LLM Meets Scene Graph: Can Large Language Models Understand and Generate Scene Graphs? A Benchmark and Empirical Study

Anonymous ACL submission

Abstract

The remarkable reasoning and generalization capabilities of Large Language Models (LLMs) have paved the way for their expanding applications in embodied AI, robotics, and other real-world tasks. To effectively support these applications, grounding in spatial and temporal understanding in multimodal environments is essential. To this end, recent works have leveraged *scene graphs*, a structured representation that encodes entities, attributes, and their relationships in a scene. However, a comprehensive evaluation of LLMs' ability to utilize scene graphs remains limited. In this work, we introduce **Text-Scene Graph (TSG) Bench**, a benchmark designed to systematically assess LLMs' ability to (1) understand scene graphs and (2) generate them from textual narratives. With TSG Bench, we evaluate 11 prominent LLMs and reveal that, while models perform well on scene graph understanding, they struggle with scene graph generation, particularly for complex narratives. Our analysis indicates that these models fail to effectively decompose discrete scenes from a complex narrative, leading to a bottleneck when generating scene graphs. These findings underscore the need for improved methodologies in scene graph generation and provide valuable insights for future research. The demonstration of our benchmark is available at https://tsg-bench.netlify.app.[1]

1 Introduction

Large language models (LLMs) have demonstrated impressive progress in various text-based tasks, such as question-answering and content generation, showcasing strong reasoning and generation capabilities (Brown et al., 2020; Touvron et al., 2023). Nevertheless, extending these abilities to multimodal environments is challenging, particularly when spatial and temporal reasoning about object relationships and physical interactions is required (Yan et al., 2023).

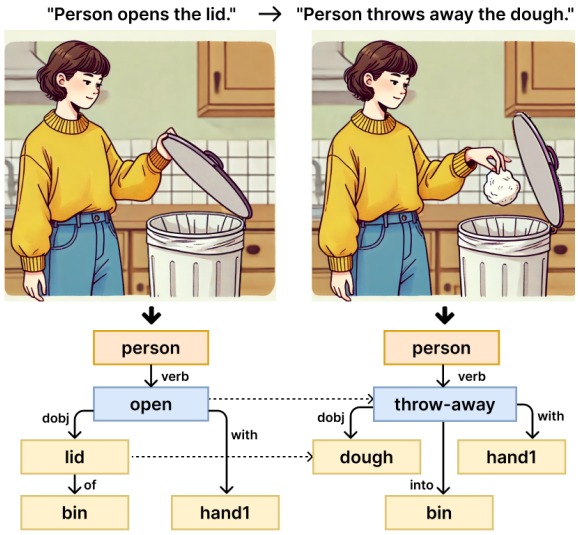

Figure 1: An illustration showing how a scene graph can represent the objects and their relationships in a scene. The illustration was created with the assistance of DALLE-3.[2]

To address this issue, researchers have explored leveraging *scene graphs* for LLMs (Chang et al., 2021). Scene graphs are structured representations that have been utilized in computer vision and embodied AI to capture key elements (*e.g.*, objects, their attributes, and relationships) in complex multimodal environments (Ji et al., 2020). As illustrated in Figure 1, by converting visual data into an interpretable representation, scene graphs enable LLMs to effectively understand spatial and semantic information, thereby allowing them to leverage their robust reasoning and generative capabilities in multimodal contexts. This integration paves the way for diverse applications ranging from dynamic scene interpretation to 3D environment modeling (Gao et al., 2023; Cong et al., 2023; Strader et al., 2024; Zhang et al., 2024).

[1] Our code and evaluation data are publicly available at https://anonymous.4open.science/r/TSG-Bench.

[2] https://openai.com/index/dall-e-3/

Despite these advancements, a comprehensive evaluation of LLMs' ability to interpret and generate scene graphs remains limited—leaving open questions such as whether these models genuinely comprehend the underlying spatial and semantic structures. Bridging this gap is essential for developing systems that can perform reliable and structured reasoning across diverse domains. For example, LLMs often struggle to identify critical nodes or edges (Huang et al., 2024) and might misinterpret the triplets in complex situations, particularly when handling long contexts (Kim et al., 2024).

We introduce **Text-Scene Graph Bench (TSG Bench)**, a benchmark designed to rigorously evaluate LLMs' ability in scene graph understanding and generation. TSG Bench comprises long-text narratives that describe real-world scenarios alongside corresponding sequences of scene graphs representing an actor's interactions with objects. In every task, contextual grounding is ensured via a preceding scenario—delivered either in text or graph format—to mirror practical applications. For understanding tasks, we assess the models' ability to interpret and reason over scene graphs. For generation tasks, we assess how accurately models can generate structured scene graphs given descriptive narratives and preceding contexts of varying complexity.

Through extensive experiments on eleven prominent LLMs using TSG Bench, we make three key observations. (1) LLMs exhibit strong performance on scene graph understanding tasks. However, they significantly underperform on generation tasks, especially when faced with narratives that should be implicitly decomposed into multiple actions (*e.g.*, implicit and repeated actions). (2) Advanced techniques, such as in-context learning and chain-of-thought (CoT) prompting, can facilitate the ability of highly capable LLMs to represent and reason over scene graphs. (3) LLMs can effectively refine errors in scene graphs when guided with error types. These findings highlight the need for improved methodologies in scene graph generation and provide critical insights for applications of LLMs in multimodal environments in future research.

2 Related Work

Scene graph representation. Scene graphs, which encode semantic information and relationships between objects in a scene, are widely used in multimodal tasks such as image captioning, 3D reconstruction, and robotics (Johnson et al., 2015; Yao et al., 2018; Chatterjee et al., 2021; Armeni et al., 2019; Rosinol et al., 2020). Previous research has leveraged scene graphs as an inductive bias to facilitate explicit reasoning in visual question answering tasks (Teney et al., 2017; Hildebrandt et al., 2020) or to provide explanations based on such knowledge (Shi et al., 2018). Scene graphs can also serve as semantic maps of real-world environments for robots, enabling higher-level reasoning for tasks like planning and navigation (Rana et al., 2023; Dai et al., 2023; Yin et al., 2024).

Scene graph datasets. Such works have been facilitated by several benchmarks and datasets, most of which is built on image-scene graph pairs (Krishna et al., 2016; Ji et al., 2020). While FACTUAL (Li et al., 2023) suggests a text-based benchmark for scene graph parsing, the dataset is focused on static scenes which restricts their applicability to dynamic, real-world scenarios. In contrast, our work extends to dynamic scenarios, where spatial, temporal relationship between scenes are actively involved. Ego-centric Action Scene Graphs (EASG) dataset (Rodin et al., 2024) explores the relationship between time-evolving actions and scene graphs in video contexts. In contrast, our benchmark, TSG Bench, is designed to evaluate the two distinct ability of LLMs – reasoning and generation – to link textual narratives and dynamic scene graphs to reflect richer context.

LLMs for scene graphs. Large Language Models, with their powerful reasoning and generation capabilities, have been employed for both understanding and generating scene graphs across various tasks. Recent works on robotics utilized LLMs to reason over scene graphs or formulate high-level plans for navigation within complex environments. (Yin et al., 2024; Rana et al., 2023). Beyond understanding, LLMs have also been applied to generate scene graphs by representing multiple objects and relations in a complex text and incorporating commonsense reasoning for 3D content generation tasks (Gao et al., 2023; Wei et al., 2024). While previous efforts have explored the use of LLMs for processing scene graphs, TSG Bench is the first benchmark to provide a unified evaluation framework for assessing LLMs' capabilities in understanding and generating scene graphs.

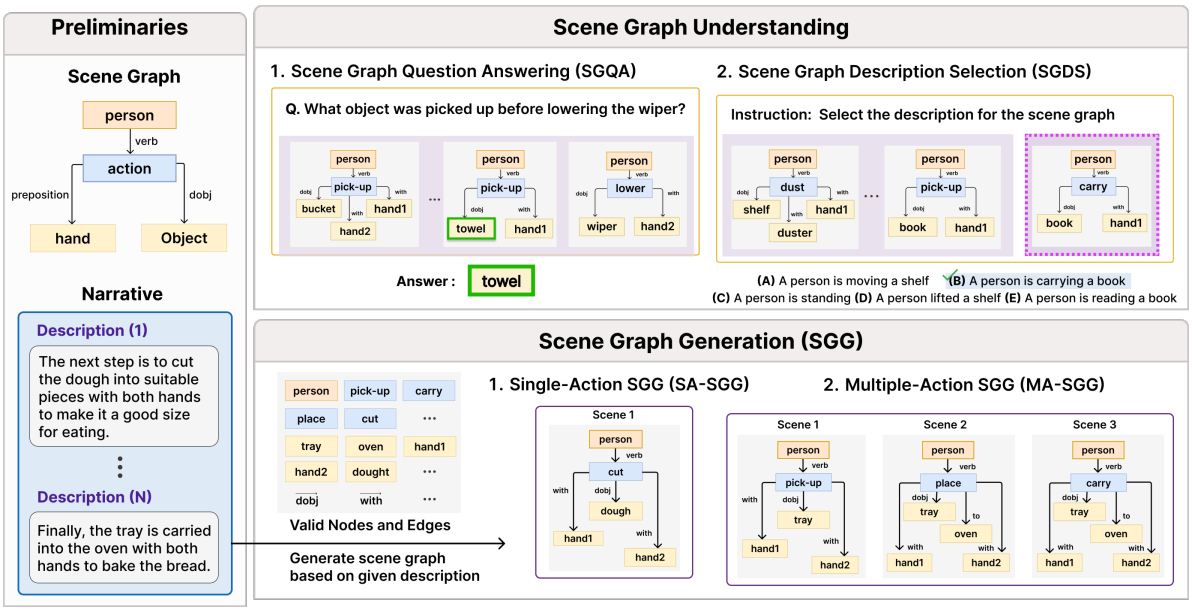

Figure 2: **Overview of TSG Bench.** Scene graph question answering aims to answer a question by reasoning over scene graphs. Scene graph description selection is a multiple-choice task of selecting the correct description of a given scene graph. Single action scene graph generation focuses on generating a scene graph based on a description representing a single action. Multiple action scene graph generation aims to generate multiple discrete scene graphs of all actions represented in the description.

3 TSG Bench

In this section, we introduce **TSG Bench**, a benchmark to evaluate LLMs' abilities to understand and generate scene graphs based on narratives. We begin by introducing the preliminaries of data representations, which form the basis of the benchmark. Next, we detail the scene graph understanding and generation tasks, followed by an explanation of benchmark construction process. Finally, we provide the statistics of TSG Bench.

3.1 Preliminaries: Text and Scene Graph Representations

We first construct a database of narratives and scene graphs representing sequential scenes of real-world scenarios, and configure understanding and generation tasks from it. The narrative is composed of multiple coherent natural language descriptions, denoted as $D = (d_1, \ldots, d_n)$ where n represents the number of descriptions in each scenario. For each description d_i, a set of scene graphs $G_i = (G_{i1}, \ldots, G_{ik})$ is aligned, where each scene graph represents an action, following the action-centric scene graph representation proposed in a previous work (Rodin et al., 2024). In the case of our database, the number of scene graphs k for a description range from 1 to 8, depending on the complexity of d_i. For example, when a description of *shaking* an object is given, k=2 because it can be decomposed into *holding* and *rapidly moving* the object. For each j such that $1 \leq j \leq k$, the individual scene graph is denoted as $G_{ij} = (V_{ij}, E_{ij})$, where V_{ij} and E_{ij} represent the nodes and edges of the scene graph, respectively. More specifically, the relationships between nodes and edges are expressed as a set of triplets in the form of (source node, edge, target node).

Nodes and edges. A node belongs to one of four categories {`person`, `action`, `object`, `hand`}, where `person` represents an actor, `action` denotes the actor's action in the scene, `object` refers to an item or a location, and `hand` corresponds to either `hand1` or `hand2` of the actor. Similar to previous works (Rodin et al., 2024; Grauman et al., 2022), we use `hand` nodes to track activities involving one or both hands. Specifically, once an item is held, it occupies `hand1`; if a second item is acquired, `hand2` is assigned. Releasing all items resets the next grasp to `hand1`. An edge belongs to one of three categories {`verb`, `dobj`, `preposition`}, where `verb` connects a `person` node to an `action` node, `dobj` links an `action` node to an `object` node only if it is the direct object of the action, and `preposition` connects any pair for representing the spatial relationship or other contextual dependencies between them. To ensure consistency and

standardize elements (*i.e.*, nodes and edges) in the graph, we define L as a predefined collection of valid node and edge values for each scenario. Consequently, any additional modifiers or redundant expressions in descriptions D are omitted to maintain a concise representation.

3.2 Scene Graph Understanding

Scene graph question answering (SGQA). This task involves reasoning over scene graphs to answer a given question. Formally, given a question and scene graphs $G = (V, E)$, the model must predict an answer, which corresponds to an element in V. As shown in Figure 2, questions in our benchmark require logically or temporally connecting a sequence of actions or object state changes, which can be solved by hopping across multiple triplets.

Scene graph description selection (SGDS). The goal of this task is to accurately interpret a scene graph within a given context and identify the correct description among distractors. We formulate SGDS as a multiple-choice question problem, consisting of the graph-based context $C_i^g = (G_1, \ldots, G_{i-1})$, a scene graph G_i, and five candidate descriptions, with one correct answer included. The model should be able to track nodes and edges from C_i^g and ensure that all elements in G_i are accurately represented. For SGDS, we use scene graphs representing a single action.

3.3 Scene Graph Generation

Scene graph generation tasks aim to generate triplets of scene graphs corresponding to a given description within a context. All valid elements are predefined as L for each scenario, and the tasks require models to identify and parse semantically similar elements from the given description to construct triplets. Each task is illustrated in Figure 2.

Single action scene graph generation (SA-SGG). SA-SGG is a task of generating a scene graph for a description that involves a single action, within a scenario. Formally, the task is to generate triplets of $G_i = (V_i, E_i)$, given the description context $C_i^d = (d_1, \ldots, d_{i-1})$, a description d_i, and valid nodes and edges of the scenario, denoted as V and E, respectively.

Multiple action scene graph generation (MA-SGG). MA-SGG aims to generate scene graphs by decomposing actions when given complex descriptions that involve multiple actions. The task formulation is identical to that of SA-SGG, except that an additional clue indicating the number of actions is provided, and the complexity of d_i is greater than 1. This makes MA-SGG more challenging than SA-SGG because the amount of information to process, especially to generate, is larger, and target actions may be implicit in the description. Also, although the number of actions is given, the task still requires the ability to accurately decompose, identify, and order valid actions from the description.

3.4 Dataset Construction

We derive TSG Bench from the Ego-centric Action Scene Graphs (EASG) dataset (Rodin et al., 2024), which represents temporally evolving actions in video contexts as scene graphs. We use the original scene graphs to build our own narratives and corresponding scene graphs through multiple rounds of a human-in-the-loop process involving three trained annotators. First, we prompt an LLM to generate a sentence from each scene graph from the EASG dataset and remove redundant sentences based on context. After human workers review the logical flow and naturalness, we prompt the LLM again to generate a scene graph for each sentence. As they often fail to generate complete scene graphs, human workers meticulously inspect and refine the graph elements one by one. Then, we prompt the model to paraphrase sentences to increase lexical diversity and to combine coherent sentences, enhancing overall complexity. As a result, we collect 120 scenarios of 2,041 descriptions and 4,289 scene graphs.

Task-related data. We additionally construct data for our understanding tasks through a similar collaboration process. For SGQA, we provide the LLM with the entire scenario narrative and prompt it to generate five questions about identifying a node that had undergone spatial and temporal transitions. Human workers then verify the validity of these questions. For SGDS, we control the difficulty of the distractors by perturbing the answer description in two ways. We put random distractors from unrelated scenarios for the half, and put distractors with LLM-perturbed nodes and edges from the answer for the other half of the problems. More detailed descriptions of our dataset construction process are in Appendix A.

Statistics	Counts
Benchmark Statistics	
# of Domains	18
# of Scenarios	120
# of Descriptions	2,041
# of Scene graphs	4,289
– avg. # nodes	4.81
– avg. # edges	3.45
# of Nodes	14,905
# of Edges	11,820
Task Statistics	
# data for SGQA	500
# data for SGDS	250
# data for SA-SGG	1,188
# data for MA-SGG	853
– avg. # scene graphs	3.64

Table 1: Statistics of TSG Bench and each task.

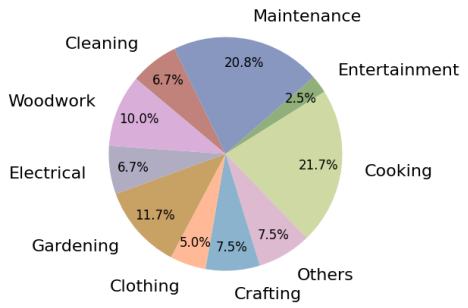

Figure 3: Domain distribution across scenarios.

3.5 Benchmark Statistics

Table 1 summarizes the statistics of TSG Bench and each task. Our benchmark provides 120 real-world scenarios, covering a wide range of domains, including maintenance, cooking, and gardening, as illustrated in Figure 3. TSG Bench contains 2,041 descriptions and 4,298 scene graphs, and 14,905 and 11,820 different nodes and edges, respectively, covering a wide semantic space. As we focus on action-centric scenarios, common action nodes include *pick-up*, *place*, *hold*, and *release*, and the most common preposition edge is *with*, involving hand movements.

For understanding tasks, we construct 500 datapoints for SGQA, and 250 for SGDS. For generation tasks, there are 1,188 data samples for SA-SGG and 853 for MA-SGG. For MA-SGG, the average number of scene graphs to generate, or the complexity k, is approximately 3.64.

4 Experiments

4.1 Setup

We conduct our experiments on eleven highly capable LLMs to provide a comprehensive assessment of current LLMs. (1) For proprietary models, we choose GPT-4o, GPT-4o-mini (OpenAI, 2024), Claude-3.5-Sonnet, and Claude-3.5-Haiku (2024, 2024); (2) For open-source models, we select LLaMA-3.3-70B (AI, 2024a), Qwen-2.5-72B, Qwen-2.5-7B (Team, 2024), DeepSeek-V3 (DeepSeek-AI, 2024), Mixtral-8x22B (AI, 2024c), Mistral-large, and Mistral-7B (AI, 2024b). Additionally, we also provide human performance on a subset of 30 examples to facilitate the interpretation of the results. Experiment details are in Appendix B.

4.2 Evaluation Protocols

We evaluate LLMs on their capabilities in scene graph understanding and generation by prompting them in a zero-shot fashion, using prompts listed in Appendix D.1. We use different metrics for each task. We assess SGQA using Exact Match (EM), which requires the model's generation to match the reference element exactly. For SGQA, we instruct LLMs to generate a single letter representing the predicted candidate and evaluate it using accuracy. Scene graph generation tasks are assessed with precision, recall, and macro F1 score. For MA-SGG, where one description yields multiple scene graphs, evaluation is conducted separately for each generated graph.

4.3 Main Results

We compare the scene graph understanding (SGDS, SGQA) and generation (SA-SGG, MA-SGG) performance of different models on TSG Bench in Table 2.

Scene graph understanding. Most models exhibit strong performance in scene graph understanding tasks, particularly in the SGDS task. Among the models, Claude-3.5-Sonnet shows relatively strong performance—98.40 in SGDS and 90.60 in QA. Even though open source models such as LLaMA-3.3-70B achieve competitive performance (97.60 in SGDS, 84.60 in SGQA), no model consistently outperforms strong proprietary models such as Claude-3.5-Sonnet and GPT-4o. For the relatively smaller open-source models (Qwen-2.5B-7B, Mistral-7B), while the models demonstrate competitive performance in SGDS task (93.60 and 90.14

Model	SGDS	SGQA	SA-SGG			MA-SGG		
	Accuracy	EM	Precision	Recall	F1	Precision	Recall	F1
Human	98.33	88.00	85.22	81.00	82.50	78.80	72.90	75.60
Proprietary models								
GPT-4o	96.40	84.80	65.84	55.04	59.23	48.88	40.84	43.99
GPT-4o-mini	96.80	76.60	20.00	21.50	19.90	23.06	18.32	20.07
Claude-3.5-Sonnet	**98.40**	**90.60**	**69.75**	**69.33**	**68.43**	**60.77**	**57.91**	**58.80**
Claude-3.5-Haiku	97.20	82.00	38.31	36.82	36.77	27.00	23.97	24.95
Open source models								
LLaMA-3.3-70B	**97.60**	**84.60**	31.52	38.90	33.37	32.43	26.58	28.92
Qwen-2.5-72B	96.80	81.40	57.96	53.22	54.42	42.64	33.29	36.78
DeepSeek-V3	96.40	79.60	55.79	55.11	54.45	**43.67**	**36.66**	**39.34**
Mixtral-8x22B	96.00	73.00	31.03	32.79	30.84	21.05	19.54	19.75
Mistral-large	96.40	82.40	**63.18**	**55.37**	**58.15**	40.17	32.12	35.13
Qwen-2.5-7B	93.60	73.40	9.58	9.95	9.39	6.61	6.77	6.34
Mistral-7B	90.14	58.20	13.60	14.64	13.14	13.86	10.57	11.67

Table 2: Main results for scene graph understanding tasks (SGDS, SGQA) and scene graph generation tasks (SA-SGG, MA-SGG). The full prompts are listed in Appendix D.1.

respectively), the performance still falls short in the SGQA task compared to other models with a larger number of parameters—73.40 for Qwen-2.5-7B and 58.20 for Mistral-7B in EM.

Scene graph generation. In contrast to scene graph understanding tasks, scene graph generation remains challenging for LLMs. Similar to the trend in understanding tasks, Claude-3.5-Sonnet achieves the highest scores among the models – F1 of 68.43 (SA-SGG) and 58.80 (MA-SGG). However, a notable difference is that Claude-3.5-Sonnet is no longer as good as a human in the scene graph generation task, scoring lower compared to human performance (82.50, 68.43). The performance gap between LLMs and humans is evident in generating multiple coherent scene graphs (MA-SGG). For example, the performance of Claude-3.5-Haiku, which is one of the most competitive models in SGQA, decreases by 58.8% in MA-SGG. Many models also show higher precision than recall in this setting (*e.g.*, Mistral-large: 40.17 vs. 32.12), indicating incomplete coverage of sub-scenes. This shortfall arises from the need to split a single description into multiple scenes and construct separate graphs for each, increasing structural complexity and reducing recall. We provide deeper analysis on generation task by disentangling the task into three distinct subtasks in Section 5.1. Open-source models perform worse, with Qwen-2.5-72B (66.15, 43.73) and Mistral-large (69.76, 37.76) being the strongest but still far behind proprietary models. Smaller models like Qwen-2.5-7B and Mistral-7B fail severely in MA-SGG (< 12 in F1).

5 Analysis

To provide further insight on leveraging LLM for scene graph tasks, we explore the following four research questions.

5.1 Which Challenges Arise When LLMs Generate Scene Graphs?

The main results in Table 2 indicate that language models struggle in generation tasks compared to understanding tasks. Hence, we aim to discover the underlying challenges by configuring three subtasks of scene graph generation: node generation, edge generation, and action decomposition. For a clearer analysis, we exclude action nodes in the reference for node generation subtask. Note that, both single and multiple action scene graph generation tasks, node and edge generation abilities are essential, and in multiple action scene graph generation task, action decomposition ability is additionally required.

Since a scene graph is represented with triplets of nodes and edges, we ablate the effect of each by conditioning on the other to generate either one. Following the notations described in Section 3, node generation task is formulated as $(C_i^d, d_i, L, E_i) \rightarrow V_i - \{action\}$. The edge generation task is formulated as $(C_i^d, d_i, L, V_i) \rightarrow G_i$,

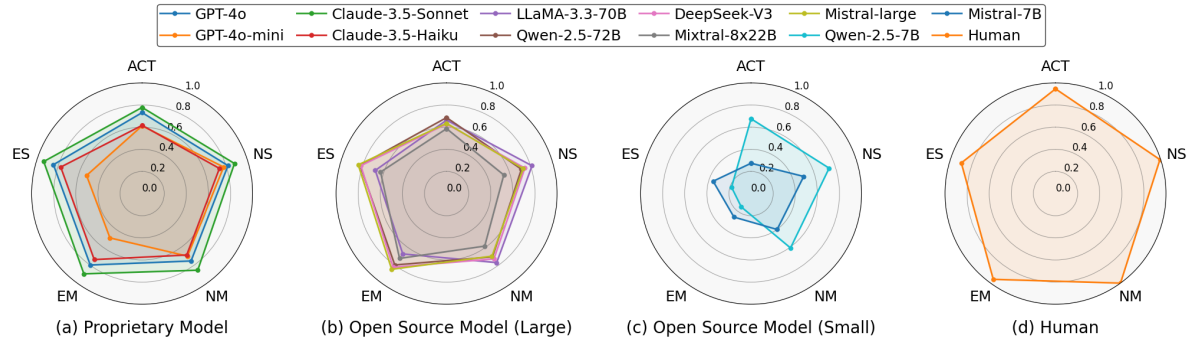

Figure 4: F1-score results on decomposed scene graph generation tasks, distinguishing between single-action and multiple-action settings. ES (Edge Single) and NS (Node Single) evaluate edge and node generation performance in SA-SGG, respectively. EM (Edge Multiple) and NM (Node Multiple) assess edge and node generation in MA-SGG. ACT (Action) measures the model's performance in action decomposition in MA-SGG.

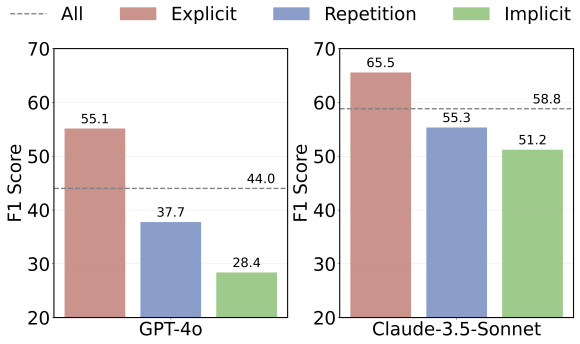

Figure 5: The results of the comparative evaluation under four conditions—Explicit, Implicit, Repetition, and All. The "All" condition comprises the entire dataset, while the other three focus on subsets featuring explicit actions, implicit actions, or repeated actions.

[Context] ... The timber was carefully dehydrated to ensure it was ready for the next stage of the woodworking project.
[Description] The timber was **cleaned** using a fabric.
[Nodes in the Reference Graph] V_1: person, **pick-up**, cloth, ... V_2: person, **wipe**, wood, ...
[Nodes in the Generated Graph] V_1: person, **wipe**, cloth, ... V_2: person, **hold**, wood, ...

Table 3: A failure case of action decomposition, which involves two consecutive actions: picking up a cloth and wiping the wood with it. V_j indicates the list of nodes in the j-th graph of the description.

as edges are designed to represent the relationships between two nodes. Lastly, action decomposition aims to generate all actions in a correct sequence, given C_i^d, d_i, and L.

According to Figure 4, we find that large models tend to generate edges better than nodes. On the other hand, small language models perform poorly on edge generation. Furthermore, in the MA-SGG, most models lack in decomposing actions correctly in complex descriptions.

5.2 Which Action Do LLMs Struggle to Generate?

We hypothesize that LLMs may struggle to represent structured graphs when descriptions contain implicit actions. To examine this, we curated a subset of descriptions in which referenced actions are not explicitly stated in any variant form. We then conducted MA-SGG experiment using GPT-4o and Claude-3.5-Sonnet. As shown in Figure 5, both models perform worse when implicit actions are present, while they perform better when generating explicit actions.

Through manual analysis on failure cases, we discovered another challenging type of actions, repetitive actions (*e.g.*, *sweep three times*). We curated another subset of descriptions containing repetitive actions by filtering for words such as *times, repeat, and repeated*. As shown in Figure 5, the models struggle with generating the same action multiple times. Consistent with previous studies (Brown et al., 2020), our results suggest that LLMs lack a strong numerical sense, which humans find trivial. Examples are shown in Appendix C.5.

5.3 Do Advanced Prompting Methods Elicit Better Performance?

We further investigate whether advanced prompting methods for improving LLM capabilities can

Method	SGDS	QA	SA-SGG	MA-SGG
GPT-4o	96.40	84.80	59.23	43.99
+ CoT	96.80	**90.00**	**67.13**	44.79
+ 10-shot	**99.20**	84.40	65.78	**57.40**
Claude-3.5-Sonnet	98.40	90.60	68.43	58.80
+ CoT	98.00	**94.00**	69.57	64.36
+ 10-shot	**98.80**	92.00	**75.29**	**71.75**
Qwen-2.5-72B	96.80	81.40	54.42	36.78
+ CoT	**97.60**	**88.00**	53.43	31.33
+ 10-shot	**97.60**	84.60	**67.87**	**53.47**
Mistral-large	96.40	82.40	58.15	35.13
+ CoT	95.20	**96.00**	62.45	32.39
+ 10-shot	**98.80**	85.40	**66.99**	**48.10**
Qwen-2.5-7B	93.60	**73.40**	9.39	6.34
+ CoT	94.00	72.00	11.56	3.88
+ 10-shot	**95.60**	73.20	**39.96**	**37.25**
Mistral-7B	90.14	58.20	13.14	11.67
+ CoT	94.00	**68.00**	10.97	6.32
+ 10-shot	**94.80**	66.20	**37.97**	**33.74**

Table 4: The results of each method are evaluated using the same setup as the main results. Rows listing only the model name correspond to the vanilla (zero-shot) setting. The detailed prompts for both CoT and few-shot approaches are provided in the Appendix D.2 and D.3.

Error Type	*w/o* Error Type		*w/* Error Type	
	GPT-4o	Claude	GPT-4o	Claude
Overall	40.04	60.03	64.80	88.28
Redundant	64.38	70.02	73.29	93.06
Missing	46.67	82.25	58.42	80.89
Mismatched	10.81	38.29	58.92	81.94
Reversed	44.24	49.55	68.55	97.22

Table 5: Refinement results with and without error-type awareness, evaluated with the F1-score. "*w/o* Error Type" denotes refinement without error type, and "*w/* Error Type" denotes refinement with error type. "Claude" refers to the Claude-3.5-Sonnet model.

benefit the models in Table 4. To assess the effectiveness of popular LLM prompting techniques, we conduct experiments with Chain-of-Thought (CoT) prompting and 10-shot in-context learning (ICL). The results indicate that 10-shot ICL generally improved performance across tasks, particularly for SGDS, SA-SGG, and MA-SGG. These tasks require a deeper understanding and structured representation of scene graphs, making ICL an effective approach. In contrast, CoT prompting proved beneficial for reasoning-intensive tasks such as SGQA. However, the performance gains varied across models. For instance, Qwen-2.5-7B, which already exhibited relatively high initial performance, showed minimal improvement with CoT, potentially due to longer prompts introducing confusion. On the other hand, Mistral-7B, which initially demonstrated lower performance, benefited significantly from CoT, suggesting that reasoning augmentation can compensate for weaker baseline capabilities.

5.4 Can LLMs Refine Errors in Scene Graphs?

We assess whether LLMs have the ability to refine an incorrect scene graph. To conduct a controlled analysis, we curated 5,940 data samples with different types of errors in the scene graph: *Redundant* (extra triplet), *Missing* (omitted triplet), *Mismatched* (perturbed element), and *Reversed* (inverted directions). Given the context, the description, and the erroneous graph, we prompt GPT-4o and Claude-3.5-Sonnet to refine the graph to align with the meaning of d. As shown in Table 5, while both models generally underperform, it stands out for the *Mismatched* type. These findings highlight the importance of precise interpretation and correction of subtle errors within scene graphs.

We further investigate whether the models' refinement challenges arise from insufficiently detecting error types or from difficulties in correcting errors they have already identified. To address this, we evaluate whether providing the models with ground-truth error-type labels can improve refinement. Our results show that with this additional guidance, both models exhibit improved performance across all types, particularly in *Mismatched*. These findings indicate that clarity regarding error types facilitates more effective LLM-based scene graph refinement.

6 Conclusion

We present TSG Bench, a benchmark for assessing large language models in scene graph understanding and generation. TSG Bench comprises textual narratives and corresponding scene graphs to evaluate interpretation, reasoning, and structured representation derivation. Our experiments with 11 LLMs revealed moderate success in understanding tasks yet difficulties in generation tasks, with action decomposition posing the largest obstacle. We believe this study bridges the gap between LLMs and structured scene representations, establishing a foundation for more effective multimodal understanding and generation.

Limitations

In the current study, both SGG tasks of TSG Bench involve generating action-centric scene graphs from textual narratives. However, the current textual narratives do not include object attributes (*e.g.*, color, size), nor does TSG Bench incorporate attributes in the scene graph generation process. As TSG Bench centers on a single actor, tasks do not involve additional individuals. Constructing a benchmark with more intricate scene graphs, including attributes and multiple actors, could provide more comprehensive insights. The primary objective of this study is to benchmark widely adopted LLMs for scene graph-related tasks, leveraging their knowledge and reasoning. Future directions may integrate multi-modal approaches, such as utilizing Vision-Language Models (VLMs) to derive textual narratives from images or videos.

References

Anthropic. 2024. 2024. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*, 1.

Meta AI. 2024a. Llama 3.3: A multilingual large language model. Accessed: 2025-02-04.

Mistral AI. 2024b. Mistral large: A cutting-edge text generation model. Accessed: 2025-02-04.

Mistral AI. 2024c. Mixtral 8x22b: A sparse mixture-of-experts language model. Accessed: 2025-02-04.

Iro Armeni, Zhi-Yang He, JunYoung Gwak, Amir Zamir, Martin Fischer, Jitendra Malik, and Silvio Savarese. 2019. 3d scene graph: A structure for unified semantics, 3d space, and camera. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5663–5672.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Xiaojun Chang, Pengzhen Ren, Pengfei Xu, Zhihui Li, Xiaojiang Chen, and Alex Hauptmann. 2021. A comprehensive survey of scene graphs: Generation and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):1–26.

Moitreya Chatterjee, Jonathan Le Roux, Narendra Ahuja, and Anoop Cherian. 2021. Visual scene graphs for audio source separation. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1184–1193.

Yuren Cong, Jinhui Yi, Bodo Rosenhahn, and Michael Ying Yang. 2023. Ssgvs: Semantic scene graph-to-video synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2555–2565.

Zhirui Dai, Arash Asgharivaskasi, Thai P. Duong, Shusen Lin, Maria-Elizabeth Tzes, George J. Pappas, and Nikolay Atanasov. 2023. Optimal scene graph planning with large language model guidance. *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14062–14069.

DeepSeek-AI. 2024. Deepseek-v3: Advancements in mixture-of-experts language modeling. Accessed: 2025-02-04.

Gege Gao, Weiyang Liu, Anpei Chen, Andreas Geiger, and Bernhard Schölkopf. 2023. Graphdreamer: Compositional 3d scene synthesis from scene graphs. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21295–21304.

Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. 2022. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18995–19012.

Marcel Hildebrandt, Hang Li, Rajat Koner, Volker Tresp, and Stephan Günnemann. 2020. Scene graph reasoning for visual question answering. *ArXiv*, abs/2007.01072.

Haoyu Huang, Chong Chen, Conghui He, Yang Li, Jiawei Jiang, and Wentao Zhang. 2024. Can llms be good graph judger for knowledge graph construction? *arXiv preprint arXiv:2411.17388*.

Jingwei Ji, Ranjay Krishna, Li Fei-Fei, and Juan Carlos Niebles. 2020. Action genome: Actions as compositions of spatio-temporal scene graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10236–10247.

Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. 2015. Image retrieval using scene graphs. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3668–3678.

Kibum Kim, Kanghoon Yoon, Jaehyeong Jeon, Yeonjun In, Jinyoung Moon, Donghyun Kim, and Chanyoung Park. 2024. Llm4sgg: Large language models for weakly supervised scene graph generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28306–28316.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. 2016. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123:32 – 73.

Zhuang Li, Yuyang Chai, Terry Yue Zhuo, Lizhen Qu, Gholamreza Haffari, Fei Li, Donghong Ji, and Quan Hung Tran. 2023. Factual: A benchmark for faithful and consistent textual scene graph parsing. In *Annual Meeting of the Association for Computational Linguistics*.

OpenAI. 2024. Gpt-4o system card. https://cdn.openai.com/gpt-4o-system-card.pdf. Accessed: 2024-09-26.

Krishan Rana, Jesse Haviland, Sourav Garg, Jad Abou-Chakra, Ian D Reid, and Niko Suenderhauf. 2023. Sayplan: Grounding large language models using 3d scene graphs for scalable task planning. *CoRR*.

Ivan Rodin, Antonino Furnari, Kyle Min, Subarna Tripathi, and Giovanni Maria Farinella. 2024. Action scene graphs for long-form understanding of egocentric videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18622–18632.

Antoni Rosinol, Arjun Gupta, Marcus Abate, J. Shi, and Luca Carlone. 2020. 3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans. *ArXiv*, abs/2002.06289.

Jiaxin Shi, Hanwang Zhang, and Juan-Zi Li. 2018. Explainable and explicit visual reasoning over scene graphs. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8368–8376.

Jared Strader, Nathan Hughes, William Chen, Alberto Speranzon, and Luca Carlone. 2024. Indoor and outdoor 3d scene graph generation via language-enabled spatial ontologies. *IEEE Robotics and Automation Letters*.

Qwen Team. 2024. Qwen2.5: A party of foundation models! Accessed: 2024-02-04.

Damien Teney, Lingqiao Liu, and Anton van Den Hengel. 2017. Graph-structured representations for visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

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*.

Yao Wei, Martin Renqiang Min, George Vosselman, Li Erran Li, and Michael Ying Yang. 2024. Planner3d: Llm-enhanced graph prior meets 3d indoor scene explicit regularization.

He Yan, Xinyao Hu, Xiangpeng Wan, Chengyu Huang, Kai Zou, and Shiqi Xu. 2023. Inherent limitations of llms regarding spatial information. *arXiv preprint arXiv:2312.03042*.

Ting Yao, Yingwei Pan, Yehao Li, and Tao Mei. 2018. Exploring visual relationship for image captioning. In *European Conference on Computer Vision*.

Hang Yin, Xiuwei Xu, Zhenyu Wu, Jie Zhou, and Jiwen Lu. 2024. Sg-nav: Online 3d scene graph prompting for llm-based zero-shot object navigation. *arXiv preprint arXiv:2410.08189*.

Hang Zhang, Zhuoling Li, and Jun Liu. 2024. Scenellm: Implicit language reasoning in llm for dynamic scene graph generation.

Appendix

A Dataset Construction Details

We build upon the Ego-centric Action Scene Graphs (EASG) dataset (Rodin et al., 2024) to create a benchmark of our database and all four tasks. EASG provides spatio-temporal scene graphs derived from first-person video, where each graph encodes actions and objects from the perspective of the camera wearer. Although EASG dataset contains text annotations, they are simple verbalizations of the graph, without any contextual interaction between texts in a scenario. We instead aim to construct a database of human-like sentences, ensuring logical flow, contextuality, and complexity. For data construction, we perform three main steps, each serving a distinct purpose. All steps are done by a collaboration with an LLM and human annotators. Three annotators worked on human annotation, and they are paid on the hourly rate exceeding $18, based on the estimated time required to complete the tasks. The screenshot of the annotation page is shown in Figure 6.

Step 1: Narrative annotation. We utilize the scene graphs in EASG dataset, which cover diverse real world scenarios, to generate the initial natural language descriptions by prompting GPT-4o-mini. Then, human annotators remove redundant sentences, add missing necessary sentences, and fix unrealistic sentences (*e.g.*, a single hand from holding multiple objects) considering the context.

Step 2: Scene graph annotation. Then, we generate scene graphs with GPT-4o based on each narrative resulted from Step 1. Here, we let the model generate elements from the predefined set of the EASG dataset. As the model often fails to generate complete graphs, human workers check each instance one by one, whether any mismatch between the description and scene graph exists, including missing elements. During this process, we add

some frequently appearing elements and obtain the set of valid nodes and edges for each scenario.

Step 3: Text quality improvement. Although we have annotations for description and scene graph pairs, we additionally work on improving the quality of the text narratives, in terms of lexical diversity, coherency, and complexity. We prompt GPT-4o to paraphrase each sentence, given the preceding context, and also identify which word changes it has made. As this process may produce a description containing a similar element in the predefined set, which may yield a misleading annotation with the reference graph, we skip the instance. After paraphrasing, we prompt the model again to increase fluency regarding the context (*e.g.*, adding a conjunction or modifier).

Data construction for understanding tasks. For generating questions for SGQA, we provide GPT-4o with the entire context in graph form to make multi-hop questions. We prompt it to make questions about common elements across multiple triplets (*e.g.*, first object that had been grabbed), the state of nodes under specific conditions (*e.g.*, before or after an action happened), etc.. For SGDS, we also prompt GPT-4o to perturb the candidates to generate sentences with similar words but that have different meanings with the reference. To finalize, human reviewers inspect and correct these questions and answers to ensure accuracy.

B Experiment Details

We used OpenRouter (https://openrouter.ai/) for LLM prompting experiments. The temperature for models is set to 0.1 after a pilot study, and the scores in the experiment tables are results of running the inference only once. We also used NLTK(https://www.nltk.org/) for data processing, like lemmatization. The human performer is proficient in English and was instructed with task guidelines.

C Additional Analysis

C.1 Actions that struggle with scene segmentation

We first explore which actions most frequently lead to segmentation errors and find that *release*, *put-down*, *hold*, *place*, and *pick-up* account for the majority of failures. These actions are often not explicitly mentioned in text, making it challenging for LLMs to infer implicit states when textual details are minimal. For instance, models overlook the moment an object is released or fail to recognize that an entity is already engaged in an action, causing them to generate incoherent action boundaries. This tendency arises because the text does not always specify when an action is completed or whether an entity is available for another action, thereby demanding implicit reasoning that LLMs have difficulty performing reliably.

C.2 Impact of Repeated Scenes

We found that scene graph generation performance suffers notably when repeated actions occur. To investigate whether poor segmentation drives this decline, we examined two main patterns of repetition: (1) a single action repeated multiple times (*e.g.*, "knead, knead, knead"), and (2) alternating actions that recur (*e.g.*, "roll, stretch, roll, stretch"). In both patterns, the model often maintains relatively high recall but exhibits a marked drop in precision, which lowers overall performance. Interestingly, when the number of repetitions is small, performance decreases even further, suggesting that the model can neither clearly merge nor distinctly separate these scenes. Nevertheless, once actions are segmented correctly, generating the corresponding scene graphs becomes less problematic. These findings highlight the critical role of reliable segmentation in handling repetitive sequences, emphasizing that errors in early segmentation steps can negatively affect downstream tasks.

C.3 Navigating Edge Formation Under Varied Information Levels

We examined how sentence length and edge density influence generation quality. In lengthy sentences with few relations, LLMs can often filter out irrelevant details while maintaining precision. However, short sentences with numerous implied relationships (*e.g.*, hand usages, repeated actions) provide limited explicit cues, requiring the model to infer missing links. While LLMs manage noisy sentences reasonably well, their performance weakens when extensive inference is needed with minimal textual guidance.

C.4 Hallucination in Scene Graph Generation

It is widely recognized that LLMs exhibit a hallucination issue, and in this study, we investigated whether the same problem arises in both our scene graph generation and understanding tasks. Each

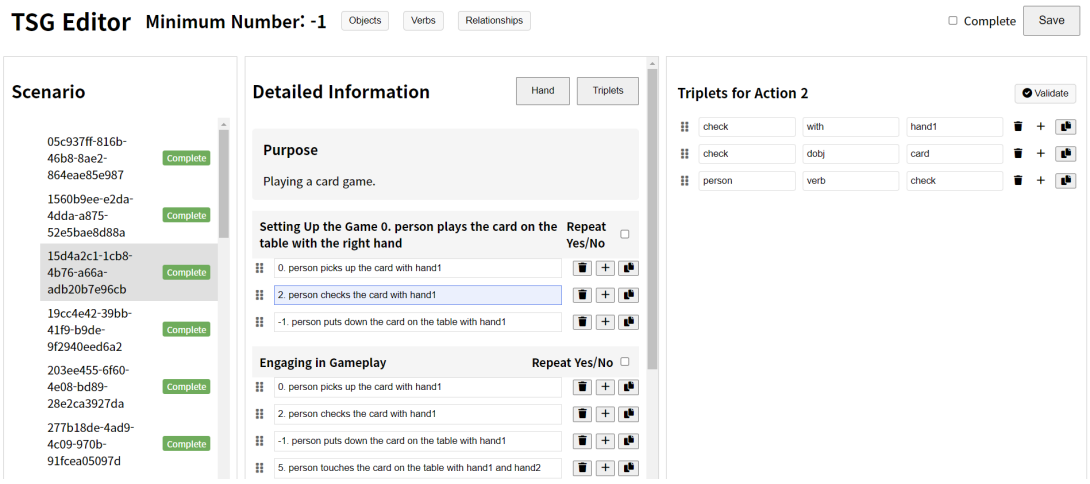

Figure 6: Example interface of the TSG Editor, designed to facilitate the creation of scene graph datasets from multiple scenarios. The left panel shows the list of scenarios with their completion statuses, while the right panel allows for specifying details such as action goals, hands used, objects, and relationship triplets.

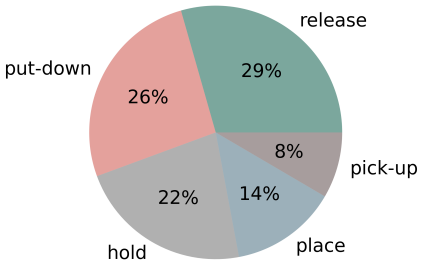

Figure 7: The distribution of the top 5 actions where segmentation difficulties occur most frequently in scene graph generation.

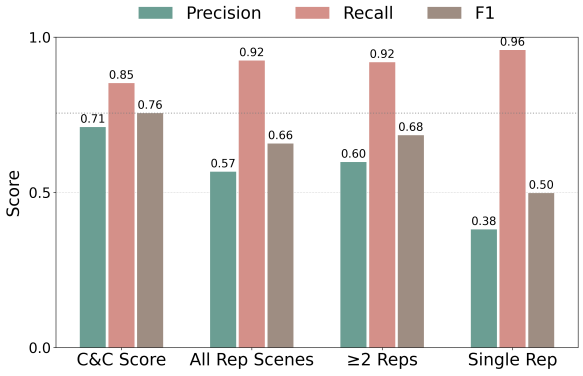

Figure 8: Comparison of aggregated ChatGPT+Claude (C&C) results under different repetition conditions: (1) All repeated scenes, (2) Multiple repetitions ($\geq$ 2 Reps), and (3) Single-repetition (Single Rep) scenarios.

task maintains its own predefined collection L, and the total number of predefined elements used in both the understanding and generation tasks is 26,725. If a word not included in these collections appears, we consider it a hallucination, and based on this criterion, we measured the hallucination occurrence rate in scene graph-related tasks.

Model	Total	Desc.	New.
Claude-3.5-Sonnet	17	14	3
GPT-4o	215	156	59
Mistral-7B	616	235	381
Qwen-2.5-7B	395	192	203

Table 6: Hallucination counts for each model, showing the total number, those semantically similar to the words in the description, and those entirely unrelated. "Desc." indicates "Description Elements." and "New." indicates "New Elements."

Our experimental results revealed that hallucinations do occur; however, when considering the total number of predefined elements, the absolute count of such occurrences was relatively low. To analyze these hallucinations more precisely, we categorized them into two groups: (1) those caused by generating words semantically similar to elements mentioned in the description, and (2) those arising from producing entirely new words.

As shown in Table 1, while larger models mostly exhibit hallucinations by producing words that are semantically similar to those in the description, smaller models not only demonstrate a higher overall hallucination rate but also tend to generate entirely new words, highlighting their increased susceptibility to hallucination.

We hypothesize that increasing the temperature in LLMs may lead to a higher incidence of hallu-

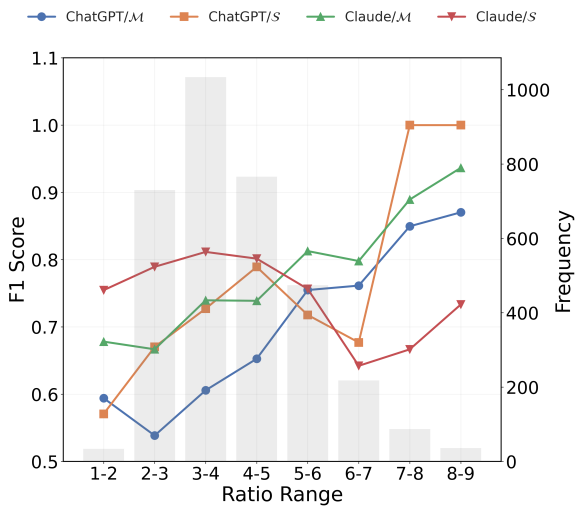

Figure 9: The results show F_1 performance between multiple action scene graph(M) and single action scene graph(S) approaches across description-to-edge ratio ranges. The histogram represents sample frequencies in each ratio range.

cinations, as the model gains more freedom in its responses. Our prior experiments employed a temperature setting of 0.1. For the new experiments, we set the temperature to 1.0. Under otherwise identical conditions, we observed a slight increase in hallucinations.

Model	Temp.	Total	Desc.	New.
Claude-3.5-Sonnet	0.1	17	14	3
	1.0	21	13	8
GPT-4o	0.1	215	156	59
	1.0	233	156	77

Table 7: Comparison of experimental results under two temperature settings (0.1 vs. 1.0), showing changes with all other conditions held constant."Temp." indicates "Temperature,", "Desc." indicates "Description Elements." and "New." indicates "New Elements."

In addition to observing an increased rate of hallucinations in both nodes and edges, we investigated whether performance deteriorates in the multiple-action scene graph generation task as well. Our experimental results show that GPT-4o's performance dropped from 43.99 to 39.72, while Sonnet's performance fell from 58.8 to 56.8, confirming a noticeable decline. In summary, these findings suggest that the increased temperature of LLMs can occasionally undermine performance when generating structured representations.

C.5 Case Study for Descriptions with Repetitive Action

The failure cases are shown in Table 8.

Case 1

[Context]
... The wood was then positioned on the cardboard, ready for the next steps. The process began by picking up the stick. The stick was then put into the paint can and used to stir the paint thoroughly.

[Description]
This stirring step was repeated two more times to ensure the paint was well-mixed.

[Nodes in the Reference Graph]
V_1: person, **stir**, stick, ...
V_2: person, **stir**, stick, ...

[Nodes in the Generated Graph]
V_1: person, **stir**, stick, ...
V_2: person, **paint**, stick, ...

Case 2

[Context]
The painting process began by preparing the paintbrush, dipping it into the paint, and applying it to the railing.

[Description]
The railing was painted with the paintbrush, ensuring even coverage. This step was repeated three times to achieve a consistent finish.

[Nodes in the Reference Graph]
V_1: person, **paint**, paintbrush, ...
V_2: person, **paint**, paintbrush, ...
V_3: person, **paint**, paintbrush, ...

[Nodes in the Generated Graph]
V_1: person, **pick-up**, paintbrush, ...
V_2: person, **dip**, paintbrush, ...
V_3: person, **paint**, paintbrush, ...

Table 8: Failure cases in generating scene graphs for sequences of repetitive actions. V_j represents the set of nodes in the j-th graph corresponding to the description.

C.6 Details for Performance

This subsection provides supplemental information for two aspects. First, we evaluate the model's ability to detect edges and nodes when explicit action is omitted (Edge and Node w/o Action Performance). Second, we present Action Segmentation Results using the Longest Common Subsequence (LCS) approach, reported in terms of Precision, Recall, and Action F1. The complete findings for these metrics can be found in Table 10 and Table 9.

C.7 Details for Data Distribution

The distribution of scene graph objects, relationships and verbs in the dataset is visually summarized in Figure 11, 10 and 12.

Model	Precision	Recall	F1 Score
Human	95.38	93.84	94.60
GPT-4o	67.45	85.24	73.15
GPT-4o-mini	56.43	73.35	61.22
Claude-3.5-Sonnet	74.58	85.12	77.85
Claude-3.5-Haiku	60.85	74.72	61.62
LLaMA-3.3-70B	63.33	76.45	66.37
Qwen-2.5-72B	68.02	75.45	68.67
DeepSeek-V3	57.11	78.28	62.93
Mixtral-8x22B	52.27	76.36	58.67
Mistral-large	60.37	73.76	63.48
Qwen-2.5-7B	66.56	72.37	67.57
Mistral-7B	19.47	64.04	27.42

Table 9: Action segmentation results using LCS for Precision, Recall, and Action F1.

D Prompts

D.1 Zero-shot Prompts

This section provides full prompt. Each prompt is formulated to clearly specify input formats, rules for processing these inputs, and the exact manner in which the output should be generated.

- **SGDS Task**: Figure 13
- **SGQA Task**: Figure 14
- **SA-SGG Task**: Figure 15
- **MA-SGG Task**: Figure 16

D.2 Chain of Thought Prompts

In addition to the base prompt, we incorporated a minimal form of Chain-of-Thought (CoT) prompting to guide the reasoning process.

- **SGDS Task**: Figure 17
- **SGQA Task**: Figure 18
- **SA-SGG Task**: Figure 19
- **MA-SGG Task**: Figure 20

D.3 Few-shot Prompts

We employed a few-shot prompt with ten examples (10-shot), allowing the model to observe multiple instances of input-output pairs.

- **SGDS Task**: Figure 21
- **SGQA Task**: Figure 22
- **SA-SGG Task**: Figure 23
- **MA-SGG Task**: Figure 24

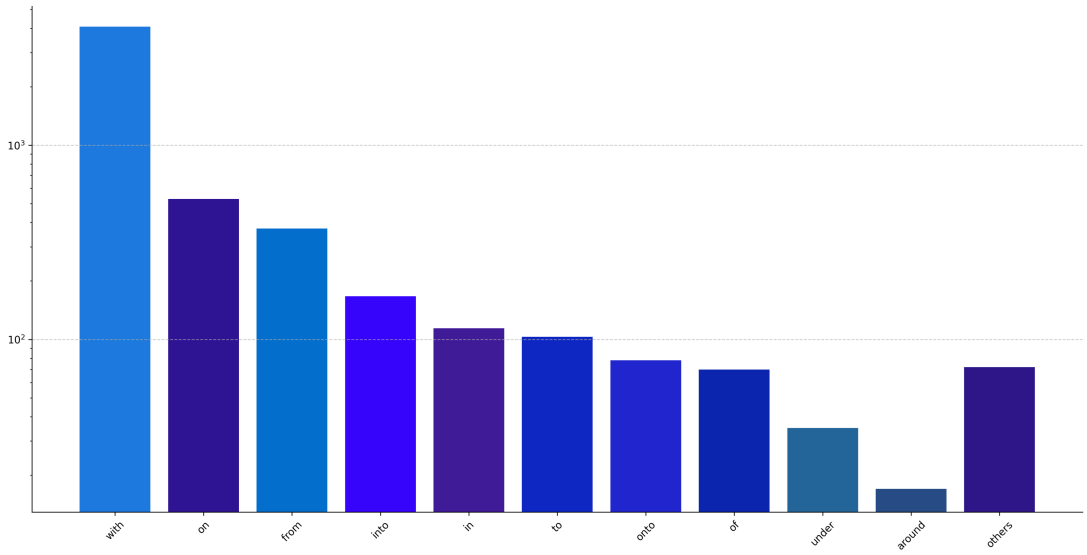

Figure 10: The distribution of relationships in the scene graph dataset, with “with” appearing most frequently, followed by “on” and “from.”

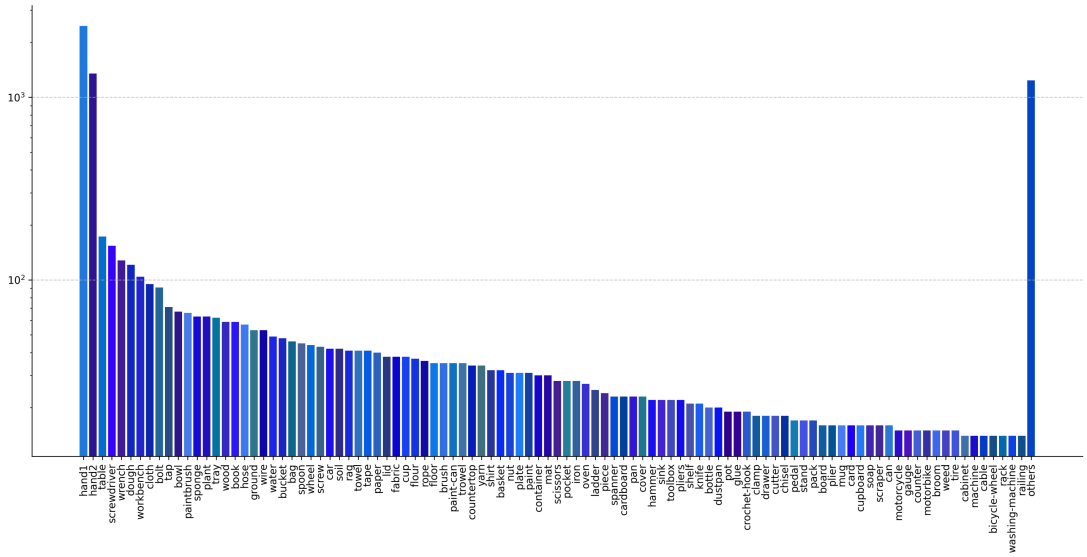

Figure 11: The frequency distribution of scene graph objects on a logarithmic scale, with “hand1” and “others” standing out as the most frequently occurring categories

Models	Edge						Node					
	Single			Multiple			Single			Multiple		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
GPT-4o	86.88	82.90	84.42	81.31	79.25	79.80	83.15	80.76	81.70	76.44	74.92	75.29
GPT-4o-mini	53.44	52.72	52.88	48.11	51.85	49.53	78.53	77.28	77.53	69.14	71.89	69.87
Claude-3.5-Sonnet	93.46	93.92	93.64	87.56	92.59	89.65	87.65	88.71	87.87	85.15	86.86	85.59
Claude-3.5-Haiku	78.97	76.52	77.47	73.04	75.06	73.72	75.96	72.23	73.53	67.78	70.54	68.61
LLaMA-3.3-70B	64.05	73.68	67.84	63.62	73.28	67.42	81.47	81.70	81.27	75.65	80.61	77.11
Qwen-2.5-72B	82.98	81.92	82.32	84.53	82.19	79.74	80.39	81.56	71.18	74.74	74.75	70.68
DeepSeek-V3	81.73	81.64	81.60	85.95	77.26	82.28	82.63	83.77	74.47	77.77	77.77	72.93
Mixtral-8x22B	62.26	64.22	62.87	79.67	64.36	72.43	68.11	75.13	54.43	59.38	59.38	58.67
mistral-large-2411	84.85	83.67	84.07	83.70	83.90	84.57	74.91	83.74	72.91	73.16	73.16	69.87
Qwen-2.5-7b	19.59	18.17	18.58	14.37	16.52	14.99	77.80	72.74	74.29	56.56	69.33	60.74
Mistral-7b	37.21	35.97	36.09	24.69	30.02	26.30	52.94	50.85	50.10	37.10	49.34	40.12

Table 10: Edge and Node w/o Action Performance (%)

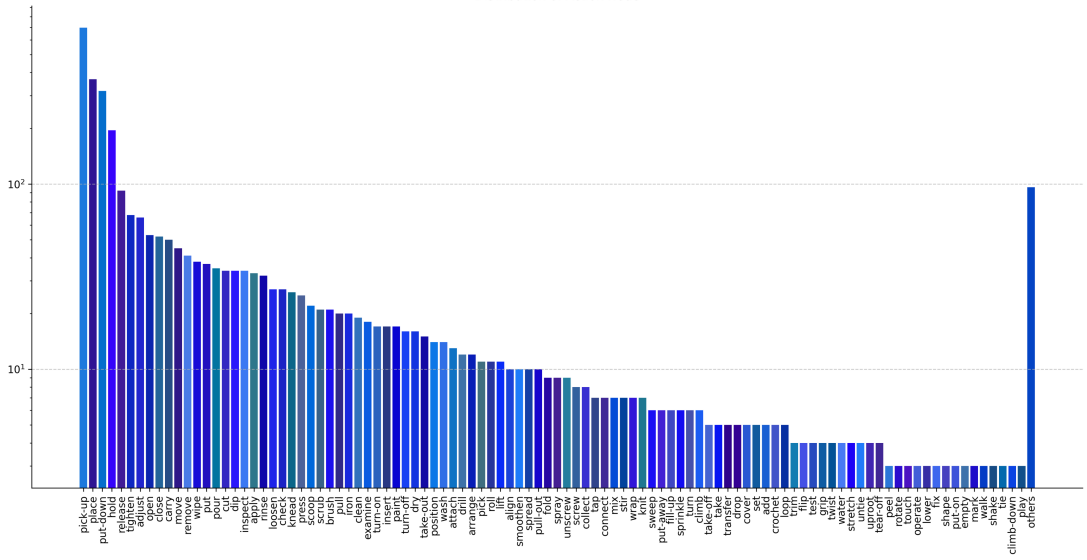

Figure 12: The frequency distribution of verbs in the scene graph, with “others” as the dominant category, followed by “pick up,” “place,” and “put down” among the most frequent specific actions.

Prompts

You are an AI that analyzes a Scene Graph based on the context and select the best text description of it among the given candidates.

1. Input:
- Context: A list of scene graphs representing the preceding context.
- Each graph is composed of a set of triplets ‘[node1, edge, node2]‘. ‘node1‘ and ‘node2‘ are one of person, action, object and hand. ‘edge‘ represents the relationship between them (e.g., ‘verb‘, ‘dobj‘, ‘from‘, ‘with‘).
- Target Scene Graph: A set of triplets that should be described into text correctly.
- Description Candidates: Candidates of sentence descriptions of the Target Scene Graph based on the Context.

2. Task:
- Determine which description best matches the Target Scene Graph.

3. Output:
- Be sure to choose only one letter of the matching description.
- Do not output any additional text or explanation. Only the letter in [ ] (e.g., [A]).

Key rules of edges in a triplet:
- ‘verb‘ describes the action performed by ‘person‘.
- ‘dobj‘ links the action to its direct object (‘node2‘).
- Other edges like ‘from‘ and ‘with‘ describe spatial relationships between nodes.

Input:
- Context: {context}
- Target Scene Graph: {triplet}
- Description Candidates:
{sentences}

Figure 13: The zero-shot prompt for scene graph description selection tasks.

Prompts

You are a highly advanced language model specialized in answering questions based on a given scene graph and question. Your task is to analyze the scene graph and provide the correct answer in a single word. Your output must strictly follow the format [answer], and nothing else should be printed. Ensure that your answer is concise, accurate, and matches the format exactly.

Scene Graph: {scene_graph}
Question: {question}

Figure 14: The zero-shot prompt for scene graph question answering tasks.

Prompts

You are an AI model tasked with generating a scene graph based on a given sentence, adhering to specific rules for the graph, nodes, and edges, while considering the provided context, available nodes, and available edges.

Rules for Scene Graph Representation:
1. A graph is composed of one or more triplets of nodes and edges.
2. A triplet starts with a node and another node is connected by an edge. (Format: node -> edge -> node)
3. Each triplet is split with a new line.
4. There must be a triplet that starts with a person node.
5. All nodes and edges must be one of "Available nodes" or "Available edges" provided.

Rules for Node:
1. A node can be person, any action, any object, or any hand.
2. A node may appear explicitly or be hidden implicitly in the given sentence. Consider the context to identify the node.
3. Map synonyms or semantically similar words to nodes in the "Available nodes" list.
4. Use default tools or body parts for actions that imply them (e.g., hands for grasping).
5. Include "person" as the starting node in the graph.

Special Rules for Hand Node:
1. If both hands are empty and a node is grasped, represent it as "hand1."
2. If one hand holds a node and another node is grasped, represent it as "hand2."
3. If all hands release their objects, reset the next grasping hand to "hand1."
4. Ensure "hand1" and "hand2" are used contextually to avoid overlap or ambiguity.
5. If the sentence implies using both hands (e.g., lifting a large object), represent both hands explicitly (e.g., hand1, hand2).

Rules for Edge:
1. An edge can be verb, dobj, or any preposition.
2. Map synonyms or semantically similar words to edges in the "Available edges" list.
1. verb: can only connect person and an action node. (e.g., person -> verb -> add)
2. dobj: connects an action and an object node, only when it is the direct object of the action (e.g., add -> dobj -> flour)
3. preposition: connects one of the four types of node pairs: action & object / action & hand / object & object / hand & object (e.g., take -> from -> table)

Output Format: The output must consist of triplets (one per line) in the format below. node -> edge -> node node -> edge -> node ...

Use only the "Available nodes" and "Available edges" provided. No additional text, explanations, or formatting should be included.

Inputs:
Context: {context}
Target sentence: {target_sentence}
Available nodes: {available_nodes}
Available edges: {available_edges}

Figure 15: The zero-shot prompt for single action scene graph generation tasks.

Prompts

You are an AI model tasked with generating scene graphs based on a given sentence. Your goal is to create exactly the specified number of scene graphs by extracting meaningful relationships between entities, actions, and objects while ensuring that the scene graphs represent actions that would visually appear in a scene.

Rules for Generating Multiple Scene Graphs:
1. Generate precisely {num_scene_graphs} scene graphs—no more, no less.
2. Each scene graph must depict an action that would be explicitly visible in a scene.
3. If the sentence contains multiple implicit actions, distribute them among the scene graphs while ensuring the total count matches {num_scene_graphs}.
4. If there are fewer visible actions than {num_scene_graphs}, **additional relevant actions may be inferred** to reach the required count.
5. However, use only the "Available Nodes" and "Available Edges" provided. **If a necessary node is missing, use the closest semantically matching node from the available list**.
6. Ensure each graph maintains logical coherence while including essential contextual elements.

Rules for A Scene Graph Representation:
1. A graph is composed of one or more triplets of nodes and edges.
2. A triplet starts with a node and another node is connected by an edge. (Format: node -> edge -> node)
3. Each triplet is split with a new line.
4. There must be exactly one triplet that starts with a person node in a graph.
5. All nodes and edges must be one of "Available nodes" or "Available edges" provided.

Rules for Node:
1. A node can be person, any action, any object, or any hand.
2. A node may appear explicitly or be hidden implicitly in the given sentence. Consider the context to identify the node from the "Available nodes" list, but do not create a new one.
3. Map synonyms or semantically similar words to nodes in the "Available nodes" list.
4. Use default tools or body parts for actions that imply them (e.g., hands for grasping).
5. Treat each action as a node.
6. Include "person" as the starting node in the graph.

Special Rules for Hand Node:
1. If both hands are empty and a node is grasped, represent it as "hand1."
2. If one hand holds a node and another node is grasped, represent it as "hand2."
3. If all hands release their objects, reset the next grasping hand to "hand1."
4. Ensure "hand1" and "hand2" are used contextually to avoid overlap or ambiguity.
5. If the sentence implies using both hands (e.g., lifting a large object), represent both hands explicitly (e.g., hand1, hand2).

Rules for Edge:
1. An edge can be verb, dobj, or any preposition.
2. Use only the edges listed under "Available edges."
3. Here are the explanations for each edge.
- verb: can only connect person and an action node. (e.g., person -> verb -> add) - dobj: connects an action and an object node, only when it is the direct object of the action (e.g., add -> dobj -> flour) - preposition: connects one of the four types of node pairs: action & object / action & hand / object & object / hand & object (e.g., take -> from -> table)

Output Format:
The output must consist of exactly {num_scene_graphs} scene graphs, each separated with a blank line. For a graph, output one triplet per line. Follow the format below (an example of three scene graphs of multiple triplets):
node -> edge -> node
node -> edge -> node
...

node -> edge -> node
node -> edge -> node
...

node -> edge -> node
node -> edge -> node
...

Use only the "Available Nodes" and "Available Edges" provided. No additional text, explanations, or formatting should be included.

Inputs:
- Context: {context}
- Target sentence: {target_sentence}
- Available nodes: {available_nodes}
- Available edges: {available_edges}
- Number of Scene Graphs: {num_scene_graphs}

Figure 16: The zero-shot prompt for multiple action scene graph generation tasks.

Prompts

You are an AI that analyzes a Scene Graph based on the context and select the best text description of it among the given candidates.

1. Input:
- Context: A list of scene graphs representing the preceding context.
- Each graph is composed of a set of triplets [node1, edge, node2]. ‘node1‘ and ‘node2‘ are one of person, action, object and hand. ‘edge‘ represents the relationship between them (e.g., ‘verb‘, ‘dobj‘, ‘from‘, ‘with‘).
- Target Scene Graph: A set of triplets that should be described into text correctly.
- Description Candidates: Candidates of sentence descriptions of the Target Scene Graph based on the Context.

2. Task:
- Think step-by-step and determine which description best matches the Target Scene Graph.

3. Output:
- Output your rationale under "Think:"
- Then, output your final answer under "Final Answer:"
- For the final answer, be sure to choose only one letter of the matching description and write it in the format of (e.g., [X]), where "X" represents a single alphabet letter.

Key rules of edges in a triplet:
- ‘verb‘ describes the action performed by ‘person‘.
- ‘dobj‘ links the action to its direct object (‘node2‘).
- Other edges like ‘from‘ and ‘with‘ describe spatial relationships between nodes.

Input:
- Context: {context}
- Target Scene Graph: {triplet}
- Description Candidates:
{sentences}

Now, think step-by-step and output the final answer.
Think:

Figure 17: CoT prompt for scene graph description selection tasks.

Prompts

You are a highly advanced language model specialized in answering questions based on a given scene graph and a question. Your task is to analyze the scene graphs and provide the correct answer in a single word. Think step-by-step under "Think:", and generate the final answer under "Final Answer:". Ensure that your final answer is a single word in the triplet, and do not generate additional explanations after it.

Scene Graph: {scene_graph}
Question: {question}

Think:

Figure 18: CoT prompt for scene graph question answering tasks.

Prompts

You are an AI model tasked with generating a scene graph based on a given sentence, adhering to specific rules for the graph, nodes, and edges, while considering the provided context, available nodes, and available edges. Think step-by-step and generate the graph in triplets. (Stick only to the target sentence and avoid over-predicting the next scene.)

Rules for Scene Graph Representation:
1. A graph is composed of one or more triplets of nodes and edges.
2. A triplet starts with a node and another node is connected by an edge. (Format: node -> edge -> node)
3. Each triplet is split with a new line.
4. There must be a triplet that starts with a person node.
5. All nodes and edges must be one of "Available nodes" or "Available edges" provided.

Rules for Node:
1. A node can be person, any action, any object, or any hand.
2. A node may appear explicitly or be hidden implicitly in the given sentence. Consider the context to identify the node.
3. Map synonyms or semantically similar words to nodes in the "Available nodes" list.
4. Use default tools or body parts for actions that imply them (e.g., hands for grasping).
5. Include "person" as the starting node in the graph.

Special Rules for Hand Node:
1. If both hands are empty and a node is grasped, represent it as "hand1."
2. If one hand holds a node and another node is grasped, represent it as "hand2."
3. If all hands release their objects, reset the next grasping hand to "hand1."
4. Ensure "hand1" and "hand2" are used contextually to avoid overlap or ambiguity.
5. If the sentence implies using both hands (e.g., lifting a large object), represent both hands explicitly (e.g., hand1, hand2).

Rules for Edge:
1. An edge can be verb, dobj, or any preposition.
2. Use only the edges listed under "Available edges."
3. Here are the explanations for each edge.
- verb: can only connect person and an action node. (e.g., person -> verb -> add)
- dobj: connects an action and an object node, only when it is the direct object of the action (e.g., add -> dobj -> flour)
- preposition: connects one of the four types of node pairs: action & object / action & hand / object & object / hand & object (e.g., take -> from -> table)

Output Format:
The output must consist of your rationale and triplets (one per line) in the format below.
Think:
(Write your rationale here, but do not predict the next scene)

Scene Graph:
node -> edge -> node
node -> edge -> node
...

Use only the "Available nodes" and "Available edges" provided, and follow the format correctly. After generating the scene graph, no additional text or explanations should be included.

Inputs:
Context: {context}
Target sentence: {target_sentence}
Available nodes: {available_nodes}
Available edges: {available_edges}

Think:

Figure 19: CoT prompt for single action scene graph generation tasks.

Prompts

You are an AI model tasked with generating scene graphs based on a given sentence. Your goal is to create exactly the specified number of scene graphs by extracting meaningful relationships between entities, actions, and objects while ensuring that the scene graphs represent actions that would visually appear in a scene. Read the rules below and think step-by-step to generate correct scene graphs that represent the target sentence.

Rules for Generating Multiple Scene Graphs:
1. Generate precisely {num_scene_graphs} scene graphs—no more, no less.
2. Each scene graph must depict an action that would be explicitly visible in a scene.
3. If the sentence contains multiple implicit actions, distribute them among the scene graphs while ensuring the total count matches {num_scene_graphs}.
4. If there are fewer visible actions than {num_scene_graphs}, additional relevant actions may be inferred to reach the required count.
5. However, use only the "Available Nodes" and "Available Edges" provided. If a necessary node is missing, use the closest semantically matching node from the available list.
6. Ensure each graph maintains logical coherence while including essential contextual elements.

Rules for A Scene Graph Representation:
1. A graph is composed of one or more triplets of nodes and edges.
2. A triplet starts with a node and another node is connected by an edge. (Format: node -> edge -> node)
3. Each triplet is split with a new line.
4. There must be exactly one triplet that starts with a person node in a graph.
5. All nodes and edges must be one of "Available nodes" or "Available edges" provided.

Rules for Node:
1. A node can be person, any action, any object, or any hand.
2. A node may appear explicitly or be hidden implicitly in the given sentence. Consider the context to identify the node from the "Available nodes" list, but do not create a new one.
3. Map synonyms or semantically similar words to nodes in the "Available nodes" list.
4. Use default tools or body parts for actions that imply them (e.g., hands for grasping).
5. Treat each action as a node.
6. Include "person" as the starting node in the graph.

Special Rules for Hand Node:
1. If both hands are empty and a node is grasped, represent it as "hand1."
2. If one hand holds a node and another node is grasped, represent it as "hand2."
3. If all hands release their objects, reset the next grasping hand to "hand1."
4. Ensure "hand1" and "hand2" are used contextually to avoid overlap or ambiguity.
5. If the sentence implies using both hands (e.g., lifting a large object), represent both hands explicitly (e.g., hand1, hand2).

Rules for Edge:
1. An edge can be verb, dobj, or any preposition.
2. Use only the edges listed under "Available edges."
3. Here are the explanations for each edge.
- verb: can only connect person and an action node. (e.g., person -> verb -> add)
- dobj: connects an action and an object node, only when it is the direct object of the action (e.g., add -> dobj -> flour)
- preposition: connects one of the four types of node pairs: action & object / action & hand / object & object / hand & object (e.g., take -> from -> table)

Output Format:
The output must consist of your rationale and exactly {num_scene_graphs} scene graphs, each separated with a blank line. For a graph, output one triplet per line. Follow the format below (an example of three scene graphs of multiple triplets):
Think:
(Write your rationale here)

node -> edge -> node
node -> edge -> node
...

node -> edge -> node
node -> edge -> node
...

node -> edge -> node
node -> edge -> node
...

Use only the "Available Nodes" and "Available Edges" provided. After generating the scene graphs, no additional text or explanations should be included.

Inputs:
- Context: {context}
- Target sentence: {target_sentence}
- Available nodes: {available_nodes}
- Available edges: {available_edges}
- Number of Scene Graphs: {num_scene_graphs}
Think:

Figure 20: CoT prompt for multiple action scene graph generation tasks.

Prompts

You are an AI that analyzes a Scene Graph based on the context and select the best text description of it among the given candidates.

1. Input:
- Context: A list of scene graphs representing the preceding context.
- Each graph is composed of a set of triplets [node1, edge, node2]. ‘node1‘ and ‘node2‘ are one of person, action, object and hand. ‘edge‘ represents the relationship between them (e.g., ‘verb‘, ‘dobj‘, ‘from‘, ‘with‘).
- Target Scene Graph: A set of triplets that should be described into text correctly.
- Description Candidates: Candidates of sentence descriptions of the Target Scene Graph based on the Context.

2. Task:
- Determine which description best matches the Target Scene Graph.

3. Output:
- Be sure to choose only one letter of the matching description.
- Do not output any additional text or explanation. Only the letter in [ ] (e.g., [A]).

Key rules of edges in a triplet:
- ‘verb‘ describes the action performed by ‘person‘.
- ‘dobj‘ links the action to its direct object (‘node2‘).
- Other edges like ‘from‘ and ‘with‘ describe spatial relationships between nodes.

Example Input 1:
- Context: [[[’arrange’, ’with’, ’hand1’], [’arrange’, ’with’, ’hand2’], [’arrange’, ’dobj’, ’book’], [’person’, ’verb’, ’arrange’]], [[’put’, ’dobj’, ’book’], [’put’, ’on’, ’floor’], [’put’, ’with’, ’hand1’], [’put’, ’with’, ’hand2’], [’person’, ’verb’, ’put’]], [[’put’, ’dobj’, ’book’], [’put’, ’on’, ’bookshelf’], [’put’, ’with’, ’hand1’], [’put’, ’with’, ’hand2’], [’person’, ’verb’, ’put’]]]
- Target Scene Graph: [[’align’, ’with’, ’hand1’], [’align’, ’with’, ’hand2’], [’align’, ’dobj’, ’book’], [’align’, ’on’, ’bookshelf’], [’person’, ’verb’, ’align’]]
- Description Candidates:
A: Finally, the flask was set down.
B: It was then aligned neatly on the shelf using both hands.
C: After achieving the desired consistency, the stick was removed from the paint can.
D: The cord was then sliced using the cord cutter.
E: Once the massaging was complete, the batter was positioned back on the counter with both hands.

Example Output 1:
[B]

...

Example Input 10:
- Context: [[[’pick-up’, ’with’, ’hand1’], [’pick-up’, ’with’, ’hand2’], [’pick-up’, ’dobj’, ’rope’], [’person’, ’verb’, ’pick-up’]], [[’tie’, ’with’, ’hand1’], [’tie’, ’with’, ’hand2’], [’tie’, ’dobj’, ’rope’], [’tie’, ’around’, ’plant’], [’person’, ’verb’, ’tie’]]]
- Target Scene Graph: [[’pull’, ’with’, ’hand1’], [’pull’, ’with’, ’hand2’], [’pull’, ’dobj’, ’rope’], [’person’, ’verb’, ’pull’], [’pull’, ’to’, ’tighten’]]
- Description Candidates:
A: The rope was then pushed loose with both hands to ensure a relaxed hold.
B: The rope was then pulled slack with one hand to ensure a loose grip.
C: The rope was then pulled tight with both hands to ensure a firm grip.
D: The rope was then dropped with both hands to ensure it stayed untightened.
E: The rope was then pulled apart with no hands to ensure it remained loose.

Example Output 10:
[C]

Input:
- Context: {context}
- Target Scene Graph: {triplet}
- Description Candidates:
{sentences}

Output:

Figure 21: Few-shot prompt for scene graph description selection tasks.

Prompts

You are a highly advanced language model specialized in answering questions based on a given scene graph and question. Your task is to analyze the scene graph and provide the correct answer in a single word. Your output must strictly follow the format [answer], and nothing else should be printed. Ensure that your answer is concise, accurate, and matches the format exactly.

Example Input 1:
Scene Graph: [[["person", "verb", "pick-up"], ["pick-up", "dobj", "screw"], ["pick-up", "from", "bowl"], ["pick-up", "with", "hand2"]], [["person", "verb", "position"], ["position", "dobj", "screw"], ["position", "on", "furniture-piece"], ["position", "with", "hand2"]]]
Question: What object was positioned immediately after being picked up from the bowl?

Example Output 1:
[screw]

...

Example Input 10:
Scene Graph: [[["person", "verb", "pick-up"], ["pick-up", "with", "hand1"], ["pick-up", "dobj", "mop-stick"]], [["person", "verb", "sweep"], ["sweep", "with", "hand1"], ["sweep", "with", "hand2"], ["sweep", "with", "mop-stick"], ["sweep", "dobj", "floor"], ["sweep", "in", "car"]], [["person", "verb", "close"], ["close", "with", "hand1"], ["close", "dobj", "door"]], [["person", "verb", "place"], ["place", "with", "hand2"], ["place", "dobj", "mop-stick"], ["place", "on", "floor"]], [["person", "verb", "open"], ["open", "dobj", "door"], ["open", "with", "hand2"]], [["person", "verb", "put"], ["put", "dobj", "cloth"], ["put", "inside", "car"], ["put", "with", "hand1"]], [["person", "verb", "move"], ["move", "to", "cabinet"]], [["person", "verb", "open"], ["open", "dobj", "cabinet"], ["open", "with", "hand1"]], [["person", "verb", "pick-up"], ["pick-up", "with", "hand1"], ["pick-up", "dobj", "cloth"]], [["person", "verb", "move"], ["move", "to", "wall"], ["hand1", "in", "cloth"], ["move", "with", "hand1"]], [["person", "verb", "pick"], ["pick", "with", "hand2"], ["pick", "from", "wall"], ["pick", "dobj", "mop-stick"]]]
Question: What object was picked up before sweeping the floor?

Example Output 10:
[mop-stick]

Input:
Scene Graph: {scene_graph}
Question: {question}

Output:

Figure 22: Few-shot prompt for scene graph qusetion answering tasks.

Prompts

You are an AI that analyzes a Scene Graph based on the context and select the best text description of it among the given candidates.

1. Input:
- Context: A list of scene graphs representing the preceding context.
- Each graph is composed of a set of triplets [node1, edge, node2]. ‘node1‘ and ‘node2‘ are one of person, action, object and hand. ‘edge‘ represents the relationship between them (e.g., ‘verb‘, ‘dobj‘, ‘from‘, ‘with‘).
- Target Scene Graph: A set of triplets that should be described into text correctly.
- Description Candidates: Candidates of sentence descriptions of the Target Scene Graph based on the Context.

2. Task:
- Determine which description best matches the Target Scene Graph.

3. Output:
- Be sure to choose only one letter of the matching description.
- Do not output any additional text or explanation. Only the letter in [ ] (e.g., [A]).

Key rules of edges in a triplet:
- ‘verb‘ describes the action performed by ‘person‘.
- ‘dobj‘ links the action to its direct object (‘node2‘).
- Other edges like ‘from‘ and ‘with‘ describe spatial relationships between nodes.

Example Input 1:
- Context: [[[’arrange’, ’with’, ’hand1’], [’arrange’, ’with’, ’hand2’], [’arrange’, ’dobj’, ’book’], [’person’, ’verb’, ’arrange’]], [[’put’, ’dobj’, ’book’], [’put’, ’on’, ’floor’], [’put’, ’with’, ’hand1’], [’put’, ’with’, ’hand2’], [’person’, ’verb’, ’put’]], [[’put’, ’dobj’, ’book’], [’put’, ’on’, ’bookshelf’], [’put’, ’with’, ’hand1’], [’put’, ’with’, ’hand2’], [’person’, ’verb’, ’put’]]]
- Target Scene Graph: [[’align’, ’with’, ’hand1’], [’align’, ’with’, ’hand2’], [’align’, ’dobj’, ’book’], [’align’, ’on’, ’bookshelf’], [’person’, ’verb’, ’align’]]
- Description Candidates:
A: Finally, the flask was set down.
B: It was then aligned neatly on the shelf using both hands.
C: After achieving the desired consistency, the stick was removed from the paint can.
D: The cord was then sliced using the cord cutter.
E: Once the massaging was complete, the batter was positioned back on the counter with both hands.

Example Output 1:
[B]

...

Example Input 10:
- Context: [[[’pick-up’, ’with’, ’hand1’], [’pick-up’, ’with’, ’hand2’], [’pick-up’, ’dobj’, ’rope’], [’person’, ’verb’, ’pick-up’]], [[’tie’, ’with’, ’hand1’], [’tie’, ’with’, ’hand2’], [’tie’, ’dobj’, ’rope’], [’tie’, ’around’, ’plant’], [’person’, ’verb’, ’tie’]]]
- Target Scene Graph: [[’pull’, ’with’, ’hand1’], [’pull’, ’with’, ’hand2’], [’pull’, ’dobj’, ’rope’], [’person’, ’verb’, ’pull’], [’pull’, ’to’, ’tighten’]]
- Description Candidates:
A: The rope was then pushed loose with both hands to ensure a relaxed hold.
B: The rope was then pulled slack with one hand to ensure a loose grip.
C: The rope was then pulled tight with both hands to ensure a firm grip.
D: The rope was then dropped with both hands to ensure it stayed untightened.
E: The rope was then pulled apart with no hands to ensure it remained loose.

Example Output 10:
[C]

Input:
- Context: {context}
- Target Scene Graph: {triplet}
- Description Candidates:
{sentences}

Output:

Figure 23: Few-shot prompt for single action scene graph generation tasks.

Prompts

You are an AI model tasked with generating scene graphs based on a given sentence. Your goal is to create exactly the specified number of scene graphs by extracting meaningful relationships between entities, actions, and objects while ensuring that the scene graphs represent actions that would visually appear in a scene.

Rules for Generating Multiple Scene Graphs:
1. Generate precisely {num_scene_graphs} scene graphs—no more, no less.
2. Each scene graph must depict an action that would be explicitly visible in a scene.
3. If the sentence contains multiple implicit actions, distribute them among the scene graphs while ensuring the total count matches {num_scene_graphs}.
4. If there are fewer visible actions than {num_scene_graphs}, additional relevant actions may be inferred to reach the required count.
5. However, use only the "Available Nodes" and "Available Edges" provided. If a necessary node is missing, use the closest semantically matching node from the available list.
6. Ensure each graph maintains logical coherence while including essential contextual elements.

Rules for A Scene Graph Representation:
1. A graph is composed of one or more triplets of nodes and edges.
2. A triplet starts with a node and another node is connected by an edge. (Format: node -> edge -> node)
3. Each triplet is split with a new line.
4. There must be exactly one triplet that starts with a person node in a graph.
5. All nodes and edges must be one of "Available nodes" or "Available edges" provided.

Rules for Node:
1. A node can be person, any action, any object, or any hand.
2. A node may appear explicitly or be hidden implicitly in the given sentence. Consider the context to identify the node from the "Available nodes" list, but do not create a new one.
3. Map synonyms or semantically similar words to nodes in the "Available nodes" list.
4. Use default tools or body parts for actions that imply them (e.g., hands for grasping).
5. Treat each action as a node.
6. Include "person" as the starting node in the graph.

Special Rules for Hand Node:
1. If both hands are empty and a node is grasped, represent it as "hand1."
2. If one hand holds a node and another node is grasped, represent it as "hand2."
3. If all hands release their objects, reset the next grasping hand to "hand1."
4. Ensure "hand1" and "hand2" are used contextually to avoid overlap or ambiguity.
5. If the sentence implies using both hands (e.g., lifting a large object), represent both hands explicitly (e.g., hand1, hand2).

Rules for Edge:
1. An edge can be verb, dobj, or any preposition.
2. Use only the edges listed under "Available edges."
3. Here are the explanations for each edge.
- verb: can only connect person and an action node. (e.g., person -> verb -> add)
- dobj: connects an action and an object node, only when it is the direct object of the action (e.g., add -> dobj -> flour)
- preposition: connects one of the four types of node pairs: action & object / action & hand / object & object / hand & object (e.g., take -> from -> table)

Output Format:
The output must consist of exactly {num_scene_graphs} scene graphs, each separated with a blank line. For a graph, output one triplet per line. Follow the format below (an example of three scene graphs of multiple triplets):
node -> edge -> node
node -> edge -> node
...

node -> edge -> node
node -> edge -> node
...

node -> edge -> node
node -> edge -> node
...

Use only the "Available Nodes" and "Available Edges" provided. No additional text, explanations, or formatting should be included.

Example Input 1:
...

Example Input 10:
- Context: The hole was carefully aligned to ensure a secure fit. A screwdriver was selected from the toolbox and positioned against the screw head.
- Target sentence: With a steady hand, the screwdriver was twisted, driving the screw into place while the piece was firmly held.
- Available nodes: person, screw, wood, hand1, hand2, hole, screwdriver, toolbox, align, select, position, hold, twist
- Available edges: verb, dobj, into, with, from, against
- Number of Scene Graphs: 2

Example Output 10:
person -> verb -> twist
twist -> dobj -> screwdriver
twist -> with -> hand2

Input:
- Context: {context}
- Target sentence: {target_sentence}
- Available nodes: {available_nodes}
- Available edges: {available_edges}
- Number of Scene Graphs: {num_scene_graphs}

Output:

Figure 24: Few-shot prompt for multiple action scene graph generation tasks.