GraphLLM – Boosting Graph Reasoning Ability of Large Language Model

Team No./Name : 09/A2V2

What this paper is all about ?

- Everyone is amazed by the huge success of LLMs..!! Even for this presentation, ChatGPT helped a lot ;-)

- LLM posses exceptional capability of understanding diverse types of information, including but limited to images, audio.

- Despite of this significant strides of processing multi modal information, LLM is in-efficient to understand and reason on graph data.

- GraphLLM provides an end-to-end approach that synergize LLM with graph learning modules to enhance Graph reasoning capabilities.

- Authors compare GraphLLM with various other methods and shows the robustness & effectiveness of GraphLLM.

- Mainly focuses on comparing with **Graph2Text** based technique.

Motivation for GraphLLM

- Despite of this significant strides of processing multi modal information, LLM is in-efficient to understand and reason on graph data.

- Previous LLM performances on fundamental graph reasoning tasks are subpar.
  - For e.g. - with tailor made prompts, LLM present 33.5% accuracy on calculating shortest path with up to 20 nodes only.

- Fine Tuning OPT-2.7B, LLaMA2 – 7B/13B results in underwhelming performance in several tasks.

Motivation for GraphLLM

- Despite of this significant strides of processing multi modal information, LLM is in-efficient to understand and reason on graph data.

- Previous LLM performances on fundamental graph reasoning tasks are subpar.
  - For e.g. - with tailor made prompts, LLM present 33.5% accuracy on calculating shortest path with up to 20 nodes only.

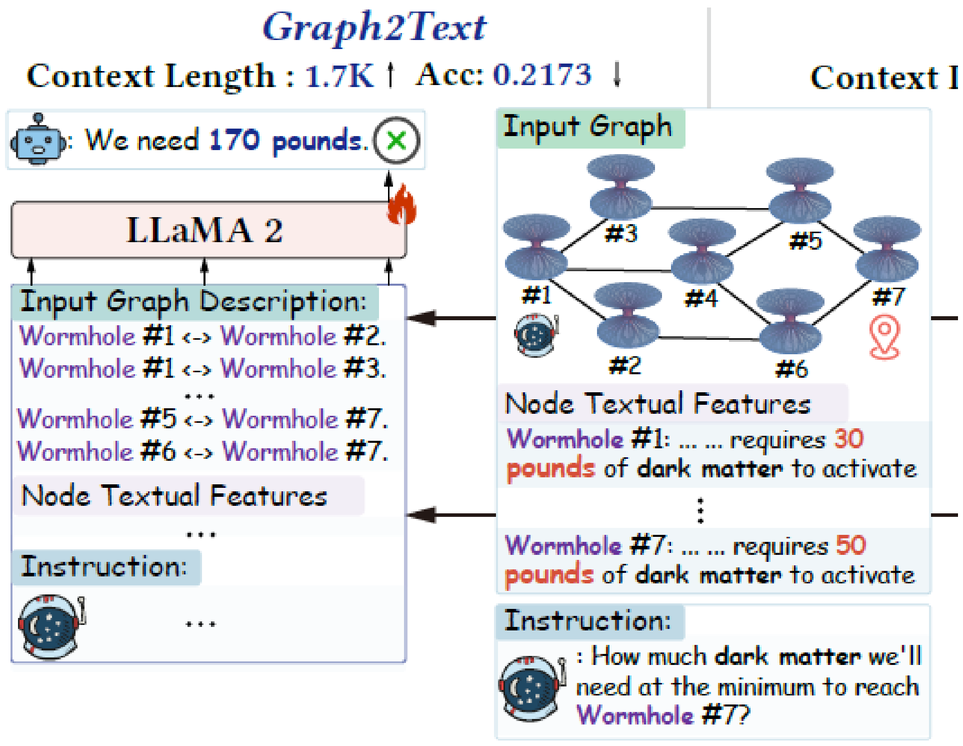
- Fine Tuning OPT-2.7B, LLaMA2 – 7B/13B results in underwhelming performance in several tasks.

What hinders the ability of LLM on graph reasoning ability ??

Possible obstacles

- Prevailing practice of converting graph to natural language descriptions – Graph2Text.
- Converting graph to text inherently incurs shortcomings that resist LLM to perform well on graph reasoning task

  - Via Graph2Text, LLM compelled to recognize structural information from text
    - *LLM may face inefficiencies*
  - Graph2Text inherently results in lengthy context of graph description –
    - *challenge for LLM to identify important information from long context.*



How GraphLLM helps ?

- GraphLLM synergistically integrate graph learning modules with LLM
  - *Harness the power of both LLM and graph reasoning modules*.

- Key advantage over Graph2Text based method –
  - **Collaborative Synergy** : Integrate graph learning module to single cohesive system by synergizing with graph learning modules.
  - **Condensed Context** : Converts graph information to concise, fixed length prefix and substantially reduces context.

- Graph LLM boosts the graph reasoning ability of LLM measured on 4 fundamental graph reasoning tasks –

text substructure counting, maximum triplet sum, shortest path, and bipartite graph matching

GraphLLM Framework

- Graph LLM consists of 3 main steps –
  - ***Node Understanding –*** *Encoder Decoder is used to extract* ***task specific*** *semantic node information from textual node description.*
  - ***Structure Understanding –*** *Graph Transformer is used to learn graph structure by aggregating task specific node representation – incorporate node semantic and graph structure information.*
  - ***Graph Enhanced Prefix Tuning for LLM –*** *Graph LLM derives graph enhanced prefix from graph representation. During prefix tuning, LLM synergizes by end to end fine tuning.*

Encoder Decoder for Node Understanding

- Goal – Extract required information form nodes based on specific graph reasoning task.
  - *For e.g. – For identifying substructures within molecule, necessary to extract atom types.*

- Encoder applied Self attention to node description and generated context vector ***Ci*** that capture semantic meaning. Decoder produce node representation ***hi*** through cross attention between ***Ci*** and **Q** – *newly initialized trainable embedding.*

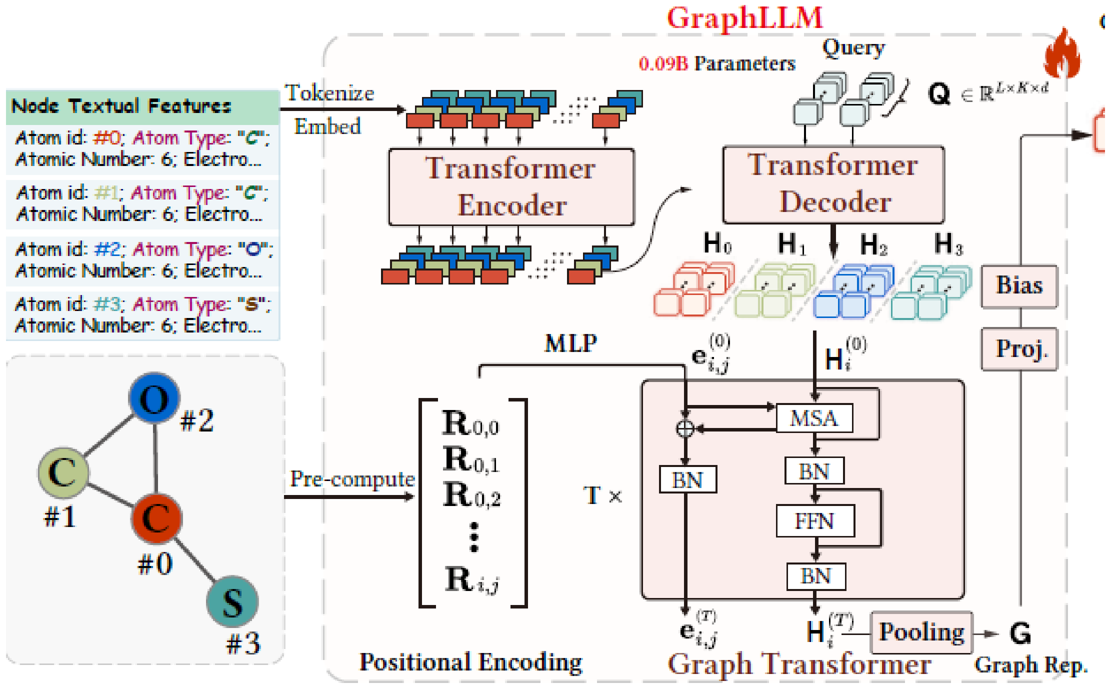
$$\boldsymbol{c}_i = \texttt{TransformerEncoder}(\boldsymbol{d}_i \boldsymbol{W}_{\mathrm{D}})$$
$$\mathbf{H}_i = \texttt{TransformerDecoder}(\mathbf{Q}; \boldsymbol{c}_i)$$

where $\boldsymbol{d}_i \in \mathbb{R}^{* \times d^{\mathrm{M}}}$ is the embeddings[3] of the textual description of node i ($*$ represents description's length). $\boldsymbol{W}_{\mathrm{D}} \in \mathbb{R}^{d^{\mathrm{M}} \times d}$ is a down-projection matrix to reduce the dimension. $\boldsymbol{c}_i \in \mathbb{R}^{* \times d}$ is node i's context vector and $\mathbf{H}_i \in \mathbb{R}^{L \times K \times d}$ is the node i' representation. $\mathbf{Q} \in \mathbb{R}^{L \times K \times d}$ is learnable query embedding, where L is the layer number of LLM transformer and K is the length of prefix.

- GraphLLM adopts lightweight encoder decoder – 0.05B parameters of LLaMA2 7B backbone

Graph Transformer for Structure Understanding

- Purpose is to effectively integrate and understand structural information of Graph.
- Core advantage of Graph Transformer over other graph learning modules is decoupling of node and structural information – empirically proves good.
- Positional Encoding initialized using random walk probabilities encoding.



$$\hat{e}_{i,j}^{(t)} = \sigma(\rho((W_{\mathrm{Q}} h_i^{(t)} + W_{\mathrm{K}} h_j^{(t)}) \odot W_{\mathrm{Ew}} e_{i,j}^{(t)}) + W_{\mathrm{Eb}} e_{i,j}^{(t)}) \in \mathbb{R}^d$$

$$\alpha_{ij} = \mathrm{Softmax}_{j\in\mathbb{V}}(W_{\mathrm{A}} \hat{e}_{i,j}^{(t)}) \in \mathbb{R}$$

$$h_i^{(t+1)} = \sum_{j\in\mathbb{V}} \alpha_{ij} \cdot W_{\mathrm{V}} h_j^{(t)} \in \mathbb{R}^d$$

Graph Enhanced Prefix Tuning for LLM

- LLM utilize graph enhanced tunable prefix derived from graph representation.
- Graph Enhanced tunable prefix ***P*** is obtained by applying linear projection to obtained graph representation.

$$\mathbf{P} = \mathbf{G}W_{\mathrm{U}} + \mathbf{B}$$

$W_{\mathrm{U}} \in \mathbb{R}^{d \times d^{\mathrm{M}}}$ is a matrix converting the dimension.

- **P** is prepended to each attention layer of LLM following the way prefix tuning is

Then $\mathbf{P} \in \mathbb{R}^{L \times K \times d^{\mathrm{M}}}$ is prepended to each attention layer of the LLM

- When **Wu = 0**, **P = B** which is vanilla Prefix Tuning. G is included while calculating P, therefore called Graph Enhanced Prefix Tuning.
- This way LLM synergize with graph transformer to incorporate additional context information crucial for graph reasoning task – by interpreting the contexts encapsulated in prefix.

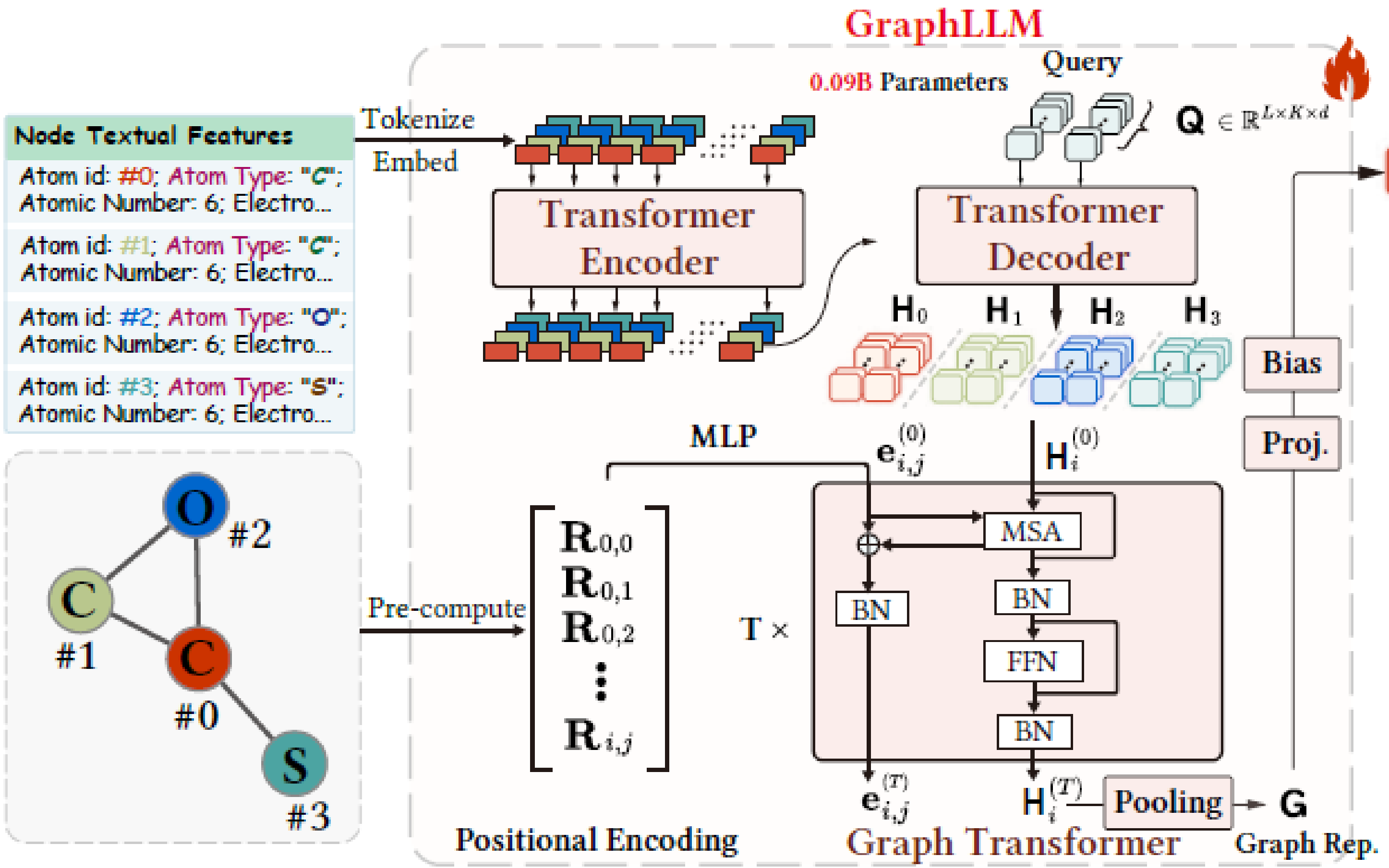


Figure 2: An illustration of reasoning on a toy molecular graph with GraphLLM. The LLM is requested to identify the number of C-C-O triangles in the molecule.

Experiments, Baselines & Results

Want to empirically substantiate -

Q1 – Does GraphLLM effectively enhance graph reasoning ability of LLM ?

Q2 – Can GraphLLM address the issue of lengthy context caused by Graph2Text strategy ?

Q3 – How GraphLLM perform in terms of computational efficiency ?

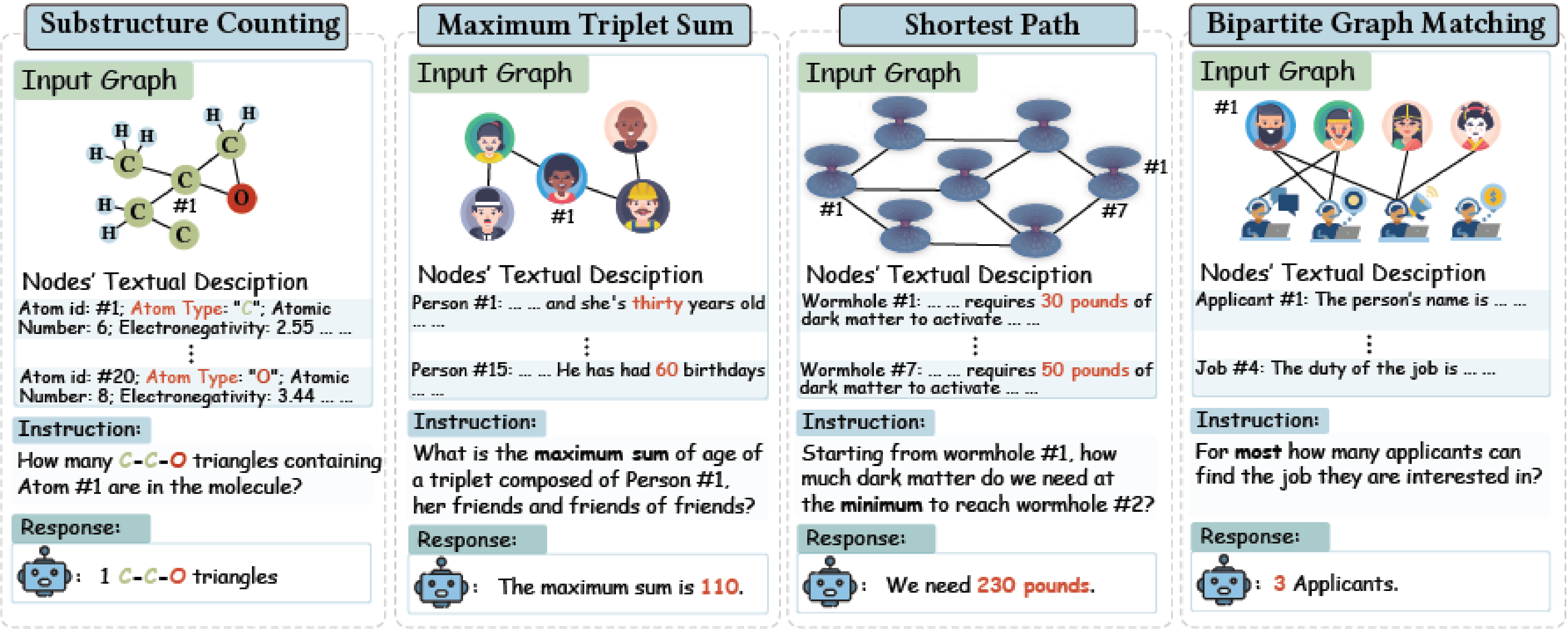
About Graph Reasoning Tasks

- Evaluation is done on 4 fundamental graph reasoning task –
  text substructure counting, maximum triplet sum, shortest path, and bipartite graph matching
- In each of the task, nodes are textual entity description of around 50 tokens.
- **Task 1** – ***Substructure Counting –*** *Given molecular graph, count the number of specific substructures e.g. carbon carbon oxygen triangle*
- **Task 2** – ***Maximum Triplet Sum –*** *Given friendship graph, each node is a person with description as age, find the maximum cumulative age among all possible triplets formed from a person, his/her direct friends and friends of those friends.*
- **Task 3** – ***Shortest Path –*** *Given graph of interconnected wormholes, compute the path from starting wormhole to destination wormhole with minimum energy. Each node has minimum energy to activate.*
- **Task 4** – ***Bipartite Graph Matching –*** *Graph depicts relationship between jobs and applicants. LLM required to compute maximum possible number of applicants who can get the job they are interested in.*

…Continued

Table 1: Statistics of the graph reasoning task datasets.

	Substructure Counting	Maximum Triplet Sum	Shortest Path	Bipartite Graph Matching
Avg. $\lvert\mathbb{V}\rvert$ / Avg. $\lvert\mathcal{E}\rvert$	15 / 22.3	15 / 26.6	20 / 32.4	20 / 14.0
No. of Tokens in Node Desc.	52-59	39-82	48-58	34-61



The textual descriptions of the nodes are generated by gpt-3.5-turbo according to specific instructions and manually verified.

Results

Table 2: Performance on Graph Reasoning Tasks. Shown is the mean ± s.d. of 3 runs with different random seeds. Highlighted are the top and second-best.

Input Format	Method	LLaMA2-7B				LLaMA2-13B			
		Substructure Counting	Maximum Triplet Sum	Shortest Path	Bipartite Graph Matching	Substructure Counting	Maximum Triplet Sum	Shortest Path	Bipartite Graph Matching
Adjacency List	Zero-shot	0.2260	0.1110	0.0000	0.3630	0.0145	0.0925	0.0010	0.1180
	Few-shot	0.2735	0.1445	0.0575	0.3280	0.2780	0.1430	0.0520	0.2675
	Few-shot CoT	0.2177	0.0585	0.1089	0.2399	0.2150	0.0544	0.1552	0.1048
	LoRA(attn)	$0.5012_{\pm.0054}$	$0.4427_{\pm.0031}$	$0.2119_{\pm.0004}$	$\mathbf{0.7383}_{\pm.1078}$	$0.4926_{\pm.0068}$	$0.4080_{\pm.0009}$	$0.1251_{\pm.0019}$	$0.7792_{\pm.0353}$
	LoRA(attn+ffn)	$\mathbf{0.5400}_{\pm.0363}$	$\mathbf{0.4723}_{\pm.0115}$	$0.1652_{\pm.0420}$	$0.6941_{\pm.0691}$	$\mathbf{0.4948}_{\pm.0035}$	$0.4274_{\pm.0459}$	$0.1181_{\pm.0051}$	$\mathbf{0.8010}_{\pm.0490}$
	Prefix Tuning	$0.5003_{\pm.0134}$	$0.3887_{\pm.0346}$	$\mathbf{0.2173}_{\pm.0078}$	$0.5534_{\pm.0739}$	$0.4610_{\pm.0444}$	$0.3377_{\pm.0038}$	$0.1608_{\pm.0376}$	$0.4640_{\pm.0314}$
Edge List (Random Order)	Zero-shot	0.2460	0.1260	0.0000	0.4325	0.0805	0.1265	0.0010	0.0055
	Few-shot	0.2610	0.1420	0.0111	0.3687	0.2655	0.1423	0.1110	0.3230
	Few-shot CoT	0.2127	0.0565	0.1069	0.1411	0.2320	0.0767	0.1351	0.0464
	LoRA(attn)	$0.5035_{\pm.0007}$	$0.4224_{\pm.0040}$	$0.2011_{\pm.0074}$	$0.6457_{\pm.0243}$	$0.4920_{\pm.0172}$	$0.4143_{\pm.0059}$	$0.1240_{\pm.0008}$	$0.6319_{\pm.0199}$
	LoRA(attn+ffn)	$0.5101_{\pm.0051}$	$0.4552_{\pm.0319}$	$0.2011_{\pm.0046}$	$0.5446_{\pm.0364}$	$0.4904_{\pm.0051}$	$\mathbf{0.4489}_{\pm.0157}$	$\mathbf{0.1958}_{\pm.0180}$	$0.6126_{\pm.0338}$
	Prefix Tuning	$0.3925_{\pm.0612}$	$0.3780_{\pm.0131}$	$0.1656_{\pm.0273}$	$0.4599_{\pm.0187}$	$0.3319_{\pm.1148}$	$0.3525_{\pm.0048}$	$0.1246_{\pm.0014}$	$0.5228_{\pm.0575}$
	GraphLLM	$\mathbf{0.9990}_{\pm.0007}$	$\mathbf{0.9577}_{\pm.0058}$	$\mathbf{0.9726}_{\pm.0011}$	$\mathbf{0.9981}_{\pm.0015}$	$\mathbf{0.9890}_{\pm.0021}$	$\mathbf{0.9392}_{\pm.0064}$	$\mathbf{0.9619}_{\pm.0038}$	$\mathbf{0.9934}_{\pm.0064}$

LLaMA 2 7B/13B used as backbone

Baselines

Analysis on Q1 -

Does GraphLLM effectively enhance graph reasoning ability of LLM ?

From the results, it is clear that –

- Zero shot, few shot and few shot Chain of Thoughts Graph2Text prompting methods deliver very low performance - need to fine tune LLM.

- Even with fine tuning, Graph2Text significantly lagged behind the performance achieved by GraphLLM – indicates that Graph2Text constitutes significantly obstacles for LLM.

- Choice between 2 graph representation does not lead to consistent enhancement in performance for Graph2Text – limitations of Graph2Text does not ties with graph representation.

- On average, GraphLLM achieves Exact Match accuracy of 98.19% over 4 tasks while Graph2Text manges to 47.35%.

… Continued

Using Stronger LLM

- Graph2Text is evaluated on more powerful gpt-3.5-turbo and gpt4.
- Results shown that even gpt4 fails on basic graph reasoning tasks.
- GraphLLM with LLaMA 2 7B as the backbone shows relative improvements of 2.61%, 99.8%, 12.22% and 15.16% compared to gpt4 few shot CoT on 4 fundamental task .

Table 3: Performance of `gpt-3.5-turbo` and `gpt-4` with *Graph2Text* strategy (converting input graph into adjacency list described in natural language), evaluated on 30 random samples due to the money cost.

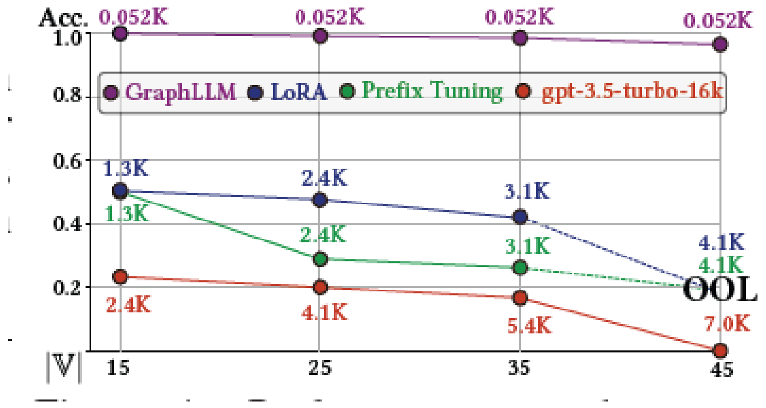
LLM	Method	Substructure Counting	Maximum Triplet Sum	Shortest Path	Bipartite Graph Matching
`gpt-3.5-turbo`	Zero-shot	0.2667	0.5667	0.2000	0.1000
	Few-shot	0.3000	0.3000	0.2667	0.0667
	Few-shot CoT	0.3667	0.7000	0.7333	0.2667
`gpt-4`	Zero-shot	0.6000	0.7333	0.6667	0.3333
	Few-shot	0.5000	0.8667	0.5667	0.5000
	Few-shot CoT	0.5000	0.9333	0.8667	0.8667
`LLaMA 2-7B`	GraphLLM	0.9990	0.9577	0.9726	0.9981

Analysis on Q2 -

Can GraphLLM address the issue of lengthy context caused by Graph2Text strategy ?

- GraphLLM reduces the context length by substantially by 96.45% across the 4 tasks.
- GraphLLM encodes both node and structure information into fixed length prefix (5 additional prefix token in implementation)
- In contrast, Graph2Text describe graph in text form and lead to extended context effecting the LLM ability on graph reasoning.
- It has been observed that on increasing the graph size, context length increases in Graph2Text and performance drops, but GraphLLM still maintains the accuracy – shows robustness.

Method	Avg. Context Length			
	Substructure Counting	Maximum Triplet Sum	Shortest Path	Bipartite Graph Matching
Zero-shot	1.3K / 1.3K	1.4K / 1.4K	1.8K / 1.7K	1.2K / 1.2K
Few-shot	2.6K / 2.5K	2.8K / 2.8K	3.1K / 2.9K	2.4K / 2.7K
Few-shot CoT	2.8K / 2.7K	3.0K / 2.9K	3.3K / 3.1K	2.5K / 2.8K
LoRA	1.3K / 1.3K	1.4K / 1.4K	1.8K / 1.7K	1.2K / 1.2K
Prefix Tuning	1.3K / 1.3K	1.4K / 1.4K	1.8K / 1.7K	1.2K / 1.2K
GraphLLM	0.040K (↓96.92%)	0.052K (↓96.29%)	0.048K (↓97.18%)	0.055K (↓95.42%)



Analysis on Q3 -

How GraphLLM perform in terms of computational efficiency ?

- GraphLLM achieves speedup of 3.42 times compared to best perfoming Graph2Text based method.

- Inference acceleration achieved by GraphLLM is due to context reduction as it surpasses the additional time overhead introduced.

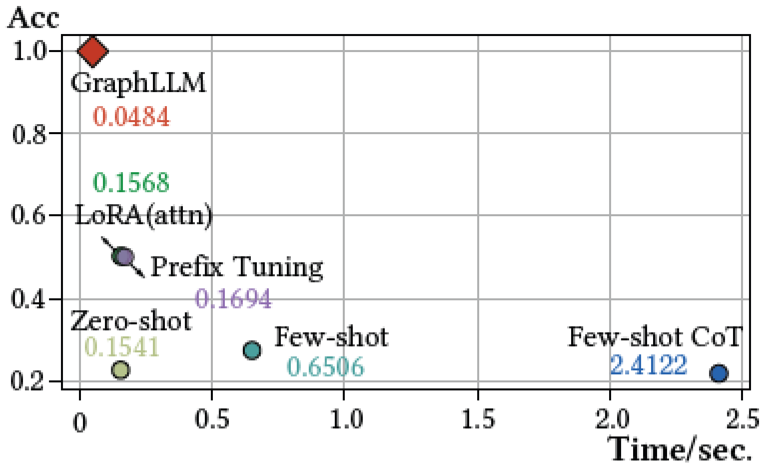


Figure 5: Avg. inference time on the sub-structure counting task on LLaMA 2 7B .

Thank You