KG-LLM-Bench: A Scalable Benchmark for Evaluating LLM Reasoning on Textualized Knowledge Graphs

Elan Markowitz[*†1], Krupa Galiya[*‡2], Greg Ver Steeg[1,3], and Aram Galstyan[1]

[1]University of Southern California
[2]Independent Researcher
[3]University of California, Riverside

Abstract

Knowledge graphs have emerged as a popular method for injecting up-to-date, factual knowledge into large language models (LLMs). This is typically achieved by converting the knowledge graph into text that the LLM can process in context. While multiple methods of encoding knowledge graphs have been proposed, the impact of this textualization process on LLM performance remains under-explored. We introduce KG-LLM-Bench, a comprehensive and extensible benchmark spanning five knowledge graph understanding tasks, and evaluate how different encoding strategies affect performance across various base models. Our extensive experiments with seven language models and five textualization strategies provide insights for optimizing LLM performance on KG reasoning tasks.

1 Introduction

The integration of knowledge graphs (KGs) with large language models (LLMs) has emerged as an important approach for enhancing contextual understanding in AI systems (Kau et al., 2024). Knowledge graphs are large structured databases of factual information that encode real world entities and their relationships. Recent surveys have highlighted the complementary nature of LLMs (static, unstructured, opaque, general) and KGs (dynamic, structured, interpretable, specific) (Pan et al., 2023; Zhu et al., 2023; Yang et al., 2023a; Li et al., 2024; Pan et al., 2024; Fan et al., 2024). This synergy has driven research on specialized architectures and algorithms for their integration (Luo et al., 2023; Wen et al., 2023; Markowitz et al., 2024). Most of these approaches rely on converting the KG to a readable text format suitable for LLM processing.

However, many of these algorithms give little consideration to the specific method of KG textualization (Fatemi et al., 2023; Chen et al., 2024; Yu et al., 2024). The most common approach simply encodes the KG as a list of edges in the form (source entity label, relation, object entity label). It is assumed that any approach will be equally effective and that using the same format ensures fair model comparison (Guo et al., 2023; Wang et al., 2024).

In this work, we challenge these assumptions and demonstrate that textualization strategy significantly impacts performance. Our experiments show that choosing the right strategy can improve overall benchmark performance by up to 17.5% absolute difference, with even larger gains on specific tasks. Figure 1 illustrates an example of the problem we are trying to analyze.

Our contributions are:

1. A scalable and extensible benchmark for analyzing how LLMs process and understand in-context knowledge graphs with five different tasks covering important KG reasoning capabilities.
2. Experiments covering five different textualization strategies using seven different popular LLM models, resulting in new insights and best practices.
3. Experiments with pseudonyms showing that LLMs do not heavily rely on memorized information when processing in-context knowledge graphs (overall difference of 0.2%).
4. A public release of the benchmark and framework so that it can be rapidly expanded.

2 Background

Knowledge Graphs are large structured databases that store factual associations as edges in a graph. They come in many varieties from general knowledge (Vrandečić and Krötzsch, 2014; Lehmann

*Denotes equal contribution
†esmarkow@usc.edu
‡krupagaliya@gmail.com

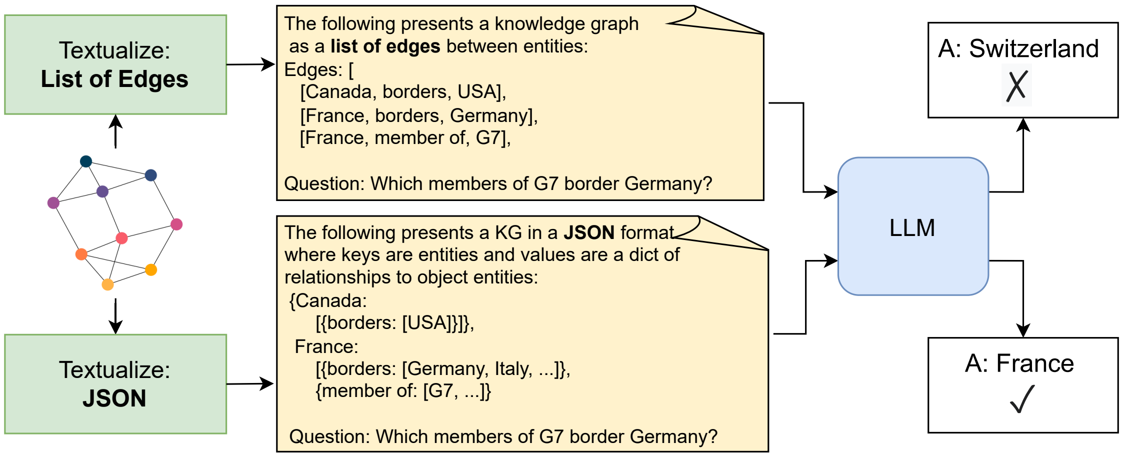

Figure 1: Different formats for graph textualization can result in highly varied performance on downstream tasks.

et al., 2015) to domain-specific variants (Finance: Liu et al., 2019; Geology: Zhu et al., 2017; Chemistry: Choi and Lee, 2019; Farazi et al., 2020; Biology: Zhang et al., 2023, and Medicine: Yang et al., 2023b).

We formally define a source knowledge graph $\mathcal{K} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ where $\mathcal{E}$ is the set of entities, $\mathcal{R}$ is the set of relation types, and $\mathcal{T}$ is the set of edges of the form $(s, r, o) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ e.g., ('Inception', 'director', 'Christopher Nolan').

We can define a subgraph of $\mathcal{K}$ as $G = (G_{\mathcal{E}}, G_{\mathcal{R}}, G_{\mathcal{T}})$ where $G_{\mathcal{E}} \subseteq \mathcal{E}$, $G_{\mathcal{R}} \subseteq \mathcal{R}$, and $G_{\mathcal{T}} \subseteq \mathcal{T}$.

Large Language Models can learn from information passed into their context window. This is used in retrieval augmented generation to produce more accurate LLM responses (Lewis et al., 2020). We can define the generation process using LLM π that responds to a query q using context text c:

$$\hat{y} = \pi(c, q) \tag{1}$$

where π generates a response $\hat{y}$. This context can include any text-format data, including various text encodings for knowledge graphs (Fig 1).

3 Related Work

Benchmarks for Graphs Reasoning Recent work has extensively evaluated LLM understanding of graph-structured data. Most benchmarks focus on simple graphs rather than knowledge graphs, including GPT4Graph (Guo et al., 2023), GraphArena (Tang et al., 2024), and GraCoRe (Yuan et al., 2024). Research has expanded to text-space graph foundation models (Chen et al., 2024) and hypergraphs (Feng et al., 2024). Particularly relevant to our work, Fatemi et al. (2023) evaluates how different natural language presentations affect graph understanding, and was later extended to learned graph encoders (Perozzi et al., 2024).

KG Question Answering (KGQA) Benchmarks The KGQA field has produced several key benchmarks, including QALD-10 (Usbeck et al., 2023), 2WikiMultiHop (Ho et al., 2020), and MetaQA (Zhang et al., 2018). While HotpotQA (Yang et al., 2018) is not KG-grounded, it is still often used for knowledge grounded evaluation. Recent work like CLR-Fact (Zheng et al., 2024) evaluates LLMs on complex logical query answering (Arakelyan et al., 2020; Galkin et al., 2024).

More focused studies have examined specific KG processing capabilities in LLMs, including KG completion (Yao et al., 2023), construction (Zhu et al., 2024), causal reasoning (Kim et al., 2024), and trustworthiness enhancement (Sui et al., 2024). While some work has explored KG formatting, such as RDF Turtle parsing (Frey et al., 2023) and natural language presentations (Dai et al., 2024), our work presents the first comprehensive evaluation of textualization strategies.

KG-Grounded Models and Algorithms Recent approaches to grounding LLMs with knowledge graphs include Think-on-Graph (Sun et al., 2023) and MindMap (Wen et al., 2023). Tree-of-Traversals (Markowitz et al., 2024) enables test-time search over KG reasoning paths. Alternative approaches like graph-to-tree text encoding (Yu et al., 2024) focus on specialized encoding strategies.

LLM Reasoning and Long Context Models Two emerging research directions could significantly impact KG reasoning capabilities. Test-time reasoning models like OpenAI's o1/o3 and DeepSeek's R1 (DeepSeek-AI et al., 2025) enable using enhanced computational resources at inference-time. Meanwhile, advances in long context models (Chen et al., 2023; Lieber et al., 2024; Hooper et al., 2024; Munkhdalai et al., 2024) allow processing of larger knowledge graphs. Our benchmark is positioned to evaluate both these developments in the context of graph reasoning.

4 KG-LLM-Bench Framework

This section details the methodology of KG-LLM-Bench, which evaluates LLMs on knowledge graph question answering tasks. In summary, the LLM answers task-specific questions based on a KG subgraph G, with responses evaluated against predefined scoring criteria. Figure 2 provides an overview of our framework.

4.1 Text Representation of KG

We define a set of textualization functions $\mathcal{F}$ that convert the structured knowledge graph G into a textual representation $x_G \in \mathcal{W}^*$, where $\mathcal{W}^*$ represents the set of all possible text strings:

$$x_G = f(G) \tag{2}$$

where $f \in \mathcal{F}$. The specific textualization functions are detailed in Sec 6.2.

4.2 Query Construction

We design a set of tasks where each task T (e.g., triple retrieval, shortest path) can be used to generate queries q and answers a for graph G:

$$q = \mathcal{Q}_T(G) \tag{3}$$

$$a = \mathcal{A}_T(G, q) \tag{4}$$

where $\mathcal{Q}_T$ formulates the natural language query and $\mathcal{A}_T$ generates the corresponding answer. These are stochastic functions but are implemented with fixed seeds to ensure deterministic behavior.

4.3 Model Generation and Evaluation

The LLM π generates an answer $\hat{y} = \pi(x_G, q)$ using the textualized graph as context. We evaluate this against the ground truth a using a scoring function $\mathcal{S}$:

$$s = \mathcal{S}(\hat{y}, a) \in \{0, 1\} \tag{5}$$

where s indicates correctness. While $\mathcal{S}$ is customizable, we use exact match in our experiments. The complete pipeline can be expressed as:

$$s = \mathcal{S}\left(\pi\left(f(G), \mathcal{Q}_T(G)\right), \mathcal{A}_T(G, \mathcal{Q}_T(G))\right) \tag{6}$$

4.4 Optimizing for an LLM

We can consider optimizing the textualization choice for a given model as an optimization of the expected performance of the LLM over the distribution of tasks, possible graph contexts, and questions and answers:

$$\max_{f \in \mathcal{F}} \mathbb{E}_{T,G,\mathcal{Q},\mathcal{A}} \Bigg[\mathcal{S}\Big(\pi\big(f(G), \mathcal{Q}_T(G), \mathcal{A}_T(G, \mathcal{Q}_T(G))\big)\Big)\Bigg] \tag{7}$$

4.5 Sampling graphs

We sample graphs $G \sim \textit{subgraph}(\mathcal{K})$ using seed entities, sampling radius, and max edges parameters. For each seed entity, we first sample ego-graphs containing all edges within the specified radius. An ego-graph for entity e with radius r is defined as $EgoGraph(e, r) =$

$$\{t = (s, r, o) | d(e, s) \leq r, d(e, o) \leq r, t \in \mathcal{T}\} \tag{8}$$

where d is the graph distance function. After combining ego-graphs, we apply a low-degree filter to remove single-edge entities, then randomly prune edges to meet the size constraint.

4.6 Pseudonymization

To ensure models rely solely on the provided knowledge graph rather than pre-trained knowledge, we introduce a pseudonymization function p that maps entities to synthetic labels. Given a set of pseudonymized entity labels $\hat{\mathcal{E}}$, we create:

$$\hat{G} = p(G, \hat{\mathcal{E}}) \tag{9}$$

where the pseudonymization creates the mapping $\left\{e : \hat{e} | e \in G_{\mathcal{E}}, \hat{e} \in \hat{\mathcal{E}}\right\}$ and applies it to G.

For our experiments with historical country entities, we generate semantically appropriate pseudonyms using a combination of a name generator tool* and LLM-generated names, filtering inappropriate or insulting samples.

*https://www.name-generator.org.uk/

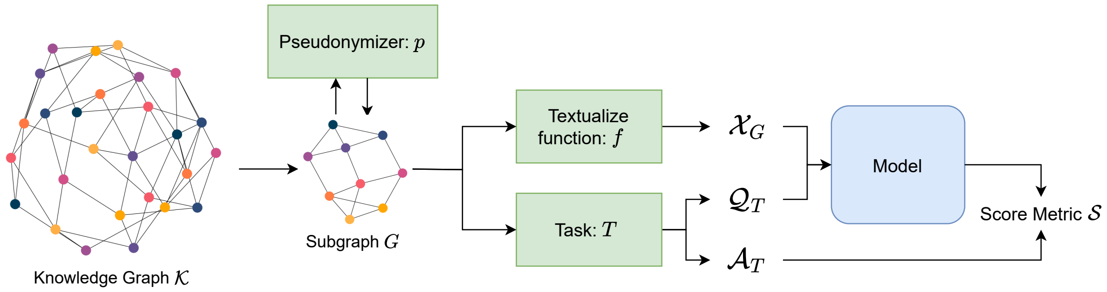

Figure 2: Framework for KG-LLM-Bench.

5 KG LLM Tasks

We present five fundamental tasks in our benchmark, each chosen to evaluate distinct aspects of KG reasoning: retrieval, path-based reasoning, local aggregation, multi-hop aggregation, and global analysis. Together, these tasks provide a comprehensive evaluation of an LLM's ability to reason over knowledge graphs.

5.1 Triple Retrieval Task

The TripleRetrieval task tests an LLM's fundamental ability to verify the presence of relationships in graph G. This capability underlies all more complex graph reasoning tasks, as models must first accurately identify existing relationships before performing higher-order reasoning.

Questions are evenly split between positive and negative cases. Positive samples are drawn directly from edges $(s, r, o) \sim G_{\mathcal{T}}$. For negative samples, we create invalid edges by replacing either the source entity, relation type, or object entity with alternatives ($s', o' \sim G_{\mathcal{E}}$ or $r' \sim G_{\mathcal{R}}$) such that the resulting edge does not exist in $G_{\mathcal{T}}$.

5.2 Shortest Path Task

The ShortestPath task evaluates a model's ability to find the shortest path between two entities in G, considering edges in either direction. This task is relevant as the shortest path between two entities is likely to represent the strongest association between those two entities. For instance, "my brother's employer" is a more direct and informative association than "my mother's sister's nephew's employer". Detailed implementation is provided in Appendix A.2.

5.3 Aggregation By Relation

The AggByRelation task tests local aggregation from anchor nodes, a common requirement in real-world queries. For example, "How many diplomatic relations does Uruguay have?" requires aggregating connections from a specific entity.

Questions in this task take the form "How many {*incoming/outgoing*} relations of type {*relation type*} does {*anchor entity*} have?". Since randomly sampling a relation type and direction and anchor entity most likely results in an aggregation over a single edge, we modify the approach to ensure variety in the questions and answers. Details of this can be found in Appendix A.3.

5.4 Aggregation of Neighbor Properties

The AggNeighborProperty task extends aggregation to two-hop paths, requiring more complex reasoning. Models must answer questions of the form "How many of the directly connected entities to {*anchor entity*} have an outgoing property of type {*relation*} in the knowledge graph?". Many real questions combine aggregation on multi-hop edges. For instance "How many actors who starred in Inception have won Academy Awards?" or "How many universities that collaborate with Stanford University have research centers focused on artificial intelligence?".

The task uses similar sampling approaches to AggByRelation but with modified aggregation functions (Appendix A.4).

5.5 Highest Degree Node by Direction

The HighestDegree task tests global graph reasoning by identifying the entity with the most (incoming/outgoing/total) edges in G. The distinction between edge directions is significant. Since many textualization functions group edges with the same source, counting outgoing degree is a more local problem than counting incoming degree.

While more difficult global tasks could be proposed (e.g. graph isomorphism or connectivity statistics), we note that at the scale of graph we use, this task already proves to be relatively difficult.

Statistic	Count
Core Entities	3,552
Attribute Entities	27,226
Core Relations	49
Attribute Relations	162
Core Facts	11,361
Attribute Facts	51,952

Table 1: Statistics of the WikiDataSets Countries dataset. Core entities represent countries while attribute entities represent related concepts such as languages or significant events. Core relations and facts involve only country-to-country relationships, while attribute relations and facts connect countries to their attributes.

6 Experiments

The following section describes the setup for our experiments. Our benchmark consists of five tasks (100 instances each, plus pseudonymized versions) and five textualization strategies, and we evaluate on seven LLMs.

6.1 Data

Our experiments use the Countries knowledge graph from WikiDataSets (Boschin and Bonald, 2019) as our source graph $\mathcal{K}$. This knowledge graph is a subgraph related to historical countries derived from Wikidata (Vrandečić and Krötzsch, 2014).

The graph contains diverse relationship types covering geographical relations (e.g., borders), political relations (e.g., diplomatic relations), and temporal relations (e.g., followed by). In addition to the 49 core relations there are 162 attribute relations that connect countries to other types of entities such as languages or significant events. Table 1 summarizes the key statistics of the dataset.

When sampling subgraphs for our tasks, we follow the procedure outlined in Section 4.5. The specific parameters were chosen to ensure reasonable questions could be generated for each task and that the subgraph is reasonable in terms of context size. Each question is asked over a subgraph with 200 edges. Full details are available in the appendix. We generate 100 sets of subgraph, question, and answer for each task.

6.2 Textualization Strategies

We evaluate five common textualization strategies for converting knowledge graphs into text:

1. **List-of-Edges**: A simple triple-based representation where each line contains a (subject, predicate, object) statement.
2. **Structured YAML**: A hierarchical representation using YAML syntax, grouping relationships by subject entities.
3. **Structured JSON**: Similar to YAML but using JSON syntax.
4. **RDF Turtle**: A W3C standard format for representing RDF graphs, using prefixes and semicolons to group statements with the same subject. This format is commonly used in semantic web applications.
5. **JSON-LD**: A JSON-based format for linked data that provides both human-readable structure and semantic web compatibility through the inclusion of contexts and URIs.

Each format represents different tradeoffs between compactness, readability, and structure, allowing us to evaluate how these characteristics affect LLM performance. Details and examples can be seen in Appendix B.

6.3 Models

We evaluate seven different language models spanning different sizes and architectures. These models are **Llama 3.3-70B** (Meta, 2024b), **Llama 3.2-1B** (Meta, 2024a), **GPT-4o-Mini** (OpenAI, 2024), **Claude-3.5-Sonnet** (Anthropic, 2024), **Amazon Nova Lite** (Intelligence, 2024), **Amazon Nova Pro** (Intelligence, 2024), and **Gemini-1.5-Flash** (Gemini Team, 2024) This selection allows us to evaluate the effect of textualization strategies across a broad range of models.

6.4 Evaluation Protocol

To construct the benchmark and run the experiments, we do the following steps for each task.

1. Sample 100 subgraphs from $\mathcal{K}$ (Sec 4.5)
2. Pseudonymize each subgraph (Eq. 9)
3. Generate questions and answers following task-specific protocols (Sec 4.2).
4. Apply textualization strategies $f \in \mathcal{F}$ (Eq. 2)
5. For each dataset, query model (Eq. 1)
6. Evaluate with exact match (Eq. 5)

7 Results

The following section presents our results. Our main finding is the best overall textualization strategies are Structured JSON and List of Edges. However, there is a complex interplay between the textualization of the graph, the model and the task.

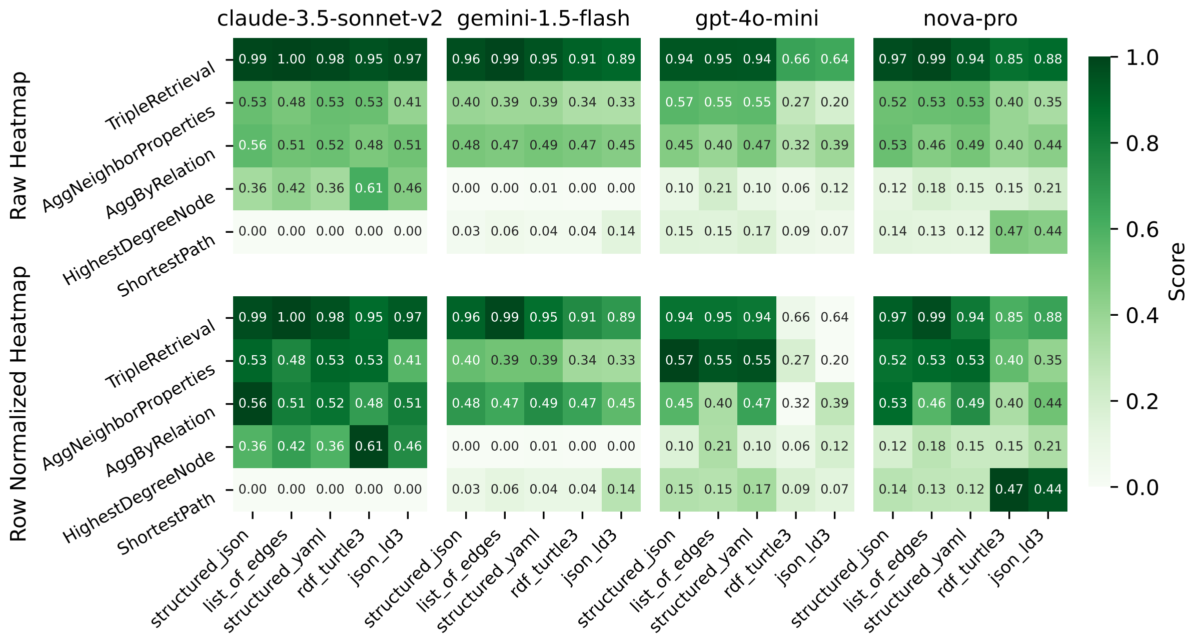

Figure 3: Heatmaps of the performance of various models. Each heatmap shows tasks as rows and textualize functions as columns. **(Top)** Heatmap colors as globally weighted from [0.0-1.0]. **(bottom)** heatmap colors normalized for each task [task minimum-task maximum]. The tasks are ordered from easiest overall to hardest. The textualization functions are ordered from best performing overall to worst. Additional models are in the appendix.

Table 2: Best Textualization Strategy for Each Model

Model	Best f
claude-3.5-sonnet-v2	RDF Turtle
gemini-1.5-flash	List of Edges
gpt-4o-mini	List of Edges
llama3.2-1b-instruct	List of Edges
llama3.3-70b-instruct	Structured JSON
nova-lite	Structured JSON
nova-pro	JSON-LD

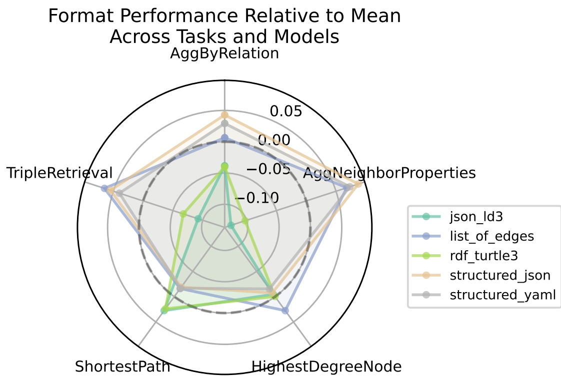

Figure 4: Mean performance across different tasks for different textualization strategies. The values show the absolute diference in performance for each textualization function compared to the mean for that task. The mean is shown as the dashed circle.

This can be seen in the different performance patterns in Figure 3. Therefore, developers must optimize textualization choice for their specific use case and model. We will present analysis on the high-level effect of textualization choice, how the selected models compare to each other, the effect of pseudonymization on suppressing memorized information, and the token efficiency of the different textualization strategies.

The full results data is presented in Table 5 in the appendix due to space constraints, and more tailored results are presented in this section.

7.1 Effect of Textualization Function

While not all global results on textualization hold true for every model, there are some global patterns that we see. Figure 4 gives a radar plot of performance by textualization function. Structured JSON performs best (0.42 average), followed by YAML and List-of-Edges, while RDF Turtle (0.35) and JSON-LD (0.34) perform worst. Part of the reason for this may be the more complex encoding strategies and use of URIs makes the format more difficult to parse. This may be further amplified by the fact that it dramatically increases the input token counts which may cause performance

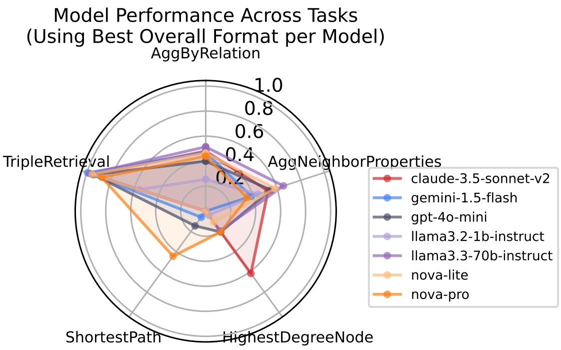

Figure 5: Mean performance across different tasks for each model.

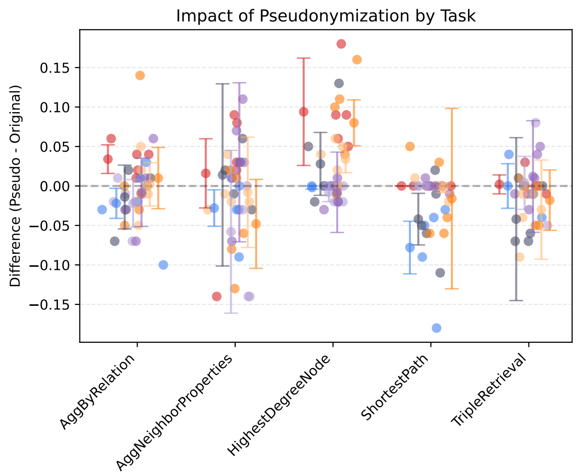

Figure 6: Impact of pseudonymization by task. Higher means that the model did better with pseudonymization. Each color represents a different model.

degradation on some of the models.

Across all tasks, List-of-Edges performs quite well. This may be due to the fact it is the most commonly used, and thus may be the most common encoding format in instruction tuning data. On the aggregation tasks Structured YAML and Structured JSON outperform. This makes sense as these structures naturally aggregate related edges together. List-of-Edges only seems better on the global task of Highest Degree Node. This, too, makes sense, as the highest degree node will appear the most times in the list of edges format, but that is not guaranteed nor likely to be the case in the structured formats.

7.2 Model Performance

We can also use KG-LLM-Bench to compare the performance of various models. We plot the comparative performances of the seven models in Figure 5. To enable a fair comparison, we use data from the best performing overall textualization strategy for each model (Table 2). We can see that the best performing approach is highly variable model to model.

Overall, task performance and model rankings align with expectations. The easiest task is Triple Retrieval and the hardest task is Shortest Path. Surprisingly, Highest Degree task appeared to be significantly harder (most models less than 20%) compared to the two aggregation tasks (most models scoring 40%-60%).

There were two notable outliers on performance. The biggest outlier was Nova-Pro scored by far the highest on the Shortest Path task, 47% with RDF Turtle and 44% with JSON-LD. The next best single Shortest Path result was gpt-4o-mini scoring 17% with Structured YAML. The other major outlier is Claude-3.5-Sonnet performance on the

Table 3: Average Input Token Usage by Format

Textualizer	Mean Input Tokens
List of Edges	2644.8 ± 390.4
Structured JSON	4504.7 ± 1123.2
Structured YAML	2903.1 ± 655.9
RDF Turtle	8171.1 ± 2284.6
JSON-LD	13503.4 ± 3611.2
Overall	6345.4 ± 1613.1

Highest Degree task. Sonnet received 61.5% with RDF-Turtle and averaged 44.3% over all formats. This is much better than the next best performer, Nova Pro at 16.2%. Partially helped by these outlier abilities, Claude-3.5-Sonnet and Nova-Pro were the top overall models.

7.3 Pseudonymization

Pseudonymization shows minimal effect, likely because questions on sampled subgraphs of $\mathcal{K}$ already prevent reliance on memorized information. This is because there is a low chance of the memorized information being present (e.g. knowing the number of countries France borders is not helpful if G only contains a subset of those edges). There is some limited evidence in Figure 6 that pseudonymization actually helped on the Highest Degree Task. This may be that the model would erroneously guess based on memorized knowledge when it saw familiar entity names.

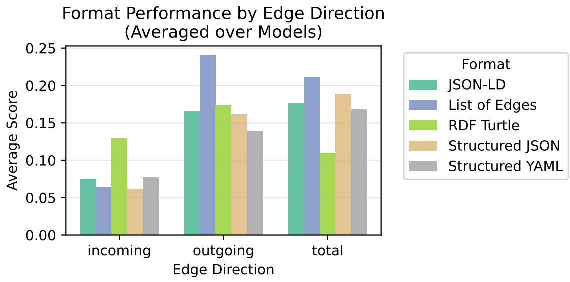

Figure 7: Performance on highest degree task by aggregation direction.

7.4 Token Efficiency

We note that not all textualization functions are equally token efficient. In fact they can vary a lot. List-of-Edges and and Structured YAML were the most token efficient with below 3000 tokens per prompt. These approaches are simple and have commonly been used with LLMs that historically had limited context windows. JSON-LD was the least token efficient, taking over 13,000 tokens per prompt followed by RDF Turtle at 8,000 tokens per prompt. Since RDF Turtle and JSON-LD are designed to be usable with semantic web technologies, they require complete and unambiguous specification of the schema. This results in many additional specifications like namespaces and URI encodings. We note that we even optimized some of these choices to reduce token usage. More naive encodings could use far more tokens.

7.5 Aggregation Direction

We find that aggregation performance significantly differs by aggregation direction. Figure 7 shows the effect of aggregation direction when performing the HighestDegree task. The models do significantly better when predicting highest degree by outgoing edges than by incoming. In all the textualization formats (including our implementation of List of Edges), outgoing edges are listed next to each other. This makes it much easier for the model to aggregate over outgoing edges than over incoming ones. This difference is least pronounced in RDF Turtle.

8 Aggregation Dependence on Degree

We analyze aggregation degree affects model performance on the AggregationByRelation task, and present the results in Figure 8. For a single edge aggregation, the model answers correctly over 80% of the time and remains above 50% for aggregations up to degree 4. Beyond that, performance

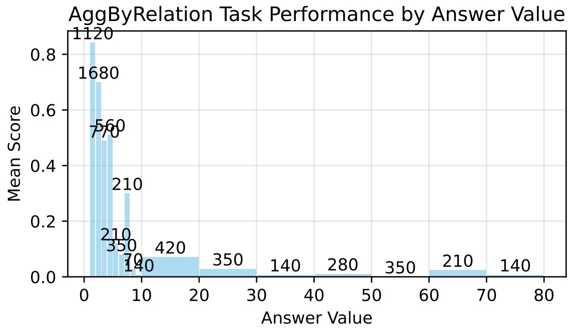

Figure 8: Effect of aggregation size on accuracy in AggByRelation task. Numbers show the count of results in each bin. We note that while the counts are high these do not represent completely i.i.d. samples as the same base question will be presented many times.

rapidly degrades to around 10%. The sharpness of the dropoff indicates these models have significant room for improvement in aggregation capability.

9 Future Work

There are a many directions for future work based on KG-LLM-Bench. Our framework's modular design allows for easy extension to new tasks, graphs, models, and textualization strategies.

In terms of research, we see two major directions to extend KG-LLM-Bench: Scale and Reasoning. In terms of scale, KG-LLM-Bench can be easily scaled by modifying the subgraph sampling parameters, and the source graph can be easily swapped should that become an issue. This enables the study of long-context reasoning over KGs.

The other major direction would be to support studying of test-time reasoning on KGs. The generation of new queries is a bottleneck in developing reasoning models for LLMs. Since KG-LLM-Bench can continuously generate new queries, it could be a useful in training KG reasoning models.

10 Conclusion

In this work, we introduced KG-LLM-Bench, a comprehensive and extensible benchmark for evaluating how LLMs process and understand textualized knowledge graphs. Through extensive experiments across five distinct tasks, seven models, and five textualization strategies, we demonstrated that the choice of textualization strategy has a significant impact on model performance. While simpler formats like List-of-Edges and Structured JSON tend to perform well overall, the best performing format varies depending on model and task.

By understanding and improving how LLMs process structured knowledge, we can ultimately develop more reliable and effective knowledge-enhanced language models.

11 Limitations

Here we note some limitations of our experiments and framework. First, our evaluation uses only the WikiDataSets Countries knowledge graph. While this provides a controlled environment, the experiments should be expanded to other domains with different relationship types. Second, we use subgraphs of 200 edges to ensure reasonable context windows, which may not accurately reflect capabilities on smaller graphs, nor capture the challenges of reasoning over larger knowledge graphs. Finally, while we evaluate different textualization strategies, our experiments are conducted in English (though some entities are in other languages).

There are a few current limitations in our framework. It currently only handles knowledge graphs with the defined triple structure. It cannot handle literal values (numbers or dates), temporal knowledge graphs, or any form of hypergraph (e.g. Wikidata edges can have qualifier edges). These would make good areas for future expansion.

12 Ethical Statement

We see no immediate ethical impact of this work. The authors believe that more factual and trustworthy AI models are ethically desirable. However, this work can be used for enhancing AI capabilities, which could present other ethical ramifications. We encourage anyone using KG-LLM-Bench to consider the ethical impact of their work or applications.

References

Anthropic. 2024. Claude 3.5 sonnet model.

Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. 2020. Complex query answering with neural link predictors. *ArXiv*, abs/2011.03459.

Armand Boschin and Thomas Bonald. 2019. Wikidatasets: Standardized sub-graphs from wikidata. *Preprint*, arXiv:1906.04536.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023. Extending context window of large language models via positional interpolation. *ArXiv*, abs/2306.15595.

Zhikai Chen, Haitao Mao, Jingzhe Liu, Yu Song, Bingheng Li, Wei Jin, Bahare Fatemi, Anton Tsitsulin, Bryan Perozzi, Hui Liu, and Jiliang Tang. 2024. Text-space graph foundation models: Comprehensive benchmarks and new insights. *Preprint*, arXiv:2406.10727.

Wonjun Choi and Hyunju Lee. 2019. Inference of biomedical relations among chemicals, genes, diseases, and symptoms using knowledge representation learning. *IEEE Access*, 7:179373–179384.

Xinnan Dai, Haohao Qu, Yifen Shen, Bohang Zhang, Qihao Wen, Wenqi Fan, Dongsheng Li, Jiliang Tang, and Caihua Shan. 2024. How do large language models understand graph patterns? a benchmark for graph pattern comprehension. *Preprint*, arXiv:2410.05298.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Jun-Mei Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiaoling Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bing-Li Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dong-Li Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Jiong Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, M. Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shao-Kang Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wen-Xia Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyu Jin, Xi-Cheng Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yi Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yu-Jing Zou, Yujia He, Yunfan Xiong, Yu-Wei Luo, Yu mei You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yao Li, Yi Zheng, Yuchen Zhu, Yunxiang Ma, Ying Tang, Yukun Zha, Yuting Yan, Zehui Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhen guo Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zi-An Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.

Wenqi Fan, Shijie Wang, Jiani Huang, Zhikai Chen, Yu Song, Wenzhuo Tang, Haitao Mao, Hui Liu, Xiaorui Liu, Dawei Yin, and Qing Li. 2024. Graph machine learning in the era of large language models (llms). *Preprint*, arXiv:2404.14928.

Feroz Farazi, Maurin Salamanca, Sebastian Mosbach, Jethro Akroyd, Andreas Eibeck, Leonardus Kevin Aditya, Arkadiusz Chadzynski, Kang Pan, Xiaochi Zhou, Shaocong Zhang, Mei Qi Lim, and Markus Kraft. 2020. Knowledge graph approach to combustion chemistry and interoperability. *ACS Omega*, 5:18342 – 18348.

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. 2023. Talk like a graph: Encoding graphs for large language models. *Preprint*, arXiv:2310.04560.

Yifan Feng, Chengwu Yang, Xingliang Hou, Shaoyi Du, Shihui Ying, Zongze Wu, and Yue Gao. 2024. Beyond graphs: Can large language models comprehend hypergraphs? *Preprint*, arXiv:2410.10083.

Johannes Frey, Lars-Peter Meyer, Natanael Arndt, Felix Brei, and Kirill Bulert. 2023. Benchmarking the abilities of large language models for rdf knowledge graph creation and comprehension: How well do llms speak turtle? *ArXiv*, abs/2309.17122.

Mikhail Galkin, Jincheng Zhou, Bruno Ribeiro, Jian Tang, and Zhaocheng Zhu. 2024. A foundation model for zero-shot logical query reasoning. In *Neural Information Processing Systems*.

et al. Gemini Team. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *Preprint*, arXiv:2403.05530.

Jiayan Guo, Lun Du, Hengyu Liu, Mengyu Zhou, Xinyi He, and Shi Han. 2023. Gpt4graph: Can large language models understand graph structured data ? an empirical evaluation and benchmarking. *Preprint*, arXiv:2305.15066.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. 2024. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *ArXiv*, abs/2401.18079.

Amazon Artificial General Intelligence. 2024. The amazon nova family of models: Technical report and model card. *Amazon Technical Reports*.

Amanda Kau, Xuzeng He, Aishwarya Nambissan, Aland Astudillo, Hui Yin, and Amir Aryani. 2024. Combining knowledge graphs and large language models. *Preprint*, arXiv:2407.06564.

Yejin Kim, Eojin Kang, Juae Kim, and H. Howie Huang. 2024. Causal reasoning in large language models: A knowledge graph approach. *Preprint*, arXiv:2410.11588.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, S. Auer, and Christian Bizer. 2015. Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6:167–195.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *ArXiv*, abs/2005.11401.

Yuhan Li, Zhixun Li, Peisong Wang, Jia Li, Xiangguo Sun, Hong Cheng, and Jeffrey Xu Yu. 2024. A survey of graph meets large language model: Progress and future directions. *Preprint*, arXiv:2311.12399.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Haim Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avshalom Manevich, Nir Ratner, Noam Rozen, Erez Shwartz, Mor Zusman, and Yoav Shoham. 2024. Jamba: A hybrid transformer-mamba language model. *ArXiv*, abs/2403.19887.

Yang Liu, Qingguo Zeng, Joaquín B. Ordieres Meré, and Huanrui Yang. 2019. Anticipating stock market of the renowned companies: A knowledge graph approach. *Complex.*, 2019:9202457:1–9202457:15.

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

Elan Markowitz, Anil Ramakrishna, J. Dhamala, Ninareh Mehrabi, Charith Peris, Rahul Gupta, Kai-Wei Chang, and A. G. Galstyan. 2024. Tree-of-traversals: A zero-shot reasoning algorithm for augmenting black-box language models with knowledge graphs. In *Annual Meeting of the Association for Computational Linguistics*.

Meta. 2024a. Llama 3.2 quantized models.

Meta. 2024b. Llama 3.3 text-only 70b instruction-tuned model.

Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. 2024. Leave no context behind: Efficient infinite context transformers with infini-attention. *ArXiv*, abs/2404.07143.

OpenAI. 2024. Gpt-4o mini: Advancing cost-efficient intelligence.

Jeff Z. Pan, Simon Razniewski, Jan-Christoph Kalo, Sneha Singhania, Jiaoyan Chen, Stefan Dietze, Hajira Jabeen, Janna Omeliyanenko, Wen Zhang, Matteo Lissandrini, Russa Biswas, Gerard de Melo, Angela Bonifati, Edlira Vakaj, Mauro Dragoni, and Damien Graux. 2023. Large language models and knowledge graphs: Opportunities and challenges. *ArXiv*, abs/2308.06374.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3580–3599.

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

Yuan Sui, Yufei He, Zifeng Ding, and Bryan Hooi. 2024. Can knowledge graphs make large language models more trustworthy? an empirical study over open-ended question answering. *Preprint*, arXiv:2410.08085.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Sai Wang, Chen Lin, Yeyun Gong, Heung yeung Shum, and Jian Guo. 2023. Think-on-graph: Deep and responsible reasoning of large language model with knowledge graph. *ArXiv*, abs/2307.07697.

Jianheng Tang, Qifan Zhang, Yuhan Li, and Jia Li. 2024. Grapharena: Benchmarking large language models on graph computational problems. *Preprint*, arXiv:2407.00379.

Ricardo Usbeck, Xi Yan, Aleksandr Perevalov, Longquan Jiang, Julius Schulz, Angelie Kraft, Cedric Möller, Junbo Huang, Jan Reineke, Axel-Cyrille Ngonga Ngomo, et al. 2023. Qald-10—the 10th challenge on question answering over linked data. *Semantic Web – Interoperability, Usability, Applicability an IOS Press Journal*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57:78–85.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2024. Can language models solve graph problems in natural language? *Preprint*, arXiv:2305.10037.

Yilin Wen, Zifeng Wang, and Jimeng Sun. 2023. Mindmap: Knowledge graph prompting sparks graph of thoughts in large language models. In *Annual Meeting of the Association for Computational Linguistics*.

Lin F. Yang, Hongyang Chen, Zhao Li, Xiao Ding, and Xindong Wu. 2023a. Chatgpt is not enough: Enhancing large language models with knowledge graphs for fact-aware language modeling. *ArXiv*, abs/2306.11489.

Peiru Yang, Hongjun Wang, Yingzhuo Huang, Shuai Yang, Ya Zhang, Liang Huang, Yuesong Zhang, Guoxin Wang, Shizhong Yang, Liang He, and Yongfeng Huang. 2023b. Lmkg: A large-scale and multi-source medical knowledge graph for intelligent medicine applications. *Knowl. Based Syst.*, 284:111323.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing*.

Liang Yao, Jiazhen Peng, Chengsheng Mao, and Yuan Luo. 2023. Exploring large language models for knowledge graph completion. *ArXiv*, abs/2308.13916.

Zhaoning Yu, Xiangyang Xu, and Hongyang Gao. 2024. G2t-llm: Graph-to-tree text encoding for molecule generation with fine-tuned large language models. *Preprint*, arXiv:2410.02198.

Zike Yuan, Ming Liu, Hui Wang, and Bing Qin. 2024. Gracore: Benchmarking graph comprehension and complex reasoning in large language models. *Preprint*, arXiv:2407.02936.

Yuan Zhang, Xing Sui, Feng Pan, Kaixian Yu, Keqiao Li, Shubo Tian, Arslan Erdengasileng, Qing Han, Wanjing Wang, Jianan Wang, Jian Wang, Donghu Sun, Henry Chung, Jun Zhou, Eric Zhou, Ben Lee, Peili Zhang, Xing Qiu, Tingting Zhao, and Jinfeng Zhang. 2023. Biokg: a comprehensive, large-scale biomedical knowledge graph for ai-powered, data-driven biomedical research. *bioRxiv*.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *AAAI*.

Tianshi Zheng, Jiaxin Bai, Yicheng Wang, Tianqing Fang, Yue Guo, Yauwai Yim, and Yangqiu Song. 2024. Clr-fact: Evaluating the complex logical reasoning capability of large language models over factual knowledge. *Preprint*, arXiv:2407.20564.

Yueqin Zhu, Wenwen Zhou, Yang Xu, Ji Liu, and Yongjie Tan. 2017. Intelligent learning for knowledge graph towards geological data. *Sci. Program.*, 2017:5072427:1–5072427:13.

Yuqi Zhu, Xiaohan Wang, Jing Chen, Shuofei Qiao, Yixin Ou, Yunzhi Yao, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Llms for knowledge graph construction and reasoning: Recent capabilities and future opportunities. *ArXiv*, abs/2305.13168.

Yuqi Zhu, Xiaohan Wang, Jing Chen, Shuofei Qiao, Yixin Ou, Yunzhi Yao, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2024. Llms for knowledge graph construction and reasoning: Recent capabilities and future opportunities. *Preprint*, arXiv:2305.13168.

A Additonal Implementation Details

A.1 Evaluation Algorithm

Algorithm 1 gives the formal algorithm for KG-LLM-Bench constructin and evaluation.

A.2 Shortest Path

We consider the shortest path between source entity e_s and destination entity e_d using edges in any direction. So for instance, we could use the edges (e_s, r_1, e_1) and (e_d, r_2, e_1) to form the path $[e_s, e_1, e_d]$. This makes the task potentially more difficult as it forces the model to rely on associations that appear in the reversed order from how they appear in the text representation x_G.

To construct these questions, we take two of the seed entities used to construct the subgraph G (e_s and e_d) and set them as the source and destination nodes for the question. We then collect the set of all shortest paths $P = p_1, ..., p_k$ from e_s to e_d in $\mathcal{K}$. We use the set of all entities in p_1 as additional seed entities for $subgraph(\mathcal{K})$ and further ensure that all edges in p_1 become part of G. This ensures that at least p_1 is present in the graph. Finally, we consider any answer in P to be a valid answer.

A.3 Aggregation by Relation

Here we present additional details on the construction of AggByRelation questions. In our work, we use `COUNT` aggregation as we do not consider edges with literal expressions (e.g. numbers or dates), only other entities.

We first compute the aggregation count for every possible question that could be constructed for G (every direction, relation type, and anchor entity). Specifically

$$Agg(s, r, dir) = \left| \left\{ t = \begin{cases} (s, r, e) & dir = 1 \\ (e, r, s) & dir = 0 \end{cases} \middle| e \in G_{\mathcal{E}}, t \in G_{\mathcal{T}} \right\} \right| \tag{10}$$

Algorithm 1: KG-LLM-Bench

Input : $\mathcal{K} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$: Knowledge graph
$\mathcal{F}$: Textualization function set
$\mathcal{T}$: Task set
π: Language model
$\hat{\mathcal{E}}$: Pseudonym entity set
N: Number of instances

Output : $\{s_{ijk}\}$: Evaluation scores where i, j, k index tasks, textualization, and question instance

```
for T ∈ 𝒯 do
    /* Construct N instances                          */
    for i ← 1 to N do
        /* Sample subgraph from
           knowledge graph                            */
        G_i ∼ subgraph(𝒦)                    ▷Sec 4.5
        /* Create anonymized version
           if requested                               */
        Ĝ_i ← p(G_i, Ê)                      ▷Eq. 9
        /* Generate task-specific
           question and ground truth
           */
        q_i ← 𝒬_T(Ĝ_i)
        a_i ← 𝒜_T(Ĝ_i, q_i)                  ▷Sec 4.2
    /* Evaluate each textualization
       strategy                                       */
    for f ∈ ℱ do
        for i ← 1 to N do
            /* Convert graph to
               textual format                         */
            x_i ← f(Ĝ_i)                     ▷Eq. 2
            /* Query LLM with context
               and question                           */
            ŷ_i ← π(x_i, q_i)                ▷Eq. 1
            /* Evaluate response with
               exact match                            */
            s_i ← 𝒮(ŷ_i, a_i)                ▷Eq. 5
```

We then collect the set of possible answers $A = \{Agg(s, r, dir) | s \in G_{\mathcal{E}}, r \in G_{\mathcal{R}}, dir \in [1, 0]\}$ and randomly select an answer $a \sim A$. After selecting the desired answer, we finally sample the s, r, and dir that would give answer a: $(s, r, dir) \sim \{(s, r, dir) | Agg(s, r, dir) = a\}$.

A.4 Aggregation of Neighbor Properties

To construct these questions we follow the same procedure as the previous task but use a different aggregation formula.

$$Agg(s, r) = \left| \left\{ e_1 \in G_{\mathcal{E}} \middle| \exists t_1, t_2 \in G_{\mathcal{T}} : t_1 \in \{(s, \_, e_1), (e_1, \_, s)\} \wedge t_2 = (e_1, r, \_) \right\} \right| \tag{11}$$

"_" indicates wildcards that could match any relation or entity that makes the edge valid in $G_{\mathcal{T}}$

A.5 Sampling Parameters

Table 4 details the sampling parameters used for each task in our benchmark.

A.6 Pseudonymization

For pseudonymization, we use a pre-defined set of country pseudonyms stored in csv file. These pseudonyms are designed to maintain the semantic naturalness of the graph while preventing the model from leveraging pre-trained knowledge about real countries.

We use a fake names generator to generate the first 100 fake country names and then use claude-3.5-sonnet to generate an additional 600 of a similar style.

B Examples of Text Formats

B.1 List of Edges

```
Your job is to answer questions using the following
     knowledge graph. The knowledge graph is
     presented as a structured JSON format. Each
     entity is a key, and the value is a dictionary
     of relations and objects.. You must rely
     exclusively on the information presented in the
      Knowledge Graph to answer questions. If the
     answer includes entities, always respond using
     the entity label rather than entity ID (if
     applicable).

Knowledge Graph:
Edges: [
(Andhra Pradesh, language used, Telugu),
(Andhra Pradesh, language used, Marathi),
(Andhra Pradesh, language used, Odia),
(Guatemala, capital of, Federal Republic of Central
     America),
(Guatemala, diplomatic relation, European Union),
(Brunei, member of, World Trade Organization),
(Brunei, member of, International Hydrographic
     Organization),
(South Korea, diplomatic relation, Ukraine),
(South Korea, diplomatic relation, Colombia),
(South Korea, member of, G20)
]
```

B.2 Structured JSON

```
Your job is to answer questions using the following
     knowledge graph. The knowledge graph is
     presented as a list of directed edges of the
     form (subject, relation, object). You must rely
      exclusively on the information presented in
     the Knowledge Graph to answer questions. If the
      answer includes entities, always respond using
      the entity label rather than entity ID (if
     applicable).
Knowledge Graph:
{
    "Andhra Pradesh": {
        "language used": [
            "Telugu",
            "Marathi",
            "Odia"
        ]
    },
    "Guatemala": {
        "capital of": [
            "Federal Republic of Central America"
        ],
        "diplomatic relation": [
            "European Union"
        ]
    },
    "Brunei": {
        "member of": [
            "World Trade Organization",
            "International Hydrographic Organization
                "
        ]
    },
    "South Korea": {
        "diplomatic relation": [
            "Ukraine",
            "Colombia"
        ],
        "member of": [
            "G20"
        ]
    }
}
```

B.3 Structured YAML

```
Your job is to answer questions using the following
     knowledge graph. The knowledge graph is
     presented as a structured YAML format. Each
     entity is a key, and the value is a dictionary
     of relations and objects.. You must rely
     exclusively on the information presented in the
      Knowledge Graph to answer questions. If the
     answer includes entities, always respond using
     the entity label rather than entity ID (if
     applicable).

Knowledge Graph:
Andhra Pradesh:
  language used:
    - Telugu
    - Marathi
    - Odia

Guatemala:
  capital of:
    - Federal Republic of Central America
  diplomatic relation:
    - European Union
```

Task	Instances	Seed Entities	Max Edges	Sample Radius	Min Degree
Triple Retrieval	100	10	200	1	1
Shortest Path	100	10	200	1	1
Highest Degree Node	100	10	200	1	1
Agg by Relation	100	1	200	2	2
Agg of Neighbor Properties	100	1	200	2	2

Table 4: Sampling parameters used for each task. Min degree filter is applied before the max edge constraint.

```
Brunei:
  member of:
    - World Trade Organization
    - International Hydrographic Organization

South Korea:
  diplomatic relation:
    - Ukraine
    - Colombia
  member of:
    - G20
```

B.4 RDF Turtle

```
Your job is to answer questions using the following
     knowledge graph. The knowledge graph is
     presented as RDF Turtle format using node IDs
     and relation IDs.. You must rely exclusively on
      the information presented in the Knowledge
     Graph to answer questions. If the answer
     includes entities, always respond using the
     entity label rather than entity ID (if
     applicable).

Knowledge Graph:
@prefix ex: <http://example.org/countries#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-
     syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema
     #> .

ex:R1 a rdf:Property ;
    rdfs:label "language used" .

ex:R2 a rdf:Property ;
    rdfs:label "member of" .

ex:R3 a rdf:Property ;
    rdfs:label "diplomatic relation" .

ex:R4 a rdf:Property ;
    rdfs:label "capital of" .

ex:1 a ex:Country ;
    rdfs:label "Andhra Pradesh" ;
    ex:R1 ex:101, ex:102, ex:103 .

ex:101 a ex:Language ;
    rdfs:label "Telugu" .

ex:102 a ex:Language ;
    rdfs:label "Marathi" .

ex:103 a ex:Language ;
    rdfs:label "Odia" .

ex:2 a ex:Country ;
    rdfs:label "Guatemala" ;
    ex:R4 ex:201 ;
    ex:R3 ex:202 .

ex:201 a ex:Country ;
    rdfs:label "Federal Republic of Central America"
         .

ex:202 a ex:Organization ;
    rdfs:label "European Union" .

ex:3 a ex:Country ;
    rdfs:label "Brunei" ;
    ex:R2 ex:301, ex:302 .

ex:301 a ex:Organization ;
    rdfs:label "World Trade Organization" .

ex:302 a ex:Organization ;
    rdfs:label "International Hydrographic
         Organization" .

ex:4 a ex:Country ;
    rdfs:label "South Korea" ;
    ex:R3 ex:401, ex:402 ;
    ex:R2 ex:403 .

ex:401 a ex:Country ;
    rdfs:label "Ukraine" .

ex:402 a ex:Country ;
    rdfs:label "Colombia" .

ex:403 a ex:Organization ;
    rdfs:label "G20" .
```

B.5 JSON-LD

```
"Your job is to answer questions using the following
     knowledge graph. The knowledge graph is
     presented as JSON-LD format using node IDs and
     relation IDs.. You must rely exclusively on the
      information presented in the Knowledge Graph
     to answer questions. If the answer includes
     entities, always respond using the entity label
      rather than entity ID (if applicable).

Knowledge Graph:
{
  "@context": {
    "@context": {
      "ex": "http://example.org/countries#",
      "label": "rdfs:label",
      "rdf": "http://www.w3.org/1999/02/22-rdf-
           syntax-ns#",
      "rdfs": "http://www.w3.org/2000/01/rdf-schema
           #",
      "type": "@type"
    }
  },
  "@graph": [
    {
      "@id": "ex:R1",
      "label": "language used",
      "type": "rdf:Property"
    },
    {
      "@id": "ex:R3",
      "label": "diplomatic relation",
      "type": "rdf:Property"
    },
    {
      "@id": "ex:R4",
      "label": "capital of",
      "type": "rdf:Property"
    },
    {
      "@id": "ex:1",
      "type": "ex:Country",
      "label": "Andhra Pradesh",
      "ex:R1": [
        { "@id": "ex:101" },
```

```
        { "@id": "ex:102" },
        { "@id": "ex:103" }
      ]
    },
    {
      "@id": "ex:101",
      "type": "ex:Language",
      "label": "Telugu"
    },
    {
      "@id": "ex:102",
      "type": "ex:Language",
      "label": "Marathi"
    },
    {
      "@id": "ex:103",
      "type": "ex:Language",
      "label": "Odia"
    },
    {
      "@id": "ex:2",
      "type": "ex:Country",
      "label": "Guatemala",
      "ex:R4": { "@id": "ex:201" },
      "ex:R3": { "@id": "ex:202" }
    },
    {
      "@id": "ex:201",
      "type": "ex:Country",
      "label": "Federal Republic of Central America"
    },
    {
      "@id": "ex:202",
      "type": "ex:Organization",
      "label": "European Union"
    }
  ]
}
```


C Full Results

We present the full results and data over the following pages.

C.1 Heatmap Results

Figure 9 presents the heatmap data for all models.

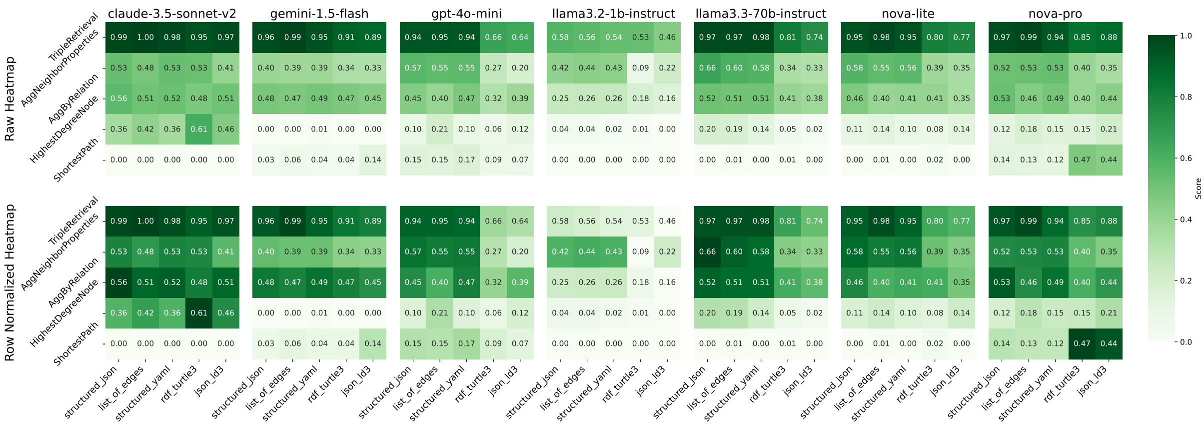

Figure 9: Heatmaps of the performance of various models. Each heatmap shows tasks as rows and textualize functions as columns. The top row of the grid shows the heatmap colors as globally weighted from [0.0-1.0]. The bottom grid shows heatmap colors normalized for each task [task minimum-task maximum].

Table 5: Full Results Summary by Format and Model

Format	Model	Agg by Relation	Agg Neighbor Properties	Highest Degree	Shortest Path	Triple Retrieval	Overall
List of Edges	**claude-3.5-sonnet-v2**	0.490	0.440	0.370	0.000	1.000	0.460
	+pseudo	0.530	0.530	0.460	0.000	1.000	0.504
	gemini-1.5-flash	0.520	0.430	0.000	0.080	1.000	0.406
	+pseudo	0.420	0.340	0.000	0.040	0.990	0.358
	gpt-4o-mini	0.400	0.520	0.140	0.150	0.980	0.438
	+pseudo	0.400	0.580	0.270	0.140	0.910	0.460
	llama3.2-1b-instruct	0.250	0.430	0.050	0.000	0.560	0.258
	+pseudo	0.260	0.450	0.030	0.000	0.560	0.260
	llama3.3-70b-instruct	0.540	0.590	0.200	0.000	0.970	0.460
	+pseudo	0.470	0.620	0.180	0.010	0.980	0.452
	nova-lite	0.390	0.560	0.120	0.000	0.990	0.412
	+pseudo	0.410	0.540	0.160	0.010	0.980	0.420
	nova-pro	0.460	0.520	0.130	0.150	0.990	0.450
	+pseudo	0.450	0.530	0.230	0.110	0.990	0.462
	Format Overall	0.436	0.499	0.144	0.054	0.927	0.412
	+pseudo	0.420	0.513	0.190	0.044	0.916	0.417
Structured JSON	**claude-3.5-sonnet-v2**	0.550	0.520	0.330	0.000	0.990	0.478
	+pseudo	0.560	0.540	0.390	0.000	0.990	0.496
	gemini-1.5-flash	0.500	0.410	0.000	0.060	0.960	0.386
	+pseudo	0.470	0.380	0.000	0.010	0.970	0.366
	gpt-4o-mini	0.490	0.560	0.100	0.140	0.950	0.448
	+pseudo	0.420	0.580	0.100	0.160	0.940	0.440
	llama3.2-1b-instruct	0.260	0.440	0.040	0.000	0.540	0.256
	+pseudo	0.240	0.410	0.040	0.000	0.620	0.262
	llama3.3-70b-instruct	0.530	0.600	0.190	0.000	0.980	0.460
	+pseudo	0.500	0.710	0.200	0.000	0.970	0.476
	nova-lite	0.490	0.590	0.100	0.000	0.960	0.428
	+pseudo	0.440	0.580	0.120	0.000	0.950	0.418
	nova-pro	0.550	0.580	0.100	0.170	0.970	0.474
	+pseudo	0.500	0.450	0.140	0.110	0.970	0.434
	Format Overall	0.481	0.529	0.123	0.053	0.907	0.419
	+pseudo	0.447	0.521	0.141	0.040	0.916	0.413
Structured YAML	**claude-3.5-sonnet-v2**	0.500	0.600	0.320	0.000	0.990	0.482
	+pseudo	0.540	0.460	0.410	0.000	0.980	0.478
	gemini-1.5-flash	0.490	0.400	0.010	0.090	0.950	0.388
	+pseudo	0.500	0.370	0.000	0.000	0.950	0.364
	gpt-4o-mini	0.490	0.540	0.070	0.200	0.940	0.448
	+pseudo	0.460	0.570	0.120	0.140	0.940	0.446
	llama3.2-1b-instruct	0.290	0.430	0.030	0.000	0.560	0.262
	+pseudo	0.220	0.430	0.010	0.000	0.510	0.234
	llama3.3-70b-instruct	0.510	0.550	0.150	0.000	0.960	0.434
	+pseudo	0.500	0.620	0.120	0.000	1.000	0.448
	nova-lite	0.410	0.540	0.100	0.000	0.950	0.400

Continued on next page

Table 5 – Continued from previous page

Format	Model	Agg by Relation	Agg Neighbor Properties	Highest Degree	Shortest Path	Triple Retrieval	Overall
	+pseudo	0.410	0.580	0.100	0.000	0.950	0.408
	nova-pro	0.510	0.520	0.070	0.110	0.930	0.428
	+pseudo	0.480	0.540	0.230	0.140	0.940	0.466
	Format Overall	0.457	0.511	0.107	0.057	0.897	0.406
	+pseudo	0.444	0.510	0.141	0.040	0.896	0.406
RDF Turtle	**claude-3.5-sonnet-v2**	0.450	0.510	0.590	0.000	0.940	0.498
	+pseudo	0.510	0.540	0.640	0.000	0.970	0.532
	gemini-1.5-flash	0.460	0.330	0.000	0.060	0.890	0.348
	+pseudo	0.490	0.340	0.000	0.030	0.930	0.358
	gpt-4o-mini	0.310	0.270	0.070	0.140	0.690	0.296
	+pseudo	0.330	0.260	0.050	0.030	0.630	0.260
	llama3.2-1b-instruct	0.190	0.160	0.010	0.000	0.530	0.178
	+pseudo	0.180	0.020	0.010	0.000	0.530	0.148
	llama3.3-70b-instruct	0.400	0.380	0.050	0.010	0.790	0.326
	+pseudo	0.410	0.310	0.050	0.000	0.840	0.322
	nova-lite	0.410	0.400	0.060	0.030	0.820	0.344
	+pseudo	0.400	0.370	0.110	0.010	0.780	0.334
	nova-pro	0.330	0.440	0.100	0.500	0.880	0.450
	+pseudo	0.470	0.360	0.210	0.440	0.830	0.462
	Format Overall	0.364	0.356	0.126	0.106	0.791	0.349
	+pseudo	0.399	0.314	0.153	0.073	0.787	0.345
JSON-LD	**claude-3.5-sonnet-v2**	0.500	0.370	0.370	0.000	0.980	0.444
	+pseudo	0.520	0.450	0.550	0.000	0.970	0.498
	gemini-1.5-flash	0.460	0.330	0.000	0.230	0.910	0.386
	+pseudo	0.440	0.330	0.000	0.050	0.870	0.338
	gpt-4o-mini	0.380	0.210	0.130	0.090	0.670	0.296
	+pseudo	0.390	0.180	0.110	0.040	0.600	0.264
	llama3.2-1b-instruct	0.170	0.290	0.000	0.000	0.460	0.184
	+pseudo	0.150	0.150	0.000	0.000	0.450	0.150
	llama3.3-70b-instruct	0.350	0.320	0.020	0.000	0.760	0.290
	+pseudo	0.410	0.330	0.020	0.000	0.730	0.298
	nova-lite	0.330	0.360	0.110	0.000	0.820	0.324
	+pseudo	0.380	0.340	0.170	0.000	0.730	0.324
	nova-pro	0.440	0.380	0.210	0.420	0.900	0.470
	+pseudo	0.440	0.320	0.200	0.470	0.850	0.456
	Format Overall	0.376	0.323	0.120	0.106	0.786	0.342
	+pseudo	0.390	0.300	0.150	0.080	0.743	0.333
All Formats	**claude-3.5-sonnet-v2**	0.498	0.488	0.396	0.000	0.980	0.472
	+pseudo	0.532	0.504	0.490	0.000	0.982	0.502
	gemini-1.5-flash	0.486	0.380	0.002	0.104	0.942	0.383
	+pseudo	0.464	0.352	0.000	0.026	0.942	0.357
	gpt-4o-mini	0.414	0.420	0.102	0.144	0.846	0.385
	+pseudo	0.400	0.434	0.130	0.102	0.804	0.374
	llama3.2-1b-instruct	0.232	0.350	0.026	0.000	0.530	0.228

Continued on next page

Table 5 – Continued from previous page

Format	Model	Agg by Relation	Agg Neighbor Properties	Highest Degree	Shortest Path	Triple Retrieval	Overall
	+pseudo	0.210	0.292	0.018	0.000	0.534	0.211
	llama3.3-70b-instruct	0.466	0.488	0.122	0.002	0.892	0.394
	+pseudo	0.458	0.518	0.114	0.002	0.904	0.399
	nova-lite	0.406	0.490	0.098	0.006	0.908	0.382
	+pseudo	0.408	0.482	0.132	0.004	0.878	0.381
	nova-pro	0.458	0.488	0.122	0.270	0.934	0.454
	+pseudo	0.468	0.440	0.202	0.254	0.916	0.456
	Overall Score	0.423	0.443	0.124	0.075	0.862	0.385
	+pseudo	0.420	0.432	0.155	0.055	0.851	0.383