TMLR

TRUSTWORTHY MACHINE LEARNING AND REASONING

# Recent Advances on LLM Reasoning with Graphs

Zhanke Zhou

PhD student @ TMLR Group, HKBU

2023 / 11 / 09

Personal Github: https://github.com/AndrewZhou924 Email: cszkzhou@comp.hkbu.edu.hk Wechat: zhouzhanke924

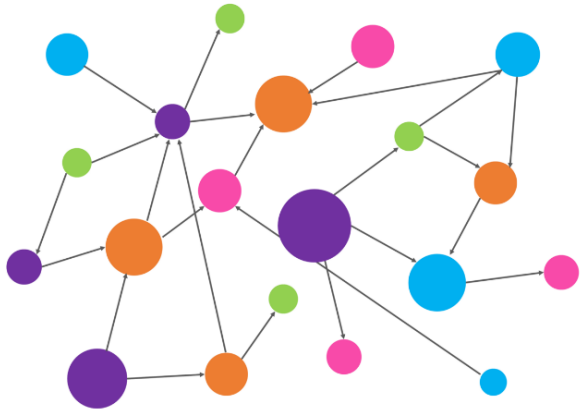
# Outlines

- Background
- An overview of LLMs & Graphs
- paper-1: Can Language Models Solve Graph Problems in Natural Language? 2023/05
- paper-2: GraphText: Graph Reasoning in Text Space. 2023/10
- paper-3: Large Language Models can Learn Rules. 2023/10
- Summary and Discussion

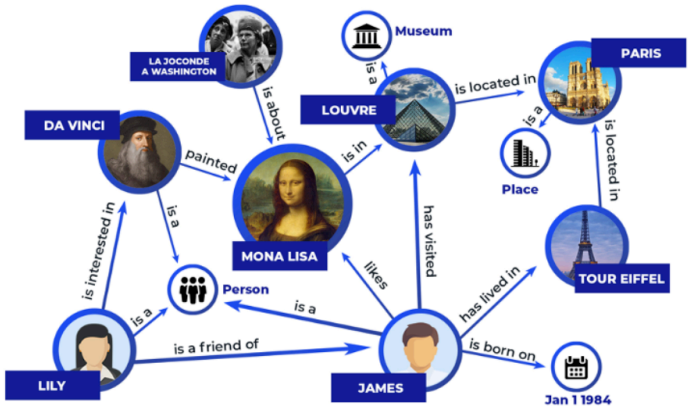
# Outlines

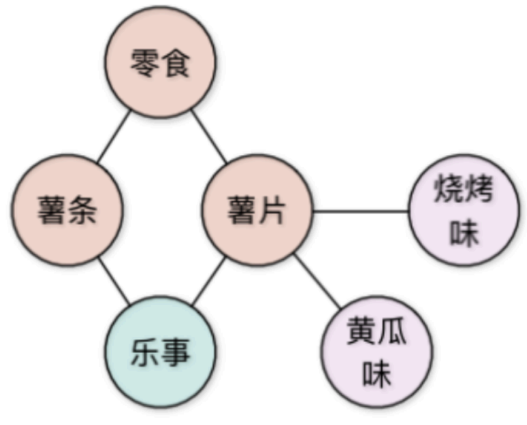
- Background (Learning on Graphs)
- An overview of LLMs & Graphs
- paper-1: Can Language Models Solve Graph Problems in Natural Language? 2023/05
- paper-2: GraphText: Graph Reasoning in Text Space. 2023/10
- paper-3: Large Language Models can Learn Rules. 2023/10
- Summary and Discussion

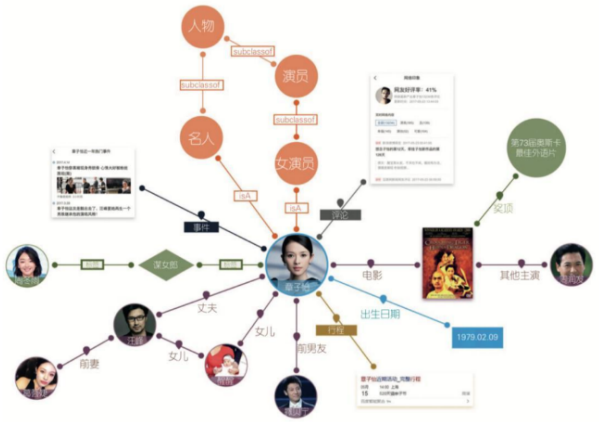
# Background

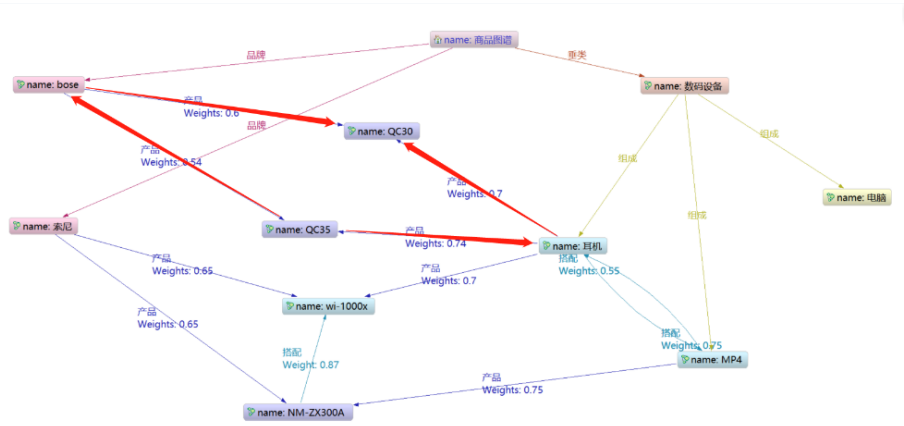


**Graph:** a general form of data expression



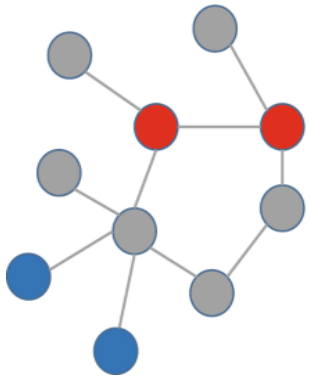



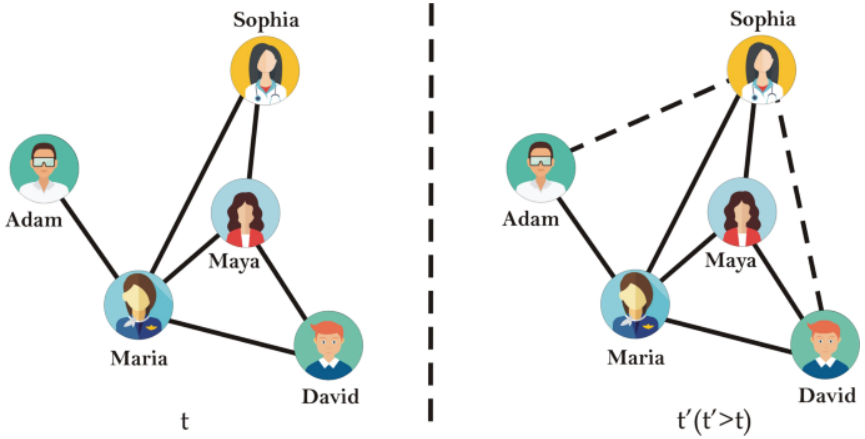



# Background

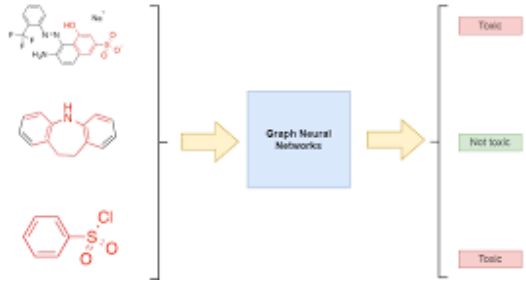
node-level



link-level

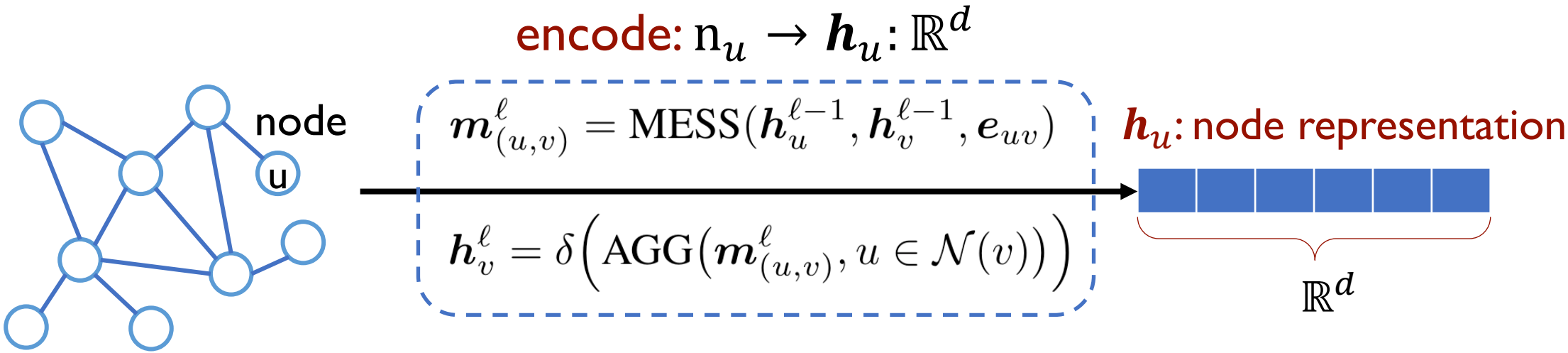



graph-level



# Background

GNN for learning on graphs



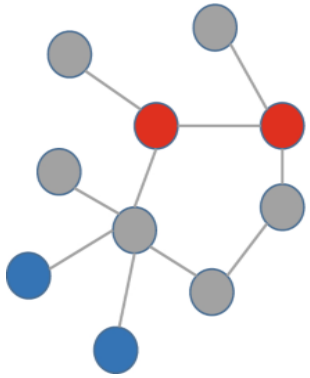

decode: $\boldsymbol{\phi}_{uv} = \mathrm{READOUT}(\boldsymbol{h}_u, \boldsymbol{h}_v) \rightarrow \mathbb{R}$

e.g.,
the link-level task

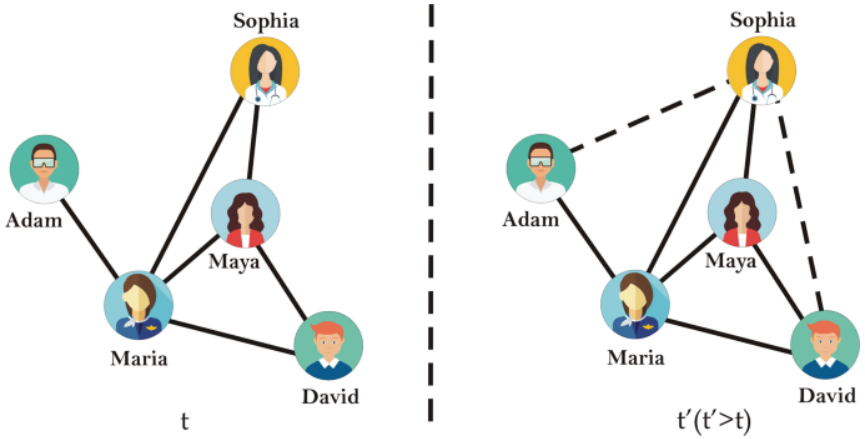
optimization: $\mathcal{L} = \sum_{e_{uv} \in \mathcal{E}^{train}} -y_{ij} \log(\boldsymbol{\phi}_{uv}) + (1 - y_{ij}) \log(1 - \boldsymbol{\phi}_{uv})$

# Background

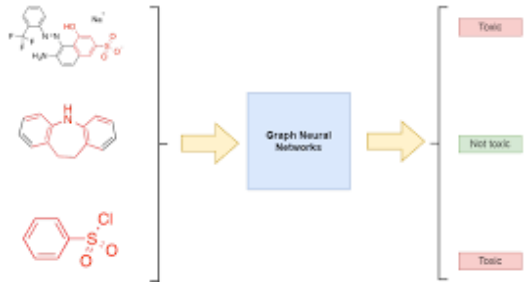
node-level



**link-level**




graph-level




**Key challenges (bottlenecks) of graph learning**

- Generalization (to test-time graphs)
- Expressiveness and Data heterogeneity (structural / semantical heterogeneity)
- Robustness (to noisy/adversarial examples) / Scalability (prone to the scale of input graph)

# Outlines

- Background
- An overview of LLMs & Graphs
  - Integrating Graphs with Large Language Models: Methods and Prospects. Arxiv 2023.10.
  - Exploring the Potential of Large Language Models (LLMs) in Learning on Graphs. Arxiv 2023.08.
- paper-1: Can Language Models Solve Graph Problems in Natural Language? 2023/05
- paper-2: GraphText: Graph Reasoning in Text Space. 2023/10
- paper-3: Large Language Models can Learn Rules. 2023/10
- Summary and Discussion

# Overview

## Integrating Graphs with Large Language Models: Methods and Prospects

Shirui Pan, *Griffith University, Gold Coast, QLD 4215, Australia*
Yizhen Zheng & Yixin Liu, *Monash University, Melbourne, VIC 3800, Australia*

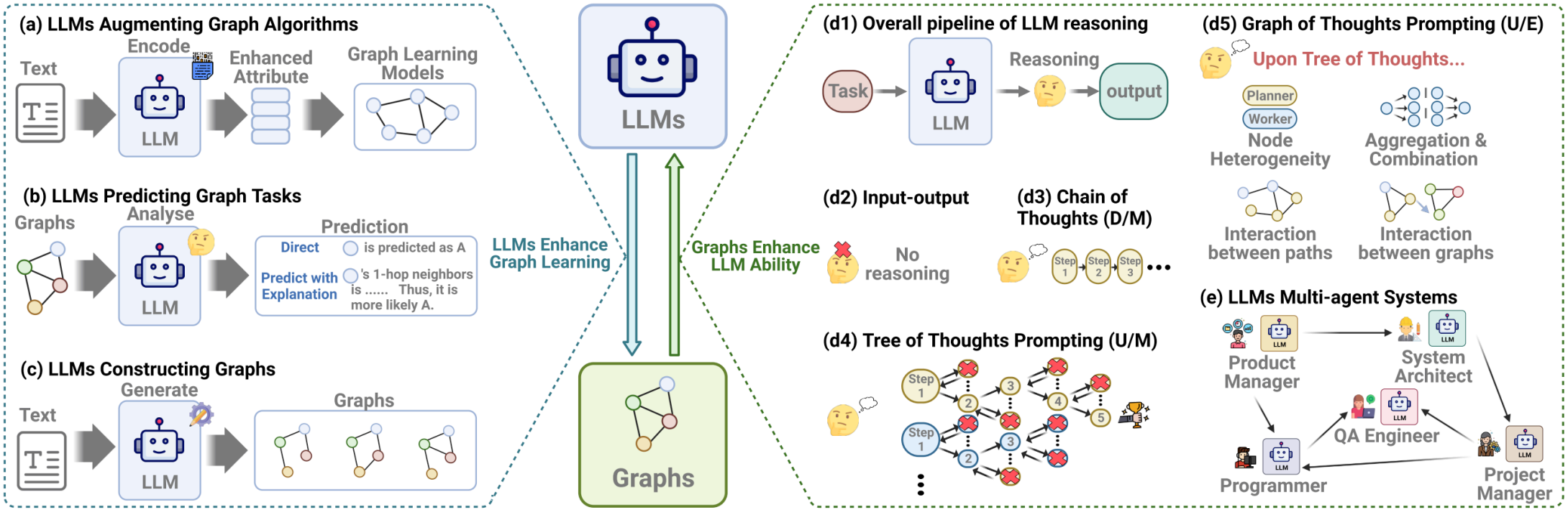
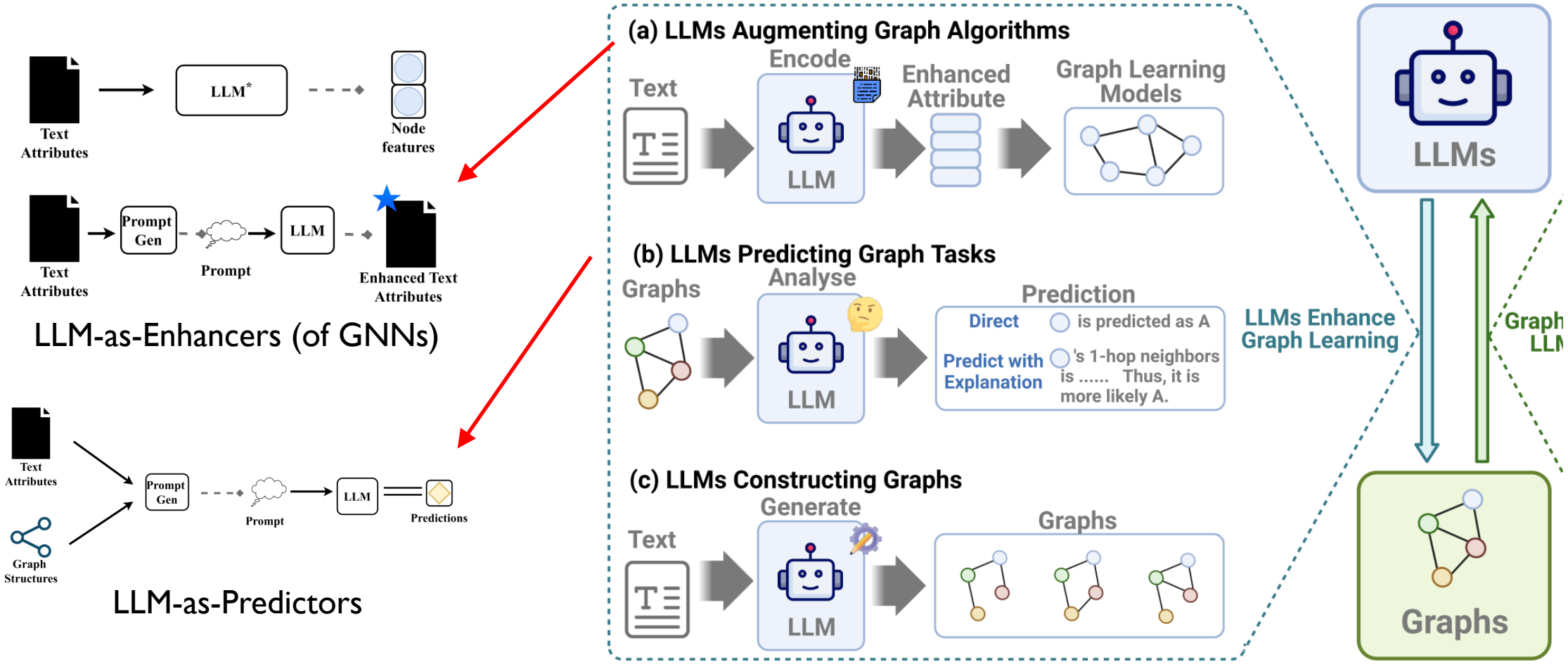



**FIGURE 1.** The overall framework of the mutual enhancement between LLMs and Graphs. (a)-(c): three pathways for LLMs to enhance graph learning. (d)-(e): techniques for graph structures enhancing LLM reasoning. Brackets after technique names indicate graph types. D, U, M and E represent directed, undirected, homogeneous and heterogeneous graphs, respectively.



Integrating Graphs with Large Language Models: Methods and Prospects. Arxiv 2023.10.

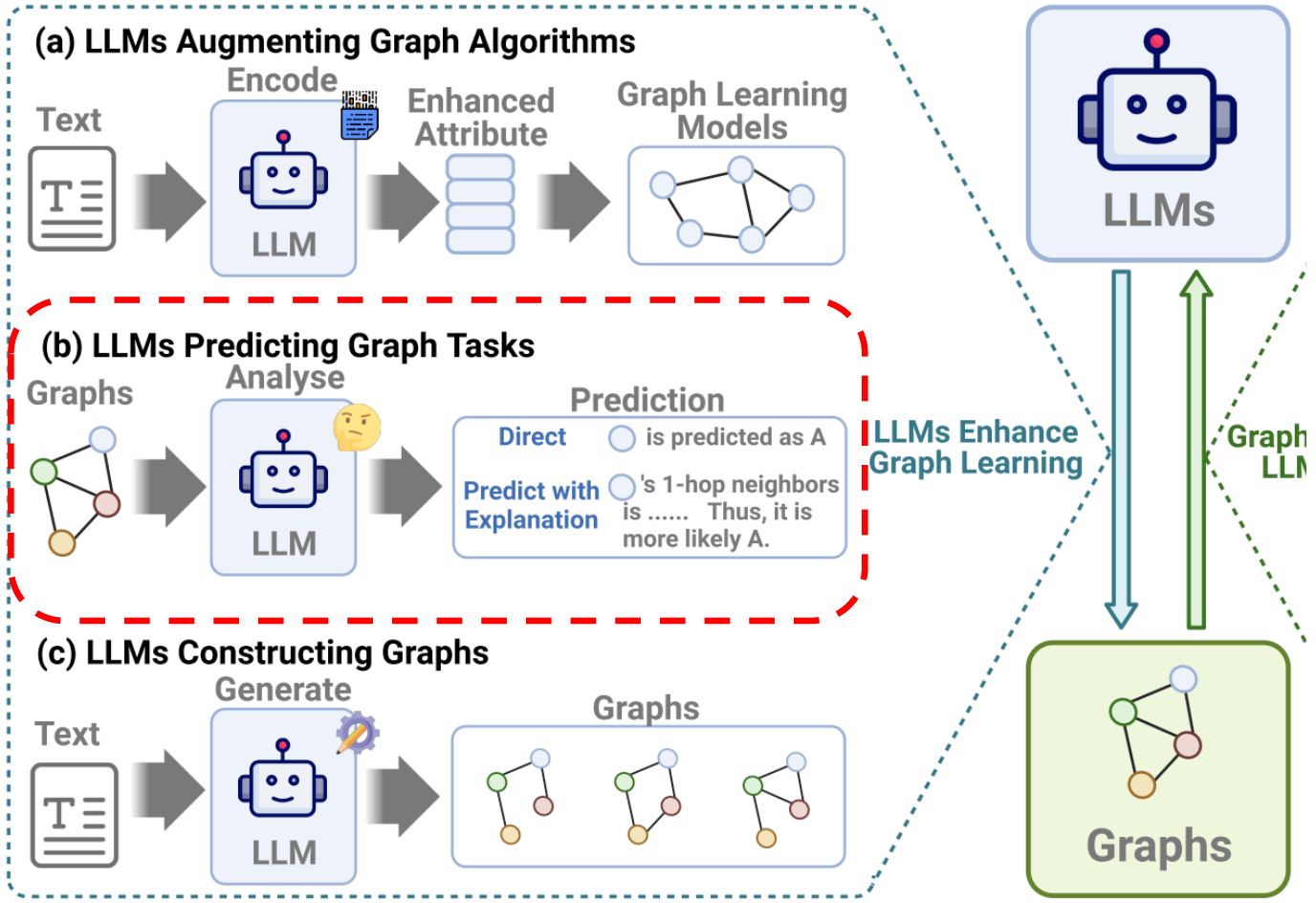
# Overview | LLMs Enhancing Graph Learning




Exploring the Potential of Large Language Models (LLMs) in Learning on Graphs. Arxiv 2023.08.
Integrating Graphs with Large Language Models: Methods and Prospects. Arxiv 2023.10.

# Overview | LLMs Enhancing Graph Learning

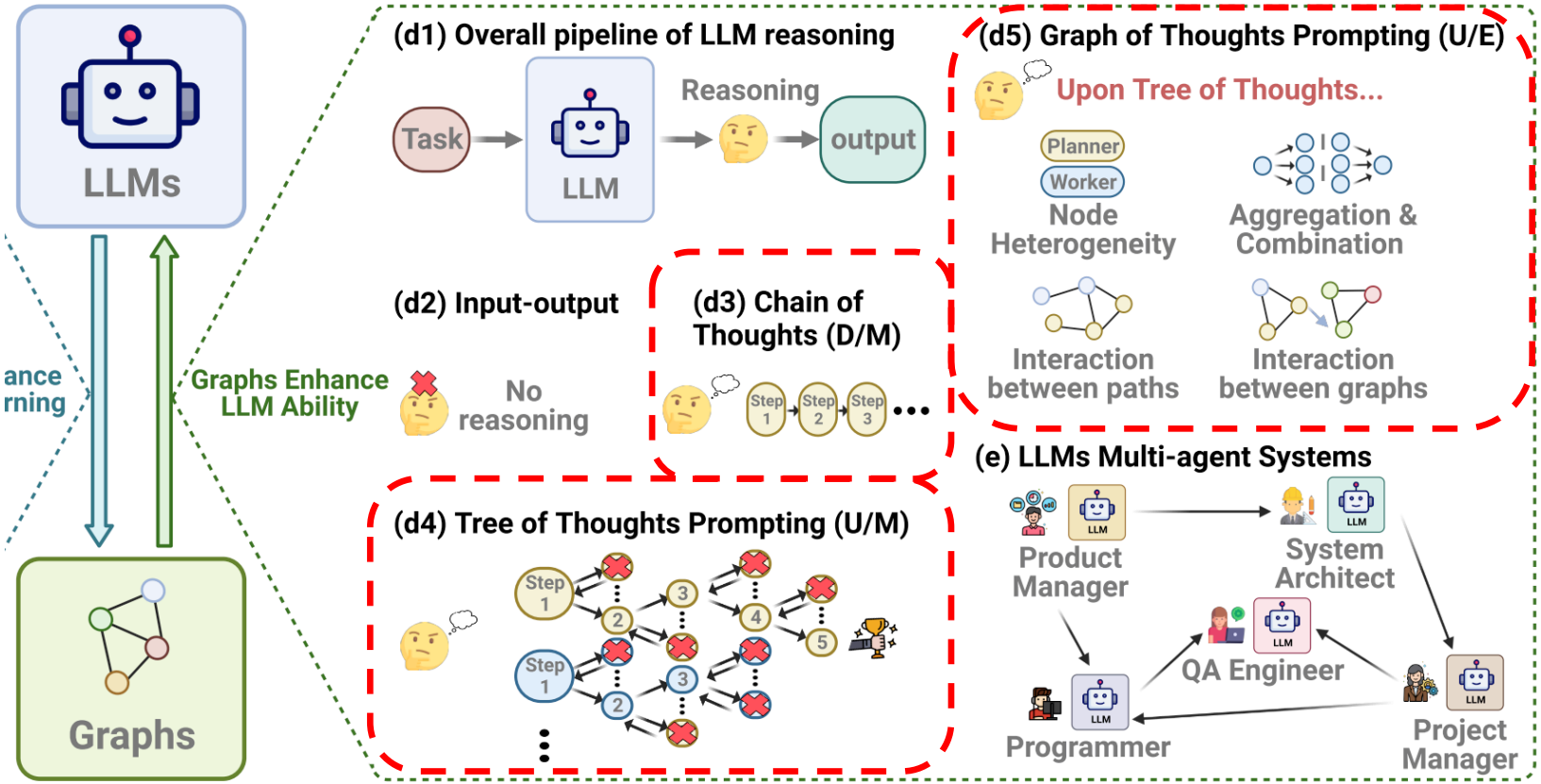
e.g.,
NLGraph (paper-1)
GRAPHTEXT (paper-2)






Integrating Graphs with Large Language Models: Methods and Prospects. Arxiv 2023.10.

# Overview | Graphs Enhance LLM Ability




e.g.,
Chain-of-thought,
Tree-of-thought,
Graph-of-thought,
Hypotheses-to-Theories (paper-3),
etc.

Integrating Graphs with Large Language Models: Methods and Prospects. Arxiv 2023.10.

# Outlines

- Background
- An overview of LLMs & Graphs
- paper-1: Can Language Models Solve Graph Problems in Natural Language? 2023/05
- paper-2: GraphText: Graph Reasoning in Text Space. 2023/10
- paper-3: Large Language Models can Learn Rules. 2023/10
- Summary and Discussion

# About the paper | NeurIPS 2023 (Spotlight)

## Can Language Models Solve Graph Problems in Natural Language?

**Heng Wang**[*1], **Shangbin Feng**[*2], **Tianxing He**[2], **Zhaoxuan Tan**[3], **Xiaochuang Han**[2], **Yulia Tsvetkov**[2]
[1]Xi'an Jiaotong University [2]University of Washington [3]University of Notre Dame
wh2213210554@stu.xjtu.edu.cn, shangbin@cs.washington.edu

The investigated problem:

Can ***LLMs*** solve ***graph algorithm problems*** in ***natural language?***

- NLGraph Benchmark → the base of evaluation
- Experiments → evaluating LLMs
- Instruction-based Approaches → enhance LLMs

- NLGraph Benchmark
- Experiments
- Instruction-based Approaches

# NLGraph Benchmark | overview

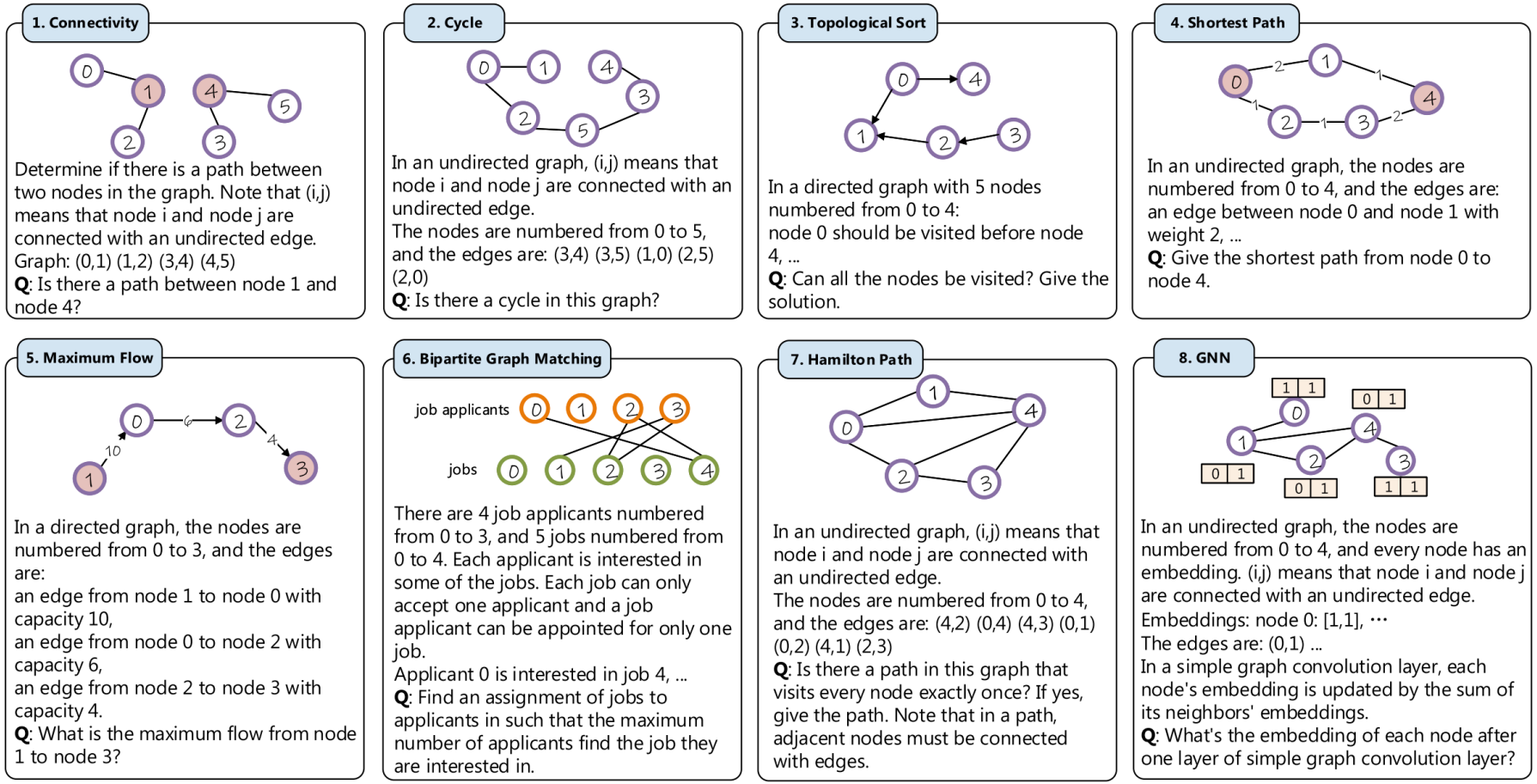
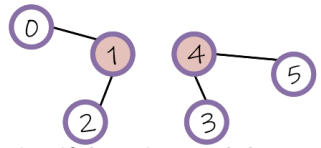



Figure 1: Overview of the NLGraph Benchmark, featuring eight tasks with varying complexity. We show an intuitive figure representing each task along with the example natural language prompts being passed to the LLMs.

# NLGraph Benchmark

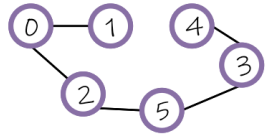
- **Task 1: Connectivity** In an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, two nodes $u$ and $v$ are *connected* if there exists a sequence of edges from node $u$ to node $v$ in $\mathcal{E}$. We randomly select two nodes in the base graphs $u, v \in \mathcal{V}$ to ask whether node $u$ and node $v$ are connected with a true/false question. We retain a balanced set of questions where half of the node pairs are connected and the other half are not connected by discarding additional questions.
- **Task 2: Cycle** In an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, a cycle is a non-empty trail $(e_1, e_2, \ldots, e_n)$ with a node sequence $(v_1, v_2, \ldots, v_n, v_1)$. We present an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ to ask whether there exists a cycle through true/false questions. We retain base graphs without cycles as the False subset, while we randomly add edges to these base graphs to generate graphs with cycles as the True subset. We retain a balanced set of cyclic and noncyclic graphs in the dataset.
- **Task 3: Topological Sort** A topological sort of a directed graph is a linear ordering of its nodes such that for every directed edge $(u, v)$ from node $u$ to node $v$, $u$ comes before $v$ in the ordering. The task is to find a valid topological sort given a directed graph and there could be multiple valid solutions. We ask LLMs to generate a valid topological sort for the given directed graph and employ an external program to examine its correctness.

**1. Connectivity**



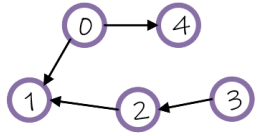

Determine if there is a path between two nodes in the graph. Note that (i,j) means that node i and node j are connected with an undirected edge.
Graph: (0,1) (1,2) (3,4) (4,5)
**Q**: Is there a path between node 1 and node 4?

**2. Cycle**




In an undirected graph, (i,j) means that node i and node j are connected with an undirected edge.
The nodes are numbered from 0 to 5, and the edges are: (3,4) (3,5) (1,0) (2,5) (2,0)
**Q**: Is there a cycle in this graph?

**3. Topological Sort**

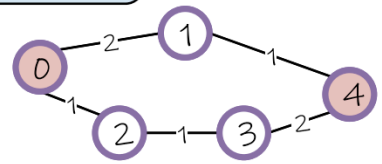



In a directed graph with 5 nodes numbered from 0 to 4:
node 0 should be visited before node 4, ...
**Q**: Can all the nodes be visited? Give the solution.

# NLGraph Benchmark

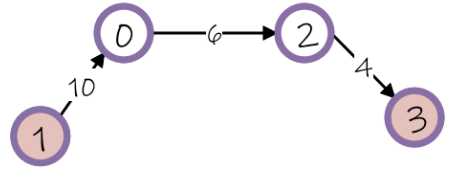
- **Task 4: Shortest Path** The shortest path between two nodes is the path with the sum of edge weights minimized. Given an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, a positive weight $w$ for each edge, and two nodes $u$ and $v$, the task is to find the shortest path between node $u$ and node $v$ and its corresponding path length. We filter the generated base graphs by specifying that the number of nodes on the correct shortest path is as least $\ell$, where $\ell$ is chosen from $\ell_{min}$ to $\ell_{max}$ for each question to sever as an additional difficulty control measure. We adopt two metrics: *exact match*, where the LLM solution is a valid path and optimal, and *partial credit*, where the LLM solution is valid and is the $\text{rank}_i$-th shortest among all the possible paths. The partial credit score for each problem can be formulated as $\text{PC} = 1/\text{rank}_i$.
- **Task 5: Maximum Flow** Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be a network with two nodes $s, t \in \mathcal{V}$ being the source and the sink. Each edge is associated with a capacity $c$, the maximum amount of flow that can pass through the edge. We ask LLMs to generate a plan to route as much flow as possible from the source to the sink. We evaluate in both exact match with the optimal plan and partial credit for this task, where partial credit can be formulated as $\text{PC} = \begin{cases} t/s, & \text{if } t \leq s \\ 0, & \text{if } t > s \end{cases}$, where $s$ is the flow value under the optimal plan, and $t$ is the flow value of the solution generated by LLMs.

**4. Shortest Path**




In an undirected graph, the nodes are numbered from 0 to 4, and the edges are:
an edge between node 0 and node 1 with weight 2, ...
**Q**: Give the shortest path from node 0 to node 4.

**5. Maximum Flow**

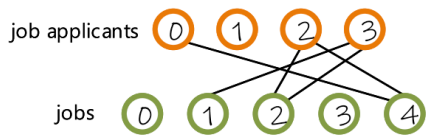



In a directed graph, the nodes are numbered from 0 to 3, and the edges are:
an edge from node 1 to node 0 with capacity 10,
an edge from node 0 to node 2 with capacity 6,
an edge from node 2 to node 3 with capacity 4.
**Q**: What is the maximum flow from node 1 to node 3?

# NLGraph Benchmark

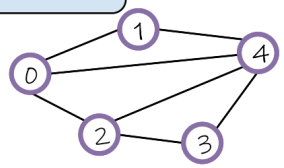
- **Task 6: Bipartite Graph Matching** In an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, a matching is a set of edges without common nodes. A bipartite graph is a graph whose nodes can be divided into two disjoint sets $\mathbf{U}$ and $\mathbf{V}$, and in each set no nodes are adjacent to each other. Given a bipartite graph, the task is to find the matching that maximizes the number of edges. We use an external program to evaluate whether the solution generated by LLMs is valid and optimal.
- **Task 7: Hamilton Path** In an undirected graph, a Hamilton path is a path that visits every node exactly once. Given an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, the task is to find a valid Hamilton path. We filter generated base graphs to ensure that at least one valid Hamilton path exists and use an external program to evaluate the LLM solution.
- **Task 8: Graph Neural Networks** Given an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, and a two-dimension node embedding $\mathbf{x}_i$ for each node, the task is to perform $\ell$ layers of message passing, *i.e.* to update the node embedding with the sum of all the neighbors' embeddings. Formally, $\mathbf{x}_i^{(\ell+1)} = \sum_{j \in \mathcal{N}_i} \mathbf{x}_j^{(\ell)}$ where $\mathcal{N}_i$ denotes the neighborhood of node $i$ and $(\ell)$ denotes the $\ell$-th layer. We use an exact match with the correct node embeddings and two types of partial credits for this task. Specifically, the first partial credit is the percentage of the nodes whose embedding is correct (PC), and the second is the average of all the values' relative errors for the standard answer (RE). The relative error is formulated as $RE = \frac{|x-y|}{\max(x,y)}$, where $x$ is the value generated by LLMs and $y$ is the value in the standard answer, averaged across all embedding dimensions.

**6. Bipartite Graph Matching**



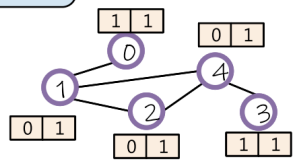

There are 4 job applicants numbered from 0 to 3, and 5 jobs numbered from 0 to 4. Each applicant is interested in some of the jobs. Each job can only accept one applicant and a job applicant can be appointed for only one job.
Applicant 0 is interested in job 4, ...
**Q**: Find an assignment of jobs to applicants in such that the maximum number of applicants find the job they are interested in.

**7. Hamilton Path**




In an undirected graph, (i,j) means that node i and node j are connected with an undirected edge.
The nodes are numbered from 0 to 4, and the edges are: (4,2) (0,4) (4,3) (0,1) (0,2) (4,1) (2,3)
**Q**: Is there a path in this graph that visits every node exactly once? If yes, give the path. Note that in a path, adjacent nodes must be connected with edges.

**8. GNN**




In an undirected graph, the nodes are numbered from 0 to 4, and every node has an embedding. (i,j) means that node i and node j are connected with an undirected edge.
Embeddings: node 0: [1,1], ⋯
The edges are: (0,1) ...
In a simple graph convolution layer, each node's embedding is updated by the sum of its neighbors' embeddings.
**Q**: What's the embedding of each node after one layer of simple graph convolution layer?

# NLGraph Benchmark | data generation

NLGraph benchmark is with
- 5,902 problems in a standard version
- 29,370 problems in an extended version

### 2.1 Random Graph Generator

We first employ a general-purpose random graph generator to generate base graphs while using the number of nodes and graph density to control for complexity. Formally, to generate a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ where $\mathcal{V}$ and $\mathcal{E}$ denote the set of nodes and edges, we specify the number of nodes $n$, thus $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$, and $|\mathcal{V}| = n$. We then specify the edge probability $p$ such that all the edges are generated according to $P(e_{ij} \in \mathcal{E}) = p$, where $e_{ij}$ is an edge from $v_i$ to $v_j$. The edges could be directed or undirected depending on the task. We use varying $n$ and $p$ values to control for the complexity of the random graph structure. Building on top of these generated base graphs, we also adopt graph edits and other difficulty control factors tailored for each task.

"easy" tasks

"hard" tasks

| Subset | Connect. | Cycle | Topo. Sort | Shortest Path | Max. Flow | Bipartite Graph | Hamilton Path | GNNs |
|---|---|---|---|---|---|---|---|---|
| # EASY | 352 / 730 | 150 / 300 | 180 / 360 | 180 / 360 | 150 / 300 | 300 / 600 | 150 / 300 | 100 / 200 |
| SPEC. | $n$: 5-10 | $n$: 5-10 | $n$: 5-10 | $n$: 5-10 | $n$: 5-10 | $n$: 6-20 | $n$: 5-10 | $n$: 5-8 |
| # MEDIUM | 1,200 / 8,580 | 600 / 1,800 | 150 / 1,350 | / | / | / | / | / |
| SPEC. | $n$: 11-25 | $n$: 11-25 | $n$: 11-25 | / | / | / | / | / |
| # HARD | 680 / 7,090 | 400 / 2,000 | 200 / 1,200 | 200 / 1,200 | 200 / 1,200 | 210 / 1,260 | 200 / 600 | 140 / 840 |
| SPEC. | $n$: 26-35 | $n$: 26-35 | $n$: 26-35 | $n$: 11-20 | $n$: 11-20 | $n$: 17-33 | $n$: 11-20 | $n$: 9-15 |

Table 1: Statistics of the NLGraph benchmark. We use A / B to indicate that there are A and B problems in the standard and extended set of NLGraph. SPEC. denotes difficulty specifications.

# Experiment | Setup

- NLGraph Benchmark
- Experiments
- Instruction-based Approaches

**Baselines** We adopt a wide range of prompting approaches as baselines. Specifically, zero-shot prompting (ZERO-SHOT), few-shot in-context learning (FEW-SHOT) [Brown et al., 2020], chain-of-thought prompting (COT) [Wei et al., 2022], zero-shot chain-of-thought (0-COT) [Kojima et al., 2022], least-to-most (LTM) [Zhou et al., 2023], and self-consistency (SC) [Wang et al., 2023] are leveraged to tackle various graph reasoning tasks in the NLGraph benchmark.

We also adopt a RANDOM baseline: for true/false questions such as connectivity and cycle, we use RANDOM to denote a baseline that randomly selects an answer from true and false with an expected accuracy of 50%; For the shortest path task, RANDOM denotes a baseline that randomly selects a valid path between the query node pair. For the maximum flow task, RANDOM denotes a baseline that randomly selects a value between 0 and the sum of all the edges' capacities. The performance comparison between different prompting techniques and the RANDOM baseline could indicate whether LLMs are capable of performing graph reasoning instead of giving randomly generated answers.

**Models and Settings** We use TEXT-DAVINCI-003 as the default large language model, while we also evaluate three additional LLMs (GPT-3.5-TURBO, CODE-DAVINCI-002, and GPT-4), presenting results in Appendix E. For all baselines except self-consistency, we set temperature $\tau = 0$; For self-consistency prompting, we sample five chain-of-thought responses with temperature $\tau = 0.7$. For few-shot prompting techniques (i.e., FEW-SHOT, COT, LTM, and COT+SC), the input prompt includes $k$ exemplars selected from the extended version before the problem of interest. For the connectivity task and cycle task, we set $k$ to 4, for the GNN task, we set $k$ to 1 due to the context size limit, while for other tasks $k$ is 5. We evaluate LLMs and various prompting techniques on the standard set of NLGraph due to monetary costs, while we encourage future research to leverage the extended version for enhanced evaluation.

# Experiment

Random < Zero-shot < Few-shot < **COT**

| Method | Connectivity | | | | Cycle | | | | Shortest Path | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Easy | Medium | Hard | Avg. | Easy | Medium | Hard | Avg. | Easy | Hard | Easy (PC) | Hard (PC) | Avg. |
| RANDOM | 50.00 | 50.00 | 50.00 | 50.00 | 50.00 | 50.00 | 50.00 | 50.00 | 6.07 | 6.69 | 14.73 | 13.81 | 17.81 |
| ZERO-SHOT | 83.81 | 72.75 | 63.38 | 71.31 | 50.00 | 50.00 | 50.00 | 50.00 | 29.40 | 21.00 | 46.00 | 26.76 | 30.79 |
| FEW-SHOT | 93.75 | 83.83 | 76.61 | 84.73 | 80.00 | **70.00** | **61.00** | **70.33** | 31.11 | 26.00 | 49.19 | 35.73 | 35.51 |
| COT | **94.32** | 82.17 | 77.21 | 84.57 | **84.67** | 63.33 | 53.25 | 66.75 | 63.89 | **29.50** | 76.84 | 35.79 | 51.51 |
| 0-COT | 79.55 | 65.83 | 68.53 | 71.30 | 55.33 | 57.67 | 49.00 | 54.00 | 8.89 | 7.50 | 62.39 | **43.95** | 32.03 |
| COT+SC | 93.18 | **84.50** | **82.79** | **86.82** | 82.00 | 63.67 | 53.50 | 66.39 | **68.89** | 29.00 | **80.25** | 38.47 | **54.15** |

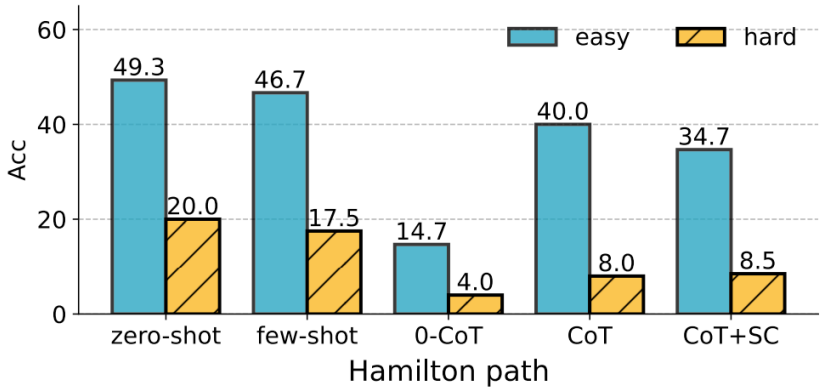
| Method | PC (↑) | ACC (↑) | RE (↓) |
|---|---|---|---|
| ZERO-SHOT | 13.61 | 0.00 | 20.04 |
| FEW-SHOT | 20.04 | 0.00 | 37.83 |
| COT | **64.55** | **31.00** | 14.34 |
| 0-COT | 13.85 | 0.00 | 44.55 |
| COT+SC | 63.92 | 28.00 | **13.28** |

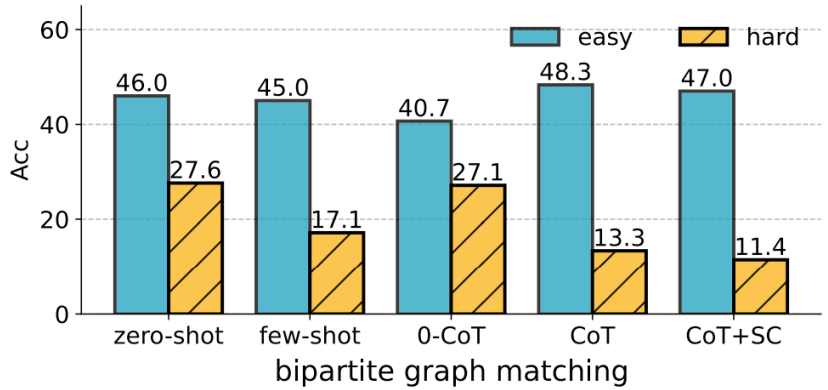
Model performance on the task of simulating graph neural networks.
PC and RE are two partial credit metrics.

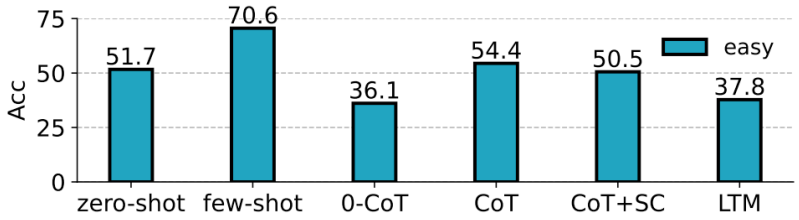
➔ LLMs Have (Preliminary) Graph Reasoning Abilities

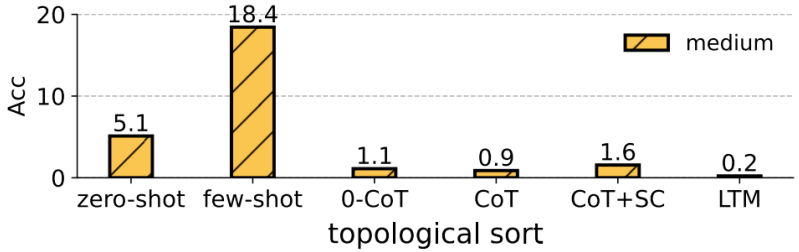
# Experiment

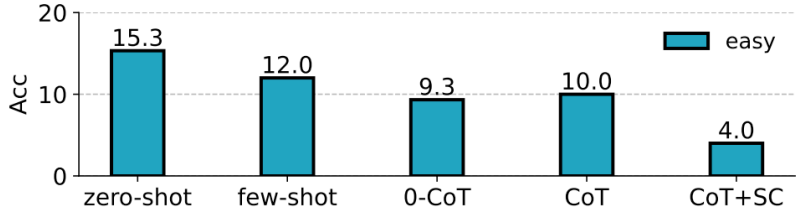
**In-Context Learning Can be Counterproductive**
➔ LLMs fall short of generating valid intermediate steps to solve the more complex graph reasoning problem

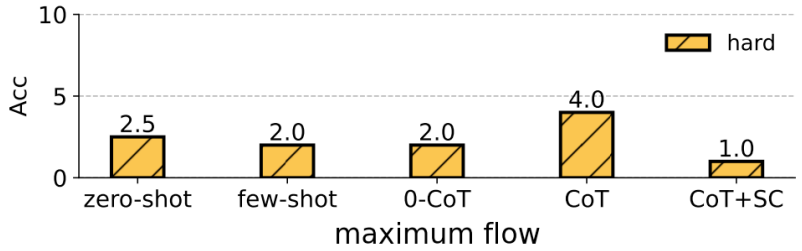


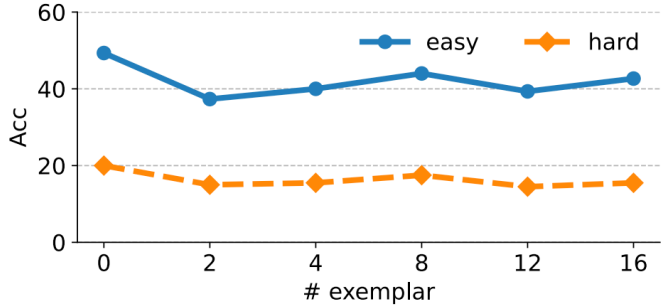




Figure 4: Model performance on the Hamilton path task with an increasing amount of exemplars. When more in-context exemplars are introduced, model performance is not trending up and is consistently lower than zero-shot prompting in both difficulty settings.

# Experiment

Two variant tasks of the connectivity task

**Chain** We firstly generate a graph with $k$ components, where each components is a chain. Formally, $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ can be divided into subgraphs $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k$, where subgraph $\mathcal{G}_i$ consists of a chain of nodes and edges $v_{i1}, e_{i1}, v_{i2}, e_{i2}, \ldots, e_{it_i}, v_{it_i}$. We then randomly select query node pairs that are at the two ends of a chain, *i.e.* $v_{i1}$ and $v_{it_i}$. These nodes only have a degree of one but they are actually connected through the chain structure. We curate a chain dataset with 120 examples.

**Clique** We first generate a graph with $k$ densely connected subgraphs. Formally, $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ can be divided into subgraphs $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k$, where subgraph $\mathcal{G}_i$ is generated by the random graph generator (§2.1) with a high edge probability $p \in \{0.7, 1.0\}$. We then randomly select query node pairs with each pair in two different densely connected subgraphs, $\mathcal{G}_i$ and $\mathcal{G}_j$ $(i \neq j)$. These nodes are associated with a large number of edges and thus are frequently mentioned in the natural language prompt, but they belong to two different subgraphs and are not connected. We curate a clique dataset with 120 examples.

| **Dataset** | ZERO-SHOT | FEW-SHOT | COT | 0-COT | COT+SC | Avg. |
|---|---|---|---|---|---|---|
| GENERAL | 74.67 | 83.33 | 85.33 | 66.00 | 82.67 | 78.40 |
| CHAIN | 51.67 (-23.00) | 45.00 (-35.33) | 40.83 (-44.50) | 92.50 (+26.50) | 44.17 (-38.50) | 54.83 (-23.57) |
| CLIQUE | 60.83 (-13.84) | 73.33 (-10.00) | 85.00 (-0.33) | 52.50 (-13.50) | 83.33 (+0.66) | 71.00 (-7.40) |

Table 4: Model performance on the chain and clique subset of the connectivity task. Large language models indeed rely on spurious correlations in problem settings, evident in the reduced performance on the two special cases compared to the general connectivity task.

LLMs are indeed vulnerable to spurious correlations in structured reasoning
(But, it that real? 🤔)

- NLGraph Benchmark
- Experiments
- Instruction-based Approaches

# Instruction-based Approaches

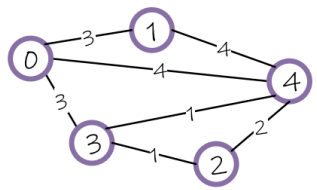
### 5.1 Methodology

We propose two instruction-based prompting techniques that improve the graph reasoning ability of LLMs, which can be used together with in-context learning and chain-of-thought prompting.

**Build-a-Graph Prompting (BAG)** We hypothesize that it might be helpful to map the textual descriptions of graphs to grounded conceptual spaces [Patel et al., 2021] before tackling the specific problem, *i.e.* visualizing the graph structure on an implicit mental sketchpad. To this end, we propose to use instruction-based prompting by appending "*Let's construct a graph with the nodes and edges first.*" after the textual description of the graph is explicitly given. We envision this straightforward instruction would provide LLMs with the buffer zone to digest graph information, map them to grounded conceptual spaces, and better prepare for the incoming query.

**Algorithmic Prompting** We hypothesize that in order to generate sound and accurate solutions, LLMs would benefit from revisiting and reciting the relevant algorithm for a given task [Sun et al., 2022]. To this end, we propose to first prepend the algorithm details before the in-context examples by adding "We can use a Depth-First Search (DFS) algorithm ..." for the shortest path task. We similarly provide the algorithm descriptions of other graph reasoning tasks in their respective prompts. We envision algorithmic prompting as empowering LLMs with a general understanding of how to solve the problem before actually solving it.

**Standard Prompting**




<in-context exemplar>
In an undirected graph, the nodes are numbered from 0 to 4, and the edges are:
an edge between node 0 and node 4 with weight 4,
an edge between node 0 and node 3 with weight 3,
an edge between node 0 and node 1 with weight 3,
...
**Q**: Give the shortest path from node 0 to node 2.

**Build-a-Graph Prompting**

<in-context exemplar>
In an undirected graph, the nodes are numbered from 0 to 4, and the edges are:
an edge between node 0 and node 4 with weight 4, ⋯

*Let's construct a graph with the nodes and edges first.*

**Q**: Give the shortest path from node 0 to node 2.
**A**: All the paths from node 0 to node 2 are:
0,3,2 with a total weight of 3 + 1 = 4,
0,1,4,2 with a total weight of 3 + 4 + 2 = 9,
0,4,3,2 with a total weight of 4 + 1 + 1 = 6.
The weight of path 0,3,2 is the smallest, so the shortest path from node 0 to node 2 is 0,3,2 with a total weight of 4.

**Algorithmic Prompting**

*We can use a Depth-First Search (DFS) algorithm to find the shortest path between two given nodes in an undirected graph.*
*The basic idea is to start at one of the nodes and use DFS to explore all of its adjacent nodes. At each node, you can keep track of the distance it takes to reach that node from the starting node.*
*Once you have explored all the adjacent nodes, you can backtrack and pick the node which has the shortest distance to reach the destination node.*
<in-context exemplar>
In an undirected graph, the nodes are numbered from 0 to 4, and the edges are:
an edge between node 0 and node 4 with weight 4, ...
**Q**: Give the shortest path from node 0 to node 2.
**A**: All the paths from node 0 to node 2 are:
0,3,2 with a total weight of 3 + 1 = 4,
0,1,4,2 with a total weight of 3 + 4 + 2 = 9,
0,4,3,2 with a total weight of 4 + 1 + 1 = 6.
The weight of path 0,3,2 is the smallest, so the shortest path from node 0 to node 2 is 0,3,2 with a total weight of 4.

Figure 5: Overview of Build-a-Graph prompting and Algorithmic prompting, aiming to make LLMs better at reasoning with graphs by encouraging conceptual grounding and algorithmic reflection. Red underline indicates our proposed instructions while green background indicates LLMs' generation.

| Method | Cycle | | | | Shortest Path | | | | | Hamilton Path | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Easy | Medium | Hard | Avg. | Easy | Hard | Easy (PC) | Hard (PC) | Avg. | Easy | Hard | Avg. |
| CoT | 84.67 | 63.33 | 53.25 | 66.75 | 63.89 | 29.50 | 76.84 | 35.79 | 51.51 | **40.00** | **8.00** | **24.00** |
| CoT+BaG | **86.00** | 69.33 | 62.00 | **72.44** | **67.78** | **33.50** | **79.20** | **42.56** | **55.76** | 38.67 | 6.00 | 22.34 |
| CoT+Algorithm | 77.33 | **74.00** | **64.00** | 71.78 | 63.89 | 28.00 | 76.06 | 38.70 | 51.66 | 36.67 | 7.50 | 22.09 |

# Take-home messages (paper-1)

Extensive experiments on the NLGraph benchmark demonstrate that:

1. LLMs do possess ***preliminary*** graph reasoning abilities.
2. The benefit of ***advanced prompting*** methods ***diminishes*** with complex problems.
3. ***Learning from examples*** ***did not happen*** on complex graph reasoning problems.
4. LLMs are (un)surprisingly brittle to ***spurious correlations*** in problem settings.

Note that the LLMs here

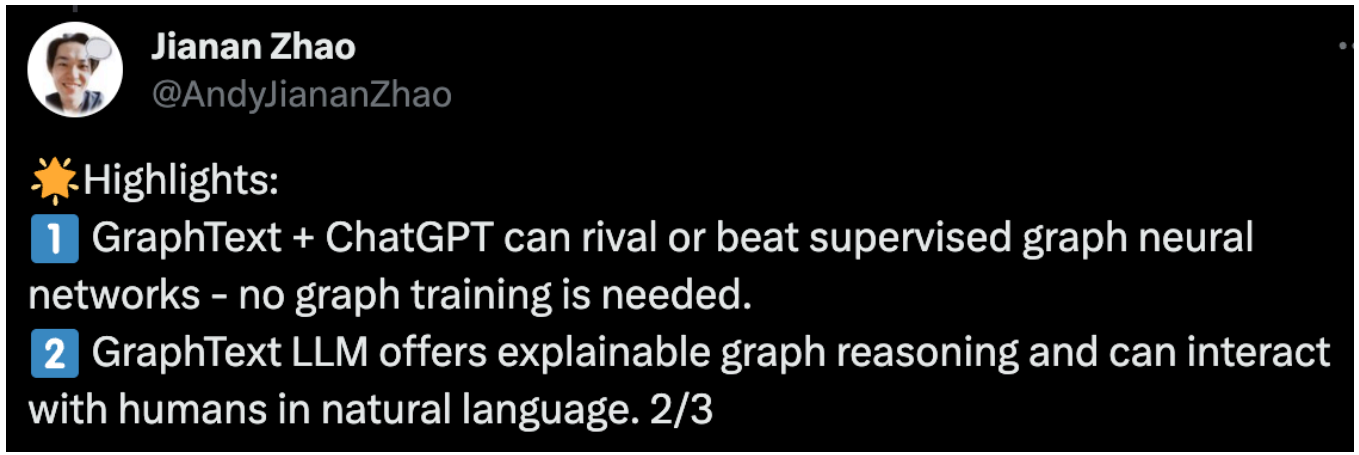
- can only rely on the ***structural*** information of graphs
- previous studies on text-attributed graphs have justified the importance of ***texts***

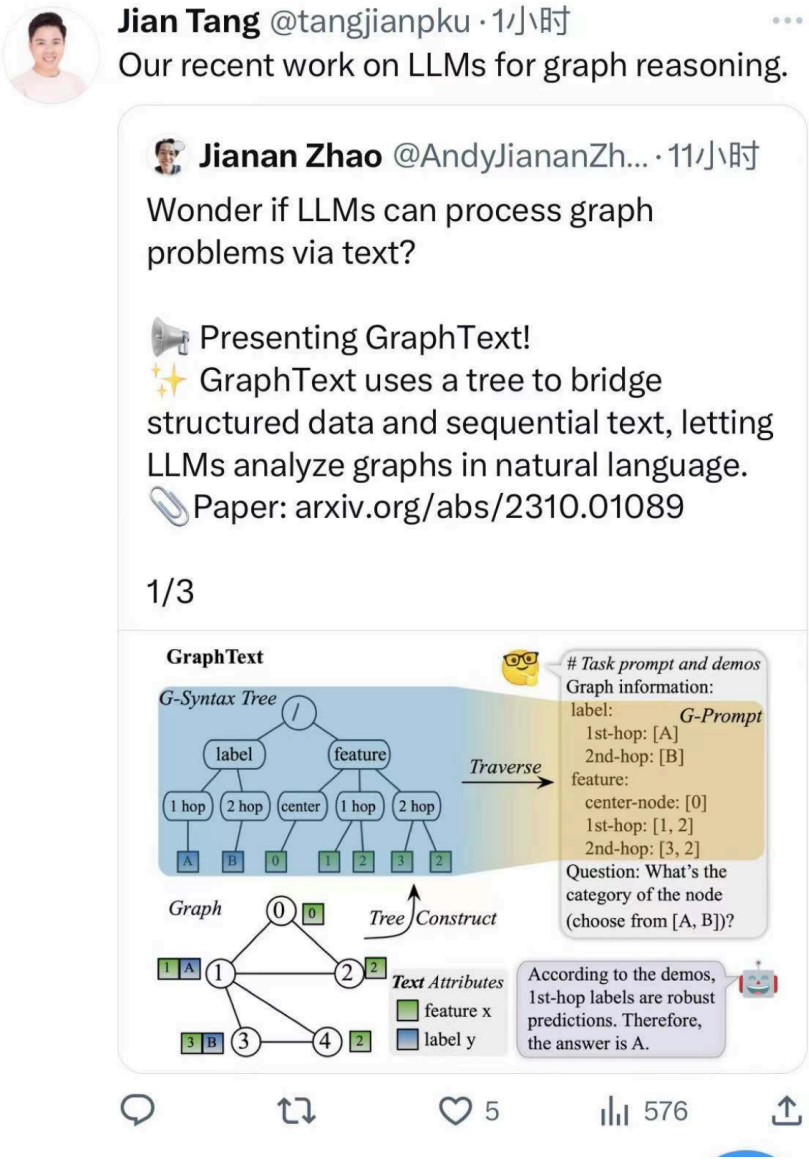
# Outlines

- Background
- An overview of LLMs & Graphs
- paper-1: Can Language Models Solve Graph Problems in Natural Language? 2023/05
- paper-2: GraphText: Graph Reasoning in Text Space. 2023/10
- paper-3: Large Language Models can Learn Rules. 2023/10
- Summary and Discussion

# GRAPHTEXT: GRAPH REASONING IN TEXT SPACE

**Jianan Zhao**[1,2], **Le Zhuo**[3], **Yikang Shen**[4], **Meng Qu**[1,2], **Kai Liu**[5]
**Michael Bronstein**[6], **Zhaocheng Zhu**[1,2], **Jian Tang**[1,7,8]
[1]Mila - Québec AI Institute, [2]Université de Montréal, [3]Beihang University,
[4]MIT-IBM Watson AI Lab, [5]Division of gRED Computational Science, Genentech Inc.,
[6]University of Oxford, [7]HEC Montréal, [8]Canadian Institute for Advanced Research (CIFAR)

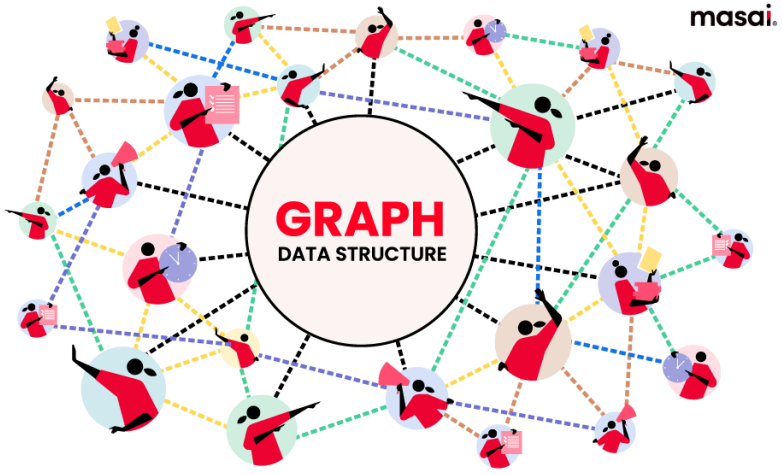


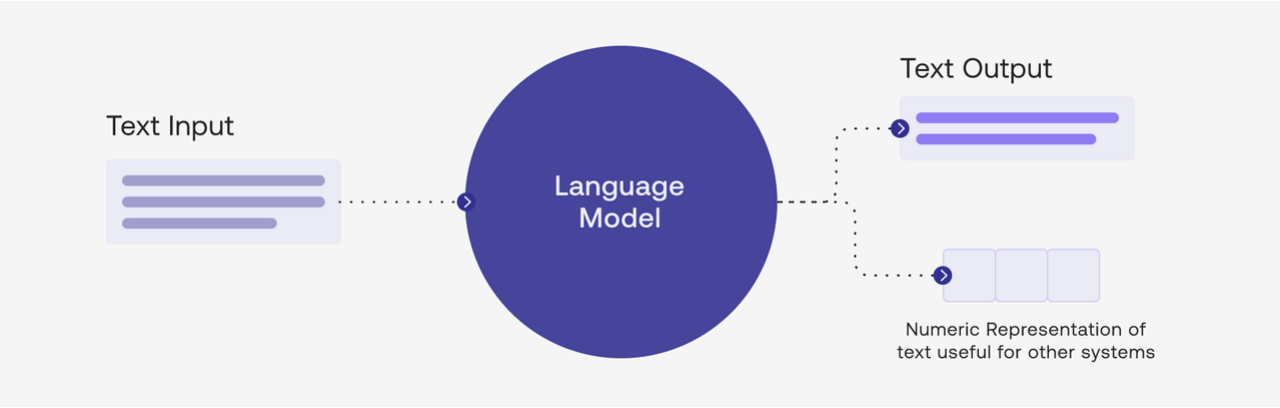




Arxiv 2023/10/02 https://arxiv.org/abs/2310.01089

# Difficulty of applying LLM to graph learning

➔The modality gap between “graph” and “text”

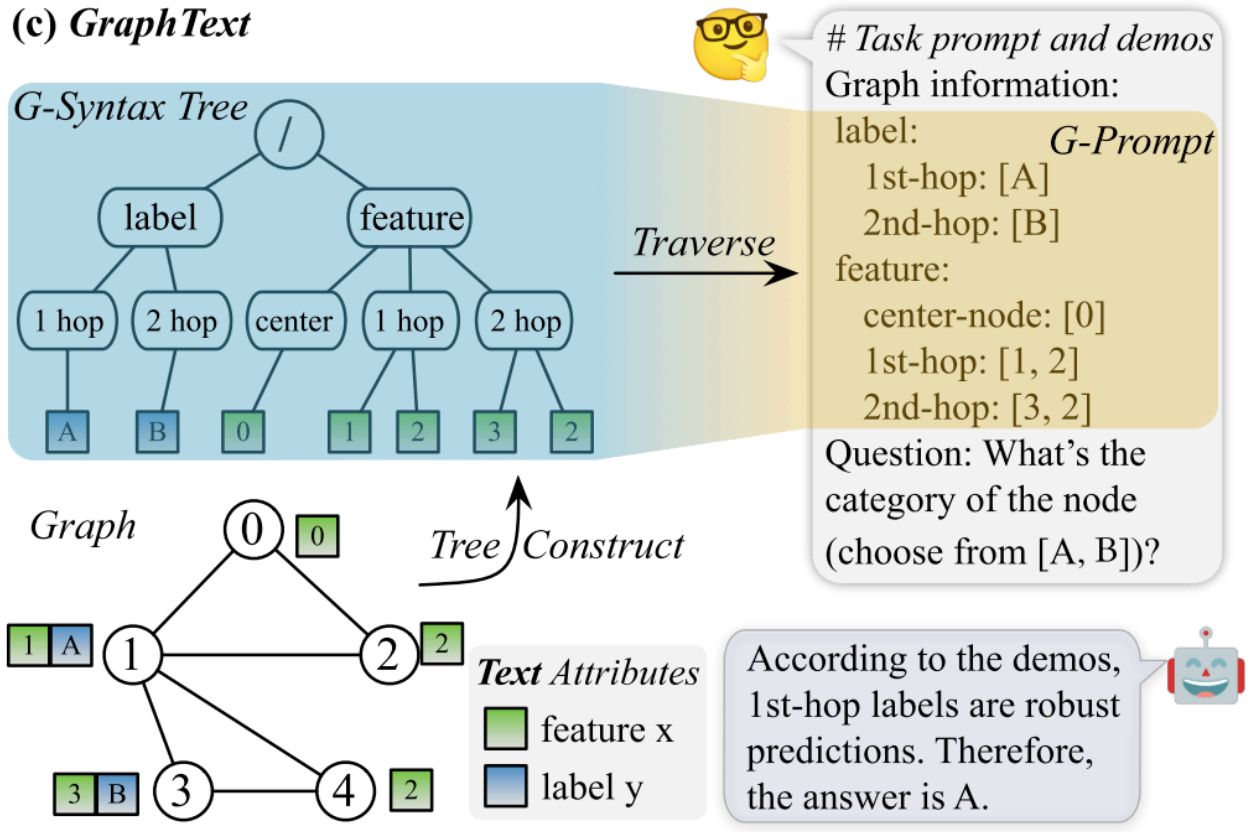



So, can we derive ***a language for graph*** in natural language? 🤔

# Difficulty of applying LLM to graph learning

**GRAPHTEXT: a graph-syntax tree** for each graph that encapsulates both the node attributes and inter-node relationships




***An example of the GRAPHTEXT framework that classifies node 0:***

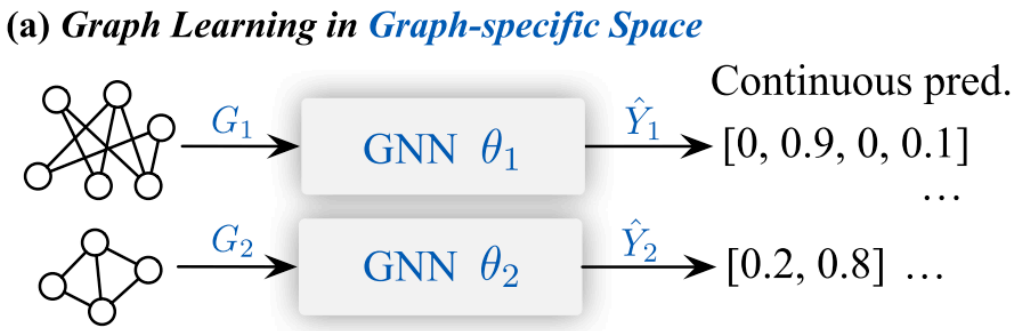
Given a graph, GRAPHTEXT ***constructs a graph-syntax tree*** that contains both node attributes (e.g. feature and label) and relationships (e.g. center-node, 1st-hop, and 2nd-hop).

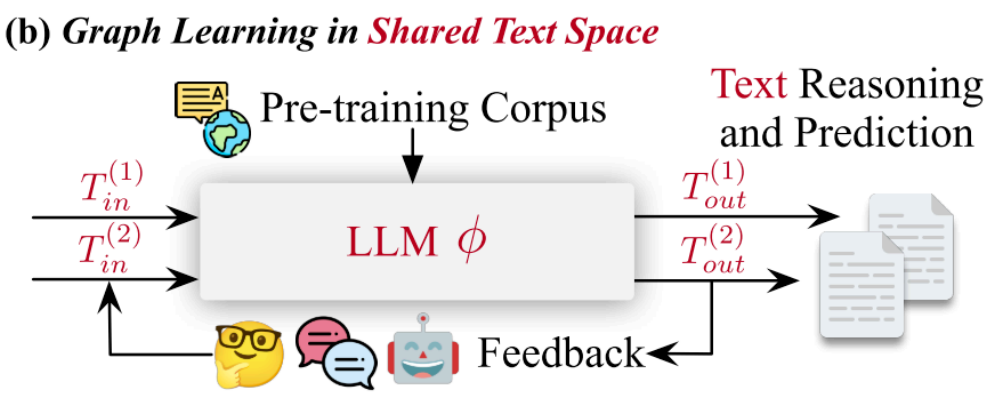
Then, GRAPHTEXT ***traverses the graph-syntax tree to obtain a sequential text***, i.e. graph prompt, and let LLM perform graph reasoning in text space.

# Comparing with GNNs

**Advantages compared with GNNs**

- equipped with commonsense knowledge
- handling text features/attributes
- generalization to few/zero-shot settings
- explainability and human-AI interaction




**GNN framework:**

- different GNNs θ1 θ2 are trained
- for different graphs G1 and G2

**GRAPHTEXT framework:**

- encodes the graph information to text
- generates text as prediction (reasoning)
- leverages a pre-trained LLM
- perform training-free graph reasoning
- enables human-AI interaction in natural language

# Application: Few-shot reasoning with interaction

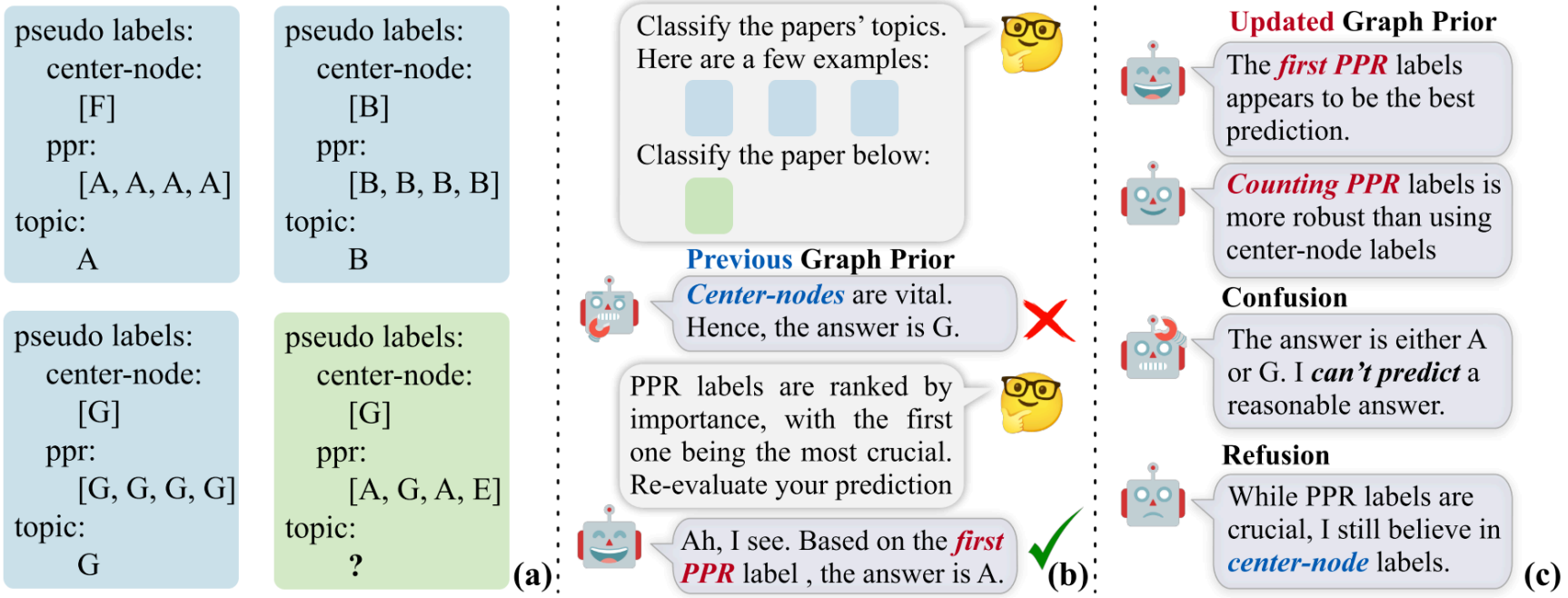



Figure 2: **(a)** Few-shot demonstrations (blue) and the target node #2188 (green) to predict on Cora. **(b)** An illustration of how human interaction changes the graph reasoning of an LLM, where the LLM previously has the prior that the center-node is vital. **(c)** Behaviors of LLMs after given demonstrations/human interaction: *update graph prior* to bias more on PPR (personalized pagerank); leads to *confusion* or *refusion*. Details are discussed in Section 4.2.

# Experiment

Table 1: Node classification results (accuracy %).

| Model | Graph Text Attributes | Relations | Cora | Citeseer | Texas | Wisconsin | Cornell |
|---|---|---|---|---|---|---|---|
| GCN | NA | original | 81.4 | 69.8 | 59.5 | 49.0 | 37.8 |
| GAT | NA | original | 80.8 | 69.4 | 54.1 | 49.0 | 45.9 |
| GCNII | NA | original | 81.2 | 69.8 | 56.8 | 51.0 | 40.5 |
| GATv2 | NA | original | **82.3** | **69.9** | 62.2 | 52.9 | 43.2 |
| GRAPHTEXT-ICL | label | original | 26.3 | 13.7 | 5.4 | 9.8 | 21.6 |
| | | ori.+synth. | 53.0 | 37.2 | 70.3 | **64.7** | **51.4** |
| | label+feat | original | 33.4 | 36.9 | 5.4 | 29.4 | 24.3 |
| | | ori.+synth. | 52.1 | 50.4 | 73.0 | 60.8 | 46.0 |
| | label+feat+synth. | original | 64.5 | 51.0 | 73.0 | 35.3 | 48.7 |
| | | ori.+synth. | 68.3 | 58.6 | **75.7** | 54.9 | **51.4** |

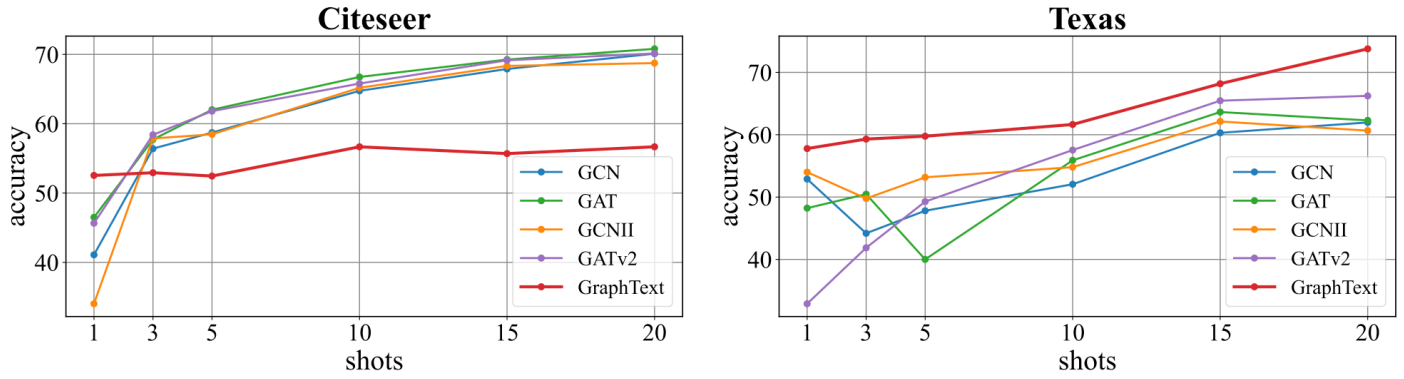



Figure 3: Few-shot in-context learning node classification accuracy. We perform 1, 3, 5, 10, 15, and 20-shot node classification on Citeseer and Texas datasets.

➔ GRAPHTEXT can deliver performance on par with, or even surpassing, supervised GNNs through in-context learning.

➔ GRAPHTEXT can out-perform GNNs in Zero-shot / Few-shot setting.

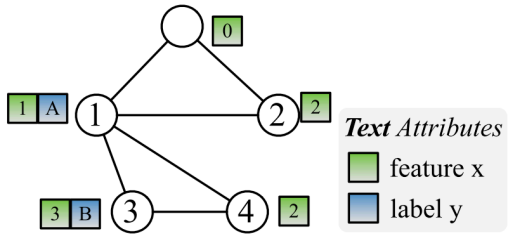
# Experiment

Table 2: Interactive graph reasoning results (accuracy %) on Cora (node # 2188). The table showcases the performance of GPT-4 and ChatGPT before and after human interactions with 15 times of evaluation. The reasoning metrics include PPR, Center-node, and instances where the model was Confused to respond or Refused (Conf./Ref.) to make their reasoning/prediction. See Figure 2 (c) for details.

| Model | Interaction | Accuracy | Reasoning | | |
|---|---|---|---|---|---|
| | | | PPR | Center-node | Conf./Ref. |
| GPT-4 | Before | 73.3 | 73.3 | 26.7 | 0 |
| | After | **100 (+26.7)** | 100 | 0 | 0 |
| ChatGPT | Before | 26.7 | 26.7 | 53.3 | 20.0 |
| | After | **63.6 (+36.9)** | 72.7 | 18.2 | 9.1 |

➔ The human-LLM interaction is essential.

# Experiment




**(a)** An example graph

label:
1st-hop: [A]
2nd-hop: [B]
feature:
center-node: [0]
1st-hop: [1, 2]
2nd-hop: [3, 2]

**(b)** GRAPHTEXT

center-node:
feature: [0]
1st-hop:
label: [A]
feature: [1, 2]
2nd-hop:
label: [B]
feature: [3, 2]

**(c)** Rev. hierarchy

attributes:
[A]
[B]
[0]
[1,2]
[3,2]

**(d)** w/o int. nodes

attributes:
[A, B, 0, 1, 2, 3, 2]

**(e)** sequence

attributes:
[2, 0, 3, B 1, 2, A]

**(f)** set

Figure 4: Ablations of graph-syntax trees. **(a)** An example graph. **(b)** GRAPHTEXT text prompt (The full example can be found in Figure 1). **(c-f)** Text prompts of different tree designs.

Table 3: Ablations of GRAPHTEXT on Cora, Citeseer and Texas.

| Model | Cora | | Citeseer | | Texas | |
|---|---|---|---|---|---|---|
| | Acc. % | Δ | Acc. % | Δ | Acc. % | Δ |
| **GRAPHTEXT** | **76.5** | - | **58.6** | - | **75.7** | - |
| rev. hierarchy | 68.3 | -10.7 % | 57.6 | -1.7 % | 73.0 | -3.6 % |
| w/o int. nodes | 67.8 | -11.4 % | 56.3 | -3.9 % | 75.7 | -0 % |
| sequence | 67.0 | -12.4 % | 53.0 | -9.6 % | 70.3 | -7.1 % |
| set | 65.9 | -13.9 % | 56.4 | -3.8 % | 67.6 | -10.6 % |

➔ the hierarchy of graph-syntax tree is important.

# Experiment

Table 4: Node classification results (accuracy %) on real-world text attributed graphs. Experiments are conducted using in-context learning with ChatGPT, as well as instruction tuning with Llama-2-7B. Note that "text" refers to raw text attributes, while "feat" represents the continuous features on the graph. The top results for each category are highlighted in bold.

| Framework | Model | Cora | Citeseer |
|---|---|---|---|
| GNNs | GCN | 89.13 | 74.92 |
| | GAT | **89.68** | **75.39** |
| GRAPHTEXT | ChatGPT-text | **67.77** | **68.98** |
| | ChatGPT-feat | 10.68 | 16.14 |
| | ChatGPT-feat+text | 65.19 | 66.46 |
| | Llama-2-7B-text | 60.59 | 49.37 |
| | Llama-2-7B-feat | **87.11** | **74.77** |
| | Llama-2-7B-feat+text | 77.53 | 73.83 |

➔ The ability of LLM on graph learning can be further enhanced.

# Take-home messages (paper-2)

- GRAPHTEXT serves as a general reasoning framework for both in-context learning and instruction tuning, on both general graphs and text-attributed graphs.

- The training-free property enables us to deploy GRAPHTEXT not only with open-source LLMs, but also with powerful closed-source LLMs.

- GRAPHTEXT fosters interactive graph reasoning. With its capacity to generate and explain predictions in natural language, humans can directly engage in.

# Outlines

- Background
- An overview of LLMs & Graphs
- paper-1: Can Language Models Solve Graph Problems in Natural Language? 2023/05
- paper-2: GraphText: Graph Reasoning in Text Space. 2023/10
- paper-3: Large Language Models can Learn Rules. 2023/10
- Summary and Discussion

# LARGE LANGUAGE MODELS CAN LEARN RULES

Google DeepMind

## LARGE LANGUAGE MODELS CAN LEARN RULES

**Zhaocheng Zhu**[1, 2, 3,*] **Yuan Xue**[1] **Xinyun Chen**[1] **Denny Zhou**[1] **Jian Tang**[2, 4, 6]
**Dale Schuurmans**[1, 5, 6] **Hanjun Dai**[1]
[1] Google [2] Mila - Québec AI Insitute [3] Université de Montréal
[4] HEC Montréal [5] University of Alberta [6] CIFAR AI Chair
zhuzhaoc@mila.quebec, hadai@google.com

https://arxiv.org/pdf/2310.07064.pdf

# Motivation

While it is common to curate a dataset and inject required knowledge into an LLM via ***supervised fine-tuning***

*However, supervised fine-tuning is quite expensive.*
*So, how can we leverage the few-shot examples without fine-tuning?* 🤔

# Motivation

While it is common to curate a dataset and inject required knowledge into an LLM via ***supervised fine-tuning***,
***a generic approach*** to ***automatically discovering and applying knowledge*** in reasoning problems would be ***more desirable***

For example:

mortal: 终有一死的，不能永生的

*Socrates is a man.* *All men are mortal.* *Therefore, Socrates is mortal.*

a fact a rule a fact

*If Scorates is a man, Socrates is mortal*

# ***Deductive*** Reasoning *or* ***Inductive*** Reasoning?

***Deductive Reasoning*** aims to ***derive new facts based on known facts and rules***.

For LLMs, most prompting techniques are designed to elicit deductive reasoning.

- chain-of-thought prompting teach an LLM to deduce conclusions from given facts
- these prompting methods rely on implicit rules stored in the LLM

***Inductive Reasoning*** focuses on ***deriving general rules from observed facts***.

- For example, if we know "Scorates is a man" and "Socrates is mortal" and the same facts hold for Aristotle, we might hypothesize a rule "All men are mortal".
- Inductive reasoning is widely studied in logic programming and KG reasoning

# ***Deductive*** Reasoning *or* ***Inductive*** Reasoning?

The proposed HtT framework

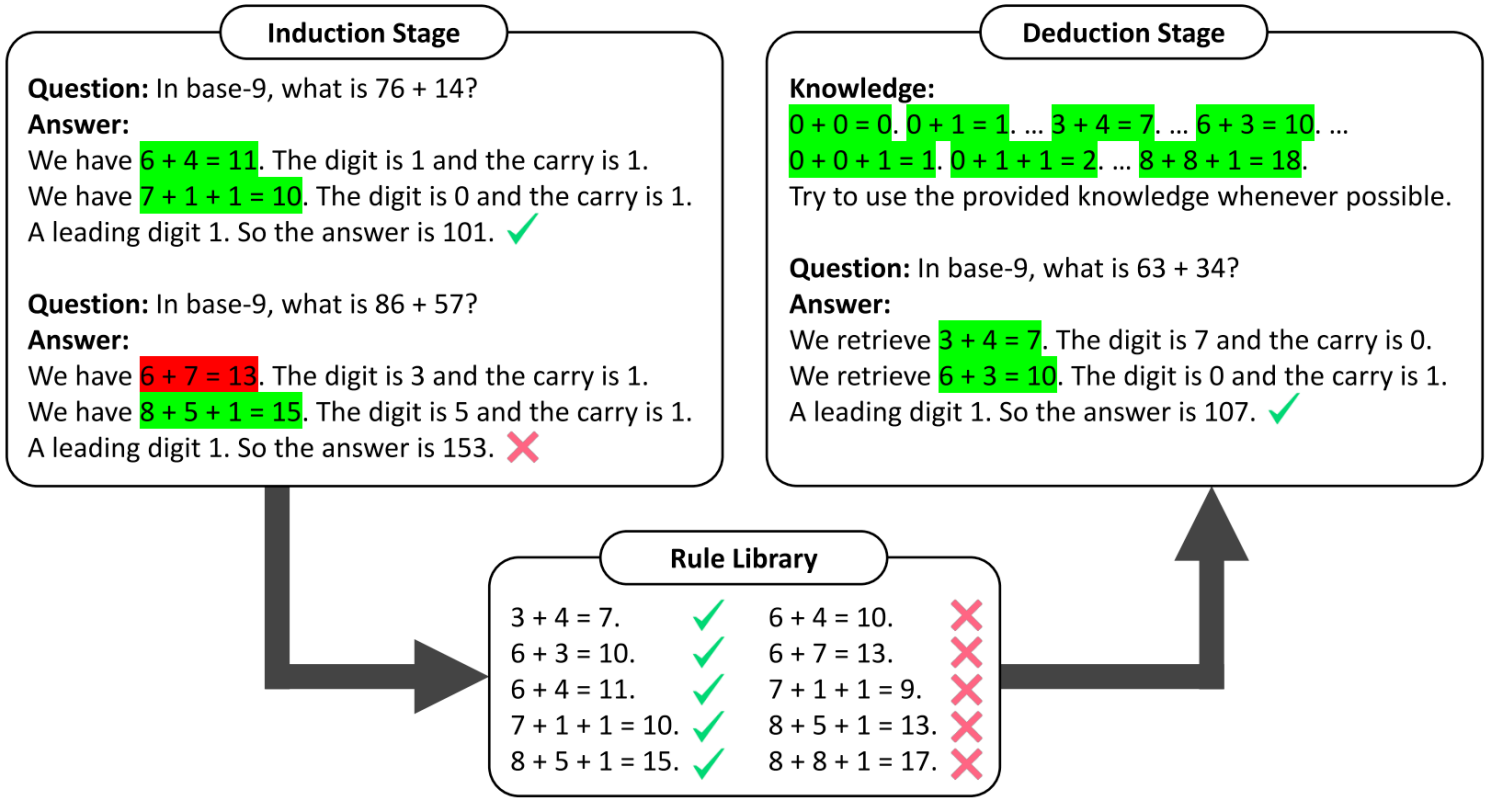



Figure 1: An example of Hypotheses-to-Theories applied to chain-of-thought for the base-9 arithmetic problem. Few-shot examples are omitted for brevity. The induction stage uses CoT to generate rules and verify them on the training samples. Rules are then collected and filtered to form the rule library. The deduction stage augments the CoT prompt with knowledge from the rule library. Correct and incorrect rules are marked with green and red respectively. Full prompts are in Appendix B.

# ***Deductive*** Reasoning *or* ***Inductive*** Reasoning?

Prompt 4: 5-shot LtM prompt for Arithmetic. LtM uses the same prompt as CoT for deductive reasoning, except that LtM calls the following prompt whenever it encounters a rule.

**Question:** In base-16, what is E + D + 1?
**Answer:** E + D + 1 = 1C.

**Question:** In base-16, what is 8 + F?
**Answer:** 8 + F = 17.

**Question:** In base-16, what is 7 + 8 + 1?
**Answer:** 7 + 8 + 1 = 10.

**Question:** In base-16, what is 6 + 4?
**Answer:** 6 + 4 = A.

**Question:** In base-16, what is 5 + D?
**Answer:** 5 + D = 12.

**Question:** In base-16, what is {{ x }} + {{ y }}?
**Answer:**

Prompt 5: 5-shot LtM+HtT prompt for Arithmetic. LtM+HtT uses the same prompt as CoT for deductive reasoning, except that LtM+HtT calls the following prompt whenever it encounters a rule.

**Instruction:** When you answer the questions, try to use the provided knowledge whenever possible. Try not to invent knowledge by yourself unless necessary.
**Knowledge:**
<no_carry>
<0><0>0 + 0 = 0.</0>... <F>0 + F = F.</F></0>
...
<F><0>F + 0 = F.</0>... <F>F + F = 1E.</F></F>
</no_carry>
<carry>
<0><0>0 + 0 + 1 = 1.</0>... <F>0 + F + 1 = 10.</F></0>
...
<F><0>F + 0 + 1 = F.</0>... <F>F + F + 1 = 1F.</F></F>
</carry>

**Question:** In base-16, what is E + D + 1?
**Answer:** We retrieve <carry><E><D>E + D + 1 = 1C.

**Question:** In base-16, what is 8 + F?
**Answer:** We retrieve <no_carry><8><F>8 + F = 17.

**Question:** In base-16, what is 7 + 8 + 1?
**Answer:** We retrieve <carry><7><8>7 + 8 + 1 = 10.

**Question:** In base-16, what is 6 + 4?
**Answer:** We retrieve <no_carry><6><4>6 + 4 = A.

**Question:** In base-16, what is 5 + D?
**Answer:** We retrieve <no_carry><5><D>5 + D = 12.

**Question:** In base-16, what is {{ x }} + {{ y }}?
**Answer:**

# ***Deductive*** Reasoning *or* ***Inductive*** Reasoning?

Prompt 7: 5-shot CoT prompt for CLUTRR.

**Context:** The relations on the path from Alan to Anthony are daughter, uncle, son.
**Question:** Anthony is Alan's what?
**Answer:**
For daughter's uncle, we have daughter's uncle is brother. So the relations are reduced to brother, son.
For brother's son, we have brother's son is nephew. So the relations are reduced to nephew.
Therefore, the answer is nephew.

**Context:** The relations on the path from Annie to Carlos are brother, mother, son.
**Question:** Carlos is Annie's what?
**Answer:**
For brother's mother, we have brother's mother is mother. So the relations are reduced to mother, son.
For mother's son, we have mother's son is brother. So the relations are reduced to brother.
Therefore, the answer is brother.

Prompt 8: 5-shot CoT+HtT prompt for CLUTRR.

**Instruction:** When you answer the questions, try to use the provided knowledge whenever possible. Try not to invent knowledge by yourself unless necessary.
**Knowledge:**
<aunt><brother>aunt's brother is uncle.</brother>. . . <aunt><son>aunt's son is cousin.</son></aunt>
. . .
<wife><brother>wife's brother is brother-in-law.</brother>. . . <son>wife's son is son.</son></wife>

**Context:** The relations on the path from Alan to Anthony are daughter, uncle, son.
**Question:** Anthony is Alan's what?
**Answer:**
For daughter's uncle, we retrieve <daughter><uncle>daughter's uncle is brother. So the relations are reduced to brother, son.
For brother's son, we retrieve <brother><son>brother's son is nephew. So the relations are reduced to nephew.
Therefore, the answer is nephew.

**Context:** The relations on the path from Annie to Carlos are brother, mother, son.
**Question:** Carlos is Annie's what?
**Answer:**
For brother's mother, we retrieve <brother><mother>brother's mother is mother. So the relations are reduced to mother, son.
For mother's son, we retrieve <mother><son>mother's son is brother. So the relations are reduced to brother.
Therefore, the answer is brother.

# Experiments

Table 1: Results on Arithmetic. For reference, GPT4 (5-shot CoT) has 99.1% accuracy on base-10.

| Model | Prompt | Base-16 2 digits | Base-16 3 digits | Base-16 4 digits | Base-11 2 digits | Base-11 3 digits | Base-11 4 digits | Base-9 2 digits | Base-9 3 digits | Base-9 4 digits | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|
| GPT3.5 | 0-shot CoT | 30.6 | 10.5 | 0.0 | 5.6 | 5.3 | 0.0 | 11.1 | 10.5 | 0.0 | 8.2 |
| | 5-shot CoT | 83.3 | 34.2 | 11.5 | 5.6 | 2.6 | 0.0 | 25.0 | 13.2 | 11.5 | 20.8 |
| | + HtT | 77.8 | 52.6 | 23.1 | 25.0 | 13.2 | 0.0 | 8.3 | 5.3 | 11.5 | **24.1 (+3.3)** |
| | + HtT (GPT4) | 63.9 | 44.7 | 34.6 | 13.9 | 7.9 | 3.8 | 25.0 | 7.9 | 11.5 | **23.7 (+2.9)** |
| | 5-shot LtM | 83.3 | 34.2 | 15.4 | 16.7 | 5.3 | 0.0 | 13.9 | 7.9 | 7.7 | 20.5 |
| | + HtT | 80.6 | 39.5 | 26.9 | 16.7 | 2.6 | 3.8 | 19.4 | 5.3 | 3.8 | **22.1 (+1.6)** |
| | + HtT (GPT4) | 72.2 | 31.6 | 30.8 | 47.2 | 15.8 | 11.5 | 44.4 | 21.1 | 15.4 | **32.2 (+11.7)** |
| GPT4 | 0-shot CoT | 72.2 | 26.3 | 7.7 | 22.2 | 10.5 | 3.8 | 30.6 | 34.2 | 23.1 | 25.6 |
| | 5-shot CoT | 83.3 | 71.1 | 61.5 | 52.8 | 47.4 | 46.2 | 75.0 | 36.8 | 42.3 | 57.4 |
| | + HtT | 100.0 | 94.7 | 84.6 | 88.9 | 71.1 | 46.2 | 86.1 | 68.4 | 65.4 | **78.4 (+21.0)** |
| | 5-shot LtM | 88.9 | 81.6 | 61.5 | 52.8 | 47.4 | 30.8 | 52.8 | 31.6 | 11.5 | 51.0 |
| | + HtT | 100.0 | 86.8 | 76.9 | 72.2 | 52.6 | 46.2 | 61.1 | 23.7 | 38.5 | **62.0 (+11.0)** |

Table 2: Results on the symbolic version of CLUTRR.

| Model | Prompt | 2 hops | 3 hops | 4 hops | 5 hops | 6 hops | 7 hops | 8 hops | 9 hops | 10 hops | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|
| EdgeTransformer | | 100.0 | 94.4 | 96.8 | 88.0 | 68.8 | 61.9 | 50.0 | 50.0 | 36.0 | 71.8 |
| GPT3.5 | 0-shot CoT | 50.0 | 22.2 | 12.9 | 8.0 | 12.5 | 9.5 | 10.0 | 3.8 | 4.0 | 14.8 |
| | 5-shot CoT | 0.0 | 27.8 | 45.2 | 36.0 | 18.8 | 19.0 | 16.7 | 11.5 | 16.0 | 21.2 |
| | + HtT | 87.5 | 38.9 | 35.5 | 44.0 | 37.5 | 14.3 | 33.3 | 11.5 | 36.0 | **37.6 (+16.4)** |
| | + HtT (GPT4) | 100.0 | 55.6 | 32.3 | 60.0 | 50.0 | 47.6 | 43.3 | 19.2 | 28.0 | **48.4 (+27.2)** |
| | 5-shot LtM | 37.5 | 22.2 | 29.0 | 36.0 | 25.0 | 14.3 | 10.0 | 23.1 | 20.0 | 24.1 |
| | + HtT | 100.0 | 33.3 | 32.3 | 48.0 | 31.3 | 33.3 | 23.3 | 34.6 | 28.0 | **40.5 (+16.4)** |
| | + HtT (GPT4) | 75 | 44.4 | 41.9 | 52.0 | 37.5 | 33.3 | 23.3 | 19.2 | 16.0 | **38.1 (+14.0)** |
| GPT4 | 0-shot CoT | 50.0 | 22.2 | 22.6 | 32.0 | 37.5 | 38.1 | 33.3 | 46.2 | 16.0 | 33.1 |
| | 5-shot CoT | 50.0 | 55.6 | 71.0 | 80.0 | 50.0 | 52.4 | 30.0 | 46.2 | 20.0 | 50.6 |
| | + HtT | 100.0 | 61.1 | 74.2 | 84.0 | 75.0 | 38.1 | 56.7 | 53.8 | 36.0 | **64.3 (+13.7)** |
| | 5-shot LtM | 62.5 | 38.9 | 58.1 | 68.0 | 50.0 | 38.1 | 43.3 | 34.6 | 28.0 | 46.8 |
| | + HtT | 100.0 | 55.6 | 77.4 | 80.0 | 75.0 | 38.1 | 36.7 | 38.5 | 20.0 | **57.9 (+11.1)** |

# Experiments

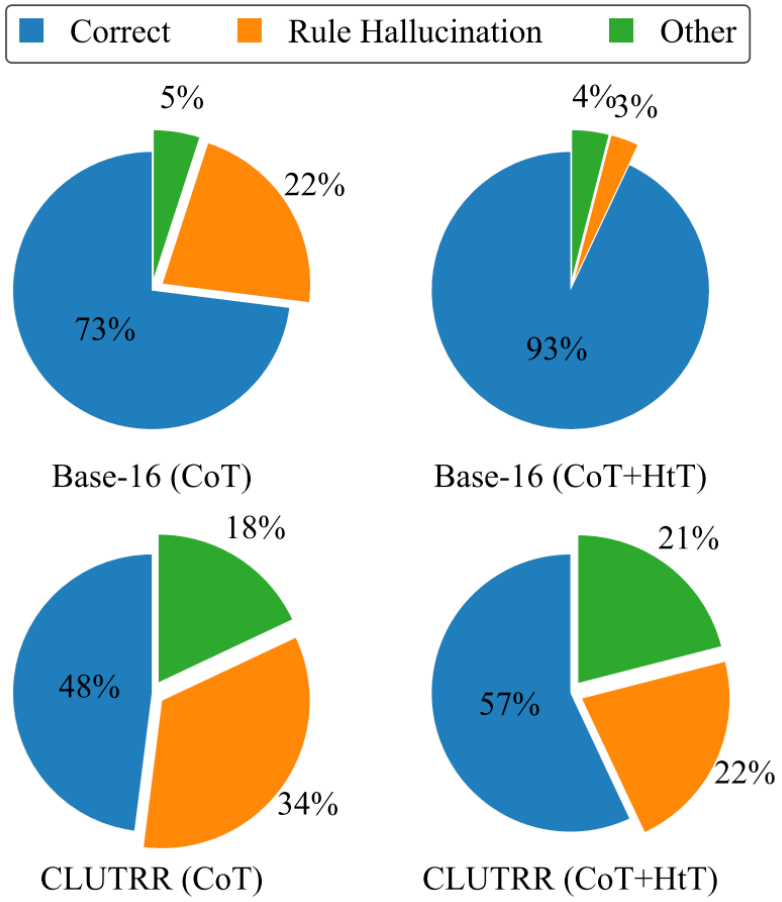



Figure 3: Statistics of different error cases.

Two types of errors related to rules:
(1) the LLM generates or retrieves ***a factually incorrect rule*** ***(rule hallucination)***
(2) the LLM retrieves a factually correct rule that is not proper for the deductive step

➔ HtT significantly reduces hallucination versus CoT

# Experiments

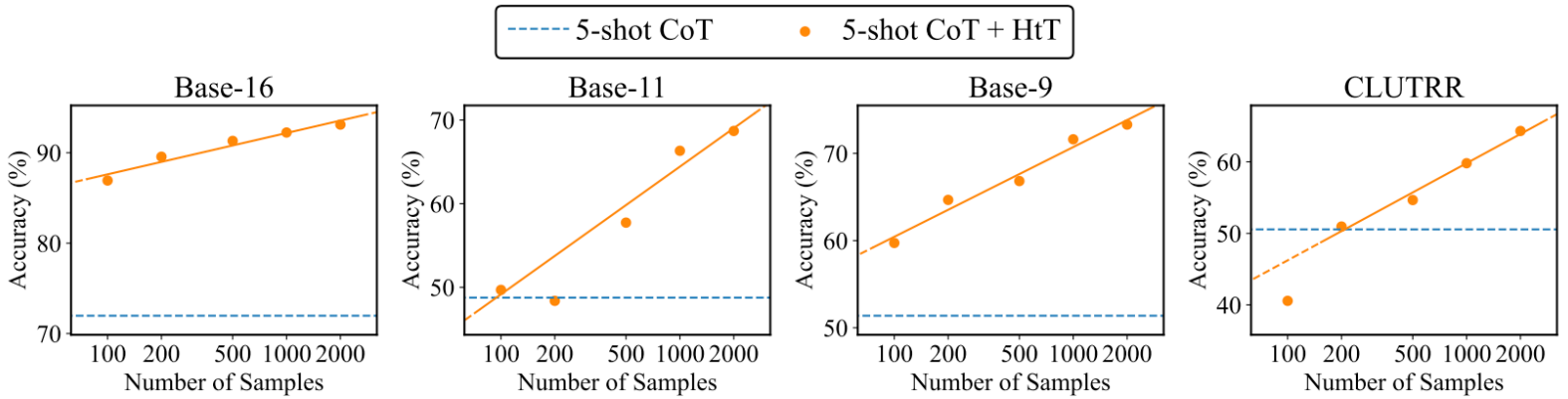



Figure 4: Performance of HtT w.r.t. the number of samples in the induction stage.

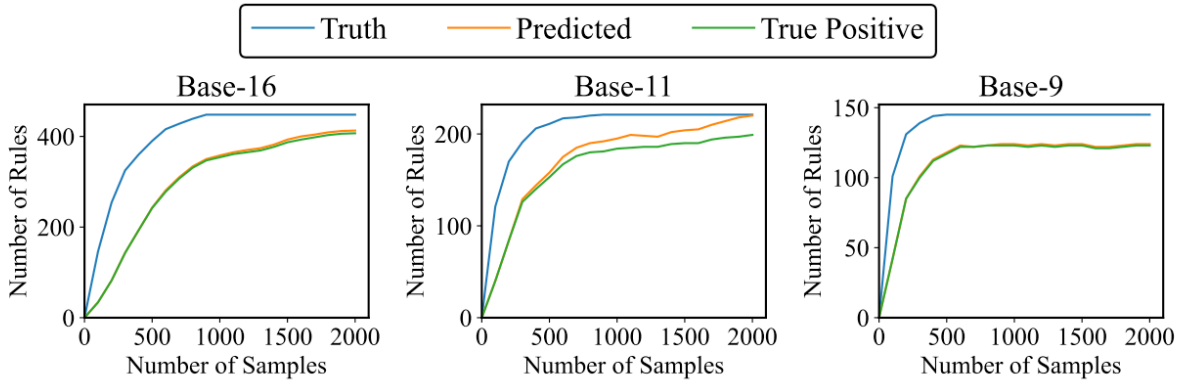



Figure 5: Number of rules discovered by HtT in the induction stage.

# Take-home messages (paper-3)

Hypotheses-to-Theories (HtT): learn explicit rules and apply them in reasoning

- ***HtT consists of two stages***:
  - an induction stage that generates rules and verifies them to construct a library
  - and a deduction stage that applies rules from the learned rule library to solve a reasoning problem.
- ***HtT significantly boosts the performance***
  - of baseline prompting methods on numerical reasoning and relational reasoning problems
- ***HtT opens up a new direction***
  - of learning textual knowledge with LLMs.

Limitations:it requires the base model to have reasonably ***strong knowledge and retrieval ability***

- the gain of HtT for GPT3.5 is very marginal due to weak knowledge of non-decimal systems
- even with a rule library induced by GPT4, GPT3.5 has issues in retrieving correct rules
- such issues may be solved by finetuning on retrieval datasets

***Retrieval-augmented generation (RAG) could be useful here?*** 🤔

# Outlines

- Background
- An overview of LLMs & Graphs
- paper-1: Can Language Models Solve Graph Problems in Natural Language? 2023/05
- paper-2: GraphText: Graph Reasoning in Text Space. 2023/10
- paper-3: Large Language Models can Learn Rules. 2023/10
- Summary and Discussion

# Overview

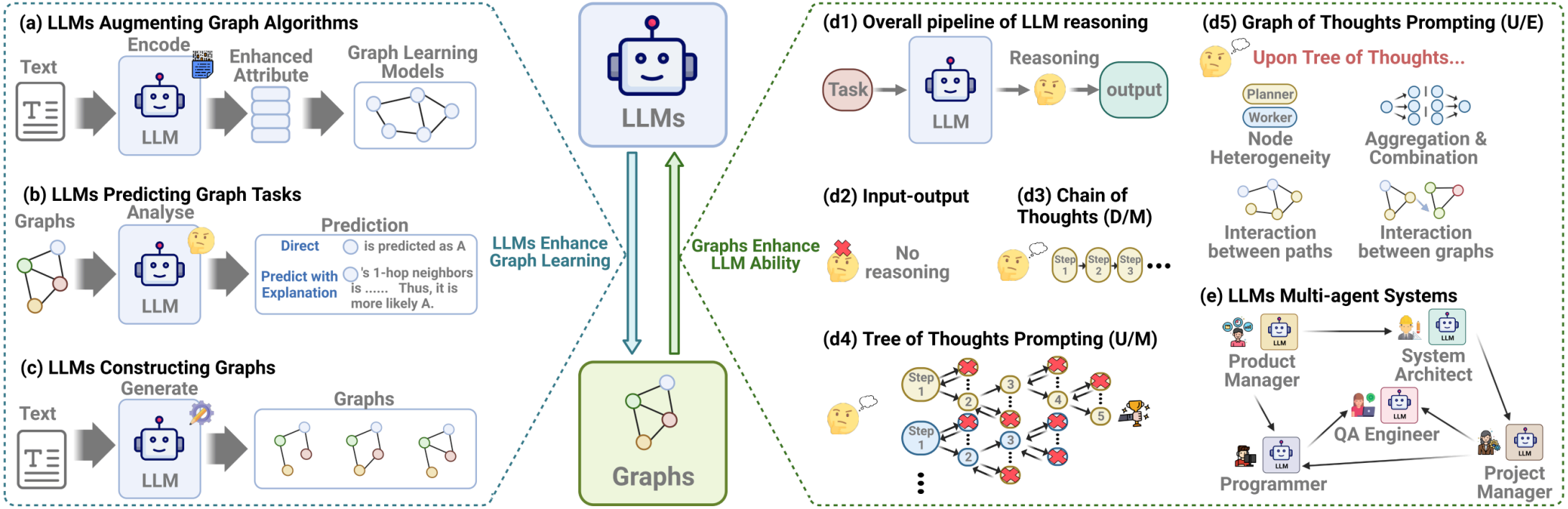
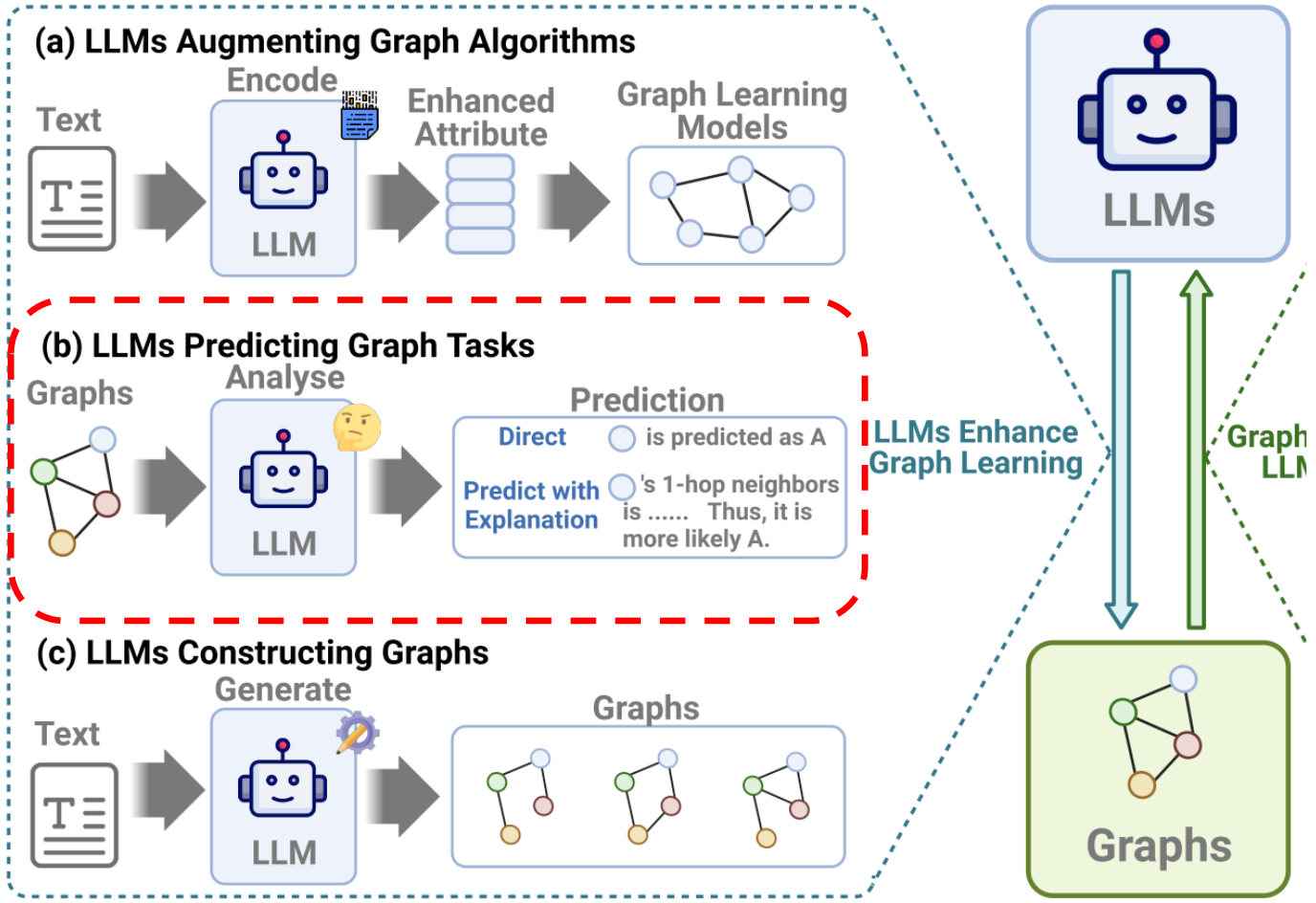



**FIGURE 1.** The overall framework of the mutual enhancement between LLMs and Graphs. (a)-(c): three pathways for LLMs to enhance graph learning. (d)-(e): techniques for graph structures enhancing LLM reasoning. Brackets after technique names indicate graph types. D, U, M and E represent directed, undirected, homogeneous and heterogeneous graphs, respectively.

Integrating Graphs with Large Language Models: Methods and Prospects. Arxiv 2023.10.

# Overview | LLMs Enhancing Graph Learning

e.g.,
NLGraph (paper-1)
GRAPHTEXT (paper-2)






Integrating Graphs with Large Language Models: Methods and Prospects. Arxiv 2023.10.

# Open Questions and Directions | LLMs Enhancing Graph Learning

**Question 1.** How to leverage LLMs to learn on other types of graphs beyond Text-attributed Graphs (TAG)?

- Direction 1. Translate diverse data types into textual format
  - For instance, a user's profile on a social network might list attributes like age, ad- dress, gender, and hobbies.
- Direction 2. Leveraging multi-modal models for graph-text alignment
  - Multi-modal LLMs have already made notable strides in domains like audio and images.

**Question 2.** How can we help LLMs understand graphs?

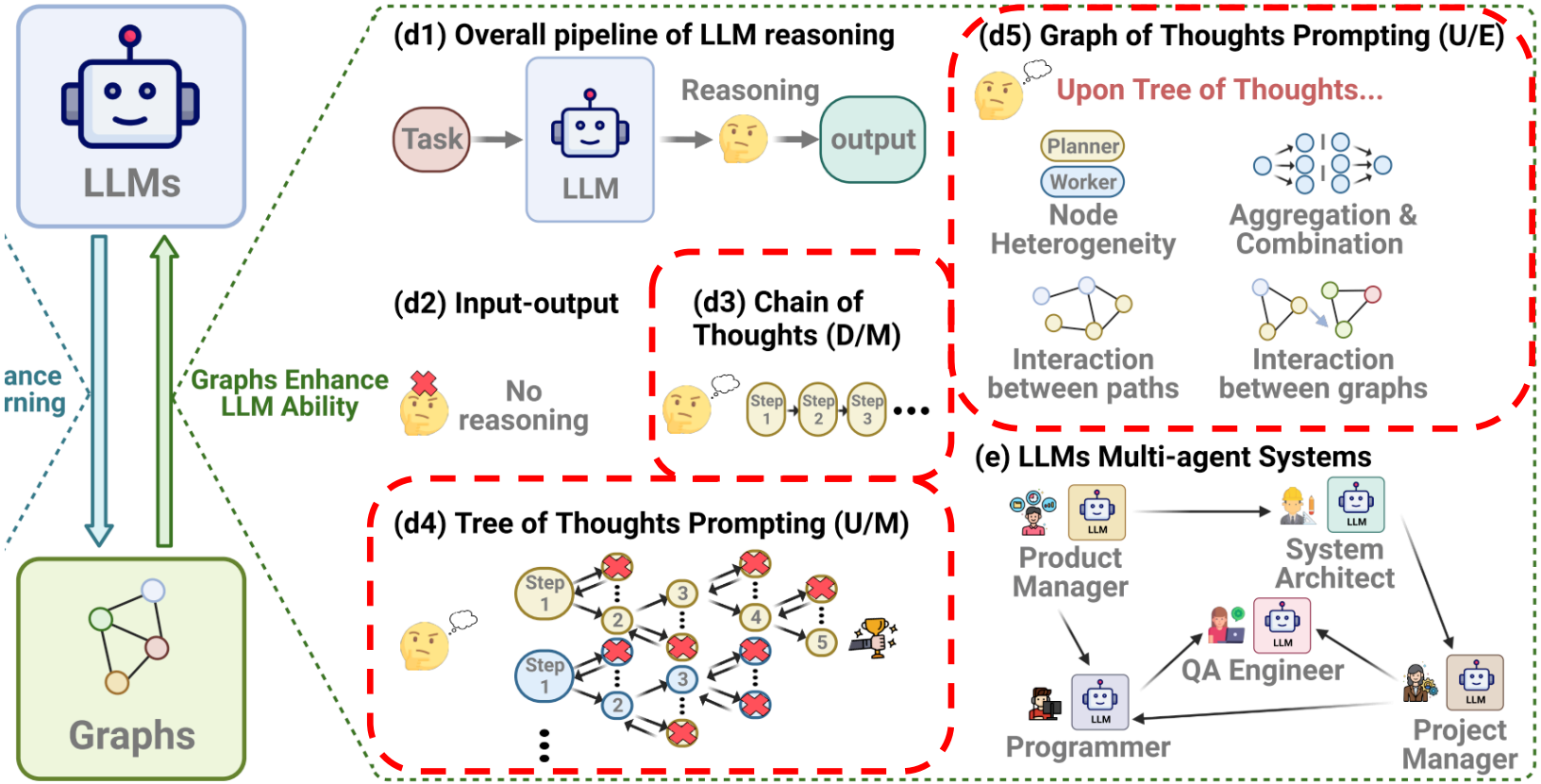
- Direction 1. Expanding graph description languages
  - Current graph description languages offer a somewhat restricted scope.
- Direction 2. Pretraining or fine-tuning LLMs on Graphs
  - Pre-training or fine-tuning LLMs on various graph data
  - converted by graph description language can help LLMs understand graphs better
- Direction 3. Foundational graph models for graph learning
  - While foundational models have made strides in areas like language, and image,
  - a gap remains in establishing large-scale foundational models for graphs.

Extensions with plug-in algorithmic modules?



Integrating Graphs with Large Language Models: Methods and Prospects. Arxiv 2023.10.

# Overview | Graphs Enhance LLM Ability




e.g.,
Chain-of-thought,
Tree-of-thought,
Graph-of-thought,
Hypotheses-to-Theories (paper-3),
etc.

Integrating Graphs with Large Language Models: Methods and Prospects. Arxiv 2023.10.

# Open Questions and Directions | Graphs Enhance LLM Ability

**Question 1.** How to elevate more sophisticated graph structures to enhance LLM reasoning?

- Directions: Expanding the types of graphs for LLM reasoning
- Diversifying the graph types, e.g., hypergraphs, probabilistic graphical models, and signed graphs

**Question 2.** How to elevate more sophisticated graph structures to enhance multi-agent systems (MLS)?

- Directions: Incorporating advanced graph structures for team-based LLM workflows
- adopting varied graph forms such as trees, traditional graphs, and even more intricate structures may help

**Question 3.** How to integrate graph structures into the pipeline of LLMs?

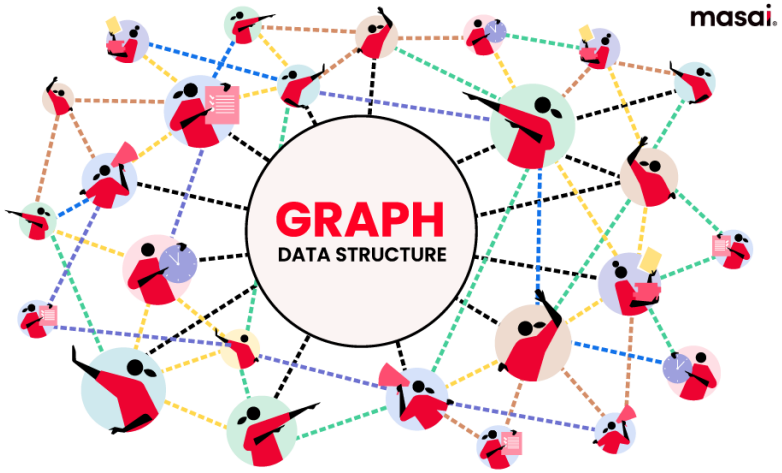
- Directions: Utilizing graph structures in training, fine-tuning, and inference.
- For example, graphs can structure the training data, enabling more effective learning.

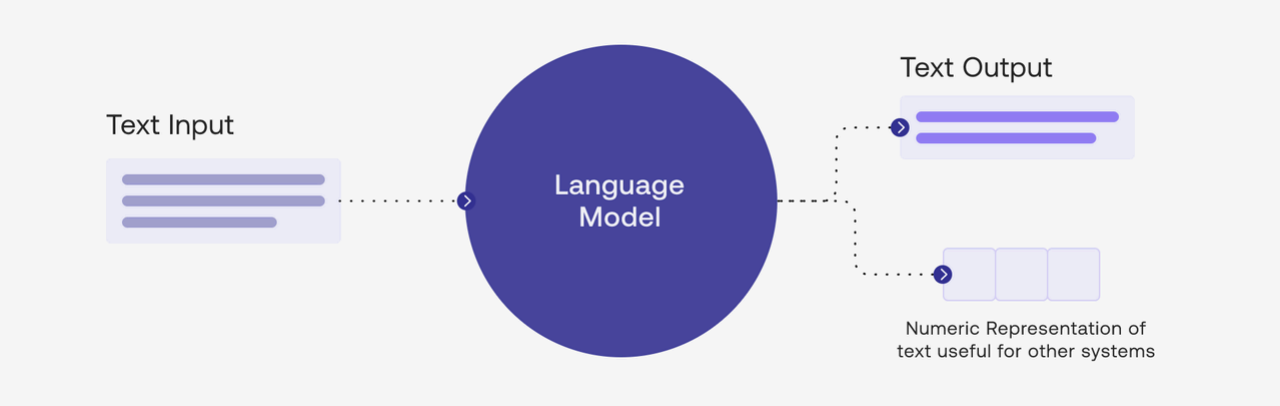


Integrating Graphs with Large Language Models: Methods and Prospects. Arxiv 2023.10.

# Future directions

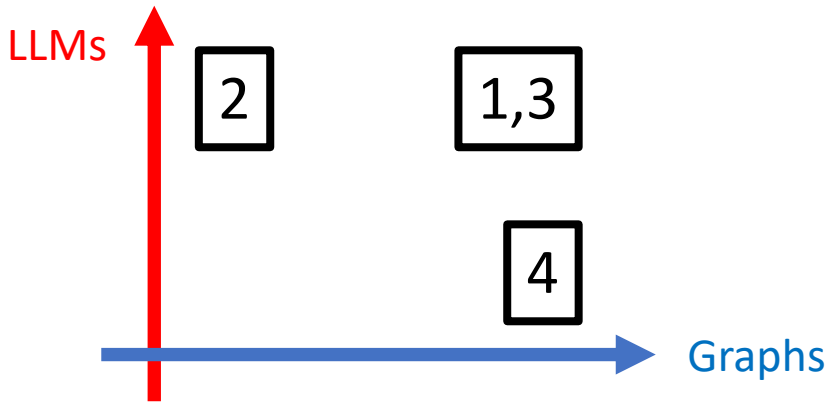
***Integrating LLM with graph data*** is ***a promising direction***
for both graph learning and complex reasoning



# Future directions

- Learning with Graphs
  - explicit with LLMs[1]: LLM-enhanced graph learning, e.g., GraphText on TAG
  - implicit with LLMs[2]: graph prompts for in-context learning, e.g., PRODIGY
- Reasoning with LLMs
  - explicit with Graphs[3]: mount with external graphs, e.g., KG-enhanced reasoning
  - implicit with Graphs[4]: progressively reasoning, e.g., COT / TOT / GOT




We are now collecting and summarizing related works, and found many works are on the way.

It will be released soon :)

# Future directions

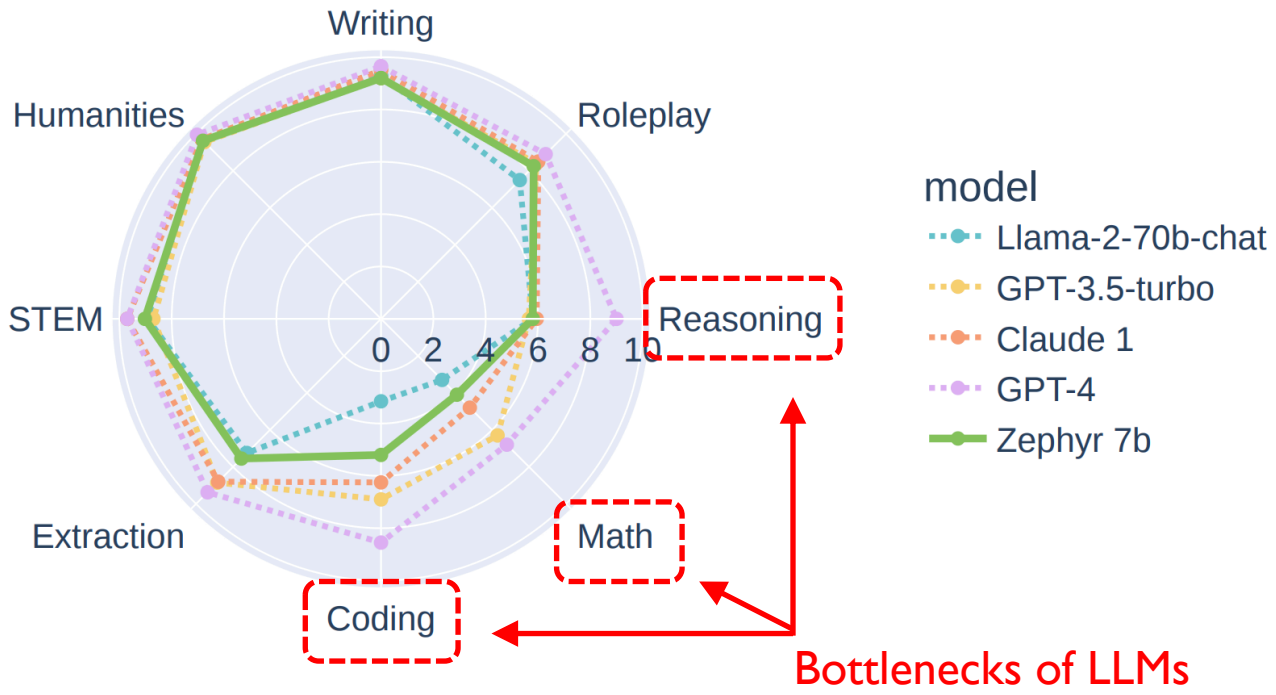
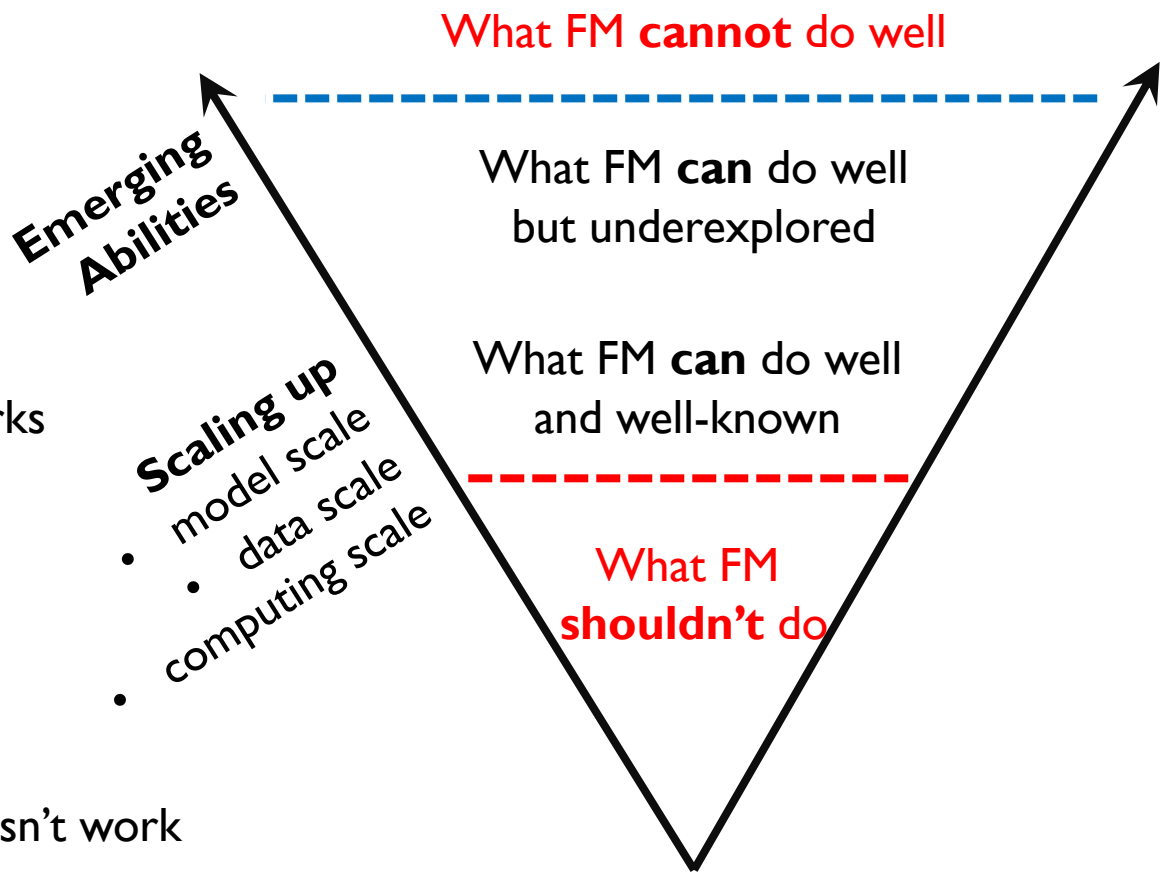



Figure 1: Model performance on MT-Bench. We compare ZEPHYR-7B, trained with distilled direct preference optimization (dDPO), to proprietary models as well as larger, open-access models like LLAMA2-CHAT-70B that were additionally trained using reinforcement learning on a large amount of human feedback.

https://arxiv.org/pdf/2310.16944.pdf

# Research scope

The idea still not works (yet)




The idea works

The idea doesn't work

e.g., algorithmic/complex reasoning

e.g., multi-agent collaboration, predicting future events

e.g., zero/few-shot with in-context learning, traditional supervised learning tasks

e.g., jailbreak, privacy leakage

[1]FM: Foundation Models, including LLM, VLM, etc.

# DeepInception | out recent work

DeepInception

## Hypnotize Large Language Model to Be Jailbreaker



Xuan Li[1*], Zhanke Zhou[1*], Jianing Zhu[1*], Jiangchao Yao[2, 3], Tongliang Liu[4], Bo Han[1],

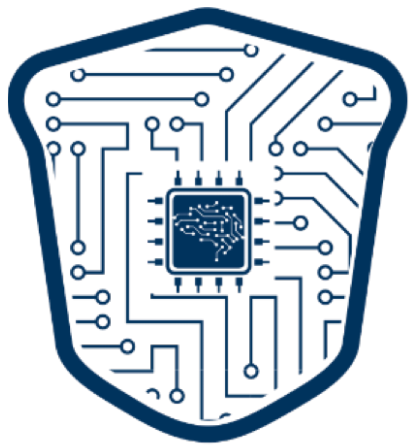
- Paper: https://arxiv.org/pdf/2311.03191.pdf
- Github: https://github.com/tmlr-group/DeepInception
- Project: https://deepinception.github.io/

***Any feedback is Welcome!***

# Q&A

## Thanks for your attention!

Email: cszkzhou@comp.hkbu.edu.hk
Wechat: zhouzhanke924



# TMLR

TRUSTWORTHY MACHINE LEARNING AND REASONING