LLM4DyG: Can Large Language Models Solve Spatial-Temporal Problems on Dynamic Graphs?

Zeyang Zhang
DCST, Tsinghua University
Beijing, China
zy-zhang20@mails.tsinghua.edu.cn

Xin Wang*
DCST, BNRist, Tsinghua University
Beijing, China
xin_wang@tsinghua.edu.cn

Ziwei Zhang
DCST, Tsinghua University
Beijing, China
zw-zhang16@tsinghua.org.cn

Haoyang Li
DCST, Tsinghua University
Beijing, China
lihy218@gmail.com

Yijian Qin
DCST, Tsinghua University
Beijing, China
qinyj19@mails.tsinghua.edu.cn

Wenwu Zhu*
DCST, BNRist, Tsinghua University
Beijing, China
wwzhu@tsinghua.edu.cn

ABSTRACT

In an era marked by the increasing adoption of Large Language Models (LLMs) for various tasks, there is a growing focus on exploring LLMs' capabilities in handling web data, particularly graph data. Dynamic graphs, which capture temporal network evolution patterns, are ubiquitous in real-world web data. Evaluating LLMs' competence in understanding spatial-temporal information on dynamic graphs is essential for their adoption in web applications, which remains unexplored in the literature. In this paper, we bridge the gap via proposing to evaluate LLMs' spatial-temporal understanding abilities on dynamic graphs, to the best of our knowledge, for the first time. Specifically, we propose the **LLM4DyG** benchmark, which includes nine specially designed tasks considering the capability evaluation of LLMs from both temporal and spatial dimensions. Then, we conduct extensive experiments to analyze the impacts of different data generators, data statistics, prompting techniques, and LLMs on the model performance. Finally, we propose Disentangled Spatial-Temporal Thoughts (**DST2**) for LLMs on dynamic graphs to enhance LLMs' spatial-temporal understanding abilities. Our main observations are: 1) LLMs have preliminary spatial-temporal understanding abilities on dynamic graphs, 2) Dynamic graph tasks show increasing difficulties for LLMs as the graph size and density increase, while not sensitive to the time span and data generation mechanism, 3) the proposed **DST2** prompting method can help to improve LLMs' spatial-temporal understanding abilities on dynamic graphs for most tasks. The data and codes are publicly available at Github.

CCS CONCEPTS

• **Information systems** → *Data mining*; • **Computing methodologies** → Natural language processing; *Knowledge representation and reasoning*.

*Corresponding Authors.

KEYWORDS

Dynamic Graph; Large Language Model; Spatial-Temporal; Benchmark; Evaluation; Disentanglement

1 INTRODUCTION

In an era marked by the increasing adoption of Large Language Models (LLMs) for various tasks beyond natural language processing, such as image recognition [1], healthcare diagnostics [58], and autonomous agents [61], there has been a growing body of research dedicated to exploring LLMs' abilities to tackle the vast troves of web data. One area of particular interest is the handling of graph data, which ubiquitously exists on the Internet. The World Wide Web itself can be seen as a colossal interconnected graph of webpages, hyperlinks, and content. For example, social media platforms like Facebook, Twitter, and Instagram generate dynamic social graphs reflecting user interactions and connections.

To leverage the in-context learning and commonsense knowledge of LLMs, several pioneer works have been dedicated to adopting LLMs on static graphs. For instance, Wang et al. [60] and Guo et al. [22] propose benchmarks to evaluate LLMs' proficiency in comprehending and reasoning about graph structures, with tasks like graph connectivity, topological sort, *etc*, demonstrating the LLMs' abilities of in-context learning and reasoning to solve static graph problems. Ye et al. [76] and Chen et al. [13] propose to fine-tune the LLMs to solve graph tasks in natural language, showing the strong potential of LLMs to leverage text information, generate human-readable explanations, and integrate commonsense knowledge to enhance the reasoning over structures.

Dynamic graphs, in comparison with static graphs, possess a wealth of temporal evolution information, which is more prevalent on the internet. For instance, on platforms such as Twitter, users engage in continuous interactions with each other, and on Wikipedia, knowledge graphs are kept updated over time. On the one hand, with the additional temporal dimension, it is possible for LLMs to interpret the ever-changing relationships and information

updates on dynamic graphs, which are ignored in static graphs. On the other hand, there exist additional research challenges for capturing the graph dynamics, and evaluating LLMs' proficiency in comprehending spatial-temporal information is critical for the applications of LLMs on dynamic graphs. Such investigations hold the potential to shed light on broader web applications such as sequential recommendation, trend prediction, fraud detection, *etc.*

To this end, in this paper, we propose to explore the following research question:

Can large language models understand and handle the spatial-temporal information on dynamic graphs in natural language?

However, this problem remains unexplored in literature, and is non-trivial with the following challenges:

- How to design dynamic graph tasks to assess the capabilities of LLMs to understand temporal and structural information both separately and simultaneously.
- How to investigate the impacts of spatial and temporal dimensions, where they have complex and mixed interactions on dynamic graphs.
- How to design the prompts for dynamic graphs and tasks, where spatial-temporal information should be taken into consideration in natural language.

To address these issues, we further propose **LLM4DyG**, a comprehensive benchmark for evaluating the spatial-temporal understanding abilities of LLMs on dynamic graphs. Specifically, we design nine specially designed tasks (illustrated in Figure 1) that consider the capability evaluation from both temporal and spatial dimensions, and question LLMs *when, what* or *whether* the spatial-temporal patterns, ranging from temporal links, and chronological paths to dynamic triadic closure, take place. To obtain a deeper analysis of the impacts of spatial and temporal dimensions for LLMs on dynamic graphs, we make comparisons on these tasks with three data generators (including Erdős-Rényi model [18], stochastic block model [25], and forest fire model [31]), various data statistics (including time span, graph size and density), four general prompting techniques (including zero/one-shot prompting, zero/one-shot chain-of-thoughts prompting [69]), and five LLMs (including closed-source GPT-3.5 and open-source LLMs Vicuna-7B, Vicuna-13B [14], Llama-2-13B [59], and CodeLlama-2-13B [49]). Inspired by the observations and dynamic graph learning literature, we further design a dynamic graph prompting technique, i.e., Disentangled Spatial-Temporal Thoughts (**DST2**), to encourage LLMs to process spatial and temporal information sequentially. We observe the following findings from conducting extensive experiments with **LLM4DyG** :

(1) **LLMs have preliminary spatial-temporal understanding abilities on dynamic graphs.** We find that LLMs significantly outperform the random baseline on the dynamic graph tasks, and the improvements range from +9.8% to +73% on average in Table 2, which shows that LLMs are able to recognize structures and time, and to perform reasoning in dynamic graph tasks.

(2) **Dynamic graph tasks exhibit increasing difficulties for LLMs as the graph size and density grow, while not sensitive to the time span and data generation mechanism.** Specifically, the performance of GPT-3.5 in the 'when link' task

drops from 48% to 27% when the density increases from 0.3 to 0.7, while the performance varies slightly as the time span changes for most tasks in Figure 3. We also find that in the 'when connect' task the performance drops from 97.7% to 17.7% when the graph size increases from 5 to 20 in Table 2.

(3) **Our proposed DST2 prompting technique can help LLMs to improve spatial-temporal understanding abilities.** We find that the results of the existing prompting techniques vary a lot for different tasks in Table 3. Inspired by dynamic graph literature, our proposed **DST2** encourages LLMs to first consider time before nodes, thus improving the performance for most tasks, particularly, from 33.7% to 76.7% in the 'when link' task in Table 6.

To summarize, we make the following contributions:

- We propose to evaluate LLMs' spatial-temporal understanding capabilities on dynamic graphs for the first time, to the best of our knowledge.
- We propose the **LLM4DyG** benchmark to comprehensively evaluate LLMs on dynamic graphs. **LLM4DyG** consists of nine dynamic graph tasks in natural language with considerations of both temporal and spatial dimensions, ranging from temporal links, and chronological paths to dynamic triadic closure and covering questions regarding *when, what* or *whether* for LLMs.
- We conduct extensive experiments taking into account three data generators, three graph parameters, four general prompts, and five different LLMs. Based on the experiments, we provide fine-grained analyses and observations about the evaluation of LLMs on dynamic graphs.
- We propose a Disentangled Spatial-Temporal Thoughts (**DST2**) prompting technique. Experimental results show that it can greatly improve the spatial-temporal reasoning ability of LLMs.

2 RELATED WORK

2.1 LLMs for tasks with graph data

Recently, there has been a surge of works about LLMs for solving tasks with graph data [29, 39, 81, 82]. He et al. [24] proposed an approach that LLMs not only execute zero-shot predictions but also generate coherent explanations for their decisions. These explanations are subsequently leveraged to enhance the features of graph nodes for node classification in text-attributed graphs. Chen et al. [13] proposes to explore LLMs-as-Enhancers and LLMs-as-Predictors for solving graph-related tasks, where the former augment the GNN with LLMs, and the latter directly adopts LLMs to make predictions. Wang et al. [60] introduced NLGraph, a benchmarking framework tailored for evaluating the performance of LLMs on traditional graph-related tasks. Simultaneously, Guo et al. [22] conducted a comprehensive empirical study focused on utilizing LLMs to tackle structural and semantic understanding tasks within graph-based contexts. Recent contributions in this line of research include InstructGLM [76], a method for fine-tuning LLMs inspired by LLaMA [59], designed specifically for node classification tasks. Zhang [80] and Jiang et al. [28] have initiated the exploration of this frontier by interfacing LLMs with external tools and enhancing their reasoning capabilities over structured data sources such as knowledge graphs (KGs) and tables. Yao et al. [75] explores

Temporal

When link

Question: Given an undirected dynamic graph with the edges [(1, 2, 0), (0, 1, 1), (3, 4, 4)]. When are node 0 and node 1 linked?
Answer: 1

When connect

Question: Given an undirected dynamic graph with the edges [(1, 2, 0), (0, 1, 1), (2, 3, 2), (3, 4, 4)]. When are node 0 and node 3 first connected?
Answer: 2

When triadic closure

Question: Given an undirected dynamic graph with the edges [(1, 2, 0), (0, 1, 1), (2, 0, 2), (3, 4, 4)]. When are node 0, 1 and 2 first close the triad?
Answer: 2

Spatial

What neighbors at time

Question: Given an undirected dynamic graph with the edges [(1, 2, 1), (0, 1, 1), (3, 4, 4)]. What nodes are linked with node 1 at time 1?
Answer: [0, 2]

What neighbors in periods

Question: Given an undirected dynamic graph with the edges [(1, 2, 0), (2, 0, 1), (2, 3, 2)]. What nodes are linked with node 2 at or after time 1?
Answer: [0, 3]

Check triadic closure

Question: Given an undirected dynamic graph with the edges [(1, 2, 0), (0, 1, 1), (2, 0, 2), (3, 4, 4)]. Did node 0, 1 and 2 form a closed triad?
Answer: Yes

Spatial-Temporal

Check temporal path

Question: Given an undirected dynamic graph with the edges [(1, 2, 1), (0, 1, 1), (3, 4, 4)]. Did nodes 0, 1, 2 form a chronological path?
Answer: Yes

Find temporal path

Question: Given an undirected dynamic graph with the edges [(1, 2, 0), (2, 0, 1), (2, 3, 2)]. Find a chronological path starting from node 1.
Answer: [1, 2, 3]

Sort edge by time

Question: Given an undirected dynamic graph with the edges [(2, 0, 2), (3, 4, 4), (1, 2, 0), (0, 1, 1)]. Sort the edges by time from earliest to latest.
Answer: [(1, 2, 0), (0, 1, 1), (2, 0, 2), (3, 4, 4)].

u Node   u Target Node   u —t— v Temporal Link   u —t— v Target Time   Temporal Triadic Closure

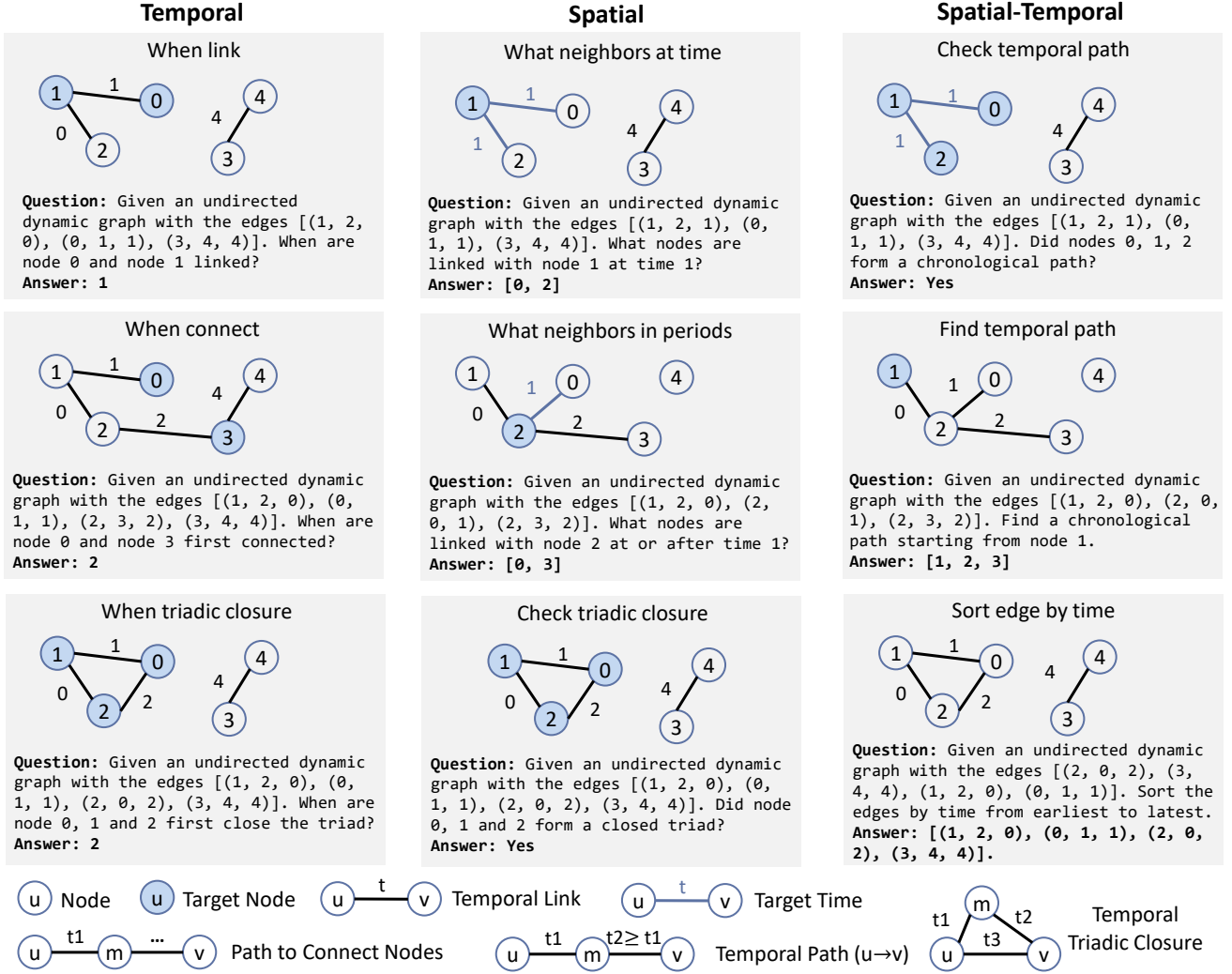u —t1— m —…— v Path to Connect Nodes   u —t1— m —t2≥t1— v Temporal Path (u→v)

Figure 1: An overview of the tasks in the LLM4DyG Benchmark. The tasks are designed to consider both temporal and spatial dimensions, and question LLMs in natural language *when, what* or *whether* the spatial-temporal patterns take place. The spatial-temporal patterns range from temporal links, and chronological paths to dynamic triadic closure. The tasks are classified based on the targets of the queries. An example prompt and graph illustration are provided for each task.

leveraging LLMs for graph generation, which have potentials for various real-world tasks like drug discoveries. Li et al. [37] utilizes LLMs to recognize and provide the possible latent causal structures given the rich domain knowledge in the textual corpus. However, these works mainly focus on static graphs, ignoring the temporal nature of graphs in real-world web applications. In this paper, we propose to explore LLMs' spatial-temporal understanding on dynamic graphs, which remains unexplored in the literature.

2.2 LLMs for other related tasks

LLMs have been recently applied to other related tasks, including time-series forecasting, recommendation, *etc.* Yu et al. [79] presents a novel study on harnessing LLMs' outstanding knowledge and reasoning abilities for explainable financial time series forecasting. Chang et al. [7] leverages pre-trained LLMs to enhance time-series forecasting and has shown exceptional capabilities as both a robust representation learner and an effective few-shot learner. Sun et al. [55] summarizes two strategies for completing time-series (TS) tasks using LLM: LLM-for-TS that designs and trains a fundamental large model for TS data and TS-for-LLM that enables the pre-trained LLM to handle TS data. Feng et al. [20] and [77] evaluate the temporal or sequential understanding abilities of LLMs in visual tasks. Lyu et al. [41] investigates various prompting strategies for enhancing personalized recommendation performance with large language models through input augmentation. However, these works do not consider the role of structures, and in this paper, we mainly focus on exploring the spatial-temporal understanding abilities of LLMs on dynamic graphs.
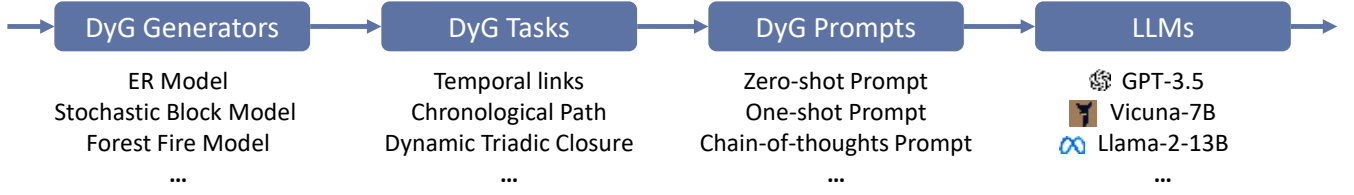
DyG Generators	DyG Tasks	DyG Prompts	LLMs
ER Model	Temporal links	Zero-shot Prompt	🪙 GPT-3.5
Stochastic Block Model	Chronological Path	One-shot Prompt	🤖 Vicuna-7B
Forest Fire Model	Dynamic Triadic Closure	Chain-of-thoughts Prompt	∞ Llama-2-13B
...	...	...	...

Figure 2: An overview of the pipeline in the LLM4DyG Benchmark, which includes various dynamic graph generators, tasks, prompt methods, and LLMs for evaluation.

2.3 Dynamic Graph Learning

Dynamic graphs are pervasive in a multitude of real-world applications, spanning areas such as event forecasting, recommendation systems, *etc* [6, 16, 32, 68, 70, 78]. This prevalence has prompted significant research interest in the development and refinement of dynamic graph neural networks [8, 54, 91, 95]. These networks are designed to model intricate graph dynamics, which incorporate evolving structures and features over time. A variety of approaches have been proposed to address the challenges posed by dynamic graphs. Some research efforts have focused on employing Graph Neural Networks (GNNs) to aggregate neighborhood information for each individual snapshot of the graph. Subsequently, these methods use a sequence module to capture and model the temporal information [23, 50, 52, 56, 73]. In contrast, other studies have proposed the use of time-encoding techniques. These methods encode the temporal links into specific time-aware embeddings, and then utilize a GNN or memory module [15, 48, 67, 71] to process and handle the structural information embedded in the graph. Recently, there are some works focusing on studying dynamic graphs under distribution shifts [83, 85–87]. However, these methods require the model to be trained every time they encounter a new dynamic graph task, limiting their widespread usage in real-world scenarios. In this paper, we explore the potential of LLMs on solving dynamic graph tasks with in-context learning skills and evaluate their spatial-temporal understanding abilities on dynamic graphs.

3 THE LLM4DYG BENCHMARK

In this section, we introduce our proposed **LLM4DyG** benchmark to evaluate whether LLMs are capable of understanding spatial-temporal information on the dynamic graph. Specifically, we first adopt a random dynamic graph generator to generate the base dynamic graphs with controllable parameters like time span. Then, we design nine dynamic graph tasks to evaluate LLMs' abilities considering both spatial and temporal dimensions. The overall pipeline is illustrated in Figure 2. Based on this pipeline, we can control the data generation, statistics, prompting methods, and LLMs for each task to conduct fine-grained analyses.

3.1 Dynamic Graph Data Generators

We first adopt a random dynamic graph data generator to control the statistics of the dynamic graph. In default, we adopt an Erdős-Rényi (ER) model to generate an undirected graph, and randomly assign a time-stamp for each edge. Denote a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with the node set $\mathcal{V} = \{v_1, v_2, \ldots v_N\}$ and edge set $\mathcal{E} = \{e_1, e_2, \ldots, e_M\}$. We first generate the graph with the ER model $\mathcal{G} = ER(N, p)$ where

N is the number of nodes in the graph, and p is the probability of edge occurrence between each node pair. In this way, N controls the graph size, and p controls the graph density. After obtaining the graph $\mathcal{G}$, we assign each edge with a random timestamp $t \sim U(\{0, 1, \ldots, T-1\})$, where T controls the time span. For a generated dynamic graph, each edge $e = (v_i, v_j, t)$ denotes that node v_i and node v_j are linked at time t. We also include other dynamic graph generators, stochastic block (SB) model, and forest fire (FF) model. For real-world datasets, we adopt a random sampler to sample ego-graphs from the real graphs for evaluation.

3.2 Dynamic Graph Tasks

To evaluate LLMs' spatial-temporal understanding abilities, we design nine tasks considering both temporal and spatial dimensions. The tasks are classified based on the targets of the queries, *e.g.*, the temporal tasks make queries about the time, the spatial tasks make queries about the nodes, while the solutions in spatial-temporal tasks are more complex and include the spatial-temporal patterns mixed together. We introduce the definition and generation of each task as follows.

- **Temporal Task 1: when link.** We ask when two nodes are linked in this task. In a dynamic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, two nodes u and v are linked at time t if there exists a temporal edge (u, v, t) in the edge set $\mathcal{E}$. We randomly select an edge from the edge set as the query.

- **Temporal Task 2: when connect.** We ask when two nodes are connected in this task. In a dynamic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, two nodes u and v are connected at time t if there exists a path $[(u, k_1, t), (k_1, k_2, t), \ldots, (k_i, v, t)]$ from node u to node v at time t in the edge set $\mathcal{E}$. We randomly select a pair of nodes that are connected at some time as the query.

- **Temporal Task 3: when triadic closure (tclosure).** We ask when the three given nodes first form a closed triad in this task. Dynamic triadic closure has been shown critical for dynamic graph analyses [94]. In a dynamic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, two nodes with a common neighbor are said to have a triadic closure, if they are linked since some time so that the three nodes have linked with each other to form a triad. We randomly select a closed triad as the query.

Note that while these temporal tasks focus on making queries about time, they also require the model to understand structures so that the model can recognize *when* some structural patterns exist, from links and paths to dynamic triads. Next, we introduce the spatial tasks that require the model to spot the specific time and discover the structures.

- **Spatial Task 1: what neighbor at time.** In this task, we ask what nodes are linked with a given node at a given time. We randomly select a time and a node not isolated in the time-related graph snapshot to construct the query.
- **Spatial Task 2: what neighbor in periods.** In this task, we ask what nodes are linked with a given node after or at a given time, but not linked before the given time. We randomly select a time and a node not isolated before the given time to construct the query. This task measures the model's abilities to understand structures within a time period, *e.g.*, the latest links.
- **Spatial Task 3: check triadic closure (tclosure).** We ask whether the three given nodes form a closed triad in the dynamic graph through true/false questions. We uniformly sample from the sets of closed triads and open triads to construct the positive and negative samples respectively. We also keep a balanced number of positive and negative samples in the dataset.

Similarly, these tasks also require the model to spot the time, from a specific time and time period to the full-time span, and then to recognize *what* structural patterns or *whether* the given structural patterns meet the requirements of the queries. Next, we introduce the spatial-temporal tasks that directly require the LLMs to process the spatial-temporal targets.

- **Spatial-Temporal Task 1: check temporal path (tpath).** In this task, we ask whether the given three ordered nodes form a chronological path. In a dynamic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a sequence of nodes $[v_1, v_2, \ldots, v_n]$ construct a chronological path if the timestamps of the edges do not decrease from source node v_1 to target node v_n in the path. We randomly select positive and negative samples from the set of chronological paths and non-chronological paths to construct a balanced dataset.
- **Spatial-Temporal Task 2: find temporal path (tpath).** In this task, we ask the model to find a chronological path starting from a given node in the dynamic graph. We randomly select a node that is a starting node at any chronological path to construct the queries. Note that any valid chronological path starting at the given node is a correct answer.
- **Spatial-Temporal Task 3: sort edge by time.** In this task, we shuffle the edges and ask the model to sort the edges by time from earliest to latest. In the cases where some edges have the same timestamp, the orders within these edges do not matter for the correct answers.

These tasks require the model to understand the spatial-temporal information at the local or global scale. The targets of the queries include both temporal and spatial information on the dynamic graph. The example prompts and illustrations are shown in Fig. 1.

4 EXPERIMENTS

In this section, we conduct experiments to evaluate LLMs' spatial-temporal understanding abilities on dynamic graphs. We conduct fine-grained analyses with various settings from different aspects, including data, prompting methods, models, *etc.*

4.1 Setups

Random baseline. To verify whether the model can understand dynamic graphs instead of outputting random answers, we adopt a random baseline that uniformly selects one of the possible solutions

Table 1: An example of prompt construction for the 'when connect' task.

Prompt	Example
DyG Instruction	In an undirected dynamic graph, (u, v, t) means that node u and node v are linked with an undirected edge at time t.
Task Instruction	Your task is to answer when two nodes are first connected in the dynamic graph. Two nodes are connected if there exists a path between them.
Answer Instruction	Give the answer as an integer number at the last of your response after 'Answer:'
Exemplar	Here is an example: Question: Given an undirected dynamic graph with the edges [(0, 1, 0), (1, 2, 1), (0, 2, 2)]. When are node 0 and node 2 first connected? Answer:1
Question	Question: Given an undirected dynamic graph with the edges [(0, 9, 0), (1, 9, 0), (2, 5, 0), (1, 2, 1), (2, 6, 1), (3, 7, 1), (4, 5, 2), (4, 7, 2), (7, 8, 2), (0, 1, 3), (1, 6, 3), (5, 6, 3), (0, 4, 4), (3, 4, 4), (3, 6, 4), (4, 6, 4), (4, 9, 4), (6, 7, 4)]. When are node 2 and node 1 first connected?
Answer	Answer:1

as the answer. The accuracies of the random baseline for the tasks can be calculated by the ratio of the number of correct solutions over the number of possible solutions, which are specifically provided as follows. For the tasks 'when connect' and 'when tclosure', the baseline accuracy is $\frac{1}{T}$. For the task 'when link', the baseline accuracy is $\frac{1}{\sum_i^T C(T,i)}$, where $C(T, i)$ is the combination number. For the tasks 'neighbor at time' and 'neighbor in periods', the baseline accuracy is $\frac{1}{\sum_i^N C(N,i)}$. For the tasks 'check tclosure' and 'check tpath', the baseline accuracies are 1/2, as the answer is either no or yes. For the tasks 'find tpath' and 'check tpath', the baseline accuracies are calculated by enumerating possible solutions and correct solutions for each instance.

Prompting methods. To investigate how different prompting techniques affect the model's abilities, we compare various prompting methods, including zero-shot prompting, few-shot prompting [5], chain-of-thought prompting (COT) [69] and few-shot prompting with COT. We adopt one example for few-shot prompting, and use one-shot prompting as the default prompting approach. For each problem instance, the prompt is constructed by sequentially concatenating dynamic graph instruction, task instruction, answer instruction, exemplar prompts, and question prompts. An example of the prompt construction is shown in Table 1.

Models. We use GPT-3.5-turbo-instruct as the default LLM, and we also include other LLMs like Vicuna-7B, Vicuna-13B, Llama-2-13B and CodeLlama-2-13B. For all models, we set temperature $\tau = 0$ for reproducibility. We adopt accuracy as the metric for all tasks.

Data. In default settings, we set $N = 10$, $p = 0.3$, $T = 5$, and ER model for generating dynamic graphs. For each task and setting, we randomly generate one hundred problem instances for evaluation.

We run the experiments three times with different seeds, and report the average performance and their standard deviations.

4.2 Results with data of different statistics

We first compare GPT-3.5 on each task with different graph sizes, where N is set to 5, 10 and 20 respectively. From Table 2, we have the following observations.

Observation 1. LLMs have preliminary spatial-temporal understanding abilities on dynamic graphs.

As shown in Table 2, on average, GPT-3.5 has shown significant performance improvement (from +9.8% to +73.0%) over the baseline for all tasks, indicating that LLMs indeed understand the dynamic graph as well as the question in the task, and are able to exploit spatial-temporal information to give correct answers instead of guessing by outputting randomly generated answers. Overall, we can find that LLMs have the ability to recognize time, structures, and spatial-temporal patterns.

Observation 2. Most dynamic graph tasks exhibit increasing difficulty for LLMs as the graph size grows.

As shown in Table 2, for most tasks, the performance of GPT-3.5 drops as the graph size N increases. For example, the performance drops from 97.7% to 17.7% on 'when connect' task, and 42.3% to 2.0% on 'neighbor in periods' task. This phenomenon may be due to two factors: 1) From the task perspective, the solution space is enlarged so that it is harder for any model to obtain the correct solution, *e.g.*, the accuracy of the random baseline also drops significantly on 'sort edge' task. 2) From the model perspective, it is harder for the model to retrieve the useful information inside the data since the input space is enlarged, *e.g.*, on 'when connect' task, the performance drops drastically while the solution space remains the same. This observation shows that it is worthy of exploring handling larger dynamic graph contexts with LLMs.

We then compare GPT-3.5 on each task with different time span T and density p, where T is set to 10, 20, and 30 respectively, and p is set to 0.3, 0.5, and 0.7 respectively. From Figure 3, we have the following observations.

Observation 3. For LLMs, the difficulties of dynamic graph tasks are not sensitive to the time span but sensitive to the graph density.

As shown in Figure 3, for most tasks, the model performance is close as the time span T increases while the density p remains the same. If we keep the time span T the same and increase the density p, the model performance drops for most tasks. One exception is the task 'find tpath' where the model performance increases as the two factors increase. Another interesting finding from the heatmap is that LLMs are relatively more sensitive with the time span T in temporal tasks while the density p in spatial tasks, possibly due to the different points of focus for these tasks. It can be also observed in spatial-temporal tasks, where the model performance mainly changes along with the diagonal of the time span T and density p.

To investigate how the performance of LLMs varies when the task requires additional temporal information other than only structural information, we make comparisons with different time span
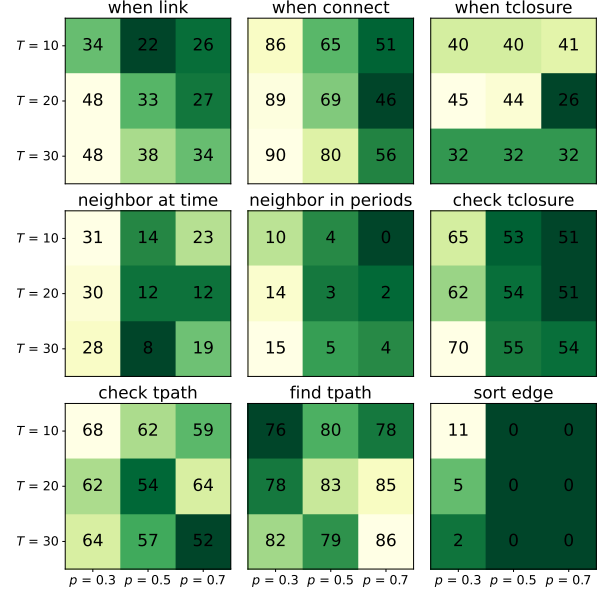

Figure 3: Performance comparisons (ACC%) on the dynamic graph tasks with different density p and time span T. (Best viewed in color)
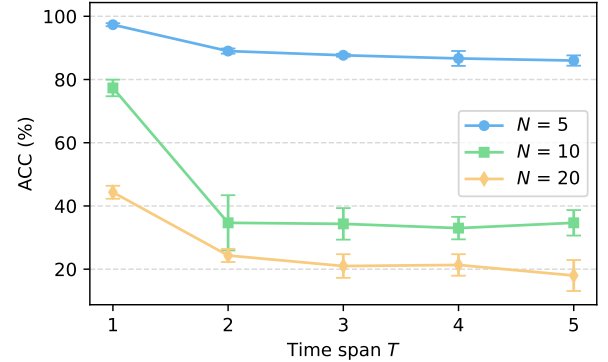

Figure 4: Performance of GPT-3.5 on the 'neighbor at time' task as the time span T increases with different network sizes N. Note that when $T = 1$, the data degenerates to a static graph, since there is only one timestamp on the graph.

T and graph size N on the 'neighbor at time' task. We have the following observation.

Observation 4. Temporal information adds additional difficulties to LLMs in comparisons with static graphs.

As shown in Figure 4, GPT-3.5 has a drastic performance drop when the time span T increases from 1 to 2. The possible reason is that the task is changed from static to dynamic, serving as a more challenging setting, since the model has to capture the additional temporal information. Similar to the results from Figure 3, the model performance is not sensitive to the time span when the task is already a dynamic graph problem.

Table 2: The overall model performance (ACC%) on the dynamic graph tasks. In the data column, 'N' denotes the number of nodes in the dynamic graph. In the model column, 'Random' denotes the random baseline which uniformly outputs one of the possible solutions, and '∆' denotes the performance improvement of GPT-3.5 over the random baseline.

Task		Temporal			Spatial			Spatial-Temporal		
Data	model	when link	when connect	when tclosure	neighbor at time	neighbor in periods	check tclosure	check tpath	find tpath	sort edge
N = 5	GPT-3.5	$68.0_{\pm2.8}$	$97.7_{\pm0.9}$	$52.7_{\pm2.4}$	$86.0_{\pm2.2}$	$42.3_{\pm1.7}$	$69.0_{\pm2.2}$	$58.7_{\pm2.1}$	$79.0_{\pm4.1}$	$78.0_{\pm1.4}$
	Random	3.2	20.0	20.0	3.2	3.2	50.0	50.0	9.3	13.1
	∆	+64.8	+77.7	+32.7	+82.8	+39.1	+19.0	+8.7	+69.7	+64.9
N = 10	GPT-3.5	$33.7_{\pm2.1}$	$77.0_{\pm2.9}$	$73.0_{\pm1.6}$	$34.0_{\pm1.4}$	$15.7_{\pm4.2}$	$66.7_{\pm4.5}$	$63.7_{\pm2.6}$	$78.3_{\pm6.0}$	$29.3_{\pm4.0}$
	Random	3.2	20.0	20.0	0.1	0.1	50.0	50.0	6.7	0.0
	∆	+30.4	+57.0	+53.0	+33.9	+15.6	+16.7	+13.7	+71.6	+29.3
N = 20	GPT-3.5	$40.3_{\pm1.7}$	$17.7_{\pm4.2}$	$63.3_{\pm0.9}$	$17.7_{\pm1.7}$	$2.0_{\pm0.8}$	$64.3_{\pm7.3}$	$57.0_{\pm2.2}$	$85.0_{\pm0.8}$	$0.0_{\pm0.0}$
	Random	3.2	20.0	20.0	0.0	0.0	50.0	50.0	7.3	0.0
	∆	+37.1	-2.3	+43.3	+17.7	+2.0	+14.3	+7.0	+77.7	0.0
Avg.	GPT-3.5	$47.3_{\pm1.2}$	$64.1_{\pm0.3}$	$63.0_{\pm1.0}$	$45.9_{\pm3.1}$	$20.0_{\pm0.8}$	$66.7_{\pm2.9}$	$59.8_{\pm0.8}$	$80.8_{\pm0.3}$	$35.8_{\pm2.0}$
	Random	3.2	20.0	20.0	1.1	1.1	50.0	50.0	7.8	4.4
	∆	+44.1	+44.1	+43.0	+44.8	+18.9	+16.7	+9.8	+73.0	+31.4

Table 3: Performance comparisons (ACC%) of various prompting methods on the dynamic graph tasks. 'Random' denotes the random baseline which uniformly outputs one of the possible solutions. The best result for each task is in bold.

Task	Temporal			Spatial			Spatial-Temporal		
Prompt Method	when link	when connect	when tclosure	neighbor at time	neighbor in periods	check tclosure	check tpath	find tpath	sort edge
zero-shot	$2.3_{\pm0.5}$	$73.3_{\pm2.1}$	$68.0_{\pm0.8}$	$\mathbf{36.0_{\pm4.3}}$	$4.3_{\pm2.1}$	$\mathbf{70.7_{\pm1.7}}$	$\mathbf{66.0_{\pm5.4}}$	$56.3_{\pm9.0}$	$33.7_{\pm7.4}$
one-shot	$\mathbf{33.7_{\pm2.1}}$	$77.0_{\pm2.9}$	$73.0_{\pm1.6}$	$34.0_{\pm1.4}$	$\mathbf{15.7_{\pm4.2}}$	$66.7_{\pm4.5}$	$63.7_{\pm2.6}$	$78.3_{\pm6.0}$	$29.3_{\pm4.0}$
zero-shot COT	$1.0_{\pm0.8}$	$58.3_{\pm1.2}$	$70.0_{\pm1.6}$	$32.0_{\pm0.8}$	$4.3_{\pm2.6}$	$55.0_{\pm1.4}$	$62.3_{\pm2.9}$	$58.0_{\pm9.1}$	$\mathbf{44.7_{\pm0.5}}$
one-shot COT	$10.3_{\pm0.5}$	$76.0_{\pm2.4}$	$\mathbf{80.0_{\pm1.6}}$	$27.7_{\pm1.9}$	$13.0_{\pm3.6}$	$57.7_{\pm2.1}$	$57.7_{\pm3.4}$	$\mathbf{81.3_{\pm2.6}}$	$24.7_{\pm2.4}$

4.3 Results with different prompting methods

We then make comparisons with different prompting methods, including zero-shot prompting, one-shot prompting, zero-shot chain-of-thoughts, and one-shot chain-of-thoughts. From Tab. 3, we have the following observations.

Observation 5. General advanced prompting techniques do not guarantee a performance boost in tackling spatial-temporal information.

As shown in Table 3, some advanced prompting methods like zero-shot COT and one-shot COT achieve higher performance than other prompting methods in the tasks 'when tclosure', 'find tpath' and 'sort edge'. Note that these tasks involve more complex dynamic graph concepts or have to tackle a large time span, which shows that the chain-of-thoughts method can, to some extent, activate the model's reasoning ability by thinking step by step on complex tasks. However, no prompting methods consistently achieve the best performance on all tasks, which calls for the need to design special advanced prompting methods to boost LLMs' performance in handling spatial-temporal information on dynamic graphs.
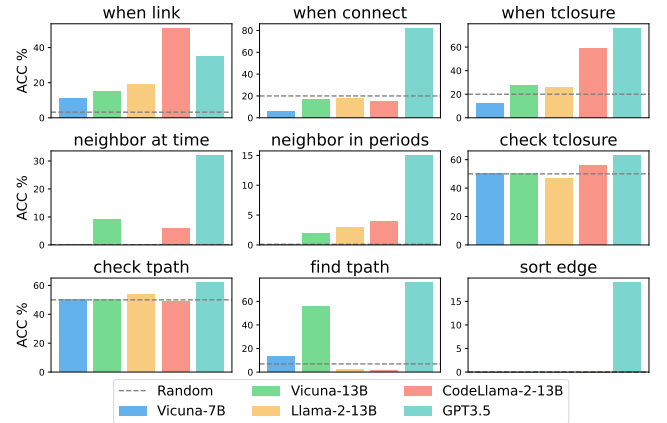

Figure 5: Performance comparisons (ACC%) of various LLMs on the dynamic graph tasks. 'Random' denotes the random baseline which uniformly outputs one of the possible solutions. (Best viewed in color)

Table 4: Valid rate (%) of different LLMs. An answer is judged as valid if it meets the requirement of the answer template and can be parsed by the evaluator program.

Valid Rate	when link	when connect	when tclosure	neighbor at time	neighbor in periods	sort edge
Vicuna-7B	100	100	100	100	99	10
Vicuna-13B	92	89	100	93	97	92
Llama-2-13B	99	97	95	21	91	74
CodeLlama-2-13B	100	100	100	100	96	100
GPT-3.5	100	100	100	100	100	100

Table 5: Performance comparisons (ACC%) of various prompting methods and dynamic graph generation models on the 'when link' task.

Generation Model	ER Model	SB Model	FF Model
zero-shot	$2.3_{\pm0.5}$	$7.7_{\pm1.7}$	$5.3_{\pm2.5}$
one-shot	$\mathbf{33.7}_{\pm2.1}$	$\mathbf{46.0}_{\pm2.9}$	$\mathbf{48.0}_{\pm7.1}$
zero-shot COT	$1.0_{\pm0.8}$	$5.7_{\pm3.1}$	$2.0_{\pm1.6}$
one-shot COT	$10.3_{\pm0.5}$	$15.3_{\pm0.9}$	$13.0_{\pm2.9}$

4.4 Results with different LLMs

We then make comparisons with different LLMs, including GPT-3.5, Llama-2-13B, Vicuna-7B, Vicuna-13B and CodeLlama-2-13B. From Figure 5, we have the following observations.

Observation 6. LLMs' abilities on dynamic graph tasks are related to the model scale.

As shown in Figure 5, smaller LLMs like vicuna-7B and Llama-2-13B have performance lower than GPT-3.5 for all tasks, and even lower than random baseline for several tasks like 'when connect' and 'check tclosure'. Overall, for these tasks, larger LLMs have better performance.

To further investigate whether the lower performance stems from the incompetence of understanding instructions or performing reasoning, we show the valid rate of the answers given by different LLMs in several tasks. An answer is judged as valid if it meets the requirement of the answer template and can be parsed by the evaluator program. From Table 4, we find that 1) smaller models have significantly lower valid rates for some tasks, *e.g.*, 21% of Llama-2-13B in the task 'neighbor at time', demonstrating their limitations in understanding human instructions for dynamic graph tasks. 2) In some tasks, the smaller models have high valid rates, while having significantly lower performance than GPT-3.5, showing their limitations in reasoning for dynamic graph tasks.

Observation 7. Training on codes may help LLMs tackle on dynamic graph tasks.

As shown in Figure 5, compared with Llama-2-13B, CodeLlama-2-13B shows significantly better results in most tasks. In particular, CodeLlama-2-13B even outperforms GPT-3.5 in the task 'when link'. Note that in comparison with Llama-2-13B, CodeLlama-2-13B is further pretrained on a large corpus of code data, which shows the potential of improving the performance of LLMs on dynamic graph tasks by training with codes. One possible reason is that the code data covers more implicit knowledge of structures

and sequences, *e.g.*, the control flows of the programs and their comments as explanations, which might be useful for LLMs to understand dynamic graphs.

4.5 Results with different data generators

We make comparisons with various prompting methods and dynamic graph generation models, including Erdős–Rényi (ER) model, Stochastic Block (SB) model, and Forest Fire (FF) model, on the 'when link' task. To keep the number of edges similar, we set the class number as 2, the in-class probability as 0.4, the cross-class probability as 0.2 for SB model, and the forward burning probability as 0.5 for FF model.

Observation 8. General prompting methods have consistent performance with different dynamic graph generators in the same task.

As shown in Table 5, the 'one-shot' prompt method consistently achieves the best performance with different dynamic graph generation models in the 'when link' task. The results indicate that the evaluation of different prompting methods on dynamic graphs may not be closely related to the dynamic graph generators.

4.6 Exploring advanced dynamic graph prompts

In this section, we aim to explore advanced dynamic graph prompting techniques to improve the reasoning ability of LLMs on dynamic graphs. The chain-of-thoughts prompting is shown as a general advanced prompting technique to activate LLMs' complex reasoning abilities, while it does not effectively improve performance on dynamic graphs as shown in Table 3.

To have further developments, we draw inspiration from dynamic graph learning literature where most works tackle spatial-temporal information separately, *e.g.*, to tackle time first and then structures, or to tackle structures first and then time. Intuitively, this thought breaks down the complex spatial-temporal information into two separate dimensions so that the difficulty can be decreased. To this end, we propose Disentangled Spatial-Temporal Thoughts (**DST2**) to improve LLMs' reasoning abilities on dynamic graphs, that is to instruct LLMs to sequentially think about the nodes or time. Specifically, we design several prompts and add the prompts after the task instruction in the one-shot prompt, which are denoted as 'v1' to 'v4' respectively in Table 6.

Observation 9. The prompting of instructing LLMs to separately tackle spatial and temporal information significantly improves the performance.

As shown in Table 6, the prompt 'v4' achieves the accuracy of 76.7% in the 'when link' task, significantly surpassing the one-shot prompt (33.7%), showing that guiding the LLM to handle time before nodes may help the model improve the spatio-temporal understanding ability on dynamic graphs. For spatial tasks, it seems that it would be better for the LLM to think about spatial information before temporal information (*e.g.*, the prompt 'v1' achieves 69.3% in the 'check tclosure' task). While our proposed methods provide performance gains in most tasks, there exist some tasks that are not positively affected. Designing specific prompting methods for LLMs on dynamic graphs is still an open research question.

Table 6: Model performance (ACC%) on the dynamic graph tasks with one-shot prompting method and our proposed DST2 prompting methods (v1 to v4). The best and the second-best results for each task are in bold and underlined respectively.

Task	Temporal			Spatial			Spatial-Temporal		
Prompting methods	when link	when connect	when tclosure	neighbor at time	neighbor in periods	check tclosure	check tpath	find tpath	sort edge
one-shot prompt	33.7±2.1	77.0±2.9	73.0±1.6	34.0±1.4	15.7±4.2	66.7±4.5	**63.7±2.6**	78.3±6.0	29.3±4.0
v1: Think (about) nodes and then time	40.0±1.6	77.0±4.1	**74.0±1.4**	34.0±0.8	15.0±4.2	**69.3±1.7**	61.0±3.3	**79.0±7.5**	30.0±3.6
v2: Think (about) time and then nodes	37.3±2.6	76.7±3.4	73.3±0.5	31.7±1.9	15.7±3.4	67.0±2.9	61.3±1.9	**79.0±7.5**	**30.7±3.9**
v3: Pick nodes and then time	59.3±2.1	77.0±2.4	68.0±0.8	35.0±2.9	**16.7±4.7**	65.0±3.7	62.3±2.9	78.0±5.4	30.0±2.9
v4: Pick time and then nodes	**76.7±1.7**	76.3±3.9	68.7±0.9	**35.7±2.5**	15.3±3.3	65.3±2.9	63.3±2.6	78.3±5.8	29.3±2.9

5 CONCLUSION

In this paper, we propose a novel **LLM4DyG** benchmark to evaluate LLMs' spatial-temporal understanding capabilities on dynamic graphs, which remains unexplored in literature. The proposed benchmark encompasses nine specially devised tasks, which assess the capabilities of LLMs to handle both temporal and spatial information on dynamic graphs. The evaluation procedure involves a diverse range of LLMs, prompting techniques, data generators, and data statistics. We also propose Disentanlged Spatio-Temporal Thoughts (**DST2**) as an advanced prompting method to enhance reasoning capabilities by guiding LLMs to think about time and structures separately. Through comprehensive experiments, we provide nine fine-grained observations that would be helpful for understanding LLMs' reasoning abilities on dynamic graphs. We hope that future work can be developed based on our proposed benchmark and observations.

ACKNOWLEDGMENTS

REFERENCES

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. 2022. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems* 35 (2022), 23716–23736.

[2] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609* (2023).

[3] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35, 8 (2013), 1798–1828.

[4] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *AAAI*, Vol. 38. 17682–17690.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*. 1877–1901.

[6] Lei Cai, Zhengzhang Chen, Chen Luo, Jiaping Gui, Jingchao Ni, Ding Li, and Haifeng Chen. 2021. Structural temporal graph neural networks for anomaly detection in dynamic graphs. In *Proceedings of the 30th ACM international conference on Information & Knowledge Management*. 3747–3756.

[7] Ching Chang, Wen-Chih Peng, and Tien-Fu Chen. 2023. LLM4TS: Two-Stage Fine-Tuning for Time-Series Forecasting with Pre-Trained LLMs. *arXiv preprint arXiv:2308.08469* (2023).

[8] Chao Chen, Haoyu Geng, Nianzu Yang, Xiaokang Yang, and Junchi Yan. 2023. EasyDGL: Encode, Train and Interpret for Continuous-time Dynamic Graph Learning. *arXiv preprint arXiv:2303.12341* (2023).

[9] Hong Chen, Yudong Chen, Xin Wang, Ruobing Xie, Rui Wang, Feng Xia, and Wenwu Zhu. 2021. Curriculum Disentangled Recommendation with Noisy Multifeedback. *NeurIPS* 34 (2021), 26924–26936.

[10] Hong Chen, Xin Wang, Yipeng Zhang, Yuwei Zhou, Zeyang Zhang, Siao Tang, and Wenwu Zhu. 2024. DisenStudio: Customized Multi-subject Text-to-Video Generation with Disentangled Spatial Control. arXiv:2405.12796 [cs.CV]

[11] Hong Chen, Yipeng Zhang, Simin Wu, Xin Wang, Xuguang Duan, Yuwei Zhou, and Wenwu Zhu. 2023. Disenbooth: Identity-preserving disentangled tuning for subject-driven text-to-image generation. In *The Twelfth International Conference on Learning Representations*.

[12] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *NeurIPS* 29 (2016).

[13] Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, and Jiliang Tang. 2023. Exploring the Potential of Large Language Models (LLMs) in Learning on Graphs. *arXiv preprint arXiv:2307.03393* (2023).

[14] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality. https://lmsys.org/blog/2023-03-30-vicuna/

[15] Weilin Cong, Yanhong Wu, Yuandong Tian, Mengting Gu, Yinglong Xia, Mehrdad Mahdavi, and Chun-cheng Jason Chen. 2021. Dynamic Graph Representation Learning via Graph Transformer Networks. *arXiv preprint* (2021).

[16] Songgaojun Deng, Huzefa Rangwala, and Yue Ning. 2020. Dynamic knowledge graph based multi-event forecasting. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1585–1595.

[17] Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. GLM: General Language Model Pretraining with Autoregressive Blank Infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 320–335.

[18] Paul Erdős, Alfréd Rényi. 1960. On the evolution of random graphs. *Publ. math. inst. hung. acad. sci* 5, 1 (1960), 17–60.

[19] Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. 2024. Talk like a graph: Encoding graphs for large language models. *ICLR* (2024).

[20] Wei Feng, Xin Wang, Hong Chen, Zeyang Zhang, Zihan Song, Yuwei Zhou, and Wenwu Zhu. 2023. LLM4VG: Large Language Models Evaluation for Video Grounding. *arXiv preprint arXiv:2312.14206* (2023).

[21] Chaoyu Guan, Ziwei Zhang, Haoyang Li, Heng Chang, Zeyang Zhang, Yijian Qin, Jiyan Jiang, Xin Wang, and Wenwu Zhu. 2021. AutoGL: A Library for Automated Graph Learning. In *ICLR 2021 Workshop GTRL*.

[22] Jiayan Guo, Lun Du, and Hengyu Liu. 2023. GPT4Graph: Can Large Language Models Understand Graph Structured Data? An Empirical Evaluation and Benchmarking. *arXiv preprint arXiv:2305.15066* (2023).

[23] Ehsan Hajiramezanali, Arman Hasanzadeh, Krishna Narayanan, Nick Duffield, Mingyuan Zhou, and Xiaoning Qian. 2019. Variational graph recurrent neural networks. *Advances in neural information processing systems* 32 (2019).

[24] Xiaoxin He, Xavier Bresson, Thomas Laurent, and Bryan Hooi. 2023. Explanations as Features: LLM-Based Features for Text-Attributed Graphs. *arXiv preprint*

arXiv:2305.19523 (2023).

[25] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. 1983. Stochastic blockmodels: First steps. *Social networks* 5, 2 (1983), 109–137.

[26] Jun-Ting Hsieh, Bingbin Liu, De-An Huang, Li F Fei-Fei, and Juan Carlos Niebles. 2018. Learning to decompose and disentangle representations for video prediction. *Advances in neural information processing systems* 31 (2018).

[27] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825* (2023).

[28] Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645* (2023).

[29] Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. 2023. Large language models on graphs: A comprehensive survey. *arXiv preprint arXiv:2312.02783* (2023).

[30] Takeshi Kojima et al. 2022. Large Language Models are Zero-Shot Reasoners. *arXiv preprint arXiv:2205.11916* (2022).

[31] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph evolution: Densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD)* 1, 1 (2007), 2–es.

[32] Haoyang Li, Peng Cui, Chengxi Zang, Tianyang Zhang, Wenwu Zhu, and Yishi Lin. 2019. Fates of Microscopic Social Ecosystems: Keep Alive or Dead?. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 668–676.

[33] Haoyang Li, Xin Wang, Zeyang Zhang, Haibo Chen, Ziwei Zhang, and Wenwu Zhu. 2024. Disentangled Graph Self-supervised Learning for Out-of-Distribution Generalization. In *International conference on machine learning*. PMLR.

[34] Haoyang Li, Xin Wang, Ziwei Zhang, Jianxin Ma, Peng Cui, and Wenwu Zhu. 2021. Intention-aware sequential recommendation with structured intent transition. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[35] Haoyang Li, Xin Wang, Ziwei Zhang, Zehuan Yuan, Hang Li, and Wenwu Zhu. 2021. Disentangled contrastive learning on graphs. *Advances in Neural Information Processing Systems* 34 (2021), 21872–21884.

[36] Haoyang Li, Ziwei Zhang, Xin Wang, and Wenwu Zhu. 2022. Disentangled Graph Contrastive Learning With Independence Promotion. *IEEE Transactions on Knowledge and Data Engineering* (2022).

[37] Peiwen Li, Xin Wang, Zeyang Zhang, Yuan Meng, Fang Shen, Yue Li, Jialong Wang, Yang Li, and Wenweu Zhu. 2024. LLM-Enhanced Causal Discovery in Temporal Domain from Interventional Data. *arXiv:2404.14786* (2024).

[38] Peiwen Li, Xin Wang, Zeyang Zhang, Yijian Qin, Ziwei Zhang, Jialong Wang, Yang Li, and Wenwu Zhu. 2024. Causal-Aware Graph Neural Architecture Search under Distribution Shifts. arXiv:2405.16489 [cs.LG]

[39] Yuhan Li, Zhixun Li, Peisong Wang, Jia Li, Xiangguo Sun, Hong Cheng, and Jeffrey Xu Yu. 2023. A survey of graph meets large language model: Progress and future directions. *arXiv preprint arXiv:2311.12399* (2023).

[40] Yanbei Liu, Xiao Wang, Shu Wu, and Zhitao Xiao. 2020. Independence promoted graph disentangled networks. In *AAAI*, Vol. 34. 4916–4923.

[41] Hanjia Lyu, Song Jiang, Hanqing Zeng, Yinglong Xia, and Jiebo Luo. 2023. LLM-Rec: Personalized Recommendation via Prompting Large Language Models. *arXiv preprint arXiv:2307.15780* (2023).

[42] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. 2019. Disentangled graph convolutional networks. In *ICML*. PMLR, 4212–4221.

[43] Liqian Ma, Qianru Sun, Stamatios Georgoulis, Luc Van Gool, Bernt Schiele, and Mario Fritz. 2018. Disentangled person image generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 99–108.

[44] Jane Pan. 2023. *What in-context learning "learns" in-context: Disentangling task recognition and task learning*. Ph. D. Dissertation. Princeton University.

[45] Yijian Qin, Xin Wang, Zeyang Zhang, and Wenwu Zhu. 2021. Graph differentiable architecture search with structure learning. In *NeurIPS*.

[46] Yijian Qin, Xin Wang, Ziwei Zhang, and Wenwu Zhu. 2023. Disentangled representation learning with large language models for text-attributed graphs. *arXiv preprint arXiv:2310.18152* (2023).

[47] Yijian Qin, Ziwei Zhang, Xin Wang, Zeyang Zhang, and Wenwu Zhu. 2022. NAS-Bench-Graph: Benchmarking Graph Neural Architecture Search. In *Advances in Neural Information Processing Systems*.

[48] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. 2020. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637* (2020).

[49] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2023. Code Llama: Open Foundation Models for Code. arXiv:2308.12950 [cs.CL]

[50] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. 2020. Dysat: Deep neural representation learning on dynamic graphs via self-attention

networks. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 519–527.

[51] Matthias Schäfer, Martin Strohmeier, Vincent Lenders, Ivan Martinovic, and Matthias Wilhelm. 2014. Bringing up OpenSky: A large-scale ADS-B sensor network for research. In *IPSN-14 Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*. IEEE, 83–94.

[52] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. 2018. Structured sequence modeling with graph convolutional recurrent networks. In *International Conference on Neural Information Processing*.

[53] Jitesh Shetty and Jafar Adibi. 2004. The Enron email dataset database schema and brief statistical report. *Information sciences institute technical report, University of Southern California* 4, 1 (2004), 120–128.

[54] Joakim Skarding, Bogdan Gabrys, and Katarzyna Musial. 2021. Foundations and Modeling of Dynamic Networks Using Dynamic Graph Neural Networks: A Survey. *IEEE Access* 9 (2021), 79143–79168.

[55] Chenxi Sun, Yaliang Li, Hongyan Li, and Shenda Hong. 2023. TEST: Text Prototype Aligned Embedding to Activate LLM's Ability for Time Series. *arXiv preprint arXiv:2308.08241* (2023).

[56] Li Sun, Zhongbao Zhang, Jiawei Zhang, Feiyang Wang, Hao Peng, Sen Su, and Philip S Yu. 2021. Hyperbolic variational graph neural network for modeling dynamic graphs. In *AAAI*, Vol. 35. 4375–4383.

[57] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnet-Miner: Extraction and Mining of Academic Social Networks. In *KDD'08*. 990–998.

[58] Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. 2023. Large language models in medicine. *Nature medicine* 29, 8 (2023), 1930–1940.

[59] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[60] Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2023. Can Language Models Solve Graph Problems in Natural Language? *arXiv preprint arXiv:2305.10037* (2023).

[61] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2023. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432* (2023).

[62] Xuezhi Wang et al. 2022. Self-Consistency Improves Chain of Thought Reasoning in Language Models. *arXiv preprint arXiv:2203.11171* (2022).

[63] Xin Wang, Hong Chen, Si'ao Tang, Zihao Wu, and Wenwu Zhu. 2023. Disentangled Representation Learning. arXiv:2211.11695 [cs.LG]

[64] Xin Wang, Hong Chen, Yuwei Zhou, Jianxin Ma, and Wenwu Zhu. 2022. Disentangled Representation Learning for Recommendation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[65] Xin Wang, Zirui Pan, Yuwei Zhou, Hong Chen, Chendi Ge, and Wenwu Zhu. 2023. Curriculum co-disentangled representation learning across multiple environments for social recommendation. In *International Conference on Machine Learning*. PMLR, 36174–36192.

[66] Xin Wang, Zihao Wu, Hong Chen, Xiaohan Lan, and Wenwu Zhu. 2023. Mixup-Augmented Temporally Debiased Video Grounding with Content-Location Disentanglement. (2023).

[67] Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. 2021. Inductive representation learning in temporal networks via causal anonymous walks. *arXiv preprint arXiv:2101.05974* (2021).

[68] Yanbang Wang, Pan Li, Chongyang Bai, and Jure Leskovec. 2021. TEDIC: Neural modeling of behavioral patterns in dynamic social interaction networks. In *Proceedings of the Web Conference 2021*. 693–705.

[69] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *NeurIPS* 35 (2022), 24824–24837.

[70] Jiapeng Wu, Meng Cao, Jackie Chi Kit Cheung, and William L Hamilton. 2020. Temp: Temporal message passing for temporal knowledge graph completion. *arXiv preprint arXiv:2010.03526* (2020).

[71] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962* (2020).

[72] Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, et al. 2023. Baichuan 2: Open large-scale language models. *arXiv preprint arXiv:2309.10305* (2023).

[73] Menglin Yang, Min Zhou, Marcus Kalander, Zengfeng Huang, and Irwin King. 2021. Discrete-time Temporal Network Embedding via Implicit Hierarchical Learning in Hyperbolic Space. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1975–1985.

[74] Yiding Yang, Zunlei Feng, Mingli Song, and Xinchao Wang. 2020. Factorizable graph convolutional networks. *Advances in Neural Information Processing Systems* 33 (2020), 20286–20296.

[75] Yang Yao, Xin Wang, Zeyang Zhang, Yijian Qin, Ziwei Zhang, Xu Chu, Yuekui Yang, Wenwu Zhu, and Hong Mei. 2024. Exploring the Potential of Large Language Models in Graph Generation. *arXiv preprint arXiv:2403.14358* (2024).

[76] Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. 2023. Natural Language is All a Graph Needs. *arXiv preprint arXiv:2308.07134* (2023).

[77] Hong Chen Jiapei Fan Weigao Wen Hui Xue Hong Mei Wenwu Zhu Yipeng Zhang, Xin Wang. 2024. Large Language Model With Curriculum Reasoning for Visual Concept Recognition. In *ACM SIGKDD*.

[78] Jiaxuan You, Yichen Wang, Aditya Pal, Pong Eksombatchai, Chuck Rosenburg, and Jure Leskovec. 2019. Hierarchical temporal convolutional networks for dynamic recommender systems. In *The world wide web conference*. 2236–2246.

[79] Xinli Yu, Zheng Chen, Yuan Ling, Shujing Dong, Zongyi Liu, and Yanbin Lu. 2023. Temporal Data Meets LLM–Explainable Financial Time Series Forecasting. *arXiv preprint arXiv:2306.11025* (2023).

[80] Jiawei Zhang. 2023. Graph-ToolFormer: To Empower LLMs with Graph Reasoning Ability via Prompt Augmented by ChatGPT. *arXiv:2304.11116* (2023).

[81] Ziwei Zhang, Haoyang Li, Zeyang Zhang, Yijian Qin, Xin Wang, and Wenwu Zhu. 2023. Graph meets llms: Towards large graph models. In *NeurIPS 2023 Workshop: New Frontiers in Graph Learning*.

[82] Ziwei Zhang, Haoyang Li, Zeyang Zhang, Yijian Qin, Xin Wang, and Wenwu Zhu. 2023. Large Graph Models: A Perspective. *Advances in Neural Information Processing Systems GLFrontiers Workshop* (2023).

[83] Zeyang Zhang, Xingwang Li, Fei Teng, Ning Lin, Xueling Zhu, Xin Wang, and Wenwu Zhu. 2023. Out-of-Distribution Generalized Dynamic Graph Neural Network for Human Albumin Prediction. In *IEEE International Conference on Medical Artificial Intelligence*.

[84] Zeyang Zhang, Xin Wang, Yijian Qin, Hong Chen, Ziwei Zhang, Xu Chu, and Wenwu Zhu. 2024. Disentangled Continual Graph Neural Architecture Search with Invariant Modularization. In *International Conference on Machine Learning*.

[85] Zeyang Zhang, Xin Wang, Ziwei Zhang, Haoyang Li, Zhou Qin, and Wenwu Zhu. 2022. Dynamic graph neural networks under spatio-temporal distribution shift. In *NeurIPS*.

[86] Zeyang Zhang, Xin Wang, Ziwei Zhang, Haoyang Li, and Wenwu Zhu. 2023. Out-of-Distribution Generalized Dynamic Graph Neural Network with Disentangled Intervention and Invariance Promotion. *arXiv preprint arXiv:2311.14255* (2023).

[87] Zeyang Zhang, Xin Wang, Ziwei Zhang, Zhou Qin, Weigao Wen, Hui Xue, Haoyang Li, and Wenwu Zhu. 2023. Spectral Invariant Learning for Dynamic Graphs under Distribution Shifts. In *NeurIPS*.

[88] Zeyang Zhang, Xin Wang, Ziwei Zhang, Guangyao Shen, Shiqi Shen, and Wenwu Zhu. 2023. Unsupervised Graph Neural Architecture Search with Disentangled Self-supervision. In *Advances in Neural Information Processing Systems*.

[89] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. *arXiv:2210.03493* (2022).

[90] Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. 2023. Multimodal chain-of-thought reasoning in language models. *arXiv preprint arXiv:2302.00923* (2023).

[91] Zeyang Zhang, Ziwei Zhang, Xin Wang, Yijian Qin, Zhou Qin, and Wenwu Zhu. 2023. Dynamic Heterogeneous Graph Attention Neural Architecture Search. In *Thirty-Seventh AAAI Conference on Artificial Intelligence*.

[92] Zeyang Zhang, Ziwei Zhang, Xin Wang, and Wenwu Zhu. 2022. Learning to Solve Travelling Salesman Problem with Hardness-adaptive Curriculum. *arXiv preprint arXiv:2204.03236* (2022).

[93] Denny Zhou et al. 2023. Teaching Small Language Models to Reason. *arXiv preprint arXiv:2301.09208* (2023).

[94] Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. 2018. Dynamic network embedding by modeling triadic closure process. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[95] Yuecai Zhu, Fuyuan Lyu, Chengming Hu, Xi Chen, and Xue Liu. 2022. Learnable Encoder-Decoder Architecture for Dynamic Graph: A Survey. *arXiv preprint arXiv:2203.10480* (2022).

A   ADDITIONAL EXPERIMENTS

Results of spatial tasks on real-world datasets. We add experiments on three larger-scale real-world datasets. The details are as follows:

- Enron [53], an email correspondence dataset containing emails exchanged among employees of the ENRON energy company with 50K edges.
- DBLP [57], an academic networks dataset containing citations between papers with 100K edges. Each node is the paper published at year y, and each edge denotes the citation between the two paper.
- Flights [51], a dynamic flight network illustrating the development of the air traffic during the COVID-19 pandemic with 2M edges. This dataset encompasses all flights documented by the

Table 7: Task instructions in DyG prompts.

Task	Task Instruction
when link	Your task is to answer when two nodes are linked in the dynamic graph.
when connect	Your task is to answer when two nodes are first connected in the dynamic graph. Two nodes are connected if there exists a path between them.
when tclosure	Your task is to answer when the three nodes in the dynamic graph first close the triad. Two nodes with a common neighbor is said to have a triadic closure, if they are linked since some time so that the three nodes have linked with each other to form a triad.
neighbor at time	Your task is to answer what nodes are linked with a given node at a given time in the dynamic graph.
neighbor in periods	Your task is to answer what nodes are linked with one node only after some time in the dynamic graph
check tclosure	Your task is to answer whether three nodes in the dynamic graph formed a closed triad. A closed triad is composed of three nodes which have linked with each other some time.
check tpath	Your task is to answer whether a path is chronological in the dynamic graph. The time of the edges in a chronological path from source node to target node must not decrease, e.g., [2, 3, 5] is a chronological path in the dynamic graph [(2, 3, 0), (3, 5, 1)]
find tpath	Your task is to find a chronological path in the dynamic graph. The time of the edges in a chronological path from source node to target node must not decrease, e.g., [2, 3, 5] is a chronological path in the dynamic graph [(2, 3, 0), (3, 5, 1)]
sort edge	Your task is to sort the edges in the dynamic graph by time from earliest to latest.

Table 8: Results (ACC) on real-world datasets in the spatial task 'neighbors at time'.

Datasets	Enron	DBLP	Flights
NLGraph	0.19	0.33	0.42
GPT4Graph	0.30	0.35	0.47
Ours	0.45	0.43	0.47

2500 members of the OpenSky network since January 1st, 2020. The nodes represent airports and there is an edge between two nodes at time t, if on day t there is a flight between two airports.

We compare the performance (ACC) of static baseline methods NLGraph [60], GPT4Graph [22] in the spatial task 'neighbors at time'. As shown in Table 8, our method has a significant performance over the baselines on larger-scale real-world dynamic graphs, which can be credited to our consideration of temporal information on dynamic graphs. Though we focus on evaluating LLMs' spatial-temporal understanding abilities on dynamic graphs, the proposed method out-performs the recent baselines for static graphs in real-world datasets.

Table 9: Comparisons (ACC) on several spatial tasks.

Task	neighbor at time	neighbor in periods
NLGraph	0.18	0.06
GPT4Graph	0.22	0.12
Ours	0.34	0.16

Table 10: Comparisons on link prediction tasks.

Dataset	Enron		DBLP	
Metric	F1	Recall	F1	Recall
NLGraph	0.11	0.12	0.26	0.29
GPT4Graph	0.17	0.22	0.27	0.35
Ours	0.29	0.60	0.33	0.74

Table 11: Comparisons (ACC) with different time formatting types on several tasks.

Time types	when tclosure	check tclosure
Original	0.59	0.56
UNIX	0.35	0.53
Date	0.47	0.57

Table 12: Comparisons (ACC) with different node formatting types on several tasks.

Node name types	when tclosure	check tclosure
Original	0.59	0.56
Random indexes	0.62	0.55
People names	0.68	0.62

Comparisons of static baselines on spatial tasks. We compare the static baselines [22, 60] on spatial tasks. As shown in Table 9, our method out-performs the recent static baselines, showing the effectiveness of our method for tackling problems on dynamic graphs.

Links prediction tasks on real-world datasets. we add experiments for link prediction tasks to further compare the baselines [22, 60] for predictive tasks. As shown in Table 10, our method out-performs static baselines NLGraph and GPT4Graph with a large margin, demonstrating the effectiveness of handling spatial-temporal information on dynamic graphs.

Effects of time formatting types. We add experiments with different types of timestamps, including original, UNIX timestamp, Date, using Codellama2-13B and two tasks. The results are shown in the table 11. The original timestamp is an integer ranging from 1 to T, where T is the time span. The UNIX timestamp ranges from 2010-01-01 00:00:00 to 2020-01-01 00:00:00. The date is formatted as '%Year%Month%Day', *e.g.*, '20200101'. The results show that using UNIX timestamp or date for time formatting reduces the performance, which may be due to the increased complexity for LLMs to infer the ordering between time.

Effects of node formatting types. We add experiments with different types of node names, including original, random indexes, people names, using Codellama2-13B and two tasks. The results are shown in the table 12. The original node formatting uses integers ranging from 0 to N to represent the nodes, where N refers to the number of nodes, while the random indexes adopt integers ranging from 0 to 1e8. For the formatting of people names, several names are adopted to represent nodes, *e.g.*, 'Aiden', 'Priya', 'Dmitri'. One interesting phenomenon is that using names to represent nodes may improve the performance, which may be due to that LLMs may be more familiar with names than integers. The results are consistent with previous literature[19].

B IMPLEMENTATION DETAILS

Task prompts. For each problem instance, the prompt is constructed by a unified template including DyG instructions describing a dynamic graph, task instructions describing the task, answer instructions describing the answer template, examplars (in few-shot learning) and questions. An example is illustrated in Table 1 and we provide the task instruction for each task in Table 7.

C ADDITIONAL RELATED WORKS

Disentangled Representation Learning. Disentangled representation learning seeks to identify and clarify the distinct latent factors underlying observable data, with each factor represented as a unique vector [3, 63]. These factors are crucial for unraveling the intrinsic processes shaping data formation and for generating robust representations for subsequent applications. This approach has demonstrated its utility in various fields, including computer vision [10–12, 26, 43, 66], and graph representation learning [9, 33–36, 40, 42, 64, 65, 74, 84, 88]. Pan [44] explores and evaluate the in-context learning in large language models by disentangling the effects of task recognition and task learning. Qin et al. [46] integrates tailored disentangled graph neural network layers to capture complex structural relationships within text for better performance and interpretability. In this paper, we focus on studying the spatial-temporal understanding abilities of LLMs on dynamic graphs, and explore a disentangled prompt to improve performance via letting the model think of spatial and temporal dimensions separately.

Chain-of-Thoughts. The concept of chain-of-thoughts (CoT) [69] has garnered significant attention in recent years. CoT refers to the process by which a model generates intermediate reasoning steps that lead to the final answer, thereby mimicking human-like reasoning patterns, which has been shown to improve the performance of language models on complex tasks that require multi-step reasoning. Subsequent research has built upon these findings, exploring various techniques to optimize CoT prompting and extend its applicability to a broader range of tasks [30, 62, 90, 93]. Besta et al. [4] model the information generated by an LLM as an arbitrary graph, enhancing thoughts using feedback loops. Zhang et al. [89] design an automated process for generating CoT prompts. Since the enhanced CoT in our framework still have mixed results, one possible future direction is design automated CoT with automated graph techniques [21, 38, 45, 47, 92]. We also leave extending the framework to recently released LLMs [2, 17, 27, 72] in future works.