A Systematic Exploration of Knowledge Graph Alignment with Large Language Models in Retrieval Augmented Generation

Shiyu Tian[1], Shuyue Xing[1], Xingrui Li[1], Yangyang Luo[1], Caixia Yuan[1], Wei Chen[2], Huixing Jiang[2*], Xiaojie Wang[1*]

[1]Beijing University of Posts and Telecommunications
[2]LI Auto Inc.
{tiansy, xsy84160158, liangy298, luoyangyang, yuancx, xjwang}@bupt.edu.cn
{jianghuixing, chenwei10}@lixiang.com

Abstract

Retrieval Augmented Generation (RAG) with Knowledge Graphs (KGs) is an effective way to enhance Large Language Models (LLMs). Due to the natural discrepancy between structured KGs and sequential LLMs, KGs must be linearized to text before being inputted into LLMs, leading to the problem of KG Alignment with LLMs (KGA). However, recent KG+RAG methods only consider KGA as a simple step without comprehensive and in-depth explorations, leaving three essential problems unclear: (1) What are the factors and their effects in KGA? (2) How do LLMs understand KGs? (3) How to improve KG+RAG by KGA? To fill this gap, we conduct systematic explorations on KGA, where we first define the problem of KGA and subdivide it into the graph transformation phase (graph-to-graph) and the linearization phase (graph-to-text). In the graph transformation phase, we study graph features at the node, edge, and full graph levels from low to high granularity. In the linearization phase, we study factors on formats, orders, and templates from structural to token levels. We conduct substantial experiments on 15 typical LLMs and three common datasets. Our main findings include: (1) The centrality of the KG affects the final generation; formats have the greatest impact on KGA; orders are model-dependent, without an optimal order adapting for all models; the templates with special token separators are better. (2) LLMs understand KGs by a unique mechanism, different from processing natural sentences, and separators play an important role. (3) We achieved 7.3% average performance improvements on four common LLMs on the KGQA task by combining the optimal factors to enhance KGA.

1 Introduction

Large language models (LLMs) (Zhao et al. 2023b) have demonstrated superior capabilities and generalizability across various tasks (Bang et al. 2023; Ye et al. 2023) and are regarded as a step toward realizing Artificial General Intelligence (AGI) (Bubeck et al. 2023; Xi et al. 2023). However, recent studies reveal that LLMs lack expertise and up-to-date knowledge (Schick et al. 2023; Peng et al. 2023), suffer from hallucination (Rawte, Sheth, and Das 2023), especially in knowledge-intensive tasks (Bang et al. 2023).

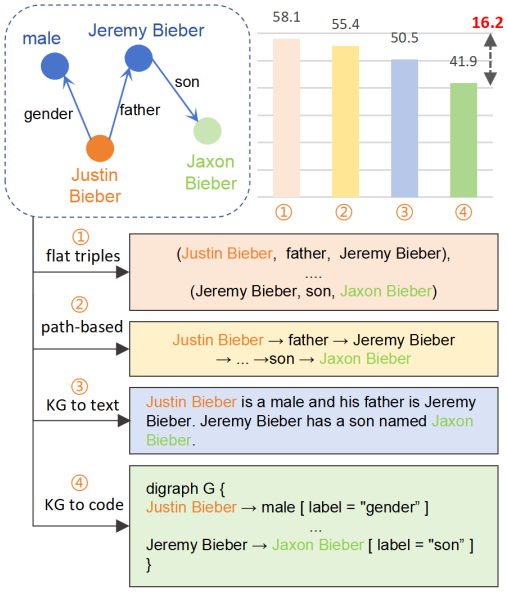

Figure 1: Examples of four KG linearization formats, one type of factor in KGA, and their average performances on the GraphextQA test set over 14 models indicate the significance of KGA in RAG.

To address these issues, the current mainstream approach builds Retrieval-Augmented Generation (RAG) systems that enable LLMs to access external knowledge to improve generation quality (Gao et al. 2024).

Knowledge Graphs (KGs) are one of the most commonly used external knowledge sources and have unique advantages: (1) They are structured and easy to retrieve (Ji et al. 2022). (2) They represent the core semantics of knowledge in the form of triple, which is aligned with human cognition (Zhao et al. 2022; Lin et al. 2024). (3) They support symbolic reasoning, which can give human-readable reasoning paths for interoperability (Abu-Salih 2021). Thus, incorporating KGs into LLMs by RAG (KG+RAG) is a promising way to elevate LLMs to the next level (Pan et al. 2024).

Since there is a natural discrepancy between structural KGs and sequential LLMs, KGs must be linearized into text to be fed into LLMs, which raises a unique problem:

*Corresponding authors.

KG Alignment with LLMs (KGA), i.e., how to input KGs into LLM. Although recent KG+RAG studies have explored some factors in KGA, such as formats (Zhang et al. 2024; Wu et al. 2023), orders (Li et al. 2023b; LUO et al. 2024a), and templates (Jiang et al. 2023b; Wen, Wang, and Sun 2024), they only consider KGA as a simple step without systematic and in-depth studies. However, we find KGA is critical and indispensable, directly affecting the efficiency of utilizing the retrieved knowledge and the quality of the final generation (see Figure 1 as an example). Moreover, a deficient KGA can even corrupt the original capabilities of LLMs, leading to knowledge conflict and worse responses (Longpre et al. 2021; Zhou et al. 2023).

Although previous works have extensively studied the KG+RAG, three key questions remain unclear: (1) What are the influencing factors and their effects in KGA? (2) How do LLMs understand KGs, and what is the underlying mechanism? (3) How to use KGA to improve KG+RAG?

To address the above problems, we systematically explore KGA. We subdivide KGA into two phases (see Figure 2): (1) the graph transformation phase, where we perform graph-to-graph transformation by modifying the retrieved sub-KG at the graph level; (2) the linearization phase, in which the sub-KG is linearized to achieve graph-to-text conversion. Then, we examine the influencing factors at each phase. In the graph transformation phase, we explore 81 graph features, according to Graph Theory (West et al. 2001), from node, edge, and full graph granularities to study their impact on the final generation and identify the most crucial features by feature analysis methods. In the linearization phase, we investigate the performance of 13 formats, 13 orders, and 14 templates across 15 models, 3 datasets, and 2 enhancement tricks (few-shot and fine-tuning), providing systematic and generalizable results. We completed more than 3,500 experiments covering almost all influencing factors in KGA and identified the effects of each kind of factor. Based on these results, we draw conclusions from the perspective of interpretability and usability. For interpretability, we uncover a distinct mechanism for LLMs to process KGs, which varies from processing general sentences. For usability, we find and utilize the combinability between different factors to enhance the KGA, which can effectively improve the performance of KG+RAG.

Our contributions are summarized as follows:

- We first define and emphasize the problem of KG Alignment with LLMs (KGA), which is a critical and indispensable part of KG+RAG, and conduct systematic experiments to fill this gap.
- We identify key factors and their effects in KGA, including (1) the centrality of the sub-KG affects the final generation, (2) formats are the most influential factor in KGA, (3) orders are model-dependent, without one that is optimal for all models, and (4) templates with special token separators work better.
- We find LLMs process KGs through a unique mechanism (circuits) by performing mechanistic interpretable analysis, where separators play an important role.
- Based on our findings, enhancing the KGA can effectively improve the average performance of four typical LLMs on the KG+RAG task by 7.3%.

2 Related Work

2.1 KG+RAG with LLMs

Based on the way of fusing KGs, recent KG+RAG studies can be categorized into two types: (1) explicit fusion, where KGs are converted to text and concatenated into prompt; (2) implicit fusion, where additional KG encoding modules are added to the LLMs so that the KGs can be directly inputted into the LLMs.

Most of the current methods are explicit fusion, where they usually focus on designing the retrieval methods (Baek, Aji, and Saffari 2023; Li et al. 2023a; Jiang et al. 2023b; LUO et al. 2024b; Zhang et al. 2024; Xiong, Bao, and Zhao 2024) and the overall pipeline (or LLM agents) (Li et al. 2023b; Wu et al. 2023; Sun et al. 2024; Wen, Wang, and Sun 2024; Sanmartin 2024; Dong et al. 2024). Although these methods achieve good results, they ignore the importance of KGA without considering the structural characteristics of KG itself, resulting in non-optimum performance.

Some studies try to integrate KGs directly into LLMs by designing additional KG encoding modules. Such as GNP (Tian et al. 2024b), KG-Adapter (Tian et al. 2024a) and G-Retriever (He et al. 2024). They design different KG encoders and use LoRa or other methods to fine-tune the LLM and KG encoders simultaneously. Although these methods bypass the KGA process by utilizing KG encoders to input the KG directly, they need training LLMs and KG encoders, resulting in high costs. Furthermore, since the KG representations and the textual representations exist in different vector spaces, suffering from the heterogeneous representation problem (Lin et al. 2019; Sun et al. 2022), causing them less effective than the explicit fusion methods.

2.2 Mechanistic Interpretability

Mechanistic interpretability is a sub-field of LLMs interpretability that understands language models by investigating individual neurons and especially their connections in terms of circuits (Zhao et al. 2023a). A foundation work conceptualizes the operation of transformers in a new way, providing a mathematical framework for transformer circuits, using that they find "induction heads" that can explain in-context learning in small models (Elhage et al. 2021). Another work (Olsson et al. 2022) validates if "induction heads" still hold true on much more complex state-of-art models. Then, many of the following works try to discover more circuits for different tasks, like "indirect object identification" (Wang et al. 2023) and "greater-than" (Hanna, Liu, and Variengien 2023). Besides, some works are trying to explain different behaviors in LLMs. For example, Chughtai, Chan, and Nanda (2023) study the universality hypothesis by presenting a novel algorithm, Nanda et al. (2023) investigate the recently-discovered phenomenon of "grokking", Meng et al. (2022) locate factual knowledge in LMs, and Todd et al. (2024) discover task vectors. Recently, new theories and analytical tools have been proposed that greatly advance the field (Nanda and Bloom 2022; Ferrando and Voita

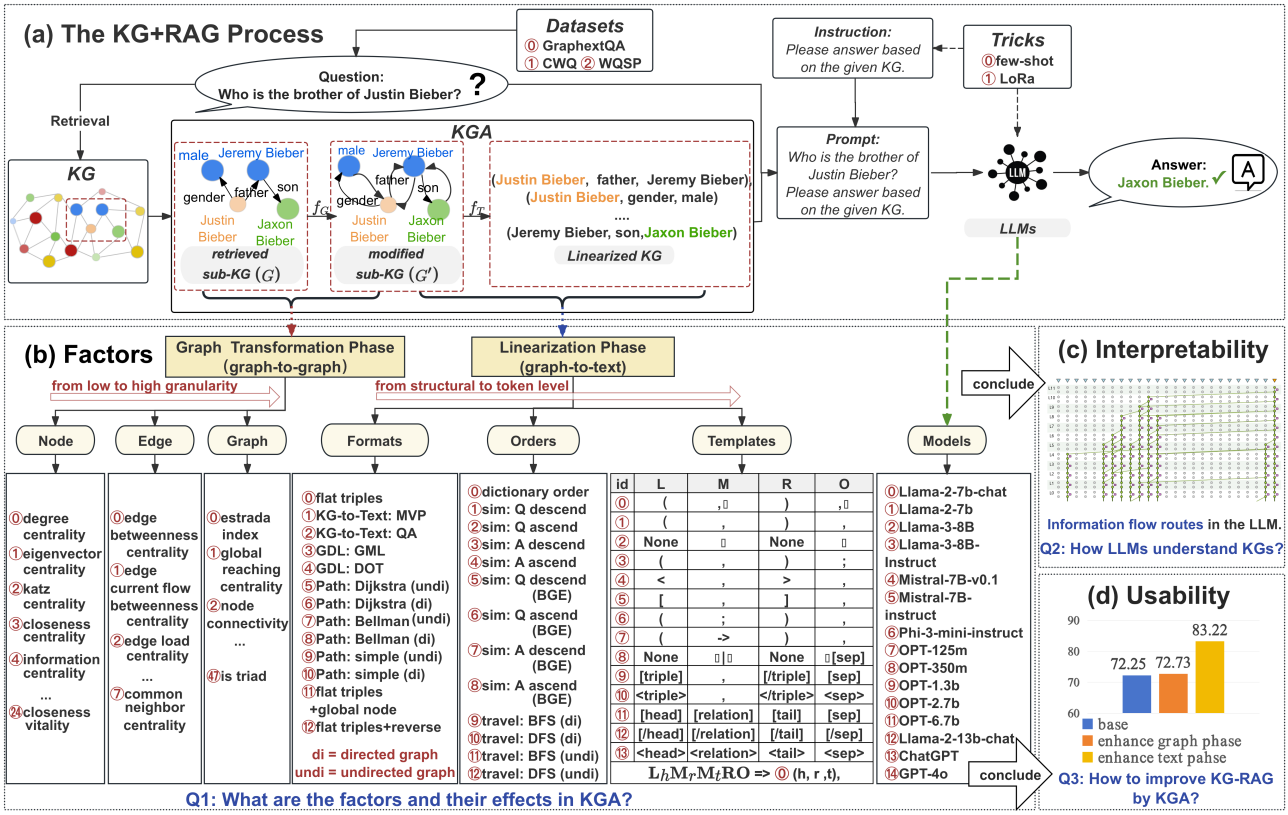

Figure 2: (a) KGA in the KG+RAG process. (b) Factors in KGA and their IDs. (c) Interpretability: how LLMs understand KGs. (d) Usability: improving KG+RAG by KGA.

2024; Tufanov et al. 2024). Although many phenomena have been explained, there is still a long way to go in fully understanding LLMs. There are no interpretability studies for how LLMs understand KGs. Hence, we take the first step to fill this gap by analyzing the information flow routes of KGs in LLMs.

3 KG Alignment with LLMs

3.1 Definitions

First, we will define Knowledge Graph (KG) and KG Alignment with LLMs (KGA).

Definition 1. Knowledge Graph. Given an entity set $E = \{e_1, ..., e_N\}$ and a relation set $R = \{r_1, ..., r_M\}$. A knowledge graph is a graph constructed by E and R: $G = \{E, R\}$. If there is a directed edge r_{ij} from e_i to e_j, then it can construct a triple: $T_m = T_{ij} = (e_i, r_{ij}, e_j), m \leq M$. KG can also be defined as a set of triples $G = \{T_1, T_2, ..., T_k, ..., T_M\}$, where each T_k is a triple.

Definition 2. KGA. KGA is the process of converting structural KGs into sequential texts while maximizing model performance. We subdivide it into the graph transformation phase and the linearization phase. Previous studies only examined the linearization without considering that LLMs may also have preferences for certain graph features, i.e., some graph features of the sub-KG will affect the final generation. Therefore, we introduce the graph transformation phase to study the impact of graph features and how to better align sub-KG with LLMs by graph-level modifications.

The first phase is to modify the graph at the graph level:

$$G' = f_G(G) \tag{1}$$

where f_G are graph operations, such as adding reverse edges or changing the graph type. In the second phase, the updated graph G' is converted into text:

$$Text = f_T(G') = f_T(\bigcup_{k}^{M}(T_k)) = f_T(\bigcup_{i,j}(e_i, r_{ij}, e_j)) \tag{2}$$

$$= f_f(f_o(\bigcup_{i,j} f_t(e_i, r_{ij}, e_j))) \tag{3}$$

where $f_T = \{f_f, f_o, f_t\}$ is a set of methods that transforms triples into texts, f_f, f_o, f_t are three specific methods in formats, orders, and templates. So, the target of KGA can be defined as below:

$$\arg\max_{f_G, f_T} log P_\theta(y|Q, f_T(f_G(G_q))) \tag{4}$$

where Q is a user question, y is the true answer, G_q is a sub-KG related to Q, θ is a frozen LLM, and the target of KGA is to find two best operations f_G and f_T to maximize the probability of generating the correct answer.

3.2 Studied Factors

After defining the KGA, we examine the influencing factors and their effects at each phase. See Figure 2 for our studied factors.

Graph Transformation Phase We aim to examine how graph features impact the final generated results, determine their relative importance, and how to use them to guide the performance improvement of KG+RAG.

Based on Graph Theory (West et al. 2001), a graph has features from three dimensions: node, edge, and full graph. Accordingly, we selected 25 node features, 8 edge features, and 48 graph features to study. We only list some typical features here (see Appendix for details of all features).

- Node Features (25): degree centrality, laplacian centrality, average neighbor degree, eccentricity, pagerank, etc.
- Edge Features (8): betweenness centrality, preferential attachment, jaccard coefficient, etc.
- Graph Features (48): vertex cover size, minimum cut, global reaching centrality, s-metric, non-randomness, dominating set num, etc.

Linearization Phase In the linearization phase, we focus on three kinds of factors from structural level to token level: (1) formats, (2) orders, and (3) templates. The formats determine the overall structure of the linearized text, and the orders determine the positional relationships between the triples, while the templates determine the specific representation of each triple. Thus, these three categories of factors encompass the complete process of KGA in the linearization phase.

The formats determine what form of text the sub-KG will be transformed into (see Figure 1 as an example). Flat triples are the most common way to convert KGs to text, which fills the entities and relations of the triples into a pre-defined template. The path-based formats fill the retrieved paths (i.e., multi-hop paths from entities in the question to entities in the possible answers) into path-style templates, and we explore three algorithms to retrieve paths: simple path, Dijkstra short path, and Bellman short path. KG-to-Text is a unique format that designs and trains models to convert KGs to natural language sentences, where we use MVP (Tang et al. 2023), the SOTA KG-to-Text method. Graph description language (GDL) is a special language that can convert a graph into code-like formal text, and we test the two most commonly used GDLs: DOT (Gansner, Koutsofios, and North 2006), and GML (Brandes et al. 2013).

Orders decide the positional relations between different triples, often used in RAG systems' post-retrieval process (i.e., re-ranking). We study two kinds of order methods: semantic similarity-based and graph travel-based. For the semantic similarity-based methods, we try two sentence embedding models (all-mpnet-base-v2 [1] and BGE-M3 (Chen et al. 2024)) and four rank strategies (ascending or descending order based on similarity to the question or answer), totaling eight combinations. For the graph travel-based methods, we test two graph traversal algorithms, BFS and DFS, combined with directed and undirected graphs, resulting in four combinations.

Templates determine the specific representation of each triple transformed into text. Basically, a template is a set of separators, and we define them as left separator (L), right separator (R), middle separator (M), and outer separator (O) based on their position in the template. Formally, a template is $\mathbf{L}h\mathbf{M}r\mathbf{M}t\mathbf{RO}$. where h, r, t are the head entity, relation, and tail entity from a triple, $\mathbf{L}, \mathbf{M}, \mathbf{R}, \mathbf{O}$ are separators in the template, which could be various punctuation marks or special tokens. We explore 14 templates used in previous methods.

4 Experimental Setup

4.1 Datasets

We use three commonly used KGQA datasets: GraphextQA (Shen et al. 2023), ComplexWebQuestions(CWQ) (Talmor and Berant 2018) and WebQuestionsSP(WQSP) (Yih et al. 2016). More details are in the Appendix.

4.2 Models

We use 15 models total from 125M to 175B+, including: Llama-2-7b-base, Llama-2-7b-chat, Llama-2-13b-chat, Llama-3-8b-base, Llama-3-8b-instruct (Touvron et al. 2023; Meta 2024); Mistral-7b-v0.1, Mistral-7b-instruct (Jiang et al. 2023a); Phi-3-mini-128k-instruct (Abdin et al. 2024); opt-125m, opt-350m, opt-1.3b, opt-2.7b, opt-6.7b (Zhang et al. 2022); ChatGPT, GPT-4o (OpenAI 2021).

4.3 Enhancement Tricks

We test the performance of different factors under few-shot (Brown et al. 2020) and fine-tuning by LoRa (Hu et al. 2022) as two enhancement tricks for generalizability.

4.4 Evaluation Metrics

We follow previous work (Jiang et al. 2023b) to use Hits@1 to evaluate the correctness of the prediction, which assesses whether the predicted answer is correct or not, suitable for evaluating LLMs (Adlakha et al. 2024).

Rank-biased Overlap (RBO) (Webber, Moffat, and Zobel 2010) measures the similarity between incomplete rankings and weight high ranks more heavily than low ranks. We use it to measure the consistency of different factors across settings to indicate generalizability.

4.5 Implementation Details

For the graph transformation phase, we use the Python library NetworkX (Hagberg, Schult, and Swart 2008) to compute the graph features[2], and use seven feature analysis methods: Permutation Importance, feature importance in Random Forests and XGBoost, Pearson Correlation, Recursive Feature Elimination, PCA, and ANOVA to calculate the importance scores for each graph feature.

[1] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

[2] https://networkx.org/documentation/stable/reference/algorithms/index.html

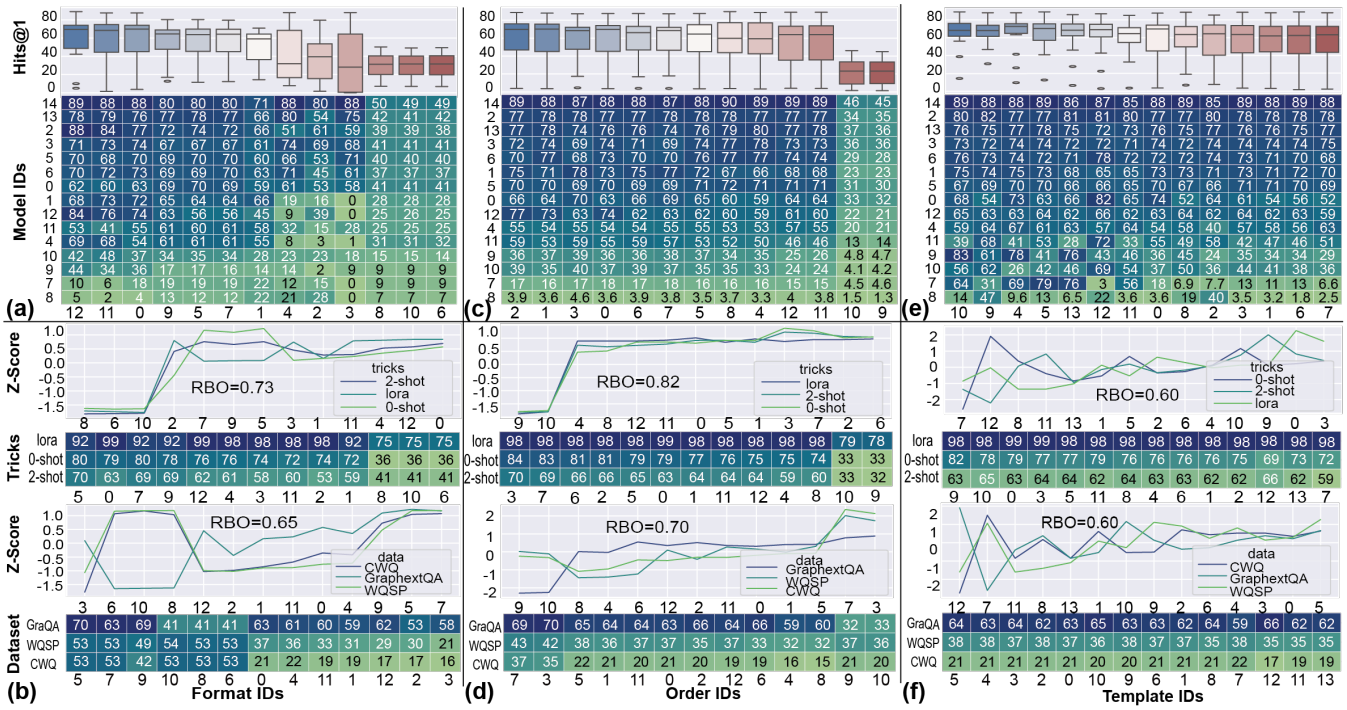

Figure 3: Effects of factors in the linearization phase. (a) 13 formats on 15 models. (b) 13 formats on diverse datasets and tricks. (c) 13 orders on 15 models. (d) 13 orders on diverse datasets and tricks. (e) 14 templates on 15 models. (f) 14 templates on diverse datasets and tricks. The dataset of (a)(c)(e) is the GraphextQA test set, and the model of (b)(d)(f) is Llama-2-7b-chat. Z-score is standardized Hits@1. The mappings of IDs are in Figure 2(b).

For the linearization phase, we use vLLM (Kwon et al. 2023) with a temperature of 0 and a top-p of 0.25 for inference, a greedy decoding strategy, and a zero-shot prompt for most experiments. We extract the model generation's first and last lines to compute the metrics. We conduct case studies of interpretability analysis using Information Flow Routes (Ferrando and Voita 2024), one of the mechanistic interpretability tools for explaining LLMs' behavior. For ChatGPT and GPT-4o, we random sample 10% data to test.

See the Appendix for more details.

5 Results and Discussions

5.1 Effects of Factors

Graph Features Table 1 shows the graph features with the top-10 importance scores averaged over seven feature analysis methods. We find that the most important features are mostly associated with centrality, which measures the importance (or how "central" a node is in the graph) of various nodes in a graph. This may be because nodes with higher centrality appear in more triples and recur more often in the linearized text. Based on that, we design two methods to improve the centrality of a sub-KG: adding a virtual global node and adding reverse edges. Surprisingly, both methods improve the final performance on different models (in Table 2), which suggests that the graph features of sub-KG can affect the generation results. However, few existing KG-RAG methods have considered graph features and added them to the system design, so there is still a lot of potential for improvement in existing methods.

Graph Features	Avg	Std	Range
degree_centrality	42.2	13.8	69.1
betweenness_centrality	45.8	11.4	69.8
average_neighbor_degree	46.4	11.2	64.4
information_centrality	48.5	11.6	69.1
current_flow_betweenness_centrality	49.6	15.0	75.6
global_reaching_centrality	49.8	11.5	74.1
closeness_centrality	50.6	13.1	70.7
katz_centrality	52.3	13.5	64.2
eccentricity	52.3	16.1	90.9
non_randomness	52.8	11.7	73.7

Table 1: Top-10 graph features, avg, std, and range representing the mean, standard deviation, and range of their rankings among seven feature analysis methods.

Formats In Figure 3(a), the flat triple methods (id=12,11,0) achieve the best average performance since they keep all information from sub-KG. Path-based approaches using different path algorithms obtained similar scores (id=9,5,7), suggesting that these algorithms are not vital influencing factors, but the graph type (directed or undirected) displays a significant influence (id=8,10,6). KG-to-Text methods (id=1,2) are not as good as in previous

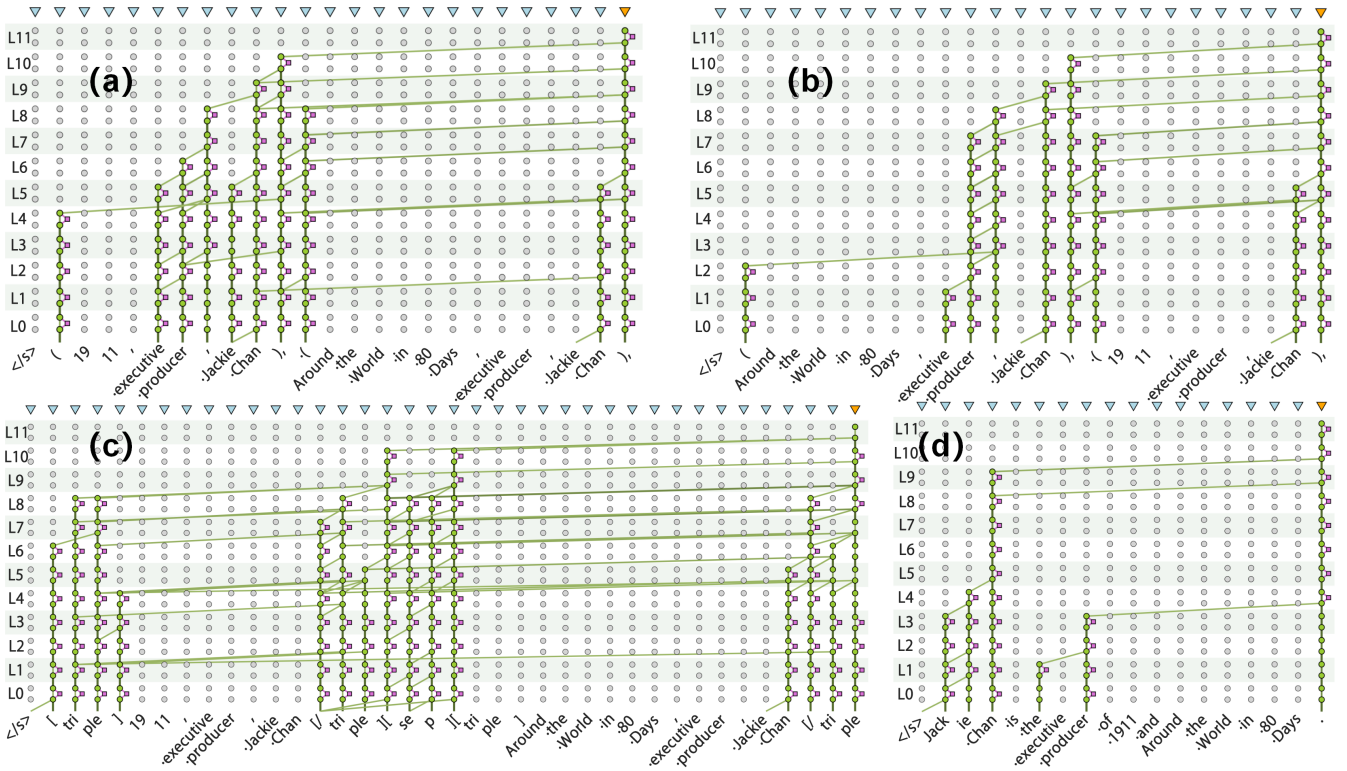

Figure 4: Case studies: information flow routes of opt-125m by different inputs to show how the model processes various factors. Nodes are representations in residual streams. Edges mean via residual (gray), attention (green), or FFN (pink).

work (Wu et al. 2023) because the KG-to-Text model (MVP) is often missing information or even generating non-existent information, implying that this approach needs training on specific datasets with low generalizability. GDL methods (id=4,3) are ineffective on small LLMs as they struggle to understand formal code but perform comparably to flat triples on larger LLMs such as Chatgpt and GPT-4o.

We compare the information flow of different formats to understand why flat triples perform better. As shown in Figure 4(a),(d), their information flow is quite distinct, and there is a solid connection between the separators (Figure 4(a)), which "guide" the direction of information flow.

For generalizability, the average RBO on three datasets is 0.65, meaning an optimal method has a roughly 65% probability of keeping optimal on another dataset, and the RBO of different tricks is 0.73, suggesting the linearization formats are generalizable across datasets and tricks (Figure 3(b)).

Orders Orders are strongly related to models' preferences without an optimal order adapts all models (Figure 3(c)) and show great generalizability across datasets and tricks (Figure 3(d)) with 0.82 and 0.7 RBO scores. However, previous studies usually rank triples in descending order based on similarity with questions, which is unsuitable for all LLMs. The orders are model-related due to similar information flow across different orders for the same model (Figure 4(a)(b)).

Templates There is a 10.7-point difference between the average score of the best and worst templates, suggesting that templates are an important consideration, which is overlooked in previous studies (Figure 3(e)). We find the templates with special tokens perform better (id=8∼13) since the information flow of these special tokens shows a BOS feature (Ferrando and Voita 2024; Cancedda 2024), where the residual stream is not activated in the middle layers (Figure 4(c)). This BOS feature benefits the model's understanding of the KG by summarizing previous information as a relayer without adding their own linguistic content since memories are stored in the middle layer FFN (Geva et al. 2021). The RBO on different datasets and tricks are both 0.6, showing decent generalizability (Figure 3(f)).

5.2 Interpretability: How LLMs understand KGs?

We find that separators play an important role in the LLMs' understanding of KG. To further investigate, we conduct quantitative and qualitative experiments. We use t-SNE (Van der Maaten and Hinton 2008) to search patterns in the information flows of the LLM and whether there are mechanisms associated with separator types.

From Figure 5, the separators (L, R, and M/O) are clustered in different groups (in most cases, M and O are the same token), while the different types of entities (Eh, Er, and Et) are not well separated, which suggests that the model has special information flow routes in processing these separators with completely different mechanisms than handling entities in triples.

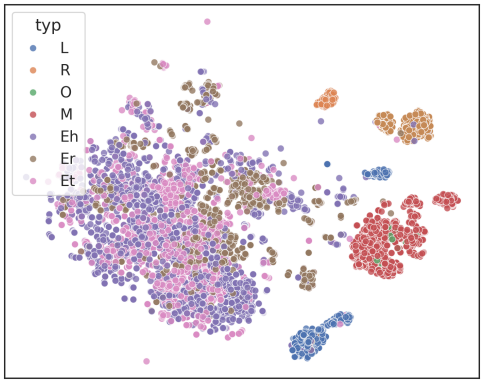

Figure 5: t-SNE of vectorized contribution weights on 100 random samples, colored by token types. L, R, O, and M are separators. Eh, Et and Er are entities and relations in triples.

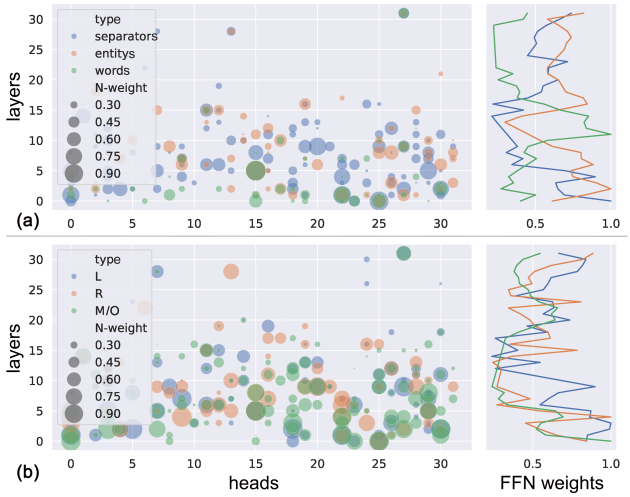

Figure 6: The normalized contribution weights of attention heads and FFN in each layer, colored by different input token types.

Following previous studies to find circuits (Ferrando and Voita 2024), we calculate the contribution weights of attention heads and FFN in each layer. In Figure 6(a), we find the separators, entities, and words activate different heads, suggesting that the model can distinguish and process them with different components. Notably, the FFN weights vary considerably, with entities and separators having low weights while words have high weights in the middle layer, indicating that the model utilizes different mechanisms for processing general sentences and KGs. Moreover, different types of separators also activate different heads, and some heads are activated by more than one type of separator (Figure 6(b)). This suggests that there are specialized heads within the model to handle different types of separators. We also note that these heads usually appear in the lower layers of the model, where the model mainly processes inputs and extracts information. The FFN block exhibits high activation in the bottom and top layers and low in the middle layer, which is consistent with the case in Figure 4(c), suggesting that some separators have the BOS feature for summarizing previous information rather than introducing new information.

5.3 Usability: How to use KGA to improve KG+RAG?

This section studies how to use KGA to improve the KG+RAG performance. According to our findings in the graph transformation phase, the centrality of sub-KG is strongly correlated to the generation performance. As shown in Table 2, improving the centrality by adding reverse edges and a virtual global node can improve the performance, showing the effectiveness of the graph transformation phase and graph features. However, many methods overlook this phase in the design of RAG systems.

We find the factors in the linearization phase are combinable, which means using two optimal methods simultaneously usually remains optimal (more in Appendix). Based on this property, we locate and combine the optimal method for each factor, which considerably improves the performance of the KG+RAG task (Table 2).

Combinations	Models			
	Llama-2-7b	Llama-3-8b	ChatGPT	GPT-4o
base	72.25	77.40	76.12	87.54
w/ reverse_edges	68.24	88.20	77.51	88.93
w/ global_node	72.73	84.01	78.55	87.54
w/ BF	72.25	77.40	79.58	87.89
w/ BO	78.41	78.20	79.93	89.62
w/ BT	78.24	82.32	77.85	88.93
w/ BF+BO	78.24	78.20	75.09	91.00
w/ BF+BT	78.24	82.32	79.58	88.58
w/ BO+BT	81.90	82.80	75.77	88.58
w/ BF+BO+BT	83.22	82.80	75.09	91.35

Table 2: Results of the four models with different combinations of factors on the GraphextQA test set. BF, BT, and BO are the best formats, templates, and orders for each model.

6 Conclusion

In this paper, we highlight the problem of KG Alignment with LLMs (KGA) and systematically explore the KGA as a critical part of KG+RAG. We subdivide it into graph transformation and linearization phases, exploring 81 graph features, 13 formats, 13 orders, and 14 templates across 15 models, 3 datasets, and 2 enhancement tricks, totaling more than 3,500 experiments. From these results, we identify key factors in KGA and their effects. For interpretability, we find a unique mechanism for LLMs to understand KGs. For usability, we show that optimizing KGA effectively improves the performance of KG+RAG by 7.3% on average. We believe that our work not only provides an in-depth and systematic study of the KGA problem to fill the gap in KG+RAG but also opens up new directions for enlightening more powerful KG+RAG designs.

Acknowledgments

We would like to thank anonymous reviewers for their suggestions and comments sincerely. The work was partially supported by the Beijing Natural Science Foundation (L247010) and NSFC (62076032).

References

Abdin, M., Jacobs, S. A., Awan, A. A.; et al. 2024. Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone. arXiv:2404.14219.

Abu-Salih, B. 2021. Domain-specific Knowledge Graphs: A survey. arXiv:2011.00235.

Adlakha, V., BehnamGhader, P., Lu, X. H.; et al. 2024. Evaluating Correctness and Faithfulness of Instruction-Following Models for Question Answering. *Transactions of the Association for Computational Linguistics*, 12: 681–699.

Baek, J., Aji, A. F.; and Saffari, A. 2023. Knowledge-Augmented Language Model Prompting for Zero-Shot Knowledge Graph Question Answering. In Dalvi Mishra, B., Durrett, G., Jansen, P.; et al., eds., *NLRSE*, 78–106. Toronto, Canada: Association for Computational Linguistics.

Bang, Y., Cahyawijaya, S., Lee, N.; et al. 2023. A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity. In *IJCNLP-AACL*, 675–718.

Brandes, U., Eiglsperger, M., Lerner, J.; and Pich, C. 2013. Graph markup language (GraphML).

Brown, T. B., Mann, B., Ryder, N.; et al. 2020. Language Models are Few-Shot Learners. arXiv:2005.14165.

Bubeck, S., Chandrasekaran, V., Eldan, R.; et al. 2023. Sparks of Artificial General Intelligence: Early experiments with GPT-4. arXiv:2303.12712.

Cancedda, N. 2024. Spectral Filters, Dark Signals, and Attention Sinks. arXiv:2402.09221.

Chen, J., Xiao, S., Zhang, P.; et al. 2024. BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. arXiv:2402.03216.

Chughtai, B., Chan, L.; and Nanda, N. 2023. A Toy Model of Universality: Reverse Engineering how Networks Learn Group Operations. In Krause, A., Brunskill, E., Cho, K.; et al., eds., *ICML*, volume 202 of *Proceedings of Machine Learning Research*, 6243–6267. PMLR.

Dong, Z., Peng, B., Wang, Y.; et al. 2024. EffiQA: Efficient Question-Answering with Strategic Multi-Model Collaboration on Knowledge Graphs. arXiv:2406.01238.

Elhage, N., Nanda, N., Olsson, C.; et al. 2021. A Mathematical Framework for Transformer Circuits. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2021/framework/index.html.

Ferrando, J.; and Voita, E. 2024. Information Flow Routes: Automatically Interpreting Language Models at Scale. arXiv:2403.00824.

Gansner, E., Koutsofios, E.; and North, S. 2006. Drawing graphs with dot.

Gao, Y., Xiong, Y., Gao, X.; et al. 2024. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv:2312.10997.

Geva, M., Schuster, R., Berant, J.; and Levy, O. 2021. Transformer Feed-Forward Layers Are Key-Value Memories. In Moens, M.-F., Huang, X., Specia, L.; and Yih, S. W.-t., eds., *EMNLP*, 5484–5495. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics.

Hagberg, A. A., Schult, D. A.; and Swart, P. J. 2008. Exploring network structure, dynamics, and function using NetworkX. In *SciPy2008*, 11–15. Pasadena, CA USA.

Hanna, M., Liu, O.; and Variengien, A. 2023. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. In Oh, A., Naumann, T., Globerson, A.; et al., eds., *Advances in Neural Information Processing Systems*, volume 36, 76033–76060. Curran Associates, Inc.

He, X., Tian, Y., Sun, Y.; et al. 2024. G-Retriever: Retrieval-Augmented Generation for Textual Graph Understanding and Question Answering. arXiv:2402.07630.

Hu, E. J., yelong shen, Wallis, P.; et al. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *ICLR*.

Ji, S., Pan, S., Cambria, E.; et al. 2022. A Survey on Knowledge Graphs: Representation, Acquisition, and Applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2): 494–514.

Jiang, A. Q., Sablayrolles, A., Mensch, A.; et al. 2023a. Mistral 7B. arXiv:2310.06825.

Jiang, J., Zhou, K., Dong, Z.; et al. 2023b. StructGPT: A General Framework for Large Language Model to Reason over Structured Data. In Bouamor, H., Pino, J.; and Bali, K., eds., *EMNLP*, 9237–9251. Singapore: Association for Computational Linguistics.

Kwon, W., Li, Z., Zhuang, S.; et al. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *SIGOPS*.

Li, T., Ma, X., Zhuang, A.; et al. 2023a. Few-shot In-context Learning on Knowledge Base Question Answering. In Rogers, A., Boyd-Graber, J.; and Okazaki, N., eds., *ACL*, 6966–6980. Toronto, Canada: Association for Computational Linguistics.

Li, X., Zhao, R., Chia, Y. K.; et al. 2023b. Chain of knowledge: A framework for grounding large language models with structured knowledge bases. *arXiv preprint arXiv:2305.13269*, 3.

Lin, B. Y., Chen, X., Chen, J.; and Ren, X. 2019. KagNet: Knowledge-Aware Graph Networks for Commonsense Reasoning. In Inui, K., Jiang, J., Ng, V.; and Wan, X., eds., *EMNLP-IJCNLP*, 2829–2839. Hong Kong, China: Association for Computational Linguistics.

Lin, W., Xu, J., Tian, Y.; et al. 2024. Cognitive Intelligence: Driven by Knowledge Graph and Big Model Collaboration. In *CNIOT*, CNIOT '24, 204–209. New York, NY, USA: Association for Computing Machinery. ISBN 9798400716751.

Longpre, S., Perisetla, K., Chen, A.; et al. 2021. Entity-Based Knowledge Conflicts in Question Answering. In Moens, M.-F., Huang, X., Specia, L.; and Yih, S. W.-t., eds., *EMNLP*, 7052–7063. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics.

LUO, L., Li, Y.-F., Haf, R.; and Pan, S. 2024a. Reasoning on Graphs: Faithful and Interpretable Large Language Model Reasoning. In *ICLR*.

LUO, L., Li, Y.-F., Haf, R.; and Pan, S. 2024b. Reasoning on Graphs: Faithful and Interpretable Large Language Model Reasoning. In *The Twelfth International Conference on Learning Representations*.

Meng, K., Bau, D., Andonian, A.; and Belinkov, Y. 2022. Locating and Editing Factual Associations in GPT. In Koyejo, S., Mohamed, S., Agarwal, A.; et al., eds., *Advances in Neural Information Processing Systems*, volume 35, 17359–17372. Curran Associates, Inc.

Meta, A. 2024. Introducing meta llama 3: The most capable openly available llm to date. *Meta AI*.

Nanda, N.; and Bloom, J. 2022. TransformerLens. https://github.com/TransformerLensOrg/TransformerLens.

Nanda, N., Chan, L., Lieberum, T.; et al. 2023. Progress measures for grokking via mechanistic interpretability. In *ICLR*.

Olsson, C., Elhage, N., Nanda, N.; et al. 2022. In-context Learning and Induction Heads. arXiv:2209.11895.

OpenAI. 2021. ChatGPT: A Large-Scale Generative Model for Open-Domain Chat. https://github.com/openai/gpt-3.

Pan, S., Luo, L., Wang, Y.; et al. 2024. Unifying Large Language Models and Knowledge Graphs: A Roadmap. *IEEE Transactions on Knowledge & Data Engineering*, (01): 1–20.

Peng, B., Galley, M., He, P.; et al. 2023. Check Your Facts and Try Again: Improving Large Language Models with External Knowledge and Automated Feedback. arXiv:2302.12813.

Rawte, V., Sheth, A.; and Das, A. 2023. A Survey of Hallucination in Large Foundation Models. arXiv:2309.05922.

Sanmartin, D. 2024. KG-RAG: Bridging the Gap Between Knowledge and Creativity. arXiv:2405.12035.

Schick, T., Dwivedi-Yu, J., Dessi, R.; et al. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. In Oh, A., Naumann, T., Globerson, A.; et al., eds., *Advances in Neural Information Processing Systems*, volume 36, 68539–68551. Curran Associates, Inc.

Shen, Y., Liao, R., Han, Z.; et al. 2023. GraphextQA: A Benchmark for Evaluating Graph-Enhanced Large Language Models. arXiv:2310.08487.

Sun, J., Xu, C., Tang, L.; et al. 2024. Think-on-Graph: Deep and Responsible Reasoning of Large Language Model on Knowledge Graph. In *ICLR*.

Sun, Y., Shi, Q., Qi, L.; and Zhang, Y. 2022. JointLK: Joint Reasoning with Language Models and Knowledge Graphs for Commonsense Question Answering. In Carpuat, M., de Marneffe, M.-C.; and Meza Ruiz, I. V., eds., *NAACL*, 5049–5060. Seattle, United States: Association for Computational Linguistics.

Talmor, A.; and Berant, J. 2018. The Web as a Knowledge-Base for Answering Complex Questions. In Walker, M., Ji, H.; and Stent, A., eds., *NAACL*, 641–651. New Orleans, Louisiana: Association for Computational Linguistics.

Tang, T., Li, J., Zhao, W. X.; and Wen, J.-R. 2023. MVP: Multi-task Supervised Pre-training for Natural Language Generation. In Rogers, A., Boyd-Graber, J.; and Okazaki, N., eds., *Findings of the Association for Computational Linguistics: ACL 2023*, 8758–8794. Toronto, Canada: Association for Computational Linguistics.

Tian, S., Luo, Y., Xu, T.; et al. 2024a. KG-Adapter: Enabling Knowledge Graph Integration in Large Language Models through Parameter-Efficient Fine-Tuning. In Ku, L.-W., Martins, A.; and Srikumar, V., eds., *Findings of the Association for Computational Linguistics ACL 2024*, 3813–3828. Bangkok, Thailand and virtual meeting: Association for Computational Linguistics.

Tian, Y., Song, H., Wang, Z.; et al. 2024b. Graph neural prompting with large language models. In *AAAI*, volume 38, 19080–19088.

Todd, E., Li, M., Sharma, A. S.; et al. 2024. Function Vectors in Large Language Models. In *ICLR*.

Touvron, H., Martin, L., Stone, K.; et al. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288.

Tufanov, I., Hambardzumyan, K., Ferrando, J.; and Voita, E. 2024. LM Transparency Tool: Interactive Tool for Analyzing Transformer Language Models. arXiv:2404.07004.

Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11).

Wang, K. R., Variengien, A., Conmy, A.; et al. 2023. Interpretability in the Wild: a Circuit for Indirect Object Identification in GPT-2 Small. In *ICLR*.

Webber, W., Moffat, A.; and Zobel, J. 2010. A similarity measure for indefinite rankings. *ACM Transactions on Information Systems (TOIS)*, 28(4): 1–38.

Wen, Y., Wang, Z.; and Sun, J. 2024. MindMap: Knowledge Graph Prompting Sparks Graph of Thoughts in Large Language Models. arXiv:2308.09729.

West, D. B.; et al. 2001. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River.

Wu, Y., Hu, N., Bi, S.; et al. 2023. Retrieve-Rewrite-Answer: A KG-to-Text Enhanced LLMs Framework for Knowledge Graph Question Answering. arXiv:2309.11206.

Xi, Z., Chen, W., Guo, X.; et al. 2023. The Rise and Potential of Large Language Model Based Agents: A Survey. arXiv:2309.07864.

Xiong, G., Bao, J.; and Zhao, W. 2024. Interactive-KBQA: Multi-Turn Interactions for Knowledge Base Question Answering with Large Language Models. arXiv:2402.15131.

Ye, J., Chen, X., Xu, N.; et al. 2023. A Comprehensive Capability Analysis of GPT-3 and GPT-3.5 Series Models. arXiv:2303.10420.

Yih, W.-t., Richardson, M., Meek, C.; et al. 2016. The Value of Semantic Parse Labeling for Knowledge Base Question Answering. In Erk, K.; and Smith, N. A., eds., *ACL*, 201–206. Berlin, Germany: Association for Computational Linguistics.

Zhang, Q., Dong, J., Chen, H.; et al. 2024. KnowGPT: Knowledge Graph based Prompting for Large Language Models. arXiv:2312.06185.

Zhang, S., Roller, S., Goyal, N.; et al. 2022. OPT: Open Pre-trained Transformer Language Models. arXiv:2205.01068.

Zhao, H., Chen, H., Yang, F.; et al. 2023a. Explainability for Large Language Models: A Survey. arXiv:2309.01029.

Zhao, H., Fu, Y., Jiang, W.; et al. 2022. Simulate Human Thinking: Cognitive Knowledge Graph Reasoning for Complex Question Answering. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 522–534. Springer.

Zhao, W. X., Zhou, K., Li, J.; et al. 2023b. A Survey of Large Language Models. arXiv:2303.18223.

Zhou, W., Zhang, S., Poon, H.; and Chen, M. 2023. Context-faithful Prompting for Large Language Models. In Bouamor, H., Pino, J.; and Bali, K., eds., *Findings of the Association for Computational Linguistics: EMNLP 2023*, 14544–14556. Singapore: Association for Computational Linguistics.