InstructGraph: Boosting Large Language Models via Graph-centric Instruction Tuning and Preference Alignment

Anonymous ACL submission

Abstract

Do current large language models (LLMs) better solve graph reasoning and generation tasks with parameter updates? In this paper, we propose **InstructGraph**, a framework that empowers LLMs with the abilities of graph reasoning and generation by instruction tuning and preference alignment. Specifically, we first propose a structured format verbalizer to unify all graph data into a universal code-like format, which can simply represent the graph without any external graph-specific encoders. Furthermore, a graph instruction tuning stage is introduced to guide LLMs in solving graph reasoning and generation tasks. Finally, we identify potential hallucination problems in graph tasks and sample negative instances for preference alignment, the target of which is to enhance the output's reliability of the model. Extensive experiments across multiple graph-centric tasks exhibit that InstructGraph can achieve the best performance and outperform GPT-4 and LLaMA2 by more than 13% and 38%, respectively.

1 Introduction

Currently, large language models (LLMs) have succeeded in reasoning on textual data (Brown et al., 2020; Zhao et al., 2023c). However, there also exists rich information in graph data, that is difficult to represent using plain text (Jin et al., 2023), such as knowledge graphs (Schneider et al., 2022), symbolic graphs (Saba, 2023), and social networks (Wang et al., 2023d), etc.

To endow LLMs with the ability to solve graph tasks, a series of works focus on designing the interface (e.g., prompt engineering) of LLMs on graph data to make them understand the semantics without parameter optimization (Ye et al., 2023; Han et al., 2023; Zhang et al., 2023b; Zhang, 2023; Kim et al., 2023; Jiang et al., 2023; Wang et al., 2023b; Luo et al., 2023), or injecting the graph embeddings into the partial parameters of LLMs through graph neural networks (GNNs) (Zhang et al., 2022; Chai et al., 2023; Tang et al., 2023; Perozzi et al., 2024). Despite significant progress, we explore these two challenges: 1) There still exists a semantic gap between graph and text, which may impede the LLM in graph reasoning and generation. 2) LLMs tend to generate hallucinations which may be caused by fabricated erroneous inputs or lack of pertinent knowledge. It can be viewed as the graph hallucination problem.

To overcome these challenges, we present a framework named **InstructGraph** that boosts LLMs by instruction tuning and preference alignment. A straightforward approach to solve the first challenge is to use a graph description (Ye et al., 2023) or graph embeddings (Chai et al., 2023), However, these methods require a large number of manual templates to describe the graph. Representing a large or complex graph via embeddings may cause information loss. In addition, the responses generated by the LLM with these methods are difficult to parse into actual graphs (Jin et al., 2023; Zhao et al., 2023c). Current investigations have demonstrated that LLMs have a great ability for code understanding and generation (Gao et al., 2023; Ma et al., 2023; Wong et al., 2023; Yang et al., 2024). Inspired by them, we can unify graph data into a code-like universal format to enhance the LLM's understanding and generation performance on graph tasks. As shown in Figure 1, each graph can be converted into a code with basic variables, such as `node_list` (or `entity_list`), `edge_list` (or `triple_list`) and optional properties. To this end, a graph instruction tuning stage is introduced to train the LLM on these formulated data.

In addition, previous works have found that LLMs generate responses with hallucination when following the instructions, typically referring to fabricated erroneous inputs or lack of intrinsic knowledge (Zhang et al., 2023a; Ji et al., 2023). For example, the LLM may derive a wrong answer

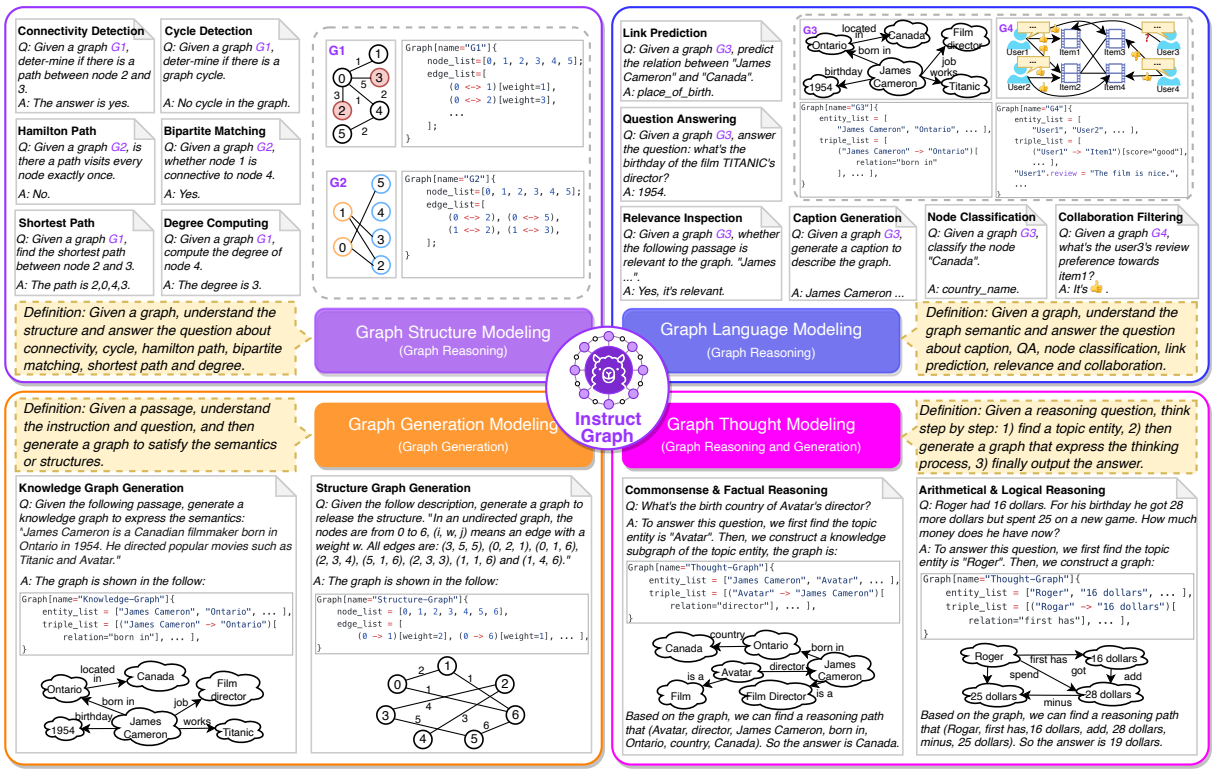

Figure 1: Four groups of graph-centric reasoning and generation tasks.

when being questioned on a graph that lacks key information, or the LLM may generate a graph with incorrect facts, conflicting, or missing information. However, how to reduce this effect in graph reasoning and generation is still under-explored. Hence, we introduce the graph preference alignment to alleviate the hallucination problem in the LLM's reasoning and generation. Specifically, we follow the direct preference optimization (DPO) algorithm (Rafailov et al., 2023) to optimize the LLM to make better preferences. To automatically sample the negative instances in DPO, we explore various scenarios, such as *unfactual graph*, *conflict graph* and *missing graph*. , to simulate the graph hallucination problem.

To evaluate the effectiveness of our framework, we perform extensive experiments on multiple graph reasoning and generation tasks. Results reveal that the proposed InstructGraph achieves the best performance on both graph-centric instruction and preference tasks and outperforms the GPT-4 (OpenAI, 2023) and LLaMA2 (Touvron et al., 2023b) by more than 13% and 38%, respectively.

2 Methodology

The skeleton is shown in Figure 2, which can be decomposed into three modules, i.e., graph input engineering, graph instruction tuning, and graph preference aligning.

2.1 Notation

Suppose that there are M graph tasks $\mathcal{D} = \{\mathcal{D}_1, \cdots \mathcal{D}_M\}$, and the corresponding dataset of each task can be denoted as $\mathcal{D}_j = \{(\mathcal{I}_i, \mathcal{G}_i, \mathcal{P}_i, \mathcal{A}_i)\}_{i=1}^{N_j}$, where N_j denotes the number of examples of $\mathcal{D}_j$, $\mathcal{I}_i$ is the corresponding instruction [1], $\mathcal{G}_i = (\mathcal{E}_i, \mathcal{R}_i, \mathcal{T}_i, \mathcal{S}_i)$ is the graph with one node (entity) set $\mathcal{E}_i$, one optional relation set $\mathcal{R}_i$, one edge (triple) set $\mathcal{T}_i$, and one optional textual property set $\mathcal{S}_i$, $\mathcal{P}_i$ is the optional passage , and $\mathcal{A}_i$ is the final answer [2].

2.2 Graph Input Engineering

The first challenge is how to align the graph to the text to meet the sequence interface of LLMs, previous works solved this issue by using graph description (Ye et al., 2023) or embedding fusion (Chai et al., 2023), which may make the generated responses difficult to parse into actual graphs.

Inspired by current LLMs that can simultaneously understand and generate code, we introduce a *structured format verbalizing* strategy to

[1] We manually design the instruction for each dataset.

[2] Especially, the answer $\mathcal{A}_i$ can be not only an independent text but also one of $\mathcal{G}_i$ and $\mathcal{P}_i$, depending on the task paradigm.

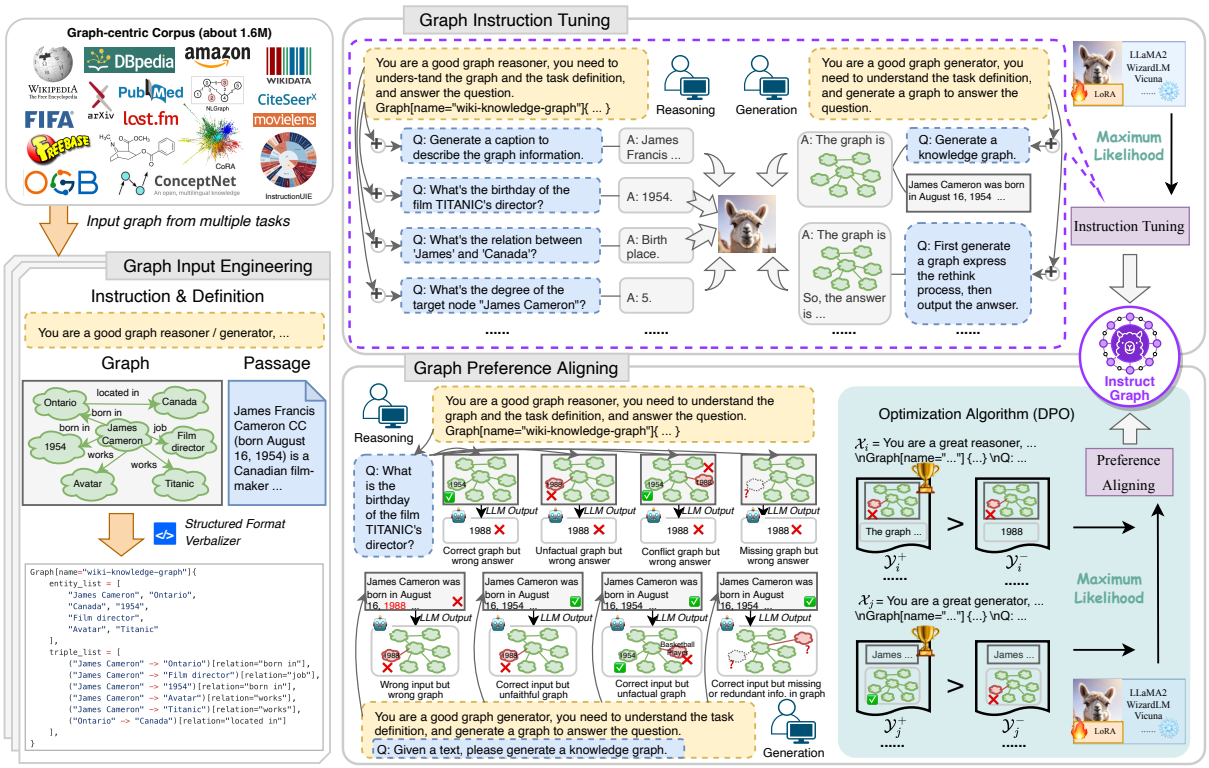

Figure 2: The InstructGraph framework. 1) We first collect multiple graph tasks, and unify them into a code-like format, along with task-specific textual data to form a graph instruction corpus. 2) Then, we perform graph instruction tuning to improve the ability of an LLM to solve graph reasoning and generation tasks. 3) Finally, we investigate multiple graph hallucination scenarios and optimize the LLM by preference alignment.

transform the graph into a simple code-like format. Formally, given one task graph $\mathcal{G}_i \in \mathcal{D}_j$, we denote $M(\cdot)$ as the structured format verbalizer, and the original graph can be mapped into a sequence as $\mathcal{C}_i = M(\mathcal{G}_i)$. For the fundamental format, all nodes (or entities) are listed as a sequence with variable `node_list` (or `entity_list`), while all edges (or triples) are listed as a sequence with variable `edge_list` (or `triple_list`). For graphs that contain side information, we can simulate the object-oriented language to express the node (or entity). Take the graph in Figure 1 as an example, the review text "The film is nice." of the node "User1" can be expressed by "User1.review=The film is nice.", where ".review" can be replaced as the property name in the graph. Therefore, we can unify all graphs into a unified format to align with textual data.

2.3 Graph Instruction Tuning

As shown in Figure 1, we first define four different groups of graph-centric instruction tasks to bolster the ability of LLMs on the graph, including graph structure modeling, graph language modeling, graph generation modeling, and graph thought modeling. The first two groups are focused on graph reasoning, the third group is typical graph generation, and the last group aims at both graph reasoning and generation [3]. After graph input engineering, we can directly reuse the standard causal language modeling (CLM) objective to continually tune the LLM on such groups. Formally, given one task dataset $\mathcal{D}_j = \{(\mathcal{I}_i, \mathcal{G}_i, \mathcal{P}_i, \mathcal{A}_i)\}_{i=1}^{N_j}$, the LLM can be optimized by *maximum likelihood* with:

$$\mathcal{L}(\mathcal{D}_j) = -\sum_{i=1}^{N_j} \log \pi_\theta(\mathcal{Y}_i = \mathcal{A}_i | \mathcal{X}_i), \quad (1)$$

where π_θ denotes the LLM with trainable parameters θ, $\mathcal{Y}_i$ is the model output, $\mathcal{X}_i$ and $\mathcal{A}_i$ respectively represent the input sequence and reference label, which depends on the specific task definition. Table 1 lists all groups of tasks and corresponding clusters to show the task definition, model input, and output. Therefore, we can obtain an instruction-based graph LLM and named InstructGraph-INS.

[3] We only choose the first three groups of tasks for instruction tuning. The tasks from graph thought modeling are only used for the evaluation.

Task Groups	Task Clusters	Task Definition	Task Input	Task Output
Graph Structure Modeling	Connection Detection, Cycle Detection, Hamilton Path, Bipartite Matching, Shortest Path, Degree Computing	The tasks in this group aim to make LLMs better understand some basic graph structures. The input only contains nodes, directed or un-directed edges, and optional weights.	$\mathcal{X}_i = [\mathcal{I}_i, \mathcal{C}_i]$	$\mathcal{Y}_i = \mathcal{A}_i$
Graph Language Modeling	Graph Caption Generation	The task aims to generate a caption passage $\mathcal{P}_i$ to describe the graph $\mathcal{G}_i$.	$\mathcal{X}_i = [\mathcal{I}_i, \mathcal{C}_i]$	$\mathcal{Y}_i = \mathcal{P}_i$
	Graph Question Answering	The task aims to reason on the whole graph $\mathcal{G}_i$ and find an entity as the final answer $\mathcal{A}_i \in \mathcal{E}_i$.	$\mathcal{X}_i = [\mathcal{I}_i, \mathcal{C}_i, \mathcal{P}_i]$	$\mathcal{Y}_i = \mathcal{A}_i$
	Graph Node Classification	The task aims to classify the target node into pre-defined classes based on $\mathcal{G}_i$.	$\mathcal{X}_i = [\mathcal{I}_i, \mathcal{C}_i, \mathcal{P}_i]$	$\mathcal{Y}_i = \mathcal{A}_i$
	Graph Link Prediction	The task aims to predict the relation between two given nodes based on $\mathcal{G}_i$.	$\mathcal{X}_i = [\mathcal{I}_i, \mathcal{C}_i, \mathcal{P}_i]$	$\mathcal{Y}_i = \mathcal{A}_i$
	Graph Relevance Inspection	The task aims to detect whether the graph $\mathcal{G}_i$ is relevant to the passage $\mathcal{P}_i$, we have $\mathcal{A}_i \in \{\text{relevant}, \text{irrelevant}\}$.	$\mathcal{X}_i = [\mathcal{I}_i, \mathcal{C}_i, \mathcal{P}_i]$	$\mathcal{Y}_i = \mathcal{A}_i$
	Graph Collaboration Filtering	The task aims to predict whether the target user prefers the target item based on the whole graph $\mathcal{G}_i$, the answer $\mathcal{A}_i$ can be set as a score.	$\mathcal{X}_i = [\mathcal{I}_i, \mathcal{C}_i, \mathcal{P}_i]$	$\mathcal{Y}_i = \mathcal{A}_i$
Graph Generation Modeling	Knowledge Graph Generation	The task aims to given a passage $\mathcal{P}_i$ that describes a piece of factual or commonsense information, the task aims to extract entities and relations from $\mathcal{P}_i$ to generate a graph $\mathcal{G}_i$.	$\mathcal{X}_i = [\mathcal{I}_i, \mathcal{P}_i]$	$\mathcal{Y}_i = \mathcal{C}_i$
	Structure Graph Generation	The task aims to generate a graph to meet the structure information described in the passage $\mathcal{P}_i$.	$\mathcal{X}_i = [\mathcal{I}_i, \mathcal{P}_i]$	$\mathcal{Y}_i = \mathcal{C}_i$
Graph Thought Modeling	Arithmetic Symbolic Robotic Logic	The task aims to solve the general reasoning task in three think steps: 1) first find the question subject, 2) then generate a thought graph $\mathcal{G}_i$ to express the rationale and 3) finally output the result $\mathcal{A}_i$ based on the graph.	$\mathcal{X}_i = \mathcal{I}_i$	$\mathcal{Y}_i = [\mathcal{C}_i; \mathcal{A}_i]$

Table 1: The overview of all groups of tasks.

2.4 Graph Preference Alignment

Recently, the NLP community has witnessed a significant decrease in hallucination through preference optimization (Ouyang et al., 2022; Zhao et al., 2023e; Rafailov et al., 2023; MacGlashan et al., 2017). Following this, we propose graph preference alignment to alleviate the hallucination of LLMs on the graph. As depicted in Figure 2, we intuitively design four typical hallucination circumstances for graph reasoning and generation and perform negative sampling for each graph task.

Hallucinations in Graph Reasoning Typically, the instruction-version LLM may be a strong instruction follower, yet, sometimes fall into hallucinations because of the erroneous input or lack of knowledge: 1) *correct graph but wrong answer* means the LLM makes a wrong prediction even though the input is legal, 2) *unfactual graph but wrong answer* means the wrong answer caused by a graph with unfaithful semantics to external knowledge, 3) *conflict graph but wrong answer* means there exists conflict information in the input graph, and 4) *missing graph but wrong answer* means that the input graph is missing some crucial information related to the answer.

To simulate the first circumstance, we can randomly choose a result from other examples to form a negative output $\mathcal{Y}_i^-$. For the rest, we can randomly *replace*, *add*, or *remove* some nodes (entities) or edges (triples) in the graph and construct a new input with the original instruction and passage. Therefore, the original answer can be viewed as the negative $\mathcal{Y}_i^-$ and the positive $\mathcal{Y}_i^+$ defined as "Sorry, the input graph contains wrong information, so the question is unanswerable directly.".

Hallucination in Graph Generation Graph generation is harder than reasoning because the LLM needs to output a complete and accurate code-like format sequence. The following are three kinds of wrong-generated graphs: *unfactual graph*, *conflict graph* and *missing graph*. We can directly construct a wrong graph as the final output $\mathcal{Y}_i^-$ by performing *replace*, *add*, and *remove* operators, which are

Clusters	Tasks	Metrics	GPT-3.5	GPT-4	LLaMA2	Vicuna	InstructGraph-INS
Structure	Conn. Dect.	ACC	81.45	80.47	54.01	54.85	**83.54**
	Cycle Dect.	ACC	59.02	61.44	50.79	52.88	**91.10**
	Hami. Path	ACC	21.03	29.10	1.23	1.23	**34.80**
	Bipt. Match	ACC	50.23	66.11	0.00	0.00	**76.36**
	Shrt. Path	ACC	38.99	49.03	0.00	0.00	**66.29**
	Degree Comp.	ACC	41.18	**70.59**	18.13	19.57	65.65
Caption	Wikipedia	BLEU	91.99	93.85	77.15	82.94	**95.81**
	WebNLG	BLEU	**99.51**	99.29	88.67	89.33	97.35
	GenWiki	BLEU	98.60	**98.65**	79.72	87.67	97.71
	EventNA	BLEU	62.66	61.75	53.39	75.52	**81.64**
	Xalign	BLEU	86.77	88.59	84.05	86.05	**93.08**
Graph QA	PathQSP	EM	52.54	68.64	42.70	31.90	**86.40**
	GrailQA	EM	43.92	60.17	15.83	17.95	**81.30**
	WebQSP	EM	53.73	61.57	40.07	26.42	**73.30**
	WikiTQ	EM	49.02	**60.78**	29.94	35.76	47.82
Node CLS	Cora	EM	74.51	64.17	83.04	84.08	**89.33**
	Citeseer	EM	70.39	**74.94**	68.24	67.94	71.65
	Pubmed	EM	74.63	77.16	79.78	80.18	**81.09**
	Arxiv	EM	70.59	74.51	45.50	57.75	**81.50**
	Products	EM	68.82	84.16	29.34	79.50	**95.20**
Link Pred.	Wikidata	Hits@1	43.73	62.94	10.75	10.38	**96.52**
	FB15K-237	Hits@1	60.34	66.88	0.00	0.00	**98.91**
	ConceptNet	Hits@1	31.33	38.30	8.30	8.19	**59.86**
Relevance	Wikipedia	ACC	94.40	**100**	69.27	68.12	**100**
RecSys	Amazon	Hits@1	27.09	59.77	44.40	16.40	**78.80**
IE	Wikipedia	F1	50.97	46.89	40.76	38.84	**83.56**
	UIE	F1	24.41	26.22	20.21	26.11	**76.82**
	InstructKGC	F1	21.44	21.86	19.26	16.6	**38.98**
Graph Gen.	NLGraph	F1	80.86	88.17	3.64	42.21	**91.05**
Avg.			59.45	66.76	41.65	46.06	**79.84**

Table 2: Main results (%) over multiple graph instruction tuning tasks under zero-shot settings. The number highlighted in bold denotes the best performance.

similar to the graph reasoning. The original graph is denoted as positive $\mathcal{Y}_i^+$. Additionally, in cases where an incorrect answer is due to a faulty input, we may substitute the original input with an unrelated one from the dataset that doesn't affect the answer graph. The original answer graph is then considered as the negative output $\mathcal{Y}_i^-$.

We next use the DPO algorithm to reduce hallucination. Specifically, given one instruction example $(\mathcal{X}_i, \mathcal{Y}_i^+)$ and a corresponding negative $(\mathcal{X}_i, \mathcal{Y}_i^-)$, we can define the preference model under the Bradley-Terry (Bradley and Terry, 1952) as:

$$
\begin{aligned}
p_\theta(\mathcal{Y}_i^+ > \mathcal{Y}_i^- | \mathcal{X}_i) &= \frac{1}{1+\exp\{r(\mathcal{Y}_i^+, \mathcal{Y}_i^-, \mathcal{X}_i)\}}, \\
r(\mathcal{Y}_i^+, \mathcal{Y}_i^-, \mathcal{X}_i) &= -\beta \log \frac{\pi_\theta(\mathcal{Y}_i^+|\mathcal{X}_i)}{\pi_{ref}(\mathcal{Y}_i^+|\mathcal{X}_i)} \\
&\quad + \beta \log \frac{\pi_\theta(\mathcal{Y}_i^-|\mathcal{X}_i)}{\pi_{ref}(\mathcal{Y}_i^-|\mathcal{X}_i)},
\end{aligned} \tag{2}
$$

where β is the balance factor, p_θ denotes the preference model, π_θ and π_{ref} respectively denotes the policy and reference model, which can be initialized from instruction-version LLM. Thus, we can optimize the LLM by *maximum likelihood* with:

$$
\begin{aligned}
&\mathcal{J}(\pi_\theta, \pi_{ref}) = -\mathbb{E}_{(\mathcal{X}_i, \mathcal{Y}_i^+, \mathcal{Y}_i^-)\sim\mathcal{D}} \\
&\left[\log \sigma\big(\beta \log \frac{\pi_\theta(\mathcal{Y}_i^+|\mathcal{X}_i)}{\pi_{ref}(\mathcal{Y}_i^+|\mathcal{X}_i)} - \beta \log \frac{\pi_\theta(\mathcal{Y}_i^-|\mathcal{X}_i)}{\pi_{ref}(\mathcal{Y}_i^-|\mathcal{X}_i)}\big)\right].
\end{aligned} \tag{3}
$$

We denote the policy π_θ as InstructGraph-PRE.

3 Experiments

In this section, we perform extensive experiments to evaluate the effectiveness of InstructGraph over graph tasks and general NLP tasks [4].

3.1 Implementation Settings

We construct about 1.6M examples for graph instruction tuning and 100K examples for graph preference alignment. In default, we choose LLaMA2-

[4]We also evaluate InstructGraph on the general NLP tasks, and perform further analysis. Due to space limitations, we move these results to Appendx B.

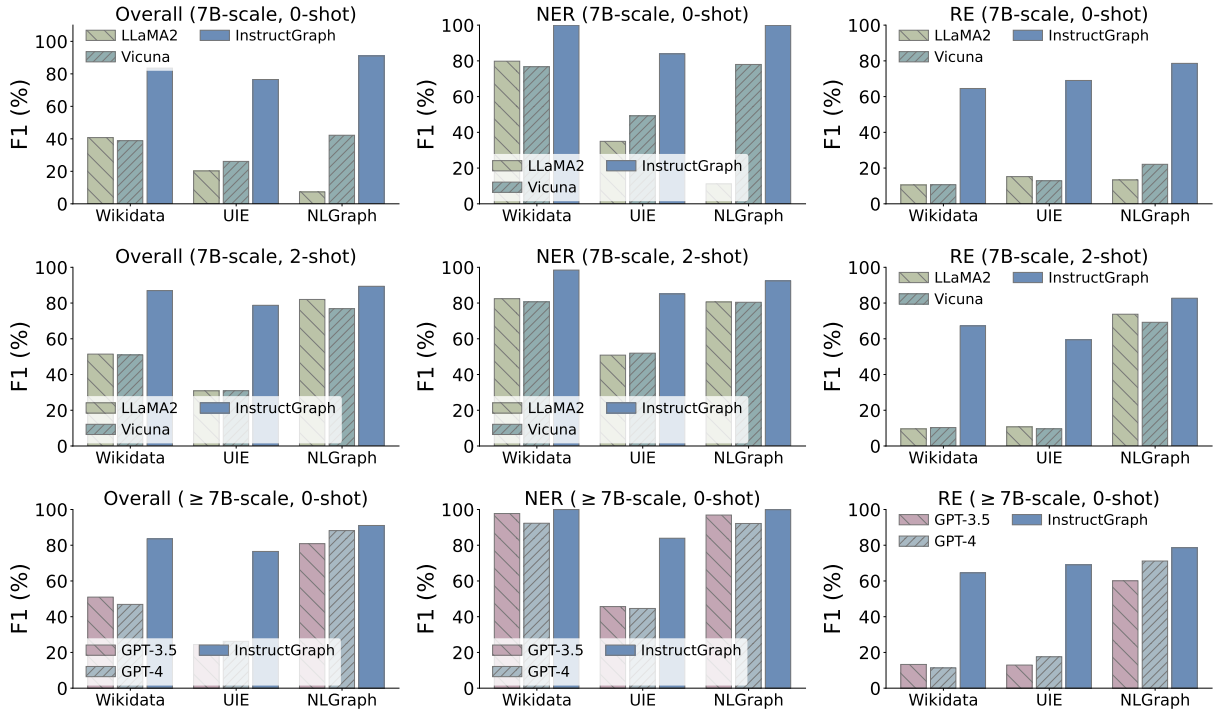

Figure 3: Performance (%) comparison with LLaMA2, Vicuna, GPT-3.5, and GPT-4 towards the overall graph, named entity recognition (NER), and relation extraction (RE) on graph generation tasks.

7B-HF (Touvron et al., 2023b) from HuggingFace[5] as the backbone. The maximum length is set as 2048. The optimizer is AdamW. The learning rate is set to $5e-5$ with a decay rate of 0.1 in the graph instruction tuning stage and will be changed to $5e-7$ in the graph preference alignment stage. To accelerate the training[6], we utilize FSDP (Zhao et al., 2023d) with CPU Offloading (Tsog et al., 2021), FlashAttention (Dao et al., 2022), and BFloat16 techniques, and utilize LoRA (Hu et al., 2022) to perform parameter-efficient learning with $rank = 32$ and $lora\_\alpha = 128$.

3.2 Main Results on Graph Instruction Tasks

In this section, we exhaustively evaluate the InstructGraph-INS on multiple graph reasoning and generation tasks in zero-shot settings. We use a code-like format to unify all graphs and construct an instruction tuning test set. Data statistics are shown in Table 6, and the details are shown in Appendix A.1. To make a comparison with a similar scale LLM, we choose the widely-used LLaMA2-7B and Vicuna-7B as the open-source baseline. In pursuit of investigating the performance level of InstructGraph in the era of AGI, we also choose GPT-3.5 (turbo) (Ouyang et al., 2022) and GPT-4 (OpenAI, 2023) as strong baselines [7].

Table 2 showcases the main results of graph reasoning and generation, we thus draw the following conclusions: 1) InstructGraph-INS achieves the best overall results 79.84% and outperforms GPT-4 by 13.08%. 2) Compared with the same scale LLMs, our framework performs the best on all graph tasks, which shows that further instruction tuning over well-designed graph tasks can better improve the reasoning and generation ability. 3) For the tasks Degree Computing, WebNLG, GenWiki, WikiTQ, and Citseer, InstructGraph-INS underperforms GPT-3.5 and GPT-4. Since the LLMs with large-scale parameters have stored more similar knowledge. Despite this, InstructGraph-INS still exhibits approximately 10% better performance on other reasoning tasks.

3.3 Effectiveness of Graph Generation

Additionally, we also expect to delve into whether InstructGraph-INS achieves the improvement on graph generation tasks, We choose two external manners to evaluate the results: 1) *NER* denotes named entity recognition, and 2) *RE* denotes relation extraction. As shown in Figure 3, we visualize

[5] https://huggingface.co/meta-llama.

[6] The implementation is referred to https://github.com/facebookresearch/llama-recipes.

[7] https://platform.openai.com/.

Methods (7B)	Is Align	Structure	Caption	Graph QA	Nodel CLS	IE	Avg.
LLaMA2	✗	38.64	57.96	70.70	74.68	37.40	55.88
Vicuna	✗	39.12	62.37	64.38	77.63	40.8	56.86
InstructGraph-INS	✗	50.32	81.15	77.85	83.16	69.14	72.32
InstructGraph-PRE	✓	**57.80**	**87.44**	**84.44**	**88.98**	**91.44**	**82.02**

Table 3: Main results (%) over multiple graph preference tasks under zero-shot settings.

Methods (7B)	Arithmetic			Symbolic		Robotic		Logic	
	GSM8K (4-shot)	SVAMP (4-shot)	AQuA (4-shot)	Letter (4-shot)	Coin (4-shot)	Termes (4-shot)	Floortile (4-shot)	ProofWriter (4-shot)	FOLIO (4-shot)
LLaMA2 w/. CoT	11.89	23.30	18.60	0.00	0.00	0.00	0.00	30.64	32.40
Vicuna w/. CoT	14.33	24.19	17.80	1.50	0.00	0.00	0.00	28.77	33.15
InstructGraph-INS w/. CoT	**17.52**	**28.80**	**22.33**	**8.70**	**6.20**	**30.00**	**50.00**	**55.80**	**41.68**
LLaMA2 w/. GTM	14.38	23.10	20.13	2.00	0.00	0.00	0.00	33.19	34.80
Vicuna w/. GTM	15.10	24.84	19.60	1.50	0.00	0.00	0.00	31.50	36.19
InstructGraph-INS w/. GTM	**19.46**	**27.10**	**23.80**	**7.40**	**9.40**	**30.00**	**50.00**	**52.77**	**43.06**

Table 4: Results (%) on thought planning tasks in few-shot scenarios.

the comparison performances on three graph generation tasks, where *Wikidata* and *UIE* belong to knowledge graph construction and *NLGraph* focus on structure graph generation. We observe that: 1) InstructGraph-INS can bring significant improvement for LLaMA2 and Vicuna, indicating the graph generation ability encompasses NER and RE. 2) We also integrate all baselines with the 2-shot exemplars, the results illustrate that the performance of InstructGraph-INS is consistently the highest. 3) RE is more challenging to NER because it involves understanding the semantics of generated nodes (entities) and making decisions on their relation or weight. Despite this, the improvement of RE is larger than NER, which signifies that graph-specific optimization can better empower the LLM in constructing triples.

3.4 Main Results on Graph Preference Tasks

We next explore whether InstructGraph can reduce the graph hallucination problem. We sample a few tasks from the corresponding cluster to build a hallucination testing set, including structure, caption, graph question answering, and node classification. The data statistics are shown in Table 6, and the details are shown in Appendix A.2. Specifically, each example consists of a correct answer and a wrong answer, we calculate the LLM's perplexity (PPL) on these answers and choose the option with the lowest PPL score as the preference results. Therefore, the accuracy metric can reflect the performance of hallucination mitigation.

As shown in Table 3, we choose LLaMA2, Vicuna, and two variants of InstructGraph to make a comparison. InstructGraph-INS outperforms LLaMA2 and Vicuna by 16.44% and 15.46%, respectively, demonstrating that our framework with only graph instruction tuning can solve the preference tasks better. This indicates that injecting task-related knowledge into the LLM's intrinsic parameter can be one of the significant factors for hallucination reduction. Furthermore, InstructGraph-PRE significantly enhances the instruction version model by about 10%, demonstrating that well-designed preference optimization can hit the upper boundary and endow the LLM with the ability to alleviate the pitfalls of hallucination.

3.5 Effectiveness of Thought Planning

Recall the graph instruction tuning, we are eager for the LLM to solve the thought planning tasks, including arithmetic, symbolic, robotic, and logic. We design two few-shot scenarios: 1) *Chain-of-Thought (CoT)* directly sampling few-shot exemplars with manually annotated sequence rationales to form a prompt. 2) *Graph Thought Modeling (GTM)* decomposes the sequence rationale into three stages, i.e., finding topic entities or keywords, building a graph to express the thought, and outputting the final answer. The comparison results are depicted in Table 4, and we can observe that InstructGraph-INS achieves the best performance when elicited by CoT and GTM prompts. In addition, GTM sometimes performs below expectations in the tasks of SVAMP, Letter, and ProofWriter. We believe that these tasks are difficult to express using

Baselines	Graph QA	Node CLS	IE
Graph Instruction Testing			
InstructGraph-INS	**72.21**	**83.75**	**66.45**
w/. only GSM	71.89	83.04	63.77
w/. only GLM	69.32	78.40	66.13
w/. only GGM	72.09	83.66	39.10
w/. only GTM	69.30	81.90	66.33
Graph Preference Testing			
InstructGraph-PRE	**84.44**	**88.98**	**91.44**
w/o. only unfactual	82.10	84.52	84.33
w/o. only conflict	83.70	85.17	81.11
w/o. only missing	79.35	83.55	78.40
w/o. ALL	77.85	83.16	69.14

Table 5: Average performance (%) of all tasks in each cluster when comparing different ablation versions. GSM, GLM, GGM, and GTM denote graph structure modeling, graph language modeling, graph generation modeling, and graph thought modeling, respectively. w/o. ALL equals to InstructGraph-INS.

an explicit graph to convey the thinking process.

3.6 Ablation Study

In this section, we focus on the ablation study to show how much each component contributes to performance. We choose three clusters for the test, i.e., Graph QA, Node CLS, and IE. As shown in Table 5, the results illustrate that the performance drops when removing one of these components. For the instruction tuning testing, we can observe that graph language modeling plays a significant role in Graph QA and Node CLS clusters, while graph generation modeling is beneficial to the performance of IE. For the preference testing, we can see that the performance of w/o. *missing graph* drops significantly, indicating that the major factor of hallucination is the lack of key information in the input graph or generated graph.

4 Related Work

4.1 LLMs for Graph Learning

A series of works have studied how to leverage LLMs to solve graph-centric tasks (Jin et al., 2023), which can be decomposed into the following categories: 1) Prompt engineering. A series of works aims to design the interface to elicit the LLM to better understand and reason on the graph (Ye et al., 2023; Han et al., 2023; Zhang et al., 2023b; Zhang, 2023; Kim et al., 2023; Wang et al., 2023b; Luo et al., 2023; Wang et al., 2023a; Guo et al., 2023; Zhao et al., 2023b). 2) Boosting LLMs with trainable GNNs. This kind of method focuses on enhancing the LLMs with trainable GNNs which can capture the arbitrary scale of the graph (Zhang et al., 2022; Chai et al., 2023; Tang et al., 2023; Zhao et al., 2023a; Tian et al., 2023; Qin et al., 2023). 3) Instruction tuning over graph data. Similar to ours, Xu et al. (2023); Jiang et al. (2023); Fang et al. (2023); Zeng et al. (2023) directly collect some graph or symbol data to form an instruction corpus, and then continually pre-train the LLM. Different from them, our InstructGraph further empowers the LLM by graph instruction tuning with the code-like universal format and well-designed hallucination alleviation strategy by preference alignment.

4.2 Hallucination in LLMs

LLMs usually generate seemingly plausible answers, which is called hallucination (Ji et al., 2023; Zhang et al., 2023a). The phenomenon of hallucination encompasses fabricating erroneous user input, unfaithful for previously generated context, and unfactual for external knowledge and commonsense. To estimate hallucination, Kryscinski et al. (2020); Li et al. (2023a); Tam et al. (2023); Min et al. (2023) leverage external tools or neural networks (e.g., BERT-NLI, GPT-4) to score the faithfulness and factuality of the model output. Recently, many works focus on suppressing this problem by retrieval-augmented generation (RAG) (Lewis et al., 2020), contrastive learning (Sun et al., 2023), contradictory evaluation (Mündler et al., 2023), and decoding strategies (Lee et al., 2022; Shi et al., 2023; Li et al., 2023b). Different from them, we aim to solve the hallucination problem on graph tasks with preference alignment.

5 Conclusion

This paper proposes a novel InstructGraph framework that empowers the LLM with the capacity to solve graph reasoning and generation tasks. To bridge the gap between graph data and textual language models, we introduce a structured format verbalizer to transform each graph into a code-like format and continually tune the LLM based on the instruction dataset, which is collected from 29 graph tasks. In addition, we also introduce a graph preference alignment stage to further mitigate the hallucination problem when reasoning on or generating a graph. Extensive experiments illustrate that InstructGraph substantially achieves the best performance. In our future work, we will further improve the performance on graph-centric and general NLP tasks and scale it to other LLMs.

Limitations

Our work is based on continual optimization for large language models and achieves outstanding performance across several benchmarks. However, it still carries the following limitations: (1) Due to resource limitations, we only conduct full experiments and analysis on a 7B scale. For 13B scales, we use 10% of the original training set for the model training. We plan to perform full parameter optimization on other backbones beyond 13B in the future. (2) The proposed structured format verbalizer aims to create a code sequence that describes a graph, but the input length may be limited when dealing with complex graphs or in a few-shot in-context learning setting.

Social Impact and Ethics

In terms of social impact, the graph data we utilize are all from publicly available data sources. Infusing this graph information into the model's reasoning process will not introduce additional bias. However, the open-source backbones we used may have some negative impacts, such as gender and social bias. Our work would unavoidably suffer from these issues. We suggest that users should carefully address potential risks when the proposed method is deployed online.

References

Tushar Abhishek, Shivprasad Sagare, Bhavyajeet Singh, Anubhav Sharma, Manish Gupta, and Vasudeva Varma. 2022. Xalign: Cross-lingual fact-to-text alignment and generation for low-resource languages. In *Companion of The Web Conference 2022, Virtual Event / Lyon, France, April 25 - 29, 2022*, pages 171–175. ACM.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom B. Brown, Jack Clark, Sam McCandlish, Chris Olah, Benjamin Mann, and Jared Kaplan. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *CoRR*, abs/2204.05862.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1533–1544. ACL.

Kurt D. Bollacker, Colin Evans, Praveen K. Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 1247–1250. ACM.

Ralph Allan Bradley and Milton E Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher 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 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Ziwei Chai, Tianjie Zhang, Liang Wu, Kaiqiao Han, Xiaohai Hu, Xuanwen Huang, and Yang Yang. 2023. Graphllm: Boosting graph reasoning ability of large language model. *CoRR*, abs/2310.05845.

Anthony Colas, Ali Sadeghian, Yue Wang, and Daisy Zhe Wang. 2021. Eventnarrative: A large-scale event-centric dataset for knowledge graph-to-text generation. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Yin Fang, Xiaozhuan Liang, Ningyu Zhang, Kangwei Liu, Rui Huang, Zhuo Chen, Xiaohui Fan, and Huajun Chen. 2023. Mol-instructions: A large-scale biomolecular instruction dataset for large language models. *CoRR*, abs/2306.08018.

Shuzheng Gao, Xin-Cheng Wen, Cuiyun Gao, Wenxuan Wang, Hongyu Zhang, and Michael R. Lyu. 2023. What makes good in-context demonstrations for code

intelligence tasks with llms? In *38th IEEE/ACM International Conference on Automated Software Engineering, ASE 2023, Luxembourg, September 11-15, 2023*, pages 761–773. IEEE.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 179–188. Association for Computational Linguistics.

C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. 1998. Citeseer: An automatic citation indexing system. In *Proceedings of the 3rd ACM International Conference on Digital Libraries, June 23-26, 1998, Pittsburgh, PA, USA*, pages 89–98. ACM.

Yu Gu, Sue Kase, Michelle Vanni, Brian M. Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond I.I.D.: three levels of generalization for question answering on knowledge bases. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pages 3477–3488. ACM / IW3C2.

Honghao Gui, Jintian Zhang, Hongbin Ye, and Ningyu Zhang. 2023. Instructie: A chinese instruction-based information extraction dataset. *CoRR*, abs/2305.11527.

Jiayan Guo, Lun Du, and Hengyu Liu. 2023. Gpt4graph: Can large language models understand graph structured data ? an empirical evaluation and benchmarking. *CoRR*, abs/2305.15066.

Jiuzhou Han, Nigel Collier, Wray L. Buntine, and Ehsan Shareghi. 2023. Pive: Prompting with iterative verification improving graph-based generative capability of llms. *CoRR*, abs/2305.12392.

Ruining He and Julian J. McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pages 507–517. ACM.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12):248:1–248:38.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. 2023. Structgpt: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 9237–9251. Association for Computational Linguistics.

Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. 2023. Large language models on graphs: A comprehensive survey. *CoRR*, abs/2312.02783.

Zhijing Jin, Qipeng Guo, Xipeng Qiu, and Zheng Zhang. 2020. Genwiki: A dataset of 1.3 million content-sharing text and graphs for unsupervised graph-to-text generation. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 2398–2409. International Committee on Computational Linguistics.

Jiho Kim, Yeonsu Kwon, Yohan Jo, and Edward Choi. 2023. KG-GPT: A general framework for reasoning on knowledge graphs using large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 9410–9421. Association for Computational Linguistics.

Wojciech Kryscinski, Bryan McCann, Caiming Xiong, and Richard Socher. 2020. Evaluating the factual consistency of abstractive text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 9332–9346. Association for Computational Linguistics.

Nayeon Lee, Wei Ping, Peng Xu, Mostofa Patwary, Pascale Fung, Mohammad Shoeybi, and Bryan Catanzaro. 2022. Factuality enhanced language models for open-ended text generation. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022,*

NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Junyi Li, Xiaoxue Cheng, Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2023a. Halueval: A large-scale hallucination evaluation benchmark for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 6449–6464. Association for Computational Linguistics.

Kenneth Li, Oam Patel, Fernanda B. Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023b. Inference-time intervention: Eliciting truthful answers from a language model. *CoRR*, abs/2306.03341.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4582–4597. Association for Computational Linguistics.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 3214–3252. Association for Computational Linguistics.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2023. Reasoning on graphs: Faithful and interpretable large language model reasoning. *CoRR*, abs/2310.01061.

Yingwei Ma, Yue Liu, Yue Yu, Yuanliang Zhang, Yu Jiang, Changjian Wang, and Shanshan Li. 2023. At which training stage does code data help llms reasoning? *CoRR*, abs/2309.16298.

James MacGlashan, Mark K. Ho, Robert Tyler Loftin, Bei Peng, Guan Wang, David L. Roberts, Matthew E. Taylor, and Michael L. Littman. 2017. Interactive learning from policy-dependent human feedback. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 2285–2294. PMLR.

Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Inf. Retr.*, 3(2):127–163.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 12076–12100. Association for Computational Linguistics.

Niels Mündler, Jingxuan He, Slobodan Jenko, and Martin T. Vechev. 2023. Self-contradictory hallucinations of large language models: Evaluation, detection and mitigation. *CoRR*, abs/2305.15852.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1470–1480. The Association for Computer Linguistics.

Bryan Perozzi, Bahare Fatemi, Dustin Zelle, Anton Tsitsulin, Mehran Kazemi, Rami Al-Rfou, and Jonathan Halcrow. 2024. Let your graph do the talking: Encoding structured data for llms. *arXiv preprint arXiv:2402.05862*.

Yijian Qin, Xin Wang, Ziwei Zhang, and Wenwu Zhu. 2023. Disentangled representation learning with large language models for text-attributed graphs. *CoRR*, abs/2310.18152.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *CoRR*, abs/2305.18290.

Walid S. Saba. 2023. Stochastic llms do not understand language: Towards symbolic, explainable and ontologically based llms. In *Conceptual Modeling - 42nd International Conference, ER 2023, Lisbon, Portugal, November 6-9, 2023, Proceedings*, volume 14320

of *Lecture Notes in Computer Science*, pages 3–19. Springer.

Phillip Schneider, Tim Schopf, Juraj Vladika, Mikhail Galkin, Elena Simperl, and Florian Matthes. 2022. A decade of knowledge graphs in natural language processing: A survey. *CoRR*, abs/2210.00105.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI Mag.*, 29(3):93–106.

Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Scott Wen-tau Yih. 2023. Trusting your evidence: Hallucinate less with context-aware decoding. *CoRR*, abs/2305.14739.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 4444–4451. AAAI Press.

Weiwei Sun, Zhengliang Shi, Shen Gao, Pengjie Ren, Maarten de Rijke, and Zhaochun Ren. 2023. Contrastive learning reduces hallucination in conversations. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pages 13618–13626. AAAI Press.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13003–13051. Association for Computational Linguistics.

Derek Tam, Anisha Mascarenhas, Shiyue Zhang, Sarah Kwan, Mohit Bansal, and Colin Raffel. 2023. Evaluating the factual consistency of large language models through news summarization. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 5220–5255. Association for Computational Linguistics.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2023. Graphgpt: Graph instruction tuning for large language models. *CoRR*, abs/2310.13023.

Yijun Tian, Huan Song, Zichen Wang, Haozhu Wang, Ziqing Hu, Fang Wang, Nitesh V. Chawla, and Panpan Xu. 2023. Graph neural prompting with large language models. *CoRR*, abs/2309.15427.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Nandinbaatar Tsog, Saad Mubeen, Fredrik Bruhn, Moris Behnam, and Mikael Sjödin. 2021. Offloading accelerator-intensive workloads in CPU-GPU heterogeneous processors. In *26th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2021, Vasteras, Sweden, September 7-10, 2021*, pages 1–8. IEEE.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2023a. Can language models solve graph problems in natural language? *CoRR*, abs/2305.10037.

Jianing Wang, Wenkang Huang, Minghui Qiu, Qiuhui Shi, Hongbin Wang, Xiang Li, and Ming Gao. 2022. Knowledge prompting in pre-trained language model for natural language understanding. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 3164–3177. Association for Computational Linguistics.

Jianing Wang, Qiushi Sun, Nuo Chen, Xiang Li, and Ming Gao. 2023b. Boosting language models reasoning with chain-of-knowledge prompting. *CoRR*, abs/2306.06427.

Xiao Wang, Weikang Zhou, Can Zu, Han Xia, Tianze Chen, Yuansen Zhang, Rui Zheng, Junjie Ye, Qi Zhang, Tao Gui, Jihua Kang, Jingsheng Yang, Siyuan Li, and Chunsai Du. 2023c. Instructuie: Multi-task instruction tuning for unified information extraction. *CoRR*, abs/2304.08085.

Xiaolei Wang, Xinyu Tang, Xin Zhao, Jingyuan Wang, and Ji-Rong Wen. 2023d. Rethinking the evaluation for conversational recommendation in the era of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 10052–10065. Association for Computational Linguistics.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *Trans. Assoc. Comput. Linguistics*, 9:176–194.

Man-Fai Wong, Shangxin Guo, Ching Nam Hang, Siu-Wai Ho, and Chee-Wei Tan. 2023. Natural language generation and understanding of big code for ai-assisted programming: A review. *Entropy*, 25(6):888.

Fangzhi Xu, Zhiyong Wu, Qiushi Sun, Siyu Ren, Fei Yuan, Shuai Yuan, Qika Lin, Yu Qiao, and Jun Liu. 2023. Symbol-llm: Towards foundational symbol-centric interface for large language models. *CoRR*, abs/2311.09278.

Ke Yang, Jiateng Liu, John Wu, Chaoqi Yang, Yi R. Fung, Sha Li, Zixuan Huang, Xu Cao, Xingyao Wang, Yiquan Wang, Heng Ji, and Chengxiang Zhai. 2024. If LLM is the wizard, then code is the wand: A survey on how code empowers large language models to serve as intelligent agents. *CoRR*, abs/2401.00812.

Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. 2023. Natural language is all a graph needs. *CoRR*, abs/2308.07134.

Zheni Zeng, Bangchen Yin, Shipeng Wang, Jiarui Liu, Cheng Yang, Haishen Yao, Xingzhi Sun, Maosong Sun, Guotong Xie, and Zhiyuan Liu. 2023. Interactive molecular discovery with natural language. *CoRR*, abs/2306.11976.

Jiawei Zhang. 2023. Graph-toolformer: To empower llms with graph reasoning ability via prompt augmented by chatgpt. *CoRR*, abs/2304.11116.

Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D. Manning, and Jure Leskovec. 2022. Greaselm: Graph reasoning enhanced language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. 2023a. Siren's song in the AI ocean: A survey on hallucination in large language models. *CoRR*, abs/2309.01219.

Zeyang Zhang, Xin Wang, Ziwei Zhang, Haoyang Li, Yijian Qin, Simin Wu, and Wenwu Zhu. 2023b. Llm4dyg: Can large language models solve problems on dynamic graphs? *CoRR*, abs/2310.17110.

Haiteng Zhao, Shengchao Liu, Chang Ma, Hannan Xu, Jie Fu, Zhi-Hong Deng, Lingpeng Kong, and Qi Liu. 2023a. GIMLET: A unified graph-text model for instruction-based molecule zero-shot learning. *CoRR*, abs/2306.13089.

Jianan Zhao, Le Zhuo, Yikang Shen, Meng Qu, Kai Liu, Michael M. Bronstein, Zhaocheng Zhu, and Jian Tang. 2023b. Graphtext: Graph reasoning in text space. *CoRR*, abs/2310.01089.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023c. A survey of large language models. *CoRR*, abs/2303.18223.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. 2023d. Pytorch FSDP: experiences on scaling fully sharded data parallel. *Proc. VLDB Endow.*, 16(12):3848–3860.

Zhiyuan Zhao, Bin Wang, Linke Ouyang, Xiaoyi Dong, Jiaqi Wang, and Conghui He. 2023e. Beyond hallucinations: Enhancing lvlms through hallucination-aware direct preference optimization. *CoRR*, abs/2311.16839.

Mantong Zhou, Minlie Huang, and Xiaoyan Zhu. 2018. An interpretable reasoning network for multi-relation question answering. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 2010–2022. Association for Computational Linguistics.

A Details of the InstructGraph Corpus

In this section, we provide some details of the corpus construction including both instruction and preference perspective.

A.1 Instruction Tuning Dataset

To merge all graph-oriented reasoning and generation tasks, we collect and construct 29 tasks to form instruction data. We do not construct training sets for graph thought modeling.

Graph Structure Modeling Graph structure modeling aims to urge the LLM to understand the structure of a graph along with the corresponding task-specific instruction. To reach this aim, we collect structure dataset NLGraph (Wang et al., 2023a). The original dataset consists of 8 different

Clusters	Tasks	Source	Sampling	Instruction Dataset		Preference Dataset	
				#Train	#Test	#Train	#Test
Structure	Conn. Dect.	(Wang et al., 2023a)	Up	3,737	237	2,227	463
	Cycle Dect.	(Wang et al., 2023a)	Up	2,877	191	863	191
	Hami. Path	(Wang et al., 2023a)	Up	1,315	55	-	-
	Bipt. Match	(Wang et al., 2023a)	Up	1,755	71	-	-
	Shrt. Path	(Wang et al., 2023a)	Up	1,580	64	948	128
	Degree Comp.	(Wang et al., 2023a)	Up	2,435	230	1,429	445
Caption	Wikipedia	(Wang et al., 2022)	Down	516,585	1,979	15,208	4,785
	WebNLG	(Gardent et al., 2017)	100%	12,237	2,000	6,040	2,616
	GenWiki	(Jin et al., 2020)	100%	99,997	1,000	-	-
	EventNA	(Colas et al., 2021)	100%	58,733	1,952	-	-
	Xalign	(Abhishek et al., 2022)	100%	30,000	470	-	-
Graph QA	PathQSP	(Zhou et al., 2018)	Down	30,530	1,000	27477	3,000
	GrailQA	(Gu et al., 2021)	Down	13,797	1,421	-	-
	WebQSP	(Berant et al., 2013)	Down	13,152	1,465	-	-
	WikiTQ	(Pasupat and Liang, 2015)	Down	2,780	688	-	-
Node CLS	Cora	(McCallum et al., 2000)	Down	548	961	166	965
	Citeseer	(Giles et al., 1998)	Down	943	995	284	990
	Pubmed	(Sen et al., 2008)	Down	9,736	1,756	2,988	1,789
	Arxiv	(Hu et al., 2020)	Down	9,710	400	2,705	325
	Products	(Hu et al., 2020)	Down	19,975	1,688	5,995	1,719
Link Pred.	Wikidata	(Wang et al., 2022)	Down	49,320	3,190	-	-
	FB15K-237	(Bollacker et al., 2008)	Down	2,988	92	-	-
	ConceptNet	(Speer et al., 2017)	Down	21,240	598	-	-
Relevance	Wikipedia	(Wang et al., 2022)	Down	39,672	1,991	-	-
RecSys	Amazon	(He and McAuley, 2016)	Down	2,424	250	-	-
IE	Wikipedia	(Wang et al., 2022)	Down	73,101	1,814	19,490	1,589
	UIE	(Wang et al., 2023c)	100%	285,877	3,000	-	-
	InstructKGC	(Gui et al., 2023)	Down	31,605	994	-	-
Graph Gen.	NLGraph	(Wang et al., 2023a)	Down	3,056	407	-	-
The total number of the corpus				1,341,885	30,959	85,820	19,005

Table 6: The data statistics of each graph task for graph instruction tuning and preference alignment.

tasks, such as *Connectivity Detection*, *Cycle Detection*, *Topological Sorting*, *Shortest Path Computing*, *Maximum Flow Computing*, *Bipartite Graph Matching*, *Hamilton Path Detection* and *GNN Embedding*. Yet, the authors Wang et al. (2023a) mentioned that the current LLMs are hard to perform on more complex graph reasoning, such as *Topological Sorting*, *Maximum Flow Computing*, and *GNN Embedding*, so we remove them. In addition, we also random sample some graphs of NLGraph, and construct a *Degree Computing* task.

- Connectivity Detection: detect whether there exists a path between two nodes in the graph. This task is a binary classification and the answer should be 'The answer is yes' or 'The answer is no'.

- Cycle Detection: determine if there is a cycle in this graph. This task is a binary classification and the answer should be 'Yes' or 'No'.

- Topological Sorting: determine if there is a path that visits every node exactly once in this graph. This task is a binary classification and the answer should be 'Yes' or 'No'.

- Bipartite Graph Matching: detect whether there exists an edge between two given nodes in a bipartite graph. This task is a binary classification and the answer should be 'Yes' or 'No'.

- Shortest Path Computing: find the shortest path between two nodes in the graph, and calculate the sum of the weights in the shortest path. The answer is a sequence of the path with a value.

- Graph Degree Computing: calculate the degree of the target node in the graph. The answer is an integer value.

Task Name	Hallucination Type	Positive Answer	Negative Answer
Conn. Dect. Cycle Detect. Shrt. Path Degree Comp.	Correct graph but wrong answer	<The original answer>	<Randomly sampled from other examples>
	Unfactual graph but wrong answer	Sorry, the graph contains some wrong knowledge in the follow: <list all unfactual triples>. So the question is unanswerable, you had better provide a correct graph.	<The original answer>
	Conflict graph but wrong answer	Sorry, the graph contains some conflict edges in the follow: <list all conflict triples>. So the question is unanswerable, you had better provide a correct graph.	<The original answer>
	Missing graph but wrong answer	Sorry, the graph does not exist node node name. So the question is unanswerable, you had better provide a correct graph.	<The original answer>
Caption	Correct graph but wrong answer	<The original answer>	<Randomly sampled from other examples>
	Unfactual graph but wrong answer	Sorry, the graph contains some wrong knowledge in the follow: <list all unfactual triples>. based on the corrected graph, the answer can be <The original answer>.	<The original answer>
	Conflict graph but wrong answer	Sorry, the graph contains some conflict edges in the follow: <list all conflict triples>. So the question is unanswerable, you had better provide a correct graph.	<The original answer>
Graph QA	Correct graph but wrong answer	<The original answer>	<Randomly sampled from other examples>
	Unfactual graph but wrong answer	Sorry, the graph contains some wrong knowledge in the follow: <list all unfactual triples>. based on the corrected graph, the answer can be <The original answer>.	<The original answer>
	Conflict graph but wrong answer	Sorry, the graph contains some conflict edges in the follow: <list all conflict triples>. So the question is unanswerable, you had better provide a correct graph.	<The original answer>
	Missing graph but wrong answer	Based on the world knowledge, the correct answer to the question is <The original answer>, but the answer does not exist in the graph.	<The original answer>
Node CLS	Correct graph but wrong answer	<The original answer>	<Randomly sampled from other examples>
IE	Wrong input but wrong graph	<The original graph>	<Randomly sampled from other examples>
	Correct input but unfaithful graph	<The original graph>	<Randomly edit entities in the original graph>
	Correct input but unfactual graph	<Randomly edit edges in the original graph>	<The original graph>
	Correct input but missing or redundant information in graph	<Randomly remove or add edges in the original graph>	<The original graph>

Table 7: The positive and negative answer of each example for preference alignment.

Graph Language Modeling Graph language modeling aims to teach the LLM to understand both the structure and semantics knowledge of the graph and answer the question. We decompose this group into 6 kinds of tasks, including *graph caption generation*, *graph question answering*, *graph node classification*, *graph link prediction*, *graph relevance inspection*, and *graph collaboration filtering*.

- Graph caption generation: generate an encyclopedia passage when given a knowledge graph with all entities and structure triples representing factual and commonsense knowledge. We directly choose the datasets from WebNLG (Gardent et al., 2017), GenWiki (Jin et al., 2020), EventNarrative (Colas et al., 2021), XAlign (Abhishek et al., 2022). In addition, we also follow (Wang et al., 2022) to collect the Wikipedia corpus and corresponding wikidata knowledge graph to build the caption task. Specifically, we use the AC automatic machine algorithm to recognize all entities in the passage and construct a 2-hop sub-graph based on the topic entity.

- Graph question answering: find an entity and a reasoning path in the graph to answer the question. We directly collect the corpus from PathQuestions (Zhou et al., 2018), GrailQA (Gu et al., 2021), WebQuestions (Berant et al., 2013), WikiTableQuestions (Pasupat and Liang, 2015). Especially, the WikiTableQuestions is a table understanding task that answers a question based on the table. To make our framework support this kind of task, we perform preprocessing that transforms each row line of the table into a single graph, where the table head is the relation name and each cell is the entity.

- Graph node classification: classify the target node based on the corresponding graph. We directly choose from Cora (McCallum et al., 2000), Citeseer (Giles et al., 1998), Pubmed (Sen et al., 2008), OGBN-ArXiv, and OGBN-Products (Hu et al., 2020). Because the graph in these tasks is too big, we only sample a 2-hop sub-graph of centering each target node. We also perform down-sampling for each task.

- Graph link prediction: classify the edge (relation) between two given nodes (entities) based on the graph. We choose three main knowl-

Methods	Is Align	HaluEval				Anthropic-HH		TruthfulQA	Avg.
		Dialogue	General	QA	Abstract	Harmless	Helpful		
GPT-3.5	✓	**72.40**	**79.44**	**62.59**	**58.53**	-	-	47.50	-
GPT-4	✓	-	-	-	-	-	-	**59.80**	-
LLaMA2-7B	✗	43.99	20.46	49.60	49.55	54.28	**60.49**	33.29	44.52
Vicuna-7B	✗	46.35	19.48	**60.34**	45.62	55.70	58.71	30.10	45.19
InstructGraph-INS	✗	44.88	21.35	52.90	51.10	56.33	59.10	35.35	45.86
InstructGraph-PRE	✓	**47.03**	**21.61**	52.88	**51.39**	**58.40**	60.12	**35.77**	**46.74**

Table 8: Main results (%) over multiple general NLP preference tasks under zero-shot settings.

edge graph, such as Wikidata (Wang et al., 2021), Freebase (Bollacker et al., 2008), ConceptNet (Speer et al., 2017). Specifically, we random sample a subset of triples, and then extract and merge two 2-hop sub-graphs that center with two entities, respectively.

- Graph relevance inspection: inspect whether the caption is relevant to the graph. The task is a binary classification with two categories, i.e., "relevant" and "irrelevant". We directly use the same corpus from wikipedia (Wang et al., 2022) in *graph caption generation* task. For the negative sampling of each graph, we directly choose other captions.

- Graph Collaboration Filtering: predict the score that the user node prefers to the target item node based on the collaboration graph. We choose the widely used Amazon (He and McAuley, 2016) as the corpus. Because the Amazon dataset does not provide any graph data, we thus perform a preprocessing stage to construct a collaboration graph. Specifically, we calculate the Jaccard similarity between each pair of users based on their preference items and then recall the top-10 similarity users for each user to form a graph. Hence, we can inject this graph into the LLM to let it know how to recommend some items based on all potential users.

Graph Generation Modeling This group aims to guide the LLM to generate a graph in a code-like format. We consider two challenging graph generation domains, including, *knowledge graph generation* and *structure graph generation*.

- Knowledge graph generation: similar to information extraction which aims to extract entities and relations when given one passage. We directly choose the corpus from unified information extraction (UIE) (Wang et al., 2023c; Gui et al., 2023), which consists of 21 used named entity recognition (NER) tasks, 10 used relation extraction (RE), and 4 used event extraction (EE).

- Structure graph generation: generate a structure graph based on the description. For example, when given a graph description is "Please generate a full-connection un-directed graph with four nodes ranging from 0 to 3.", the expected code-like format graph is "Graph[name='structure-graph']node_list=[0, 1, 2, 3]; edge_list=[(0 <-> 1), (0 <-> 2), (0 <-> 3), (1 <-> 2), (1 <-> 3), (2 <-> 3)];". We can directly reuse the corpus from NLGraph (Wang et al., 2023a) and sample a subset to build this task.

A.2 Preference Alignment Dataset

We have selected a partial dataset from the graph instruction tuning dataset for preference alignment. This dataset includes Connection Detection, Cycle Detection, Shortest Path Computing, Degree Computing, Graph Caption with Wikipedia and WebNLG, Graph QA with PathQSP, Node CLS with Cora, Citeseer, Pubmed, Arxiv, and Products, and IE with Wikipedia.

For each task, we design positive and negative answers to support preference alignment. Details are shown in Table 7.

B Further Analysis

B.1 Effectiveness on General Preference Tasks

We also delve into whether the preference optimization on the graph data hinders the effectiveness in the general domains. To reach this goal, we choose three external preference and hallucination tasks. 1) HaluEval (Li et al., 2023a) [8] focuses on hallucina-

[8] https://github.com/RUCAIBox/HaluEval.

Methods	BBH (3-shot)	MMLU (5-shot)
GPT-3.5	-	70.00
GPT-4	-	86.40
MPT-7B	31.00	26.80
Falcon-7B	28.00	26.20
LLaMA-7B	30.30	35.10
LLaMA2-7B	32.58	45.65
Vicuna-7B	31.54	50.34
InstructGraph-INS	**33.06**	**51.62**

Table 9: Results (%) over multiple general NLP tasks under few-shot in-context learning settings.

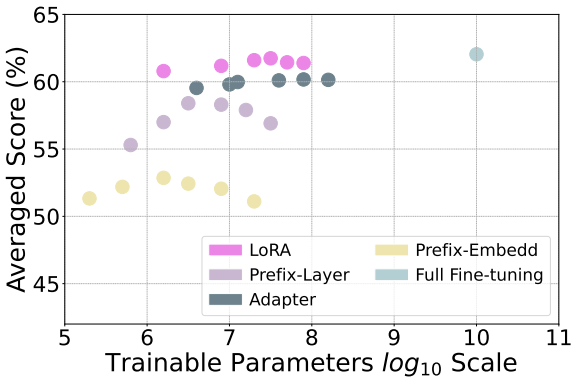

Figure 4: Results (%) of balance between trainable parameters and performances over graph tasks.

Methods	PathQSP	WebNLG	CoRA	UIE
GPT-4				
Template	58.20	96.13	58.58	0.00
Code Format	68.64	99.29	64.17	26.22
LLaMA2				
Template	20.36	59.15	27.44	0.00
Code Format	42.70	88.67	83.04	20.21

Table 10: Results (%) comparison with different prompt engineering during the inference.

tion evaluation in dialogue, general understanding, question answering, and text summarization (abstract). 2) TruthfulQA (Lin et al., 2022) [9] aims to test the factuality of LLMs on knowledge-intensive tasks. We choose MC1 as the test. 3) Anthropic-HH (Bai et al., 2022) [10] has released the evaluation set for both harmless and helpful perspective. For these tasks, we do not perform task-specific fine-tuning to show the zero-shot performance. Results in Table 8 showcase that our framework occasionally outperforms the sample scale baselines on some tasks, which meets our desiderata.

B.2 Performance on General NLP Tasks

We next evaluate the performance of InstructGraph on the general NLP tasks. We choose Big-Bench-Hard (BBH) (Suzgun et al., 2023) and Massive Multitask Language Understanding (MMLU) (Hendrycks et al., 2021) benchmarks with few-shot exemplars to perform reasoning. As shown in Table 9, even though these tasks do not belong to graph domains, we can still obtain competitive results compared with other same-scale open-source LLMs.

B.3 Parameter-Efficient Learning Study

To accelerate the training speed and reduce memory usage under the limitation of sources, we leverage parameter-efficient learning (PEL) techniques to equip the original LLM with only a few trainable parameters. To study the choice of different PEL methods, we compare LoRA with other PEL methods, such as Prefix-tuning (Li and Liang, 2021) [11], and Adapter (Houlsby et al., 2019). For each method, we choose six different scales and perform graph instruction tuning over 10% training data. The balance between trainable parameters and averaged results is visualized in Figure 4. We can see that LoRA can achieve the best performance and is similar to full fine-tuning regardless of the scale of trainable parameters.

B.4 Effectiveness of Code Format Graph

In this part, we evaluate the use of the structured format verbalizer when aligning the graph structure to the textual LLM. We choose four classic graph reasoning and generation tasks, i.e., PathQSP, WebNLG, CoRA, and UIE. To compare with the structured format verbalizer, we directly choose the heuristic template introduced by InstructGLM (Ye et al., 2023) to describe each path in the graph. For example, the path "$(e_1, r_1, e_2), (e_2, r_2, e_3)$" can be formulated as "$e_1$ is connected with e_3 within tow hops through e_2, and featured relations r_1 and r_2". We use this template to prompt GPT-4 and LLaMA2 to show the performance. The results in Table 10 demonstrate that our structured format verbalizer outperforms traditional templates in all tasks. Especially, the LLM with traditional templates cannot support graph generation, while the structured format verbalizer can reach this goal.

[9] https://github.com/sylinrl/TruthfulQA.
[10] https://github.com/anthropics/hh-rlhf.
[11] Prefix-Embedd: only tune the input embeddings layer; Prefix-Layer: tune each transformer layer.

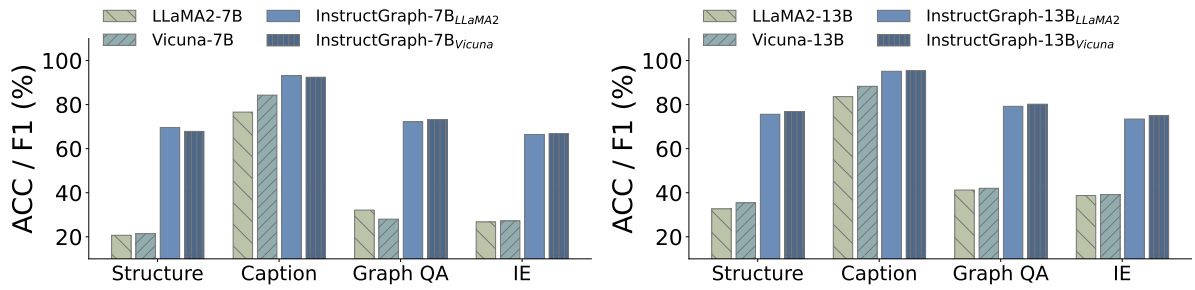

Figure 5: Effectiveness (%) of InstructGraph towards different scales and backbones.

Instruction (Graph Caption)	**LLaMA2 Output**	**InstructGraph Output**	**Reference**
You are a good graph reasoner. Give you a graph language that describes a graph structure and node information. You need to understand the graph and the task definition and answer the question. \n"'\nGraph[name='wikipedia-knowledge-graph'] \n entity_list = ['clown beetle', 'euspilotus cribrum', 'america', 'the family', 'species', 'histeridae'];\n triple_list = [('euspilotus cribrum' -> 'species')[relation='taxon rank'], ('the family' -> 'america')[relation='country']];\n\n"'\n Task definition: given a knowledge graph with all entities and structure triples representing factual and commonsense knowledge. Please leverage this graph to generate an encyclopedia passage. Note that do not list all knowledge in a running account.\n Q: Please generate an encyclopedia passage for the knowledge graph.\n A:	The relation between Euspilotus cribrum and species is 'taxon rank', and the family country is America. **[Human Evaluation]**: *This answer directly states each fact triple from the given graph separately. It is accurate but not natural and satisfied as a whole.*	Euspilotus cribrum is a species of clown beetle in the family Histeridae which is found in North America. **[Human Evaluation]**: *This answer naturally describes the factual knowledge. Overall, it is very similar to an encyclopedia document and makes the reading experience better.*	Euspilotus cribrum is a species of clown beetle in the family Histeridae. It is found in North America.
Instruction (Graph Generation)	**LLaMA2 Output**	**InstructGraph Output**	**Reference**
You are a good graph generator. You need to understand the task definition and generate a graph language to answer the question. \n Task definition: given a passage, extract some corresponding structure factual triples from the passage to form a knowledge graph, and generate a graph language to describe the graph. Note that: 1) the graph is a directed graph and the name is 'knowledge-graph'. 2) The entities must be mentioned in the passage. 3) The candidate relation type must be one of 'publication date', 'country of origin', 'publisher', 'performer'. 4) The generated graph language should be a code-like structure, and the skeleton format can be expressed as the following:\n "'\n Graph[name='knowledge-graph'] {\n entity_list = ['xxx', ···];\n triple_list = [('xxx' -> 'xxx')[relation='xxx'], ···];\n }\n"'\n Passage: "'Bluesman' is the 12th original album of Tak Matsumoto, the guitarist of the Japanese music group B'z. It was released by VERMILLION RECORDS on September 2, 2020.".\n Q: Given you a passage, please generate a corresponding knowledge graph.\n A:	Graph[name='knowledge-graph'] {\n entity_list = ['Bluesman', 'Tak Matsumoto', 'B'z'];\n triple_list = [('Bluesman' -> 'Tak Matsumoto) [relation='performer'], \n ('Bluesman' -> 'B'z') [relation='performer'], \n (Tak Matsumoto' -> 'B'z') [relation='performer'],\n ('Tak Matsumoto' -> 'VERMILLION RECORDS') [relation='publisher']];} **[Human Evaluation]**: *This answer sometimes contains wrong triples or missing some entities.*	Graph[name=knowledge-graph] {\n entity_list = [Bluesman, Tak Matsumoto, B'z, VERMILLION RECORDS, September 2, 2020];\n triple_list = [(Bluesman-> September 2, 2020) [relation=publication date], (Bluesman-> Tak Matsumoto) [relation=performer], (Bluesman-> VERMILLION RECORDS) [relation=publisher], (Bluesman-> Japan) [relation=country of origin]];\n} **[Human Evaluation]**: *This answer looks satisfying.*	Graph[name=knowledge-graph] {\n entity_list = [country of origin, performer, publication date, Bluesman, publisher];\n triple_list = [(Bluesman-> Japan) [relation=country of origin], (Bluesman-> VERMILLION RECORDS) [relation=publisher], (Bluesman-> September 2, 2020) [relation=publication date], (Bluesman-> Tak Matsumoto) [relation=performer]];\n}

Table 11: Human evaluation for the generation of LLaMA2 and InstructGraph.

B.5 Effectiveness of Different Backbones

To investigate whether the proposed InstructGraph can consistently improve the graph reasoning and generation ability with different LLMs, we select LLaMA2-7B, LLaMA2-13B, Vicuna-7B, and Vicuna-13B as the start checkpoints. To make the experiment efficient, we randomly choose 10% training data to perform graph instruction tuning and make a comparison with the corresponding vanilla LLMs. Results in Figure 5 show that InstructGraph can consistently achieve substantial improvement for arbitrary backbones and scales. Additionally, we observe that Vicuna has better performance than LLaMA2 initially. However, after graph instruction tuning, this trend is reversed. Upon further analysis, we find that both LLaMA2 and Vicuna were re-optimized based on LLaMA (Touvron et al., 2023a). Vicuna's optimization involves using supervised fine-tuning (SFT) to inject domain knowledge with massive conversation data into LLaMA. Meanwhile, LLaMA2 focuses on refactoring the model architecture and pre-training strategy to improve the model's versatility. Thus, Vicuna may have a better ability to understand instructions than LLaMA2. Despite this, LLaMA2 can be the better starting checkpoint for boosting LLMs on graph reasoning and generation tasks with parameter updates.

B.6 Human Evaluation

We end this section with a case study to demonstrate the performance of LLMs when solving graph reasoning and generation tasks. We choose LLaMA2 (7B) to make a comparison and respectively choose one example from graph caption generation and knowledge graph generation. For the answer, we perform a human evaluation to estimate the effectiveness of InstructGraph. As shown in Table 11, InstructGraph can outperform all the baselines. Specifically, compared with LLaMA2, InstructGraph can generate more natural and readable captions to describe factual information. For the graph generation, InstructGraph can provide accurate entities and triples.