, Halcrow, Perozzi
https://arxiv.org/pdf/2310.04560.pdf

Let your Graph Do the Talking: Encoding Structured Data for LLMs
Perozzi, Fatemi, Zelle, Tsitsulin, Kazemi, Al-Rfou, Halcrow
https://arxiv.org/pdf/2402.05862.pdf