GNN-RAG: Graph Neural Retrieval for Efficient Large Language Model Reasoning on Knowledge Graphs

Costas Mavromatis
University of Minnesota
mavro016@umn.edu

George Karypis
University of Minnesota
karypis@umn.edu

Abstract

Retrieval-augmented generation (RAG) in Knowledge Graph Question Answering (KGQA) enhances the context of Large Language Models (LLMs) by incorporating information retrieved from the Knowledge Graph (KG). Most recent approaches rely on costly LLM calls to generate executable relation paths or traverse the KG, which is inefficient in complex KGQA tasks, such as those involving multi-hop or multi-entity questions. We introduce the GNN-RAG framework, which utilizes lightweight Graph Neural Networks (GNNs) for effective and efficient graph retrieval. The GNN learns to assign importance weights to nodes based on their relevance to the question, as well as the relevance of their neighboring nodes. This enables the framework to effectively handle context from distant nodes in the graph, improving retrieval performance. GNN-RAG retrieves the shortest paths connecting question entities to GNN answer candidates, providing this information as context for the LLM. Experimental results show that GNN-RAG achieves effective retrieval on two widely used KGQA benchmarks (WebQSP and CWQ), outperforming or matching GPT-4 performance with a 7B tuned LLM. Additionally, GNN-RAG excels on multi-hop and multi-entity questions outperforming LLM-based retrieval approaches by 8.9–15.5% points at answer F1. Furthermore, it surpasses long-context inference while using 9× fewer KG tokens. The code is provided in https://github.com/cmavro/GNN-RAG.

1 Introduction

Large Language Models (LLMs) (Brown et al., 2020; Bommasani et al., 2021; Chowdhery et al., 2023) are the state-of-the-art models in many NLP tasks due to their remarkable ability to understand natural language. The power of LLM comes from pretraining in a large corpus of textual data to obtain general human knowledge (Kaplan et al., 2020; Hoffmann et al., 2022). However, because pretraining is costly and time-consuming (Gururangan et al., 2020), LLMs cannot easily adapt to new or in-domain knowledge and are prone to hallucinations (Zhang et al., 2023b).

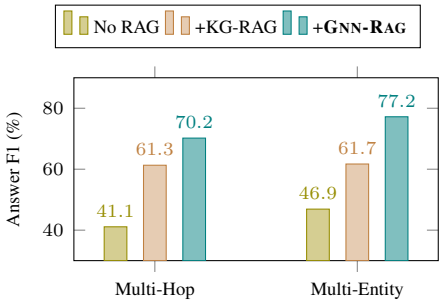

Figure 1: Retrieval effect on multi-hop/entity KGQA. Our **GNN-RAG outperforms** existing KG-RAG methods by 8.9–15.5% points at F1 (ref: Table 2).

Knowledge Graphs (KGs) (Vrandečić and Krötzsch, 2014) store information in structured form that can be easily updated. KGs represent human-crafted factual knowledge in the form of triplets *(head, relation, tail)*, e.g., <Jamaica → language_spoken → English>, which collectively form a graph. In the case of KGs, the stored knowledge is updated by fact addition or removal. As KGs capture complex interactions between stored entities, e.g. multi-hop relations, they are widely used for knowledge-intensive tasks, such as Question Answering (QA) (Pan et al., 2024) with Retrieval-Augmented Generation (RAG) (Lewis et al., 2020).

RAG performance is highly dependent on the KG facts that are retrieved (Wu et al., 2023). The challenge in KGQA is that KGs store complex graph information, usually consisting of millions of facts, and retrieving the right information requires effective graph processing (Mavromatis and Karypis, 2022). Retrieval methods that rely on off-the-shelf NLP retrievers (Baek et al., 2023) or classical graph algorithms (He et al., 2024) are lim-

Findings of the Association for Computational Linguistics: ACL 2025, pages 16682–16699

ited as retrieval is not tailored for KGQA. Most recent methods rely on LLMs for semantic parsing (Luo et al., 2024a) or for traversing the KG (Sun et al., 2024). However, LLMs can be ineffiecient in complex KGQA tasks, such as those involving multi-hop or multi-entity questions. In such cases, leveraging context from distant nodes in the graph is crucial, and LLMs may struggle to process the context of the graph which expands exponentially at deeper levels (Liu et al., 2024a).

To address retrieval efficiency in complex KGQA, we present GNN-RAG, a graph neural framework which utilizes context from deeper parts of the graph, such as information from distant nodes, during retrieval. GNN-RAG relies on Graph Neural Networks (GNNs) (He et al., 2021; Mavromatis and Karypis, 2022), which are powerful graph representation learners, able to handle the complex graph information stored in the KG. GNN-RAG consists of a *graph neural* phase, where the GNN *learns* to embed KG-specific semantics and to identify relevant nodes, given a question. In GNN-RAG, we implement deep (multi-layer) GNN models to leverage graph context. The GNN learns to assign importance weights to nodes based on their relevance to the question, as well as the relevance of their neighboring nodes. This enables the framework to effectively handle context from deeper parts of the graph, improving retrieval performance. At retrieval, the GNN scores answer candidates for the given question, and the shortest paths that connect question entities and answer candidates are retrieved, which are verbalized and given as context to the LLM. Experimental results show GNN-RAG's superiority over competing RAG-based systems for KGQA by outperforming them by up to 15.5% points at complex KGQA performance (Figure 1). Furthermore, we show the effectiveness of GNN-RAG in retrieval augmentation, while outperforming long-context retrieval using $9\times$ fewer KG tokens. Our **contributions** are summarized below:

- **Framework**: GNN-RAG utilizes GNN models to leverage context from distant nodes in the graph, while the LLM leverages its natural language processing ability for ultimate KGQA. Additionally, we propose retrieval augmentation and routing techniques that utilize GNN-RAG to enhance overall end-to-end efficiency.
- **Effectiveness**: GNN-RAG achieves effective performance on two widely used KGQA benchmarks (WebQSP and CWQ), improving complex question answering by 8.9–15.5% points at F1 (Figure 1).
- **Efficiency**: GNN-RAG improves vanilla LLMs on KGQA performance without incurring additional LLM calls as previous state-of-the-art RAG systems for KGQA require. In addition, GNN-RAG outperforms or matches GPT-4 performance with a 7B tuned LLM, while outperforming long-context retrieval using $9\times$ fewer KG tokens.

2 Related Work

KGQA Methods. KGQA methods fall into two categories (Lan et al., 2022): (i) semantic parsing (SP) methods and (ii) information retrieval (IR) methods. SP methods (Sun et al., 2020; Lan and Jiang, 2020; Ye et al., 2022) learn to transform the given question into a query of logical form, e.g., SPARQL query. The transformed query is then executed over the KG to obtain the answers. However, SP methods require ground-truth logical queries for training, which are time-consuming to annotate in practice, and may lead non-executable queries due to syntactical or semantic errors (Das et al., 2021; Yu et al., 2022). IR methods (Sun et al., 2018, 2019; Zhang et al., 2022b) focus on the weakly-supervised KGQA setting, where only question-answer pairs are given for training. IR methods retrieve KG information, e.g., a KG subgraph (Zhang et al., 2022a), which is used as input during KGQA reasoning. GNN-RAG falls in the IR category.

GNNs & LMs. Combining GNNs with LMs has been the subject of a substantial body of existing literature (Jin et al., 2023), with various applications ranging from QA (Yasunaga et al., 2021; Wang et al., 2021; Zhang et al., 2022c; Christmann et al., 2023; Tian et al., 2024; He et al., 2024; Zhang et al., 2024a) to training LMs on graphs (Zhao et al., 2022; Yasunaga et al., 2022; Huang et al., 2024). Such approaches seek to combine the natural language and graph reasoning into a single model by fusing *latent* GNN information with the LM. However, due to the modality mismatch of GNNs and LMs, fusing graph and natural language information is challenging for many knowledge-intensive tasks, even in supervised settings (Mavromatis et al., 2024). To alleviate this challenge, GNN-RAG divides KGQA in two stages. The GNN

first retrieves useful information from the graph modality, which is then converted into natural language for effective LLM reasoning.

GraphRAG. GraphRAG usually refers to the general approach of inserting *verbalized* graph information at the context of LLMs (Peng et al., 2024; Wei et al., 2024) or leveraging additional graph information when retrieving context for RAG (Edge et al., 2024; Gutiérrez et al., 2024). For instance, verbalizing graph information obtained by KGs has been widely applied in GraphRAG (Xie et al., 2022; Baek et al., 2023; Jiang et al., 2023a; Jin et al., 2024; Liu et al., 2024b; Zhao et al., 2024; Li et al., 2024a; Luo et al., 2024b). However, GraphRAG performance downgrades when the graph information retrieved is noisy and irrelevant to the question (Wu et al., 2023; He et al., 2024). To improve retrieval in KGQA, GNN-RAG employs a graph neural framework, which tailors graph retrieval for the KG at hand. By optimizing GNNs to identify the right graph information for answering the questions, GNN-RAG achieves superior retrieval performance compared to existing approaches in KGQA.

3 Problem Statement & Background

KGQA. We are given a KG $\mathcal{G}$ that contains facts represented as (v, r, v'), where v denotes the head entity, v' denotes the tail entity, and r is the corresponding relation between the two entities. Given $\mathcal{G}$ and a natural language question q, the task of KGQA is to extract a set of entities $\{a_q\} \in \mathcal{G}$ that correctly answer q. Following previous works (Lan et al., 2022), question-answer pairs are given for training, but not the ground-truth paths that lead to the answers.

As KGs usually contain millions of facts and nodes, a smaller question-specific subgraph $\mathcal{G}_q$ is retrieved for a question q. Entity linking identifies question entities $\{e_q\}$ and subgraph $\mathcal{G}_q$ is extracted by following connections that start from these question entities (Yih et al., 2015). Ideally, all correct answers for the question are contained in the retrieved subgraph, $\{a_q\} \in \mathcal{G}_q$. The retrieved subgraph $\mathcal{G}_q$ along with the question q are used as input to a KGQA model, which outputs the correct answer(s). The prevailing KGQA methods studied are GNNs and LLMs.

GNNs. KGQA can be regarded as a node classification problem, where KG entities are classified as answers vs. non-answers for a given question. GNNs (Kipf and Welling, 2016; Veličković et al., 2017; Schlichtkrull et al., 2018) are powerful graph representation learners suited for tasks such as node classification. GNNs update the representation $\boldsymbol{h}_v^{(l)}$ of node v at layer l by aggregating messages $\boldsymbol{m}_{vv'}^{(l)}$ from each neighbor v'. During KGQA, the message passing is also conditioned to the given question q (He et al., 2021). For readability purposes, we present the following GNN update for KGQA,

$$\boldsymbol{h}_v^{(l)} = \psi\Big(\boldsymbol{h}_v^{(l-1)}, \sum_{(v,r,v')\in\mathcal{N}_v} \omega(q,r)\cdot \boldsymbol{m}_{vv'}^{(l)}\Big), \quad (1)$$

where function $\omega(\cdot)$ is typically a LM that measures how relevant relation r of fact (v, r, v') is to question q. $\mathcal{N}_v$ denotes the set of neighboring triplets of node v. Neighbor messages $\boldsymbol{m}_{vv'}^{(l)}$ are aggregated by a sum-operator $\sum$ and function $\psi(\cdot)$ combines representations from consecutive GNN layers.

Previous GNN approaches perform query-to-KG similarity using pretrained encoders (function ω in Equation 1). However, such encoder models are not as accurate as LLMs. GNN-RAG addresses this limitation by employing an LLM as a post-retrieval step to accurately select the relevant KG evidence. The importance of GNN-RAG's framework is extensively validated in Section 6.

LLM RAG. Retrieval-Augment Generation (RAG) is a method aiming to reduce LLM hallucinations (Lewis et al., 2020). Given a query q, RAG retrieves relevant information (e.g, documents from the given corpus), which is inserted as additional context c to the LLM's input. In KGs, the context c consists of graph information relevant to the question, such KG triplets, paths, or subgraphs. The retrieved graph information is first converted into natural language so that it can be processed by the LLM. The input given to the LLM contains the KG factual information along with the question and a prompt. For instance, the input becomes "`Knowledge: Jamaica` $\rightarrow$ `language_spoken` $\rightarrow$ `English \n Question: Which language do Jamaican people speak?`", where the LLM has access to KG information for answering the question.

Landscape of KGQA methods. Figure 2 presents the landscape of existing KGQA methods with respect to KG retrieval and reasoning. GNN-based methods, such as GraftNet (Sun et al., 2018), NSM (He et al., 2021), and ReaRev (Mavromatis and Karypis, 2022), reason over a dense KG

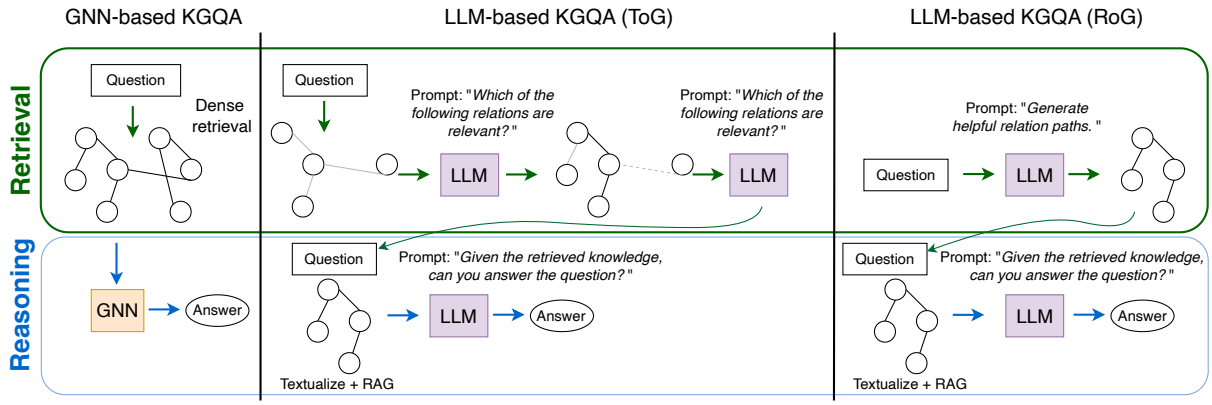

Figure 2: The landscape of existing KGQA methods. GNN-based methods reason on dense subgraphs as they can handle complex and multi-hop graph information. LLM-based methods employ the same LLM for both retrieval and reasoning due to its ability to understand natural language.

subgraph leveraging the GNN's ability to handle complex graph information. Recent LLM-based methods leverage the LLM's power for both retrieval and reasoning (Gu et al., 2023). For instance, ToG (Sun et al., 2024) uses the LLM to retrieve relevant facts hop-by-hop. RoG (Luo et al., 2024a) uses the LLM to generate plausible relation paths which are then queried on the KG to retrieve the relevant information, similar to semantic parsing. ToG utilizes an LLM to selectively expand and then prune the graph context. Meanwhile, RoG does not rely on any graph context. As a result, LLM-based approaches may overlook or incorrectly prune important graph information for KGQA tasks.

4 GNN-RAG

We present GNN-RAG, a new graph neural retrieval method for KGQA that leverages GNNs to improve retrieval performance when questions require complex graph information. GNN-RAG's primary contribution lies in its use of GNNs for efficient graph retrieval, distinguishing it from previous methods that rely on long-context retrieval based on text similarity or retrieval via LLMs. We provide the overall framework at inference time in Figure 3. First, the GNN processes a dense subgraph to utilize deep graph context and identify relevant answer nodes for a given question. Second, the shortest paths in the KG that connect question entities and GNN-based answers are extracted to represent useful KG reasoning paths. The extracted paths are verbalized and given as context for LLM reasoning via RAG. In our GNN-RAG framework, the GNN acts as a dense subgraph processor, while the LLM leverages its natural language processing ability for downstream KGQA.

4.1 GNN

In order to retrieve high-quality reasoning paths, we prefer deep GNNs over other shallow KGQA methods, e.g., embedding-based methods (Saxena et al., 2020), due to their ability to handle deep graph interactions. GNNs mark themselves as good candidates for retrieval due to their architectural benefit of exploring diverse reasoning paths (Choi et al., 2024) that result in high answer recall.

GNNs consist of L updates via Equation 1 (L is hyperparameter), where the node embbeddings in the subgraph $\mathcal{G}_q$ are updated to $\boldsymbol{h}_v^{(L)} \in \mathbb{R}^d$, where d is the embedding dimension. In our framework, at each layer, the GNN assigns importance to nodes based on their likelihood of forming a path to the answer. We use their KG relations to assess their relevance to the question. Unlike LLM-based KG traversal, the GNN learns to aggregate information relevant to the question within the embedding space.

At layer $l+1$, the GNN aggregates relation embeddings $\boldsymbol{r}, r \in \mathcal{N}_v$, where $\mathcal{N}_v$ is the set of neighboring triplets of node v, based on their relevance to the question as

$$\sum_{(v,r,v') \in \mathcal{N}_v} p_{v'}^{(l)} \cdot \sigma(\boldsymbol{q}^k \odot \boldsymbol{r}). \tag{2}$$

In Equation 2, $\boldsymbol{q}^k$ is a question embedding, $\odot$ is the pairwise multiplication, $\sigma(\cdot)$ is a ReLU activation, and $p_{v'}^{(l)}$ is the node importance weight. Message $\boldsymbol{m}_{vv'} = \sigma(\boldsymbol{q}^k \odot \boldsymbol{r})$ is activated if the corresponding relation is relevant to the question, e.g., relation

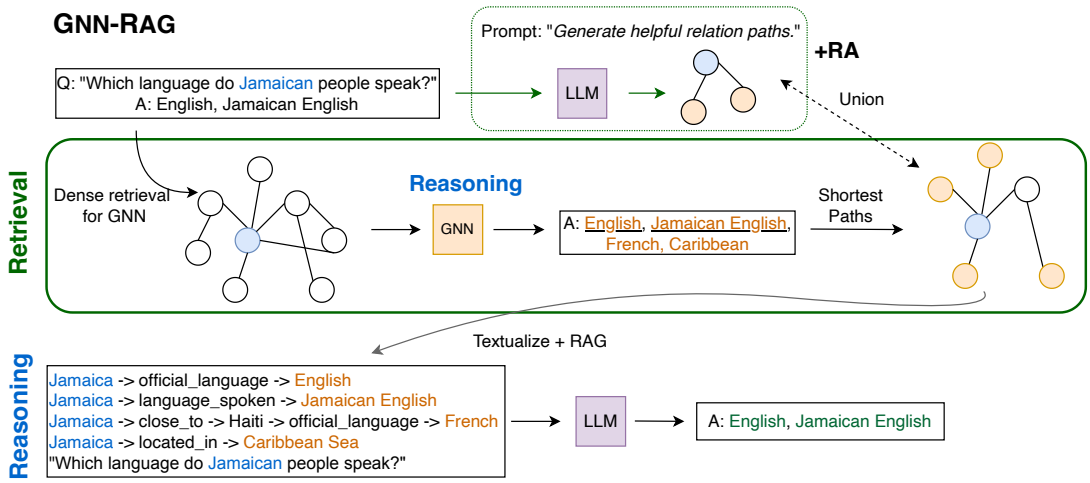

Figure 3: GNN-RAG: The GNN reasons over a dense subgraph to retrieve candidate answers, along with the corresponding reasoning paths (shortest paths from question entities to answers). The retrieved reasoning paths –optionally combined with retrieval augmentation (RA)– are verbalized and given to the LLM for RAG.

'language_spoken' is relevant to question 'Which language do Jamaican people speak?'.

As complex questions consist of multiple sub-questions, we obtain K different question embeddings $\boldsymbol{q}^k, k \in K$, that capture different question parts (Qiu et al., 2020). Question embeddings $\boldsymbol{q}^k$ and KG relation embeddings $\boldsymbol{r}$ are encoded via a shared pretrained LM (Jiang et al., 2023b), such as SBERT (Reimers and Gurevych, 2019). We obtain

$$\boldsymbol{q}_k = \gamma_k\big(\mathrm{LM}(q)\big), \quad \boldsymbol{r} = \gamma_c\big(\mathrm{LM}(r)\big), \tag{3}$$

where γ_k is an attention-based pooling neural network that attends to question tokens, and γ_c is the [CLS] token pooling. We provide the implementation of these networks in Appendix A.

We implement a multi-head GNN layer which aggregates neighboring messages based on different question parts as

$$\boldsymbol{h}_v^{(l+1)} = \psi\Big(\boldsymbol{h}_v^{(l)}, \big|\big|_{k=1}^{K} \sum_{(v,r,v')\in\mathcal{N}_v} p_{v'}^{(l)} \cdot \sigma(\boldsymbol{q}^k \odot \boldsymbol{r})\Big). \tag{4}$$

Operation $||_{k=1}^{K}$ denotes the concatenation of K heads, and $\psi(\cdot) : \mathbb{R}^{(K+1)d} \rightarrow \mathbb{R}^d$ is a multilayer perceptron that combines KG semantics relevant to the question in the embedding space from consecutive layers.

In Equation 4, the importance of each node is measured by $p_v^{(l)} \in [0, 1]$, which is the probability of visiting a node at layer l. In the first layer, $p_v^{(0)} = 1$ if node v is a question entity $v \in \{e_q\}$ and $p_v^{(0)} = 0$, otherwise. At each layer, the probability vector is updated based on the GNN node embeddings followed by a $\mathrm{softmax}(\cdot)$ operation as

$$\boldsymbol{p}^{(l+1)} = \mathrm{softmax}(\boldsymbol{H}^{(l)}\boldsymbol{w}), \tag{5}$$

where $\boldsymbol{w} \in \mathbb{R}^d$ is a learnable vector. The GNN described above incorporates deep graph information up to L hops from the question entities. At each layer, it assigns importance to nodes based on their relations relevance to the question, as well as the probability of visiting their neighbor node in the previous layer. This approach parallels LLM-based KG traversal (Sun et al., 2024), where, starting from the question entities, the LLM iteratively selects relations and neighboring nodes to explore.

Furthermore, to enable iterative KG traversal we reset the probability vector $\boldsymbol{p}^{(l)}$ by

$$\boldsymbol{p}^{(l)} = \begin{cases} \boldsymbol{p}^{(0)} & \text{if } l = \frac{L}{2} \\ \boldsymbol{p}^{(l)} & \text{else.} \end{cases} \tag{6}$$

Here, we impose the constraint that L must be even. As shown in Equation 6, this mechanism allows us to re-evaluate node importance using deep graph embeddings $\boldsymbol{H}^{(\frac{L}{2})}$, effectively restarting the reasoning process from the question entities (Mavromatis and Karypis, 2022). By default, we implement $L = 6$ GNN layers, so the reasoning process restarts from the question entities at layer $l = 3$. This approach ensures that node embeddings capture deep graph information, enabling a revisited and refined KG traversal. Details on node embedding initialization can be found in Appendix A.

4.2 GNN Optimization and RAG

Given training question-answer pairs, the GNN is trained via node classification, where nodes have

label $y_v = 1$ if they belong to the answer set $v \in \{a_q\}$ and $y_v = 0$, otherwise. The GNN parameters are optimized via the KL-divergence loss so that $p_v^{(L)}$ is close to 1 if $v \in \{a_q\}$, and zero otherwise.

During inference, nodes with the highest probability scores are identified as candidate answers. To filter out noisy information, we select nodes whose cumulative probability exceeds a threshold of 0.95. Subsequently, the shortest paths connecting the question entities to the candidate answers (i.e., reasoning paths) are extracted. These reasoning paths are then verbalized using a predefined template and serve as input for the LLM-based RAG process.

For the downstream LLM, we opt to follow KGQA prompt tuning (Lin et al., 2023; Zhang et al., 2024b), where the Llama-7B-Chat model is trained to generate a list of answers for KGQA (Luo et al., 2024a), given the input reasoning paths. We provide the details in Appendix A.

4.3 GNN-RAG Augmentation and Routing

GNN-RAG employs GNNs that are specialized in retrieving context from deeper parts of the graphs, such as context from distant nodes in the graph. However, the effectiveness of GNNs depends on the training data, and they may not always generalize well to unseen contexts. In contrast, retrieval based on natural language, such as using text embedding models or LLMs, leverages the pretraining data of these models, enabling better generalization. To address the generalization limitation of GNNs, we propose the following two techniques.

GNN-RAG+RA. First, retrieval augmentation (RA) combines information retrieved from KGs using different approaches to enhance diversity and improve answer recall. GNN-RAG+RA combines GNN-RAG with LLM-based semantic parsing, which specialized in question-relation matching. We use RoG (Luo et al., 2024a) as the semantic parsing approach, which fine-tunes a 7B LLM to generate executable relation paths given a question. During inference, we take the union of the reasoning paths retrieved by the two retrievers.

GNN-RAG+Route. Second, text-based approaches (Baek et al., 2023) retrieve a large number of KG triplets by using text embedding similarity to the input question, subsequently utilizing long-context LLMs for KGQA. With access to a more extensive graph context, the LLM can provide more accurate responses. GNN-RAG+Route integrates GNN-RAG for fetching relevant graph information that may not be captured through long-context retrieval alone. Specifically, the context retrieved by GNN-RAG is fed into the downstream LLM to generate answers. If any of the generated responses are not included in the long-context, e.g., in the top-$k = 100$ triplets retrieved by text similarity, this indicates that GNN-RAG has provided additional deep graph context that is beneficial for answering the question. In contrast, when the answers align with the long-context information, the system routes the inference to long-context processing, thereby capitalizing on the increased graph data. For text embedding retrieval, we employ the SubgraphRAG method (Li et al., 2024b) which uses training data to fine-tune the retriever.

5 Experimental Setup

KGQA Datasets. We experiment with widely used KGQA benchmarks: WebQuestionsSP (WebQSP) (Yih et al., 2015), Complex WebQuestions 1.1 (CWQ) (Talmor and Berant, 2018), and MetaQA-3 (Zhang et al., 2018). **WebQSP** contains 4,737 natural language questions that are answerable using a subset Freebase KG (Bollacker et al., 2008). The questions require up to 2-hop reasoning within this KG. **CWQ** contains 34,699 total complex questions that require up to 4-hops of reasoning over the KG. **MetaQA-3** consists of 3-hop questions in the domain of WikiMovies (Miller et al., 2016). We provide the detailed dataset statistics in Appendix C.

Implementation & Evaluation. For subgraph retrieval, we use the linked entities and the pagerank algorithm to extract dense graph information (He et al., 2021). The default GNN implementation is to use $L = 6$ layers, $K = 3$ decomposed question embeddings, and iterative reasoning (Equation 6). For GNN training, we employ standard training procedure and hyperparameters as in the literature (Mavromatis and Karypis, 2022). As the defualt downstream LLM, we use the Llama2-Chat-7B model finetuned for KGQA (Luo et al., 2024a). For KGQA evaluation, we adopt Hit and F1 metrics. Hit measures if any of the true answers is found in the generated response, which is typically employed when evaluating LLMs. Hit is based on exact match (Sun et al., 2024) rather than fuzzy matching as in Tan et al. (2023). F1 penalizes incorrect or missing answers. We also report H@1, which is the accuracy of the first predicted answer. For retrieval evaluation, we use Hit@k, which evaluates whether a correct answer is retrieved in the

Table 1: Performance comparison of different methods on the two KGQA benchmarks.

Type	Method	WebQSP		CWQ	
		Hit	F1	Hit	F1
Embedding	KV-Mem (Miller et al., 2016)	46.7	38.6	21.1	–
	EmbedKGQA (Saxena et al., 2020)	66.6	–	–	–
	TransferNet (Shi et al., 2021)	71.4	–	48.6	–
	Rigel (Sen et al., 2021)	73.3	–	48.7	–
GNN	GraftNet (Sun et al., 2018)	66.7	62.4	36.8	32.7
	PullNet (Sun et al., 2019)	68.1	–	45.9	–
	NSM (He et al., 2021)	68.7	62.8	47.6	42.4
	SR+NSM(+E2E) (Zhang et al., 2022a)	69.5	64.1	50.2	47.1
	NSM+h (He et al., 2021)	74.3	67.4	48.8	44.0
	SQALER (Atzeni et al., 2021)	76.1	–	–	–
	UniKGQA (Jiang et al., 2023b)	77.2	72.2	51.2	49.1
	ReaRev (Mavromatis and Karypis, 2022)	76.4	70.9	52.9	47.8
	ReaRev + LM_{SR}	77.5	72.8	53.3	49.7
LLM	Flan-T5-xl (Chung et al., 2024)	31.0	–	14.7	–
	Alpaca-7B (Taori et al., 2023)	51.8	–	27.4	–
	LLaMA2-Chat-7B (Touvron et al., 2023)	64.4	–	34.6	–
	ChatGPT	66.8	–	39.9	–
	ChatGPT+CoT	75.6	–	48.9	–
KG+LLM	KAPING (Baek et al., 2023)	73.9	–	–	–
	KD-CoT (Wang et al., 2023)	68.6	52.5	55.7	–
	StructGPT (Jiang et al., 2023a)	72.6	–	–	–
	KB-BINDER (Li et al., 2023)	74.4	–	–	–
	ToG+Llama2-70B (Sun et al., 2024)	68.9	–	57.6	–
	ToG+ChatGPT (Sun et al., 2024)	76.2	–	58.9	–
	ToG+GPT-4 (Sun et al., 2024)	82.6	–	**69.5**	–
	RoG-7B (Luo et al., 2024a)	85.7	70.8	62.6	56.2
GNN+LLM	G-Retriever (He et al., 2024)	70.1	–	–	–
	GNN-RAG	85.7	71.3	66.8	59.4
	GNN-RAG+RA	**90.7**	**73.5**	68.7	**60.4**
KG-LC	SubgraphRAG (Li et al., 2024b)	89.4	–	68.6	–
	GNN-RAG+Route	90.1	–	72.4	–
	GNN-RAG+RA+Route	91.0*	–	73.3*	–

We denote the **best** and second-best methods, as well as the best* method with long-context (KG-LC).
GNN-RAG, RoG, KD-CoT, and G-Retriever use 7B fine-tuned Llama2 models. KD-CoT employs ChatGPT as well. For KG-LC, methods use Llama-3.1-8B.

top-k retrieved nodes. Further experimental setup details are provided in Appendix C.

Competing Methods. We compare with SOTA GNN and LLM methods for KGQA (Mavromatis and Karypis, 2022; Li et al., 2023). We also include earlier embedding-based methods (Saxena et al., 2020) as well as zero-shot/few-shot LLMs (Taori et al., 2023). We do not compare with semantic parsing methods (Yu et al., 2022; Gu et al., 2023) that use ground-truth SPARQL annotations for training, which are difficult to obtain in practice. Furthermore, we compare GNN-RAG with LLM-based retrieval (Luo et al., 2024a; Sun et al., 2024) and long-context (Li et al., 2024b) approaches in terms of efficiency and effectiveness.

6 Results

Main Results. Table 1 presents performance results of different KGQA methods. The results show that equipping LLMs with GNN-based retrieval enhances KGQA performance compared to previous approaches (GNN+LLM vs. KG+LLM). Specifically, GNN-RAG+RA outperforms RoG by 5.0–6.1% points at Hit, while it outperforms or matches ToG+GPT-4 performance, using an LLM with only 7B parameters and much fewer LLM calls, while GNN-RAG can be deployed on a single 24GB GPU. GNN-RAG+RA outperforms ToG+ChatGPT by up to 14.5% points at Hit and the best performing GNN by 5.3–9.5% points at Hits@1 and by 0.7–10.7% points at F1. In long-context retrieval (KG-LC), GNN-RAG+Route outperforms SubgraphRAG by 3.5% points at Hit on CWQ, while being more efficient when queries are routed to GNN-RAG inference.

Table 2: Performance analysis on multi-hop (hops$\geq$ 2) and multi-entity (entities$\geq$ 2) questions.

Method	**WebQSP** (F1)		**CWQ** (F1)		**MetaQA-3** (H@1)
	multi-hop	multi-entity	multi-hop	multi-entity	multi-hop
LLM (No RAG)	48.4	61.5	33.7	32.3	29.7
GNN	58.8	70.4	57.7	54.2	98.6
RoG	63.3	65.1	59.3	58.3	84.8
SubgraphRAG	65.8	54.9	55.8	52.3	–
GNN-RAG	69.8	82.3	68.2	64.8	98.6
GNN-RAG+RA	71.1	88.8	69.3	65.6	98.6
Absolute Improv.	**+5.3**	**+23.7**	**+10.0**	**+7.3**	**+13.8**

Table 3: Comparison of different retrieval methods on CWQ. '#KG Tokens' denotes the median number of KG tokens retrieved as context for the LLM.

	Retrieval Metrics			*KGQA*
	#KG Tokens	**Hit@1** (%)	**Hit@10** (%)	**F1** (%)
RoG	201	25.9	54.5	56.2
SubgraphRAG	1,442	26.8	58.7	47.2
GNN-RAG	114	52.9	64.1	59.4
GNN-RAG+RA	362	52.9	71.1	60.4
GNN	–	52.9	63.8	47.8

Complex KGQA. Table 2 compares complex KGQA performance results on multi-hop questions, where answers are more than one hop away from the question entities, and multi-entity questions, which have more than one question entities. GNN-RAG leverages GNNs to handle complex graph information and outperforms RoG (LLM-based retrieval) and SubgraphRAG (long-context retrieval) by 4.0–17.2% points at F1 on WebQSP, by 8.5–8.9% points at F1 on CWQ, and by 13.8% points at H@1 on MetaQA-3. In addition, GNN-RAG+RA offers an additional improvement by up to 6.5% points at F1 over RoG. The results show that GNN-RAG is an effective retrieval method when the questions involve complex graph information.

Retrieval Results. Table 3 presents an evaluation of the retrieval performance across different graph retrieval methods, alongside their impact on downstream KGQA performance. Based on these results, we make the following observations: First, GNN-based retrieval is both more efficient

Table 4: Ablation study on different GNN retrievers for KGQA in terms of number of layers L, number of question embeddings K, and iterative reasoning.

Retriever	Setting		WebQSP			CWQ		
	L	K	Hit	H@1	F1	Hit	H@1	F1
Dense Subgraph	–	–	70.2	68.7	54.3	47.1	45.5	41.9
GNN	2	1	82.8	78.6	69.8	58.2	51.9	49.4
GNN	3	1	85.0	79.6	70.4	58.5	52.5	50.1
GNN	3	3	85.2	80.1	70.6	62.5	57.5	53.3
GNN iterative	6	3	**85.7**	**80.6**	**71.3**	**66.8**	**61.7**	**59.4**

(in terms of the number of KG tokens) and more effective (in terms of F1 score) than both RoG and SubgraphRAG, particularly for complex questions such as those from CWQ. Second, GNN-based retrieval demonstrates superior performance, surpassing RoG and SubgraphRAG by 26.1–27.0% points in terms of Hit@1. Third, retrieval augmentation (denoted as GNN-RAG+RA) enhances both Hit@k and KGQA F1 scores, as it enables the combination of non-overlapping knowledge graph information from GNN-based and semantic parsing (RoG) retrieval, leading to an increase in the number of input tokens and thereby improving the LLM's contextual understanding. GNN-RAG+RA is more efficient than long-context alternatives as SubgraphRAG.

GNN Ablation. In Table 4, we present an ablation study of GNN hyperparameters (number of layers L and number of question embeddings K) and techniques (with or without iterative reasoning) for GNN-RAG. The results indicate that shallow GNNs with fewer layers exhibit suboptimal performance. Additionally, increasing the number K of question embeddings used leads to improvements, particularly for complex questions (CWQ), which consist of multiple subquestions. Furthermore, iterative GNN reasoning, as described in Equation 6, further enhances performance. This suggests that the GNN effectively revisits node importance by incorporating deeper graph context.

Retrieval Effect on LLMs. Table 5 presents performance results of various LLMs using different retrievers. GNN-RAG is the retrieval approach that achieves the overall best performance. For instance, GNN-RAG improves ChatGPT by up to 5.2% points at Hit over the best competing approach . Moreover, GNN-RAG substantially improves the KGQA performance of weaker LLMs, such as Alpaca-7B and Flan-T5-xl. The improvement over RoG is up to 13.2% points at Hit, while GNN-RAG outperforms Llama-70B approaches using a lightweight Llama-7B model. The results demonstrate that GNN-RAG can be integrated with other LLMs to improve their KGQA performance without retraining.

Table 5: Retrieval effect on performance (% Hit) using various LLMs.

Method	WebQSP	CWQ
ChatGPT	51.8	39.9
+ ToG	76.2	58.9
+ RoG	81.5	52.7
+ SubgraphRAG	83.1	56.2
+ GNN-RAG	85.3	64.1
Alpaca-7B	51.8	27.4
+ RoG	73.6	44.0
+ GNN-RAG	76.2	54.5
Llama2-Chat-7B	64.4	34.6
+ RoG	84.8	56.4
+ GNN-RAG	85.2	62.7
Llama2/3.1-Chat-70B	57.4	39.1
+ ToG	68.9	57.6
+ SubgraphRAG	86.2	57.9
Flan-T5-xl	31.0	14.7
+ RoG	67.9	37.8
+ GNN-RAG	74.5	51.0

Table 6: GNN-RAG routing effect on efficiency and KGQA performance. We evaluate results on the routed subset (CWQ-sub). '#Routed' denotes the number of routed questions for efficient retrieval with GNN-RAG.

	CWQ-sub		
	#Routed	#LLM Calls	Hit (%)
RoG	0	4	59.6
w/ GNN-RAG route	2,377 (78%)	1	66.7
	#Routed	#KG Tokens	Hit (%)
SubgraphRAG	0	1,400	41.4
w/ GNN-RAG route	970 (28%)	153	55.1

Routing Efficiency. Table 6 presents results demonstrating that GNN-RAG routing significantly reduces inference costs while enhancing the performance of KGQA. For RoG routing, we route to GNN-RAG if the GNN retrieves a multi-hop reasoning path. For SubgraphRAG routing, we route to GNN-RAG if any of the LLM response is not present in the long-context. As the results indicate, GNN-RAG improves the performance of 78% of questions by 7.1% points in Hit, while requiring only a single LLM call, in contrast to RoG, which necessitates 4 LLM calls. Furthermore, GNN-RAG enhances 28% of questions by 13.7% points relative to SubgraphRAG, while retrieving approximately 9× fewer KG tokens, leading to more efficient LLM inference.

Table 7: Time latency across methods. LLM latency time is measured on an A10G GPU with fp16 inference.

	Retrieval Time (mins)	Generation Time (mins)	Total Time (mins)
RoG	11	31	42
SubgraphRAG	0.1	58	58.1
GNN-RAG	0.9	29	30

6.1 Time Latency

We primarily measure #KG Tokens as a metric of efficiency, as the end-to-end (retrieval+generation) latency is predominantly determined by the LLM generation. To provide more context, we present the modularized latency cost of each method on WebQSP, using a 7B Llama model for generation in Table 7. As the results show, GNN-RAG is the most efficient method in terms of end-to-end latency.

Appendix. Case studies are provided in Appendix B and further ablation studies in Appendix D.

7 Conclusion

We introduce GNN-RAG, a novel graph neural method for enhancing RAG in KGQA with GNNs. Our **contributions** are the following. (1) **Framework**: GNN-RAG tailors GNNs for KG retrieval due to their ability to handle complex graph information. (2) **Effectiveness**: GNN-RAG achieves superior performance when multi-hop information is needed for KGQA. (3) **Efficiency**: GNN-RAG improves vanilla LLMs on KGQA performance without incurring additional LLM calls as existing RAG systems for KGQA require. In addition, GNN-RAG outperforms long-context retrieval using $9\times$ fewer KG tokens.

8 Limitations

GNN-RAG assumes that the KG subgraph, on which the GNN reasons, contains answer nodes. However, this may not be true for all questions or when errors in entity linking happen. In addition, GNN-RAG employs simple prompting with the shortest paths from question entities to candidate answers as context. As an extension, GNN-RAG can be combined with prompt optimization (Wen et al., 2023; Zhang et al., 2023a) so that the LLM understands the graph better. Moreover, the scope of our GNN-RAG contributions is to improve the retrieval results over the KG without specialized GNN-LLM interactions. However, the GNN and the LLM could be coupled via iterative retrieval (Asai et al., 2023) to further improve KGQA.

Acknowledgements

This work was supported in part by NSF (1447788, 1704074, 1757916, 1834251, 1834332), Army Research Office (W911NF1810344), Intel Corp, and Amazon Web Services. Access to research and computing facilities was provided by the Minnesota Supercomputing Institute.

References

Reid Andersen, Fan Chung, and Kevin Lang. 2006. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*.

Mattia Atzeni, Jasmina Bogojeska, and Andreas Loukas. 2021. Sqaler: Scaling question answering by decoupling multi-hop and logical reasoning. *Advances in Neural Information Processing Systems*.

Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. *arXiv preprint arXiv:2306.04136*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

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

Hyeong Kyu Choi, Seunghun Lee, Jaewon Chu, and Hyunwoo J Kim. 2024. Nutrea: Neural tree search for context-guided multi-hop kgqa. *Advances in Neural Information Processing Systems*, 36.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Philipp Christmann, Rishiraj Saha Roy, and Gerhard Weikum. 2023. Explainable conversational question answering over heterogeneous sources via iterative graph neural networks. In *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval*, pages 643–653.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay-Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew McCallum. 2021. Case-based reasoning for natural language queries over knowledge bases. *arXiv preprint arXiv:2104.08762*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Yu Gu, Xiang Deng, and Yu Su. 2023. Don't generate, discriminate: A proposal for grounding language models to real-world environments. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Toronto, Canada. Association for Computational Linguistics.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. *arXiv preprint arXiv:2405.14831*.

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 553–561.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *arXiv preprint arXiv:2402.07630*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Xuanwen Huang, Kaiqiao Han, Yang Yang, Dezheng Bao, Quanjin Tao, Ziwei Chai, and Qi Zhu. 2024. Can gnn be good adapter for llms? *arXiv preprint arXiv:2402.12984*.

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

Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, and Ji-Rong Wen. 2023b. Unikgqa: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. In *International Conference on Learning Representations*.

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

Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Suhang Wang, Yu Meng, and Jiawei Han. 2024. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. *arXiv preprint arXiv:2404.07103*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Complex knowledge base question answering: A survey. *IEEE Transactions on Knowledge and Data Engineering*.

Yunshi Lan and Jing Jiang. 2020. Query graph generation for answering multi-hop complex questions from knowledge bases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 969–974. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Kun Li, Tianhua Zhang, Xixin Wu, Hongyin Luo, James Glass, and Helen Meng. 2024a. Decoding on graphs: Faithful and sound reasoning on knowledge graphs through generation of well-formed chains. *arXiv preprint arXiv:2410.18415*.

Mufei Li, Siqi Miao, and Pan Li. 2024b. Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation. *arXiv preprint arXiv:2410.20724*.

Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. 2023. Few-shot in-context learning on knowledge base question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6966–6980. Association for Computational Linguistics.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Rich James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, et al. 2023. Ra-dit: Retrieval-augmented dual instruction tuning. *arXiv preprint arXiv:2310.01352*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the middle: How language models use long contexts. *TACL*.

Yang Liu, Xiaobin Tian, Zequn Sun, and Wei Hu. 2024b. Finetuning generative large language models with discrimination instructions for knowledge graph completion. *arXiv preprint arXiv:2407.16127*.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024a. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *International Conference on Learning Representations*.

Linhao Luo, Zicheng Zhao, Chen Gong, Gholamreza Haffari, and Shirui Pan. 2024b. Graph-constrained reasoning: Faithful reasoning on knowledge graphs with large language models. *arXiv preprint arXiv:2410.13080*.

Costas Mavromatis and George Karypis. 2022. ReaRev: Adaptive reasoning for question answering over knowledge graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2447–2458, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Costas Mavromatis, Petros Karypis, and George Karypis. 2024. Sempool: Simple, robust, and interpretable kg pooling for enhancing language models. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 154–166. Springer.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

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*.

Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2024. Graph retrieval-augmented generation: A survey. *arXiv preprint arXiv:2408.08921*.

Yunqi Qiu, Yuanzhuo Wang, Xiaolong Jin, and Kun Zhang. 2020. Stepwise reasoning for multi-relation question answering over knowledge graph with weak supervision. In *Proceedings of the 13th international conference on web search and data mining*, pages 474–482.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*, pages 593–607. Springer.

Priyanka Sen, Amir Saffari, and Armin Oliya. 2021. Expanding end-to-end question answering on differentiable knowledge graphs with intersection. *arXiv preprint arXiv:2109.05808*.

Jiaxin Shi, Shulin Cao, Lei Hou, Juanzi Li, and Hanwang Zhang. 2021. Transfernet: An effective and transparent framework for multi-hop question answering over relation graph. *arXiv preprint arXiv:2104.07302*.

Haitian Sun, Tania Bedrax-Weiss, and William W Cohen. 2019. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. *arXiv preprint arXiv:1904.09537*.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4231–4242. Association for Computational Linguistics.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-graph: Deep and

responsible reasoning of large language model with knowledge graph. In *International Conference on Learning Representations*.

Yawei Sun, Lingling Zhang, Gong Cheng, and Yuzhong Qu. 2020. Sparqa: skeleton-based semantic parsing for complex questions over knowledge bases. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8952–8959.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*.

Yiming Tan, Dehai Min, Yu Li, Wenbo Li, Nan Hu, Yongrui Chen, and Guilin Qi. 2023. Can chatgpt replace traditional kbqa models? an in-depth analysis of the question answering performance of the gpt llm family. In *International Semantic Web Conference*, pages 348–367. Springer.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Yijun Tian, Huan Song, Zichen Wang, Haozhu Wang, Ziqing Hu, Fang Wang, Nitesh V Chawla, and Panpan Xu. 2024. Graph neural prompting with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19080–19088.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International conference on machine learning*. PMLR.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Junxing Wang, Xinyi Li, Zhen Tan, Xiang Zhao, and Weidong Xiao. 2021. Relation-aware bidirectional path reasoning for commonsense question answering. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 445–453.

Keheng Wang, Feiyu Duan, Sirui Wang, Peiguang Li, Yunsen Xian, Chuantao Yin, Wenge Rong, and Zhang Xiong. 2023. Knowledge-driven cot: Exploring faithful reasoning in llms for knowledge-intensive question answering. *arXiv preprint arXiv:2308.13259*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yanbin Wei, Qiushi Huang, James T Kwok, and Yu Zhang. 2024. Kicgpt: Large language model with knowledge in context for knowledge graph completion. *arXiv preprint arXiv:2402.02389*.

Yilin Wen, Zifeng Wang, and Jimeng Sun. 2023. Mindmap: Knowledge graph prompting sparks graph of thoughts in large language models. *arXiv preprint arXiv:2308.09729*.

Yike Wu, Nan Hu, Guilin Qi, Sheng Bi, Jie Ren, Anhuan Xie, and Wei Song. 2023. Retrieve-rewrite-answer: A kg-to-text enhanced llms framework for knowledge graph question answering. *arXiv preprint arXiv:2309.11206*.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle, Ansong Ni, Ziyu Yao, Dragomir Radev, Caiming Xiong, Lingpeng Kong, Rui Zhang, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2022. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. *EMNLP*.

Michihiro Yasunaga, Antoine Bosselut, Hongyu Ren, Xikun Zhang, Christopher D Manning, Percy S Liang, and Jure Leskovec. 2022. Deep bidirectional language-knowledge graph pretraining. *Advances in Neural Information Processing Systems*, 35:37309–37323.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2022. RNG-KBQA: Generation augmented iterative ranking for knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6032–6043. Association for Computational Linguistics.

Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the Joint Conference of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing of the AFNLP*.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Wang, Zhiguo Wang, and Bing Xiang. 2022. Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases. *arXiv preprint arXiv:2210.00063*.

Geng Zhang, Jin Liu, Guangyou Zhou, Kunsong Zhao, Zhiwen Xie, and Bo Huang. 2024a. Question-directed reasoning with relation-aware graph attention network for complex question answering over knowledge graph. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*.

Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. 2022a. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. *arXiv preprint arXiv:2202.13296*.

Qinggang Zhang, Junnan Dong, Hao Chen, Xiao Huang, Daochen Zha, and Zailiang Yu. 2023a. Knowgpt: Black-box knowledge injection for large language models. *arXiv preprint arXiv:2312.06185*.

Qixuan Zhang, Xinyi Weng, Guangyou Zhou, Yi Zhang, and Jimmy Xiangji Huang. 2022b. Arl: An adaptive reinforcement learning framework for complex question answering over knowledge base. *Information Processing & Management*, 59(3):102933.

Tianjun Zhang, Shishir G Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, and Joseph E Gonzalez. 2024b. Raft: Adapting language model to domain specific rag. *arXiv preprint arXiv:2403.10131*.

Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D Manning, and Jure Leskovec. 2022c. Greaselm: Graph reasoning enhanced language models. In *International Conference on Learning Representations*.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023b. Siren's song in the ai ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. 2022. Learning on large-scale text-attributed graphs via variational inference. *arXiv preprint arXiv:2210.14709*.

Ruilin Zhao, Feng Zhao, Long Wang, Xianzhi Wang, and Guandong Xu. 2024. Kg-cot: Chain-of-thought prompting of large language models over knowledge graphs for knowledge-aware question answering. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI-24)*, pages 6642–6650. International Joint Conferences on Artificial Intelligence.

Appendix

A GNN-RAG Implementation

Classification layer: After L GNN layers, we obtain node representation matrix $\boldsymbol{H}^{(L)} \in \mathbb{R}^{|\mathcal{V}| \times d}$. To perform classification, we obtain the node probability matrix $P = \text{softmax}(\boldsymbol{H}^{(L)}\boldsymbol{W})$, where $\boldsymbol{W} \in R^{d \times 1}$ is a learnable projection layer followed by softmax normalization. Answer nodes should have larger probability $p_v \in [0, 1]$ than non-answer nodes.

Node and relation embeddings: We use pretrained models, such as SBERT or other LMs, to encode relation embeddings. We obtain node embeddings by aggregating the adjacent relation embeddings of nodes, which has been shown to generalize better to new entities (He et al., 2021; Choi et al., 2024). The formula is $\boldsymbol{h}_v^{(0)} = ReLU(\sum_{\boldsymbol{r} \in N_r(v)} \boldsymbol{W}_r \boldsymbol{r})$, where $\boldsymbol{r}$ is the relation embedding and $\boldsymbol{W}$ is learnable. During training, we optimize the GNN parameters, but not the relation embeddings obtained via the pretrained models.

Question Embeddings: As complex questions might consist of multiple subquestions, we obtain K question embeddings to better capture different question parts (Qiu et al., 2020), as shown in Equation 3. To capture multiple question's contexts, each question representation $\boldsymbol{q}_k \in \mathbb{R}^d, k \in K$, is initialized by dynamically attending to different question's tokens. First, we derive a representation $\boldsymbol{q}_j \in \mathbb{R}^d$ for each token j of the question and a question representation, e.g., via CLS pooling, $\boldsymbol{q}_c \in \mathbb{R}^d$ with pre-trained language models, such as SBERT. Equation 3 becomes

$$\boldsymbol{q}_k = \gamma_k(\text{LM}(q)) = \sum_j a_{k,j} \boldsymbol{q}_j, \tag{7}$$

where j denotes is the j-th token position and $a_{k,j} \in [0, 1]$ is an attention weight. At each iteration k, weight $a_{k,j}$ is dynamically adjusted by encouraging attention to new question parts via:

$$a_{k,j} = \text{softmax}_j(\boldsymbol{W}_a(\tilde{\boldsymbol{q}_k} \odot \boldsymbol{q}_j) \tag{8}$$

$$\tilde{\boldsymbol{q}_k} = \boldsymbol{W}_k(\boldsymbol{q}_{k-1} || \boldsymbol{q}_c || \boldsymbol{q}_{k-1} \odot \boldsymbol{q}_c || \boldsymbol{q}_c - \boldsymbol{q}_{k-1}), \tag{9}$$

where $\boldsymbol{W}_a \in \mathbb{R}^{d \times d}$ and $\boldsymbol{W}_k \in \mathbb{R}^{d \times 4d}$ are learnable parameters.

Downstream LLM: For the downstream LLM, we opt to follow prompt tuning (Lin et al., 2023;

Zhang et al., 2024b), where the Llama-7B-Chat model is trained to generate to generate a list of answers for KGQA. We follow RoG (Luo et al., 2024a) fine-tuning approach. The LLM is fine-tuned based on the training question-answer pairs to generate a list of correct answers, given the prompt:

"`Based on the reasoning paths, please answer the given question.\n Reasoning Paths: {Reasoning Paths} \n Question: {Question}`".

The reasoning paths are verbalized as "`{question entity}` → `{relation}` → `{entity}` → ⋯ → `{relation}` → `{answer entity} \n`" (see Figure 3).

During training, the reasoning paths are the shortest paths from question entities to answer entities. During inference, the reasoning paths are obtained by GNN-RAG.

B Case Studies

Figure 4 illustrates two case studies from the CWQ dataset, showing how GNN-RAG improves LLM's faithfulness, i.e., how well the LLM follows the question's instructions and uses the right information from the KG.

In both cases, GNN-RAG retrieves multi-hop information, which is necessary for answering the questions correctly. In the first case, GNN-RAG retrieves both crucial facts `<Gilfoyle → characters_that_have_lived_here → Toronto>` and `<Toronto → province.capital → Ontario>` that are required to answer the question, unlike the KG-RAG baseline (RoG) that fetches only the first fact. In the second case, the KG-RAG baseline incorrectly retrieves information about `<Erin Brockovich → person>` and not `<Erin Brockovich → film_character>` that the question refers to. GNN-RAG uses GNNs to explore how `<Erin Brockovich>` and `<Michael Renault Mageau>` entities are related in the KG, resulting into retrieving facts about `<Erin Brockovich → film_character>`. The retrieved facts include important information `<films_with_this_crew_job → Consultant>`.

Figure 5 illustrates one case study from the WebQSP dataset, showing how RA (Section 4.3) improves GNN-RAG. Initially, the GNN does not retrieve helpful information due to its limitation to understand natural language, i.e., that `<jurisdiction.bodies>` usually "`make the laws`". GNN-RAG+RA retrieves the right information, helping the LLM answer the question correctly.

Table 8: Datasets statistics. "avg.$|\mathcal{V}_q|$" denotes average number of entities in subgraph, and "coverage" denotes the ratio of at least one answer in subgraph.

Datasets	Train	Dev	Test	avg. $\lvert\mathcal{V}_q\rvert$	coverage (%)
WebQSP	2,848	250	1,639	1,429.8	94.9
CWQ	27,639	3,519	3,531	1,305.8	79.3
MetaQA-3	114,196	14,274	14,274	497.9	99.0

C Experimental Setup

KGQA Datasets. We experiment with two widely used KGQA benchmarks: WebQuestionsSP (WebQSP) (Yih et al., 2015), Complex WebQuestions 1.1 (CWQ) (Talmor and Berant, 2018). We also experiment with MetaQA-3 (Zhang et al., 2018) dataset. We provide the dataset statistics Table 8. **WebQSP** contains 4,737 natural language questions that are answerable using a subset Freebase KG (Bollacker et al., 2008). This KG contains 164.6 million facts and 24.9 million entities. The questions require up to 2-hop reasoning within this KG. Specifically, the model needs to aggregate over two KG facts for 30% of the questions, to reason over constraints for 7% of the questions, and to use a single KG fact for the rest of the questions. **CWQ** is generated from WebQSP by extending the question entities or adding constraints to answers, in order to construct more complex multi-hop questions (34,689 in total). There are four types of questions: composition (45%), conjunction (45%), comparative (5%), and superlative (5%). The questions require up to 4-hops of reasoning over the KG, which is the same KG as in WebQSP. **MetaQA-3** consists of more than 100k 3-hop questions in the domain of movies. The questions were constructed using the KG provided by the WikiMovies (Miller et al., 2016) dataset, with about 43k entities and 135k triples. For MetaQA-3, we use 1,000 (1%) of the training questions.

Implementation. For subgraph retrieval, we use the linked entities to the KG provided by (Yih et al., 2015) for WebQSP, by (Talmor and Berant, 2018) for CWQ. We obtain dense subgraphs by (He et al., 2021). It runs the PageRank Nibble (Andersen et al., 2006) (PRN) method starting from the linked entities to select the top-m ($m = 2,000$) entities to be included in the subgraph.

For GNN training, we employ standard train-

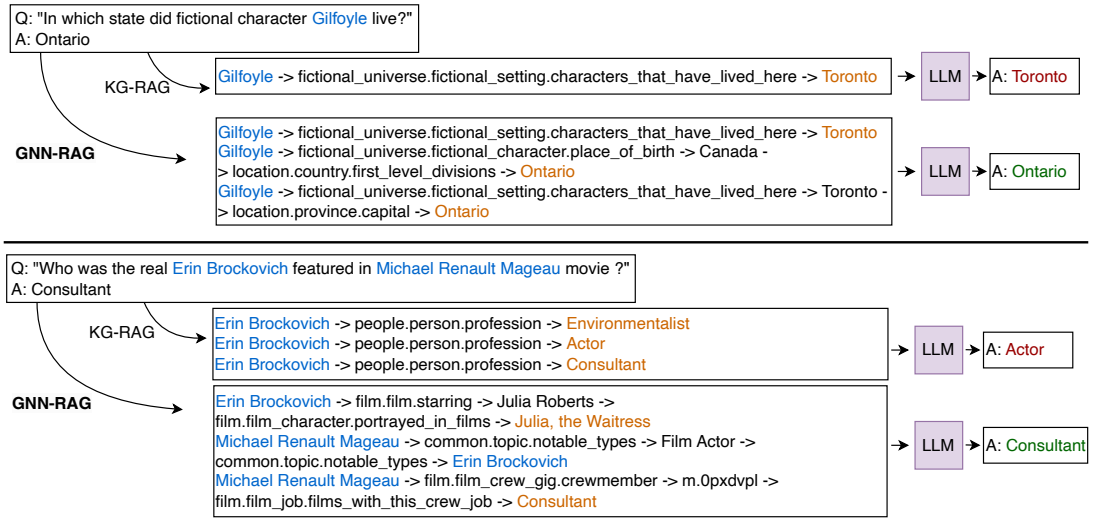

Figure 4: Two case studies that illustrate how GNN-RAG improves the LLM's faithfulness. In both cases, GNN-RAG retrieves *multi-hop* information that is necessary for answering the complex questions.

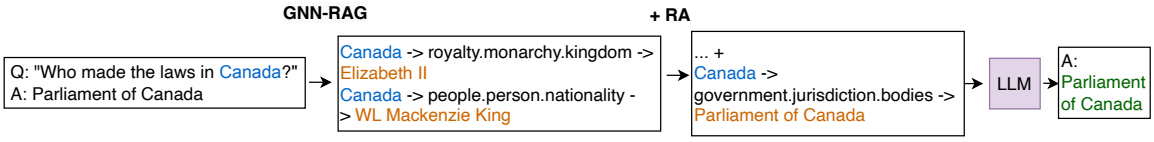

Figure 5: One case study that illustrates the benefit of retrieval augmentation (RA). RA uses LLMs to fetch semantically relevant KG information, which may have been missed by the GNN.

ing procedure and hyperparameters as in the literature (Mavromatis and Karypis, 2022). As the defualt downstream LLM, we use the Llama2-Chat-7B model finetuned for KGQA (Luo et al., 2024a). For both training and inference, we use suggested hyperparameters, without performing further hyperparameter search. Model selection is performed based on the validation data. Experiments with GNNs were performed on a Nvidia Geforce RTX-3090 GPU over 128GB RAM machine. Experiments with LLMs were performed on 4 A100 GPUs connected via NVLink and 512 GB of memory. The experiments are implemented with PyTorch.

For LLM prompting during KGQA, we use the following prompt:

```
Based on the reasoning paths, please
answer the given question. Please keep
the answer as simple as possible and
return all the possible answers as a
list.\n
Reasoning Paths: {Reasoning Paths} \n
Question: {Question}
```

During GNN inference, each node in the subgraph is assigned a probability of being the correct answer, which is normalized via softmax. To retrieve answer candidates, we sort the nodes based on the their probability scores, and select the top nodes whose cumulative probability score is below a threshold. We set the threshold to 0.95. To retrieve the shortest paths between the question entities and answer candidates for RAG, we use the NetworkX library[1].

Competing Approaches.

We evaluate the following categories of methods: 1. Embedding, 2. GNN, 3. LLM, 4. KG+LLM, 5. GNN+LLM, 6. Long-context (KG-LC).

[1]https://networkx.org/

1. KV-Mem (Miller et al., 2016) is a key-value memory network for KGQA. Embed-KGQA (Saxena et al., 2020) utilizes KG pre-trained embeddings (Trouillon et al., 2016) to improve multi-hop reasoning. TransferNet (Shi et al., 2021) improves multi-hop reasoning over the relation set. Rigel (Sen et al., 2021) improves reasoning with questions of multiple entities.

2. GraftNet (Sun et al., 2018) uses a convolution-based GNN (Kipf and Welling, 2016). PullNet (Sun et al., 2019) is built on top of GraftNet, but learns which nodes to retrieve via selecting shortest paths to the answers. NSM (He et al., 2021) is the adaptation of GNNs for KGQA. NSM+h (He et al., 2021) improves NSM for multi-hop reasoning. SQALER (Atzeni et al., 2021) learns which relations (facts) to retrieve during KGQA for GNN reasoning. Similarly, SR+NSM (Zhang et al., 2022a) proposes a relation-path retrieval. UniKGQA (Jiang et al., 2023b) unifies the graph retrieval and reasoning process with a single LM. ReaRev (Mavromatis and Karypis, 2022) explores diverse reasoning paths in a multi-stage manner.

3. We experiment with instruction-tuned LLMs. Flan-T5 (Chung et al., 2024) is based on T5, while Aplaca (Taori et al., 2023) and LLaMA2-Chat (Touvron et al., 2023) are based on LLaMA. ChatGPT[2] is a powerful closed-source LLM that excels in many complex tasks. ChatGPT+CoT uses the chain-of-thought (Wei et al., 2022) prompt to improve the ChatGPT. We access ChatGPT '`gpt-3.5-turbo`' through its API (as of May 2024).

4. KD-CoT (Wang et al., 2023) enhances CoT prompting for LLMs with relevant knowledge from KGs. StructGPT (Jiang et al., 2023a) retrieves KG facts for RAG. KB-BINDER (Li et al., 2023) enhances LLM reasoning by generating logical forms of the questions. ToG (Sun et al., 2024) uses a powerful LLM to select relevant facts hop-by-hop. RoG (Luo et al., 2024a) uses the LLM to generate relation paths for better planning.

[2]https://openai.com/blog/chatgpt

Table 9: Performance analysis (F1) based on the number of maximum hops that connect question entities to answer entities.

Method	WebQSP			CWQ		
	1 hop	2 hop	≥3 hop	1 hop	2 hop	≥3 hop
RoG	73.4	63.3	–	50.4	60.7	40.0
GNN-RAG	72.0	69.8	–	47.4	69.4	51.8
GNN-RAG +RA	74.6	71.1	–	48.2	70.9	47.7

Table 10: Performance analysis (F1) based on the number of answers (#Ans).

Method	WebQSP				CWQ			
	#Ans=1	2≤#Ans≤4	5≤#Ans≤9	#Ans≥10	#Ans=1	2≤#Ans≤4	5≤#Ans≤9	#Ans≥10
RoG	67.89	79.39	75.04	58.33	56.90	53.73	58.36	43.62
GNN-RAG	71.24	76.30	74.06	56.28	60.40	55.52	61.49	50.08
GNN-RAG +RA	71.16	82.31	77.78	57.71	62.09	56.47	62.87	50.33

5. G-Retriever (He et al., 2024) augments LLMs with GNN-based prompt tuning.

6. SubgraphRAG (Li et al., 2024b) uses a text encoder to encode KG triplets and trains an MLP classifier to select the top$k$ triplets based on the question.

D Additional Experimental Results

D.1 Question Analysis

Following the case studies presented in Figure 4 and Figure 5, we provide numerical results on how GNN-RAG improves multi-hop question answering and how retrieval augmentation (RA) enhances simple hop questions. Table 9 summarizes these results. GNN-RAG improves performance on multi-hop questions (≥2 hops) by 6.5–11.8% F1 points over RoG. Furthermore, RA improves performance on single-hop questions by 0.8–2.6% F1 points over GNN-RAG.

Table 10 presents results with respect to the number of correct answers. As shown, RA enhances GNN-RAG in almost all cases as it can fetch correct answers that might have been missed by the GNN.

D.2 Prompt Ablation

When using RAG, LLM performance depends on the prompts used. To ablate on the prompt impact, we experiment with the following prompts:

Prompt A:

Table 11: Performance comparison (%Hit) based on different input prompts.

		WebQSP	CWQ
Prompt A	RoG	84.8	56.4
	GNN-RAG	86.8	62.9
Prompt B	RoG	84.3	55.2
	GNN-RAG	85.2	61.7
Prompt C	RoG	81.6	51.8
	GNN-RAG	84.4	59.4

```
Based on the reasoning paths,
please answer the given question.
Please keep the answer as simple
as possible and return all the
possible answers as a list.\n
Reasoning Paths: {Reasoning
Paths} \n
Question: {Question}
```

Prompt B:

```
Based on the provided knowledge,
please answer the given question.
Please keep the answer as simple
as possible and return all the
possible answers as a list.\n
Knowledge: {Reasoning Paths} \n
Question: {Question}
```

Prompt C:

```
Your tasks is to use the
following facts and answer the
question. Make sure that you use the
information from the facts provided.
Please keep the answer as simple as
possible and return all the possible
answers as a list.\n
The facts are the following:
{Reasoning Paths} \n
Question: {Question}
```

We provide the results based on different input prompts in Table 11. As the results indicate, GNN-RAG outperforms RoG in all cases, being robust at the prompt selection.

D.3 Effect of Training Data

Training Cost. GNN-RAG requires only fine-tuning the GNN for retrieval. The downstream LLM can be fine-tuned (our default implementation) or not (as we experimented with in Table 5). Fine-tuning the downstream LLM is memory-intensive. For example, if we use 2 A100-80G GPUs, 1 epoch of 30k training data requires more than 12 hours. GNN training is much more efficient: On a GeForce RTX 3090, 1 epoch of 30k training data needs less than 15 minutes and less than 8GB of GPU memory.

Data Size Impact. Fine-tuning the downstream LLM generally improves performance. In Table 14, we compare LLaMa2-Chat-7B and LLaMa2-Chat-7B fine-tuned. As shown (Hit metric), GNN-RAG demonstrates a more stable performance when switching between the two LLMs. Specifically, GNN-RAG experiences a relatively small drop of 0.5-5.0 points, whereas RoG suffers from a larger performance degradation of 0.9-6.2 points under the same conditions. CWQ has more data (27.6k) than WebQSP (2.8k) and thus, performance improvement when using the tuned LLM is larger.

In Table 15, we provide results when we use 10k training data of CWQ when training the GNN. As shown, although GNN-RAG uses approximately 3x less data, it still outperforms RoG (which uses 30k data from both CWQ and WebQSP for training).

Table 12 compares performance of different methods based on the training data used for training the retriever and the KGQA model. For example, GNN-RAG trains a GNN model for retrieval and uses a LLM for KGQA, which can be fine-tuned or not. As the results show, GNN-RAG outperforms the competing methods (RoG and UniKGQA) by either fine-tuning the KGQA model or not, while it uses the same or less data for training its retriever.

D.4 Graph Effect

GNNs operate on dense subgraphs, which might include noisy information. A question that arises is whether removing irrelevant information from the subgraph would improve GNN retrieval. We experiment with SR (Zhang et al., 2022a), which learns to prune question-irrelevant facts from the KG. As shown in Table 13, although SR can improve the GNN reasoning results – see row (a) vs. (b) at CWQ –, the retrieval effectiveness deteriorates; rows (c) and (d). After examination, we found that the sparse subgraph may contain disconnected KG parts. In this case, GNN-RAG's extraction of the shortest paths fails, and GNN-RAG returns empty KG information.

D.5 Threshold Ablation

As an additional ablation study, we set the threshold θ, which controls the number of candidate answer nodes for entity selection, to 0.99 (retrieves

Table 12: Performance results based on different training data.

Method	WebQSP			CWQ		
	Training Data (Retriever)	Training Data (KGQA Model)	Hit	Training Data (Retriever)	Training Data (KGQA Model)	Hit
UniKGQA	WebQSP	WebQSP	77.2	CWQ	CWQ	51.2
RoG	WebQSP	WebQSP	81.5	CWQ	CWQ	59.1
	WebQSP+CWQ	None	84.8	WebQSP+CWQ	None	56.4
	WebQSP+CWQ	WebQSP+CWQ	85.7	WebQSP+CWQ	WebQSP+CWQ	62.6
GNN-RAG	WebQSP	None	86.8	CWQ	None	62.9
	WebQSP	WebQSP+CWQ	**87.2**	CWQ	WebQSP+CWQ	**66.8**

Table 13: Performance comparison on different subgraphs.

Retriever	KGQA Model	WebQSP			CWQ		
		Hit	H@1	F1	Hit	H@1	F1
a) Dense Subgraph	(A) GNN	–	77.5	72.8	–	52.7	49.1
b) Sparse Subgraph (Zhang et al., 2022a)	(B) GNN	–	74.2	69.8	–	53.3	49.7
c) GNN-RAG: (A)	LLaMA2-Chat-7B (tuned)	85.0	80.3	71.5	66.2	61.3	58.9
d) GNN-RAG: (B)		83.4	78.9	69.8	60.6	55.6	53.3

Table 14: Impact of LLM tuning.

Retrieval	LLM	WebQSP	CWQ
RoG	LLaMa2-Chat-7B (untuned)	84.8	56.4
RoG	LLaMa2-Chat-7B (tuned)	85.7	62.6
GNN-RAG	LLaMa2-Chat-7B (untuned)	85.2	62.7
GNN-RAG	LLaMa2-Chat-7B (tuned)	85.7	66.7

Table 15: Number of training data impact on CWQ.

Retrieval	# Training Data	CWQ Hit (%)
RoG	30k	62.6
GNN-RAG	27.6k	66.7
GNN-RAG	10k	63.7

more candidate answers), to 0.95 (default), and to 0.75 (retrieves less candidate answers). GNN-RAG performance is shown in Table 16. Increasing the threshold (0.99) to retrieve more context, can further increase performance to 85.9%. Lower threshold (0.75) might miss some answers and the performance drops to 83.5%.

Table 16: Threshold θ impact for answer node selection (WebQSP Hit %).

	$\theta = 0.99$	$\theta = 0.95$	$\theta = 0.75$
GNN-RAG	85.9	85.7	83.8

D.6 Text Retrieval

Table 17 presents retrieval performance against additional text-based baselines. We conduct experiments comparing GNN-RAG with triplet embedding similarity methods, Retrieve-Rewrite-Answer (), and UniHGKR-base () (KG instruction, top-50) on WebQSP. As Table 17 demonstrates, GNN-RAG outperforms these methods in both retrieval and downstream QA metrics.

Table 17: GNN-RAG vs. text-based retrieval methods.

	Ret. Hit@1 (%)	Ret. Hit@10 (%)	QA Hit (%)
UniHGKR	24.5	56.0	76.3
Ret-Rewrite-Ans	50.7	85.9	81.1
GNN-RAG	76.5	92.3	85.7