GRAG: Graph Retrieval-Augmented Generation

Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, Liang Zhao
Department of Computer Science
Emory University
Atlanta, GA 30322, USA
{yuntong.hu,liang.zhao}@emory.edu

Abstract

Naive Retrieval-Augmented Generation (RAG) focuses on individual documents during retrieval and, as a result, falls short in handling networked documents which are very popular in many applications such as citation graphs, social media, and knowledge graphs. To overcome this limitation, we introduce Graph Retrieval-Augmented Generation (GRAG), which tackles the fundamental challenges in retrieving textual subgraphs and integrating the joint textual and topological information into Large Language Models (LLMs) to enhance its generation. To enable efficient textual subgraph retrieval, we propose a novel divide-and-conquer strategy that retrieves the optimal subgraph structure in linear time. To achieve graph context-aware generation, incorporate textual graphs into LLMs through two complementary views—the text view and the graph view—enabling LLMs to more effectively comprehend and utilize the graph context. Extensive experiments on graph reasoning benchmarks demonstrate that in scenarios requiring multi-hop reasoning on textual graphs, our GRAG approach significantly outperforms current state-of-the-art RAG methods.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities in a variety of reasoning tasks, including on graph-based data (Hu et al., 2023b; Chen et al., 2024; Fatemi et al., 2023). However, LLMs themselves struggle with factual errors due to limitations in their training data and a lack of real-time knowledge (Mallen et al., 2023; Min et al., 2023). Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Guu et al., 2020; Ram et al., 2023; Gao et al., 2023), which integrates external data retrieval into the generative process, has been widely used to help LLMs access relevant information from external sources to generate more relevant answers and hence reduce factual errors (Tang and Yang, 2024). Naive RAG approaches focus solely on individual documents and retrieve relevant ones based on text similarity. However, real-world documents, such as social media postings, research papers, knowledge items, and product reviews, are typically not isolated but networked as *textual graphs* (He et al., 2023; Jin et al., 2023; Li et al., 2023). Importantly, such network information is typically crucial in both retrieving relevant documents and prompting LLMs for text generation (Yang et al., 2024; Tang and Yang, 2024). For example, research papers form a citation graph, so when a solar physicist wants to learn state-of-the-art techniques in solar flare prediction, paper mutual referencing links need to be considered to pursue comprehensive retrieval coverage and insightful technical evolution understanding of this research community (as shown in Figure 1). Similarly, social interaction among social media postings, entity relations in knowledge graphs, and purchasing relations in product review systems are indispensable when LLMs want to leverage these external data. So the question is how LLMs could harness this type of networked documents when performing RAG?

To address it, we propose **Graph Retrieval-Augmented Generation (GRAG)**, which extends beyond the traditional RAG method to incorporate graph context. Unlike RAG, which focuses on individual documents during retrieval and generation, GRAG requires to consider the networking of documents in both stages, leading to two fundamental challenges: *1) For retrieval: How to efficiently retrieve relevant textual subgraph?* Textual subgraph retrieval is particularly challenging due to the high dimensionality of textual features within nodes and edges. *2) For generation: How to deliver textual subgraph's joint textual and topological information into LLMs?* The generation phase poses additional complexities, as it requires effectively passing networked documents to LLMs while pre-

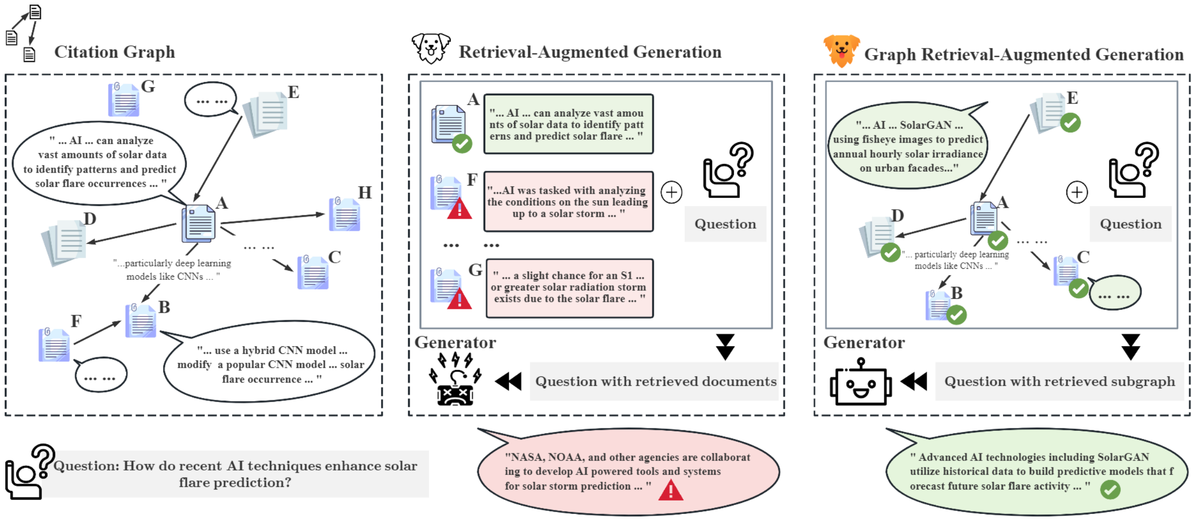

Citation Graph
G
E
" ... AI ... can analyze vast amounts of solar data to identify patterns and predict solar flare occurrences ... "
... ...
A
D
H
"...particularly deep learning models like CNNs ... "
... ...
C
F
B
"... use a hybrid CNN model ... modify a popular CNN model ... solar flare occurrence ... "
... ...
Retrieval-Augmented Generation
A
"... AI ... can analyze vast amounts of solar data to identify patterns and predict solar flare ... "
F
"...AI was tasked with analyzing the conditions on the sun leading up to a solar storm ... "
... ...
G
" ... a slight chance for an S1 ... or greater solar radiation storm exists due to the solar flare ... "
Question
Generator
Question with retrieved documents
Graph Retrieval-Augmented Generation
"... AI ... SolarGAN ... using fisheye images to predict annual hourly solar irradiance on urban facades..."
E
A
D
"... particularly deep learning models like CNNs ... "
... ...
C
B
... ...
Question
Generator
Question with retrieved subgraph
Question: How do recent AI techniques enhance solar flare prediction?
"NASA, NOAA, and other agencies are collaborating to develop AI powered tools and systems for solar storm prediction ... "
" Advanced AI technologies including SolarGAN utilize historical data to build predictive models that forecast future solar flare activity ... "

Figure 1: GRAG retrieves textual subgraphs relevant to the query, rather than discrete entities as in RAG. Entities with similar topics tend to have connections, which improves the precision and robustness of the retrieval phase.

serving both textual and topological information, along with their interdependencies.

To address these two challenges, we propose a computational framework for GRAG. Specifically, to achieve efficient textual subgraph retrieval, we propose a new divide-and-conquer strategy that breaks the high-dimensional combinatorial optimization problem into first retrieving the most relevant ego-graphs and then refining and unioning them with a graph soft pruning mechanism. This provides an approximate solution to identifying the most relevant textual subgraph structure, thereby avoiding the NP-hard problem of exhaustively searching all subgraphs (Johnson and Garey, 1979). To integrate the retrieved textual subgraphs into LLMs, we feed the LLM both text view via hard prompts (*text tokens*) and graph view via soft prompts (*graph tokens*). Retrieved textual subgraphs are transformed into hierarchical text descriptions by our proposed graph algorithms to form hard prompts, which encode the topological information in texts. Soft prompts are generated by encoding the graph's topological information directly via graph encoders, encoding text information as node/edge attributes in graphs. Finally, the generation process is guided by both hard and soft prompts for LLM to gain a deeper understanding of the relationships between entities, leading to responses that are well-aligned with the underlying textual graph context.

Empirical results on multi-hop graph reasoning tasks demonstrate that our GRAG approach significantly outperforms RAG-based retrievers and LLM baselines in graph reasoning scenarios. In particular, Frozen LLM with GRAG outperforms fine-tuned LLM on all tasks.

The main contributions of this article are summarized as follows:

- We formulate the problem of Graph Retrieval-Augmented Generation (GRAG) and propose an efficient computational framework for GRAG, addressing the limitations of RAG methods in handling graph-based contexts.
- We propose a novel prompting method that converts textual graphs into hierarchical text descriptions without losing topological information.
- We propose an approximate solution for retrieving the most relevant textual subgraphs, efficiently avoiding the NP-hard problem of exhaustive subgraph searches.
- Extensive experiments on graph multi-hop reasoning benchmarks demonstrate that GRAG significantly outperforms current state-of-the-art RAG methods in graph-related scenarios.

2 Related Work

2.1 Prompt Tuning

Unlike traditional fine-tuning methods, such as Low-rank Adaptation (LoRA) (Hu et al., 2021), which require updating a model's parameters, prompt tuning focuses on modifying inputs to guide the model's responses more effectively (Liu et al., 2023; Jia et al., 2022). Approaches like AutoPrompt (Shin et al., 2020) and Prompt Tuning (Lester et al., 2021) have introduced automated techniques for crafting effective prompts without manual intervention. In particular, Lester et al. propose learning soft prompts directly as embeddings, allowing task-specific adaptations while preserving the model's original parameters. Building on this foundation, recent studies have explored adapting prompt embeddings for multi-modal contexts (Zhou et al., 2022; Khattak et al., 2023; Yang et al., 2022; Ge et al., 2023), providing a flexible mechanism for integrating LLMs into diverse domains through prompt tuning.

2.2 LLMs in Graph Related Tasks

On the one hand, the text embedding capability of LLMs helps encode textual node & edge attributes, which directly benefits the classification task (Hu et al., 2023b; Chen et al., 2023, 2024) and knowledge graph creation (Trajanoska et al., 2023; Yao et al., 2023). On the other hand, the contextual reasoning capabilities of the LLM benefits the graph reasoning (Wang et al., 2024; Jiang et al., 2023; Luo et al., 2023) and graph answering in zero-shot scenarios (Baek et al., 2023; Hu et al., 2023a). While training on large text corpora enables LLMs to develop robust language understanding for simple graph structures, it does not inherently equip them to understand or reason about complex graph-structured data, as textual data lacks explicit topological information (Huang et al., 2023; Chen et al., 2024; Merrer and Trédan, 2024). Recently, graph prompt tuning (Perozzi et al., 2024; Tian et al., 2024) has emerged as a powerful tool to help LLMs process and comprehend topological information.

2.3 Retrieval on Graphs

Yasunaga et al. retrieve relevant nodes and create a joint graph that includes the QA context and the relevant nodes. Kang et al. and Kim et al. focus on retrieving triples rather than individual nodes and edges to capture more complex relational data. Particularly, some retrieval problems can be solved by reasoning chains, which can be simplified to retrieve the path between the question and the target entity (Lo and Lim, 2023; Choudhary and Reddy, 2023). Edge et al. leverage community detection algorithms to partition the graph into communities, then retrieve and aggregate relevant communities to generate the final answer to the query. Li et al. enhance retrieval processes by incorporating both textual and topological information, allowing models to better capture the structural relationships within graph-structured data.

3 Problem Formalization

Textual Graphs are graphs consisting of text-attributed nodes and edges, which can be formally defined as $G = (V, E, \{T_n\}_{n\in V}, \{T_e\}_{e\in E})$. V and E represent the node set and edge set. T_n and T_e represent the natural language attributes of the corresponding nodes and edges in the graph.

Textual Subgraphs are subgraph structures in a textual graph, e.g., G with finite node set V and edge set E, we have its subgraph set $\mathcal{S}(G) = \{g = (V', E', \{T_n\}_{n\in V'}, \{T_e\}_{e\in E'}) | V' \in \mathcal{P}(V), E' \in \mathcal{P}(E)\}$, where $\mathcal{P}(V)$ and $\mathcal{P}(E)$ represent the power set of V and E, respectively.

Graph Retrieval Augmented Generation (GRAG) aims to integrate graph context into both the retrieval and generation phases, improving the relevance of generated content to the knowledge embedded within the textual graph. Given a specific query q over a textual graph G, there exists an optimal textual subgraph $\hat{g} \in \mathcal{S}(G)$ that leads the LLM to generate answers that align with expectations, where $\mathcal{S}(G)$ denotes the set of all subgraphs of G. The objective of GRAG is to retrieve the optimal subgraph $\hat{g}$ and incorporate its information into an LLM$_\theta$ parameterized by θ to enhance the generation process. Formally, the probability distribution of the final output sequence Y is defined as follows:

$$p_\theta(Y \mid [q, G]) = \prod_{i=1}^{n} p_\theta(y_i \mid y_{<i}, [q, \hat{g}]), \quad (1)$$

where $y_{<i}$ represents the prefix tokens, and $[q, \hat{g}]$ indicates the concatenation of the query and optimal subgraph information, respectively.

4 Methodology

Overview. In this section, we introduce our solution of GRAG. As illustrated in Figure 2(a), **to address the challenge of textual subgraph retrieval**, we propose a divide-and-conquer strategy, based on the assumption that important subgraph consists of important nodes and some of their neighbors. specifically, we search for important ego-graphs. We then merge the top-N most relevant ego-graphs and perform soft pruning operations to reduce the impact of redundant nodes and edges, yielding an approximately optimal subgraph structure. In contrast to direct subgraph searching, which has a total search space of $2^{|V|+|E|}$, our retrieval-then-pruning approach ensures efficiency by limiting the retrieval

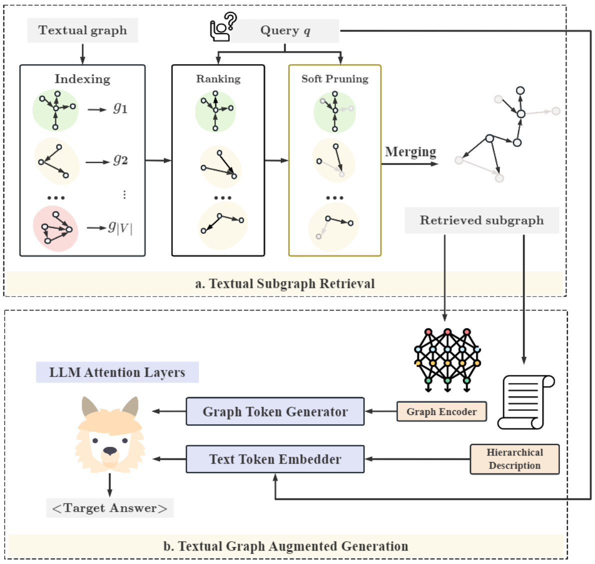


Figure 2: Illustration of our GRAG approach.

space to only $|V|$ ego-graphs. **To address the challenge of preserving both textual and topological information**, as shown in Figure 2(b), we pursue two complementary views of textual graphs: 1) *graph view of textual graphs*, learning representations of textual graphs as soft prompts to preserve how texts are connected, and 2) *text view of textual graphs*, which converts the textual graph into hierarchical text descriptions as hard prompts to retain how connections are narrated. We then present the detailed description of the retrieval and generation processes in the following sections.

4.1 Textual Subgraph Retrieval

Given a textual graph G, the optimal textual subgraph $\hat{g} \in S(G)$ should be retrieved to maximize generation quality. Formally, let $f(\cdot)$ be the function that evaluates generation quality based on the retrieved subgraph, such that $\max_{\hat{g}} f(\hat{g})$, this problem is NP-hard and to work around it, we pursue an alternative understanding of a retrieved textual subgraph. A retrieved textual subgraph can be considered as a union of the (partial) neighborhoods of a number of important nodes, which can be formulated as:

$$\max_{\hat{g}} f(\hat{g}) = \max_{V_{\text{key}}} f(\bigcup_{v \in V_{\text{key}}} G[\mathcal{N}_K^*(v)]), \quad (2)$$

where $G[\mathcal{N}_K^*(v)] \in S(G)$ denotes the induced subgraph on node v and its selected K-hop neighbors, i.e., $\mathcal{N}_K^*(v) \subseteq \mathcal{N}_K(v)$, and V_{key} represents the set of key nodes that form the backbone of $\hat{g}$. Hence, instead of the original NP-hard problem, we approach the problem in Equation 2 via a novel divide-and-conquer strategy that leverages the approximation:

$$\max_{V_{\text{key}}} f(\bigcup_{v \in V_{\text{key}}} G[\mathcal{N}_K^*(v)]) \approx \max_{V_{\text{key}}} \sum_{v \in V_{\text{key}}} f(G[\mathcal{N}_K^*(v)]) \quad (3)$$

Hence, solving the original problem turns into selecting the top-ranked key nodes to form V_{key} which has linear time complexity. More importantly, we can further accelerate it by first encoding the neighborhood surrounding each node in an offline manner. During textual subgraph retrieval, we can quickly index a pool of promising candidate subgraphs $\{G[\mathcal{N}_K(v)]\}$, from which we further rank, retain and refine the top-ranked ones. This process is followed by a learnable pruner that carves the selected neighborhoods into subgraphs that are relevant to the query and most beneficial to the task, i.e., $\{G[\mathcal{N}_K(v)]\} \rightarrow \{G[\mathcal{N}_K^*(v)]\}$.

Textual Subgraph Indexing. For any node v, $G[\mathcal{N}_K(v)]$ is equivalent to the K-hop ego-graph centered around v. Consequently, each K-hop ego-graph in G is assigned a unique identifier and subsequently pooled into a graph embedding. Specifically, we leverage a pre-trained language model (PLM)[1] to convert the text attributes of nodes and edges into embeddings. We then apply a mean pooling operation on these embeddings to obtain a graph embedding, denoted as $z_g \in \mathbb{R}^d$ for each subgraph $g \in \mathcal{S}(G)$, where d represents the dimension of the graph embedding:

$$z_g = \text{POOL}(\text{PLM}(\{T_n\}_{n \in V_g}, \{T_e\}_{e \in E_g})), \quad (4)$$

where V_g and E_g represent the node set and edge set of the subgraph g. All indexed embeddings are stored for the subsequent retrieval process.

Textual Subgraph Ranking. The same PLM encoder is used to encode the query as:

$$z_q = \text{PLM}(q) \in \mathbb{R}^d. \quad (5)$$

We then calculate the semantic relevance between the query and each K-hop ego-graph to find the top-N most relevant subgraphs:

$$\mathcal{S}_N(G) = \underset{g \in \mathcal{S}(G)}{\text{Top-}N} \cos(z_q, z_g), \quad (6)$$

where $\cos(\cdot, \cdot)$ represents the cosine similarity function. The subset $\mathcal{S}_N(G) \subseteq \mathcal{S}(G)$ contains the N

[1] SentenceBERT (Reimers and Gurevych, 2019) is used to encode the query and text attributes.

subgraphs with the highest semantic relevance to the query.

Textual Subgraph Soft Pruning. Although we retrieve relevant subgraphs, some irrelevant nodes and edges may still be present, which can negatively impact the final generation. Therefore, we leverage a soft pruning approach to minimize the influence of these irrelevant entities. Specifically, we use two Multilayer Perceptrons (MLPs) to learn a scaling factor based on the distance between nodes & edges and the query as follows:

$$z_n = \text{PLM}(T_n), \quad \alpha_n = \text{MLP}_{\phi_1}\left(z_n \ominus z_q\right), \quad (7)$$

$$z_e = \text{PLM}(T_e), \quad \alpha_e = \text{MLP}_{\phi_2}\left(z_e \ominus z_q\right), \quad (8)$$

where $\ominus$ represents the operator to measure the element-wise distance. This scalar adaptively mask some tokens of nodes& edges. The farther a node or edge is from the query, the closer its scalar value is to 0, effectively masking these nodes or edges. Finally, we merge the adaptively masked subgraphs in $\mathcal{S}_N(G)$ to obtain the optimal subgraph structure, $\hat{g}$, tailored to the query q, achieving this with linear complexity.

4.2 Textual Graph Augmented Generation

In this section, we introduce our approach to provide LLMs with two complementary views of a textual graph: *text view* and *graph view*.

Text View of Textual Graphs. LLMs demonstrate reasoning capabilities on graphs, particularly when interpreting texts organized in hierarchical structures, such as tree structures (Saad-Falcon et al., 2023). While representing retrieved textual subgraphs in a hierarchical structure helps preserve topological information, automating this transformation remains an open challenge. Here, we propose a novel algorithm that leverages graph and tree traversals to achieve this conversion. The distinction between an ego-graph and a tree lies in the presence of additional edges connecting nodes within the same level or across multiple levels, beyond the typical parent-child connections found in a hierarchical tree structure. To overcome this challenge, we split each retrieved ego-graph into two parts, denoted by $g = \mathcal{T}_g \cup \mathcal{E}_g$ where $\mathcal{T}_g$ indicates a partially ordered set that forms a tree rooted at the ego node and $\mathcal{E}_g$ is an edge set consisting of edges not included in the tree. We leverage Breadth-First Search (BFS) on each ego-graph to find its $\mathcal{T}_g$, and then $\mathcal{E}_g$ can be easily obtained. Afterwards, we perform pre-order traversal on $\mathcal{T}_g$ and append the texts of visited node & edge with a relation template. Then, we insert the texts of triples in $\mathcal{E}_g$ into the current hierarchical description. The final description of textual graph, denoted by D_g, retains both textual information and topological information with a hierarchical structure, enabling lossless conversion between the K-hop ego-graphs and text descriptions. An example of this interconversion is provided in the Appendix A.1. Finally, we provide the LLM with a concatenation of the query and the hierarchical description of the textual subgraph (i.e., $[q, D_g]$) as a hard prompt.

Graph View of Textual Graphs. We utilize a Graph Neural Network (GNN) to encode the graph's topological information. To minimize the influence of irrelevant entities on generation in the encoding process, we propose to learn the representation of the soft pruned subgraph as the soft prompt. This strategy controls the message passing in the graph encoder, GNN_Φ, through learned relevance scaling factors (α). Then, an MLP_{ϕ_3} is used to align the graph embeddings with the LLM tokens. This approach enables controlled message passing within GNN_Φ, guided by the relevance between nodes & edges and the query as follows:

$$m_u^{(l)} = \text{MSG}^{(l)}\left(\alpha_u \cdot h_u^{(l-1)}, \alpha_{uv} \cdot e_{uv}\right), \quad (9)$$

where $u \in \{\mathcal{N}(v) \cup v\}$, $h_u^{(0)} = z_n$ and $e_{uv} = z_{uv}$, $\mathcal{N}(v)$ represents the set of neighboring nodes of v, $h_u^{(l-1)}$ are the node features from the previous layer, e_{uv} denotes the attributes of the edge connecting nodes u and v, α_u and α_{uv} are scaling factors.

Generation Phase. The generation is guided by the retrieved subgraph $\hat{g}$ and the original query q. The modalities of these two prompts are not the same. Therefore, to bridge the gap between graph embeddings and the LLM_θ's text vector space, we use an MLP_{ϕ_3} to align the graph embeddings accordingly, as follows:

$$\mathbf{h}_{\hat{g}} = \text{MLP}_{\phi_3}\left(\text{GNN}_\Phi(\hat{g})\right) \in \mathbb{R}^{d_{\text{LLM}}}, \quad (10)$$

where d_{LLM} represents the dimension of the text vectors in LLM_θ. $\mathbf{h}_{\hat{g}}$ aggregates topological information to enhance LLM_θ's awareness of the graph's structure during the generation stage. We utilize the text embedder of LLM_θ to convert the hard prompt $[q, D_g]$ into text embeddings $\mathbf{h}_T$. The final generation Y is given as follows:

$$p_{\theta,\phi_1,\phi_2,\phi_3,\Phi}(Y|q,G) = p_{\theta,\phi_1,\phi_2,\phi_3,\Phi}(Y|q,\hat{g})$$
$$= \prod_{i=1}^{r} p_{\theta,\phi_1,\phi_2,\phi_3,\Phi}(y_i|y_{<i},[\mathbf{h}_{\hat{g}};\mathbf{h}_T]), \qquad (11)$$

where $[\cdot;\cdot]$ denotes the concatenation of token embeddings before feeding them through transformer layers of LLM$_\theta$.

5 Experiments

5.1 Experiment Setup

Datasets. We conduct experiments on the GraphQA benchmark (He et al., 2024). The statistics of the dataset is shown in Table 1. Each textual graph corresponds to at least one question-answer pair. Answering the question requires the LLM to comprehend the graph's context. WebQSP (Yih et al., 2016; Luo et al., 2023) is a large-scale, multi-hop knowledge graph QA dataset, while ExplaGraphs (Saha et al., 2021) is a common-sense reasoning dataset focused on predicting positions in debates.

Table 1: Average Dataset Statistics: the average number (#) of graphs, nodes, edges, and tokens.

Dataset	WebQSP	ExplaGraphs
# Graphs	4,700	2,766
# Nodes	1370.89	5.17
# Edges	4252.37	4.25
# Tokens	100,627	1,396

Evaluation Metrics. For the large-scale dataset WebQSP, we utilize the F_1 Score, Hit@1, and Recall metrics to comprehensively evaluate performance of models. For ExplaGraphs which focuses on common-sense reasoning, we employ Accuracy (Acc) as the primary metric.

Comparison Methods. To demonstrate the effectiveness of GRAG, we compare its performance to widely used retrievers on graph multi-hop reasoning tasks .We compare GRAG with RAG using different retrievers: BM25 (Robertson et al., 2009), MiniLM-L12-v2 (Reimers and Gurevych, 2019), LaBSE (Feng et al., 2022), mContriever (Izacard et al., 2021), E5 (Wang et al., 2022), and G-Retriever (He et al., 2024). Detailed introduction of comparison retrievers are presented in Appendix A.2. Additionally, we establish two LLM baselines without retrieved external knowledge: (1) a frozen LLM, and (2) a fine-tuned LLM using LoRA (Hu et al., 2021). The LLM used is the Llama2-7b model (Touvron et al., 2023). Detailed experimental settings are provided in Appendix A.3.

5.2 Main Results

Table 2 reports the overall results across datasets. We compare the performance of GRAG with comparison retrievers and baselines introduced in Section 5.1 and make the following key observations.

GRAG surpasses RAG and LLM baselines. Notably, GRAG significantly outperforms the fine-tuned LLM in all metrics across both datasets without fine-tuning the LLM. Fine-tuning offers only marginal performance gains when GRAG is employed, as evidenced by the limited improvement on the WebQSP dataset, with the Hit@1 metric increasing from 0.7236 to 0.7275. This suggests that GRAG is a more effective strategy for enhancing the graph reasoning capabilities of LLMs than mere fine-tuning. This can significantly reduce the cost of training LLMs for graph-related tasks.

Soft pruning boosts LLM performance in graph-related tasks. When all textual information from graphs is integrated into the prompt, the LLM exhibits suboptimal performance, even on the ExplaGraphs dataset, which features smaller graph sizes. This underscores the critical need to implement retrieval operations to mitigate the negative impact of redundant information in graphs. Notably, fine-tuning yields significant improvements in the performance of the LLM when reasoning on small graphs, with a notable increase from 33.94% to 89.27% accuracy on ExplaGraphs. However, the benefits of fine-tuning diminish with larger graph sizes, with Hit@1 on WebQSP only increasing from 0.4148 to 0.6186.

GRAG demonstrates strong transferability to transfer learned textual graph encoding capabilities across datasets. As shown in Table 3, when trained on a large dataset, GRAG can enhance generation on a smaller dataset using the trained model. Notably, GRAG trained on WebQSP on ExplaGraphs outperforms the naive LLM, with an accuracy improvement of 33.77%.

Larger LLMs don't necessarily outperform smaller ones in graph-related tasks without retrieval. Beyond the performance comparison of GRAG and RAG models, we evaluated the impact of LLM scale on graph-related tasks, specifically examining the 7B and 13B versions of the Llama model. Our findings indicate

Table 2: Performance comparison across `WebQSP` and `ExplaGraphs` datasets. **Bold** numbers indicate the best performance among all models. Highlight numbers demonstrate the performance improvement achieved by our GRAG approach compared to the LLM baselines.

Model	Prompt tuning	Fine-tuning	`WebQSP`			`ExplaGraphs`
			F_1 **Score** ↑	**Hit@1** ↑	**Recall** ↑	**Acc** ↑
`Baselines`						
LLM only	✗	✗	0.2555	0.4148	0.2920	0.3394
LLM$_{LoRA}$	✗	✓	0.4295	0.6186	0.4193	0.8927
`Compared Retrievers`						
BM25	✗	✗	0.2999	0.4287	0.2879	0.6011
MiniLM-L12-v2	✗	✗	0.3485	0.4730	0.3289	0.6011
LaBSE	✗	✗	0.3280	0.4496	0.3126	0.6011
mContriever-Base	✗	✗	0.3172	0.4453	0.3047	0.5866
E5-Base	✗	✗	0.3421	0.4705	0.3254	0.6011
G-Retriever	✓	✗	0.4674	0.6808	0.4579	0.8825
G-Retriever$_{LoRA}$	✓	✓	0.5023	0.7016	0.5002	0.9042
`Our Retrieval Approach`						
GRAG	✓	✗	0.5022	0.7236	0.5099	0.9223
Δ_{LLM}			↑ 96.56%	↑ 74.45%	↑ 74.62%	↑ 171.74%
GRAG$_{LoRA}$	✓	✓	**0.5041**	**0.7275**	**0.5112**	**0.9274**
Δ_{LoRA}			↑ 17.37%	↑ 17.60%	↑ 21.92%	↑ 3.89%

Table 3: Cross-Dataset Transfer Learning Performance.

Transferability	**Acc**	Δ_{LLM}
`WebQSP` → `ExplaGraphs`	0.4540	↑ 33.77%
	Hit@1	Δ_{LLM}
`ExplaGraphs` →`WebQSP`	0.4237	↑ 2.15%

that larger LLMs may underperform relative to smaller models. In the absence of retrieval techniques, larger LLMs fail to yield superior performance in these tasks. For example, the `llama2-7b-chat-hf` model achieves an accuracy of 33.94% on the commonsense reasoning task in the `ExplaGraphs` dataset, marginally outperforming the `llama2-13b-chat-hf` model, which records an accuracy of 33.57%. A similar trend is observed on the `WebQSP` dataset, where the 13B model's Hit@1 score of 0.4112 is slightly lower than the 0.4148 achieved by the 7B model.

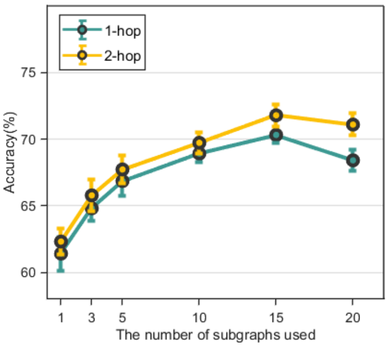


Figure 3: Performance of our GRAG approach on `WebQSP` as the ego-graph size and number of ego-graphs used vary.

5.3 Discussion

Impact of Subgraph Size K. The retrieval efficiency of our GRAG approach is preserved by constraining the search space to only $|V|$ K-hop ego-graphs. However, as the subgraph size K increases, a broader range of graph context is integrated during the generation process, resulting in longer training and inference times. Additionally, embeddings of larger subgraphs (i.e., over 3-hop) are more susceptible to oversmoothing, which can diminish their distinctiveness for retrieval. Therefore, the subgraph size must be carefully controlled to avoid excessive growth. Figure 3 shows the performance of GRAG on `WebQSP` as the number of 1-hop and 2-hop ego-graphs changes. With the same number of retrieved ego-graphs, using 2-hop ego-graphs consistently outperform using 1-hop ego-graphs. Increasing the number of retrieved subgraphs does not necessarily improve generation quality due to the introduction of more irrelevant information. A drop in performance is observed when the number of ego-graphs increases from 15 to 20. Moreover, using a larger number of subgraphs results in more robust generation, as indicated by a smaller standard deviation.

Evaluation on Hallucinations. We conduct a small-scale human evaluation of GRAG outputs to assess hallucinations. Specifically, we randomly select and manually review 100 samples from the `WebQSP` and `ExplaGraphs` results. The LLM is prompted to generate answers to the questions, along with the referenced nodes and edges. Following Menick et al. and He et al., human annotators evaluate whether the model output is reasonable and supported, verifying whether nodes and edges referenced in the output exist in the actual graph. GRAG's outputs reference 79% of valid entities in the graph, compared to MiniLM-L12-v2 and G-Retriever, which reference 62% and 71% of valid entities, respectively.

Thorough Comparisons with RAG. Overall, the LLM can generate better responses with retrieved entities by all tested retrievers. However, even if advanced retrievers use more data for training and increase the embedding dimension to obtain better embeddings, their focus remains exclusively on the text domain, creating a performance bottleneck as no topological information is retrieved. As shown in Table 2, when graph context is not considered, there is only a slight difference in the enhancement achieved by various retrievers. This phenomenon is further discussed in Appendix A.4. G-Retriever, which aggregates topological information as soft prompts, outperforms other retrievers, but it also fails to consider topology during the retrieval process. Our GRAG approach addresses this limitation by directly retrieving subgraphs instead of individual entities and incorporating topological information into the LLM during the generation phase, thereby achieving optimal performance on both datasets.

5.4 Ablation Study

We conducted a series of ablations to our GRAG framework to identify which components play a key role. We evaluate four model variants trained differently, where fine-tuning is used and 2-hop ego-graphs are retrieved in all settings: *w/o Retrieval* trains the LLM without retrieving subgraphs, instead providing the entire graph to the LLM. *w/o Graph Encoder* trains the LLM using the text on the retrieved textual subgraphs, but does not generate graph tokens to provide the graph context; *w/o Soft Pruning* indicates that irrelevant entities are not pruned when retrieved subgraphs are encoded to the graph tokens; *w/o Graph Description* trains the LLM without the hierarchical text descriptions of retrieved textual subgraphs. Table 4 shows the main results. Our main findings are as follows:

Importance of Graph Context. When the graph context is not encoded (*w/o Graph Encoder*), the LLM's generation quality significantly declines (Hit@1: 0.7275 → 0.5835). This suggests that merely describing relationships between nodes and edges in text is insufficient for LLMs to fully comprehend the graph context. Embedding the graph enables the LLM to capture the graph's context at a deeper level.

Table 4: Ablation study on `WebQSP`. Δ_{GRAG} represents the change in Hit@1 performance relative to our full GRAG approach.

Setting	Hit@1	Δ_{GRAG}
w/o Retrieval	0.6093	↓ 16.25%
w/o Graph Encoder	0.5835	↓ 19.79%
w/o Soft Pruning	0.5671	↓ 22.05%
w/o Graph Descriptions	0.4496	↓ 38.20%

Impact of Pruning. When irrelevant entities in retrieved textual subgraphs are not pruned (*w/o Soft Pruning*), the performance on `WebQSP` is worse compared to the *w/o Retrieval* and *w/o Graph Encoder* variant. This suggests that pruning is crucial, especially in dense graphs, to improve the quality of graph tokens and avoid negative impacts from irrelevant entities.

Importance of Text Attributes. When the text attributes of retrieved subgraphs are excluded, relying solely on the soft token does not enhance the generation process in graph-related tasks. This variant performs worse than the *w/o Retrieval* setup, with its Hit@1 score dropping to 0.4496—a 38.2% decrease. This finding highlights the importance of node and edge textual attributes for effective generation. While soft tokens aggregate these text attributes, incorporating the text attributes remains essential for optimal LLM generation.

6 Conclusion

In this paper, we introduce Graph Retrieval-Augmented Generation (GRAG) to extend Retrieval-Augmented Generation (RAG) to graph-based scenarios. We present a computational framework for GRAG that enhances the generation capabilities of Large Language Models (LLMs) by retrieving query-relevant textual subgraphs. To ensure efficient subgraph retrieval, we propose a divide-and-conquer strategy that leverages K-hop ego-graphs and soft pruning to approximate the optimal textual subgraph. Our approach provides LLMs with two complementary views of a textual graph: *graph view* and *text view*, enabling a comprehensive understanding of the graph context. Empirical results demonstrate that GRAG significantly outperforms LLM baselines and RAG-based LLMs, particularly in scenarios requiring detailed, multi-hop reasoning on textual graphs. Our approach not only addresses the NP-hard challenge of exhaustive subgraph searches but also shows that a frozen LLM enhanced by GRAG can outperform fine-tuned LLMs at a reduced training cost.

References

Jinheon Baek, Alham Aji, and Amir Saffari. 2023. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. In *The 61st Annual Meeting Of The Association For Computational Linguistics*.

Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, et al. 2024. Exploring the potential of large language models (llms) in learning on graphs. *ACM SIGKDD Explorations Newsletter*, 25(2):42–61.

Zhikai Chen, Haitao Mao, Hongzhi Wen, Haoyu Han, Wei Jin, Haiyang Zhang, Hui Liu, and Jiliang Tang. 2023. Label-free node classification on graphs with large language models (llms). In *The Twelfth International Conference on Learning Representations*.

Nurendra Choudhary and Chandan K Reddy. 2023. Complex logical reasoning over knowledge graphs using large language models. *arXiv preprint arXiv:2305.01157*.

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*.

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. 2023. Talk like a graph: Encoding graphs for large language models. In *The Twelfth International Conference on Learning Representations*.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2022. Language-agnostic bert sentence embedding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 878–891.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Chunjiang Ge, Rui Huang, Mixue Xie, Zihang Lai, Shiji Song, Shuang Li, and Gao Huang. 2023. Domain adaptation via prompt learning. *IEEE Transactions on Neural Networks and Learning Systems*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, pages 3929–3938.

Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. 2023. Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning. In *The Twelfth International Conference on Learning Representations*.

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*.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Nan Hu, Yike Wu, Guilin Qi, Dehai Min, Jiaoyan Chen, Jeff Z Pan, and Zafar Ali. 2023a. An empirical study of pre-trained language models in simple knowledge graph question answering. *World Wide Web*, 26(5):2855–2886.

Yuntong Hu, Zheng Zhang, and Liang Zhao. 2023b. Beyond text: A deep dive into large language models' ability on understanding graph data. In *NeurIPS 2023 Workshop: New Frontiers in Graph Learning*.

Jin Huang, Xingjian Zhang, Qiaozhu Mei, and Jiaqi Ma. 2023. Can llms effectively leverage graph structural information: when and why. *arXiv preprint arXiv:2309.16595*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. 2022. Visual prompt tuning. In *European Conference on Computer Vision*, pages 709–727. Springer.

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

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*.

David S Johnson and Michael R Garey. 1979. *Computers and intractability: A guide to the theory of NP-completeness*. WH Freeman.

Minki Kang, Jin Myung Kwak, Jinheon Baek, and Sung Ju Hwang. 2023. Knowledge graph-augmented language models for knowledge-grounded dialogue generation. *arXiv preprint arXiv:2305.18846*.

Muhammad Uzair Khattak, Hanoona Rasheed, Muhammad Maaz, Salman Khan, and Fahad Shahbaz Khan. 2023. Maple: Multi-modal prompt learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19113–19122.

Jiho Kim, Sungjin Park, Yeonsu Kwon, Yohan Jo, James Thorne, and Yoonjae Choi. 2023. Factkg: Fact verification via reasoning on knowledge graphs. In *61st Annual Meeting of the Association for Computational Linguistics, ACL 2023*, pages 16190–16206. Association for Computational Linguistics (ACL).

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

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.

Yuhan Li, Zhixun Li, Peisong Wang, Jia Li, Xiangguo Sun, Hong Cheng, and Jeffrey Xu Yu. 2023. A survey of graph meets large language model: Progress and future directions. *arXiv preprint arXiv:2311.12399*.

Zijian Li, Qingyan Guo, Jiawei Shao, Lei Song, Jiang Bian, Jun Zhang, and Rui Wang. 2024. Graph neural network enhanced retrieval for question answering of llms. *arXiv preprint arXiv:2406.06572*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.

Pei-Chi Lo and Ee-Peng Lim. 2023. Contextual path retrieval: A contextual entity relation embedding-based approach. *ACM Transactions on Information Systems*, 41(1):1–38.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

LinHao Luo, Yuan-Fang Li, Reza Haf, and Shirui Pan. 2023. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations*.

Alex Troy Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *The 61st Annual Meeting Of The Association For Computational Linguistics*.

Jacob Menick, Maja Trebacz, Vladimir Mikulik, John Aslanides, Francis Song, Martin Chadwick, Mia Glaese, Susannah Young, Lucy Campbell-Gillingham, Geoffrey Irving, et al. 2022. Teaching language models to support answers with verified quotes. *arXiv preprint arXiv:2203.11147*.

Erwan Le Merrer and Gilles Trédan. 2024. Llms hallucinate graphs too: a structural perspective. *arXiv preprint arXiv:2409.00159*.

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

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

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331.

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)*, pages 3982–3992.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Jon Saad-Falcon, Joe Barrow, Alexa Siu, Ani Nenkova, Ryan A Rossi, and Franck Dernoncourt. 2023. Pdftriage: Question answering over long, structured documents. *arXiv preprint arXiv:2309.08872*.

Swarnadeep Saha, Prateek Yadav, Lisa Bauer, and Mohit Bansal. 2021. Explagraphs: An explanation graph generation task for structured commonsense reasoning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7716–7740.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*.

Yixuan Tang and Yi Yang. 2024. Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries. *arXiv preprint arXiv:2401.15391*.

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*.

Milena Trajanoska, Riste Stojanov, and Dimitar Trajanov. 2023. Enhancing knowledge graph construction using large language models. *arXiv preprint arXiv:2305.04676*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2024. Can language models solve graph problems in natural language? *Advances in Neural Information Processing Systems*, 36.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Hao Yang, Junyang Lin, An Yang, Peng Wang, Chang Zhou, and Hongxia Yang. 2022. Prompt tuning for generative multimodal pretrained models. *arXiv preprint arXiv:2208.02532*.

Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. 2024. Do large language models latently perform multi-hop reasoning? *arXiv preprint arXiv:2402.16837*.

Liang Yao, Jiazhen Peng, Chengsheng Mao, and Yuan Luo. 2023. Exploring large language models for knowledge graph completion. *arXiv preprint arXiv:2308.13916*.

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)*.

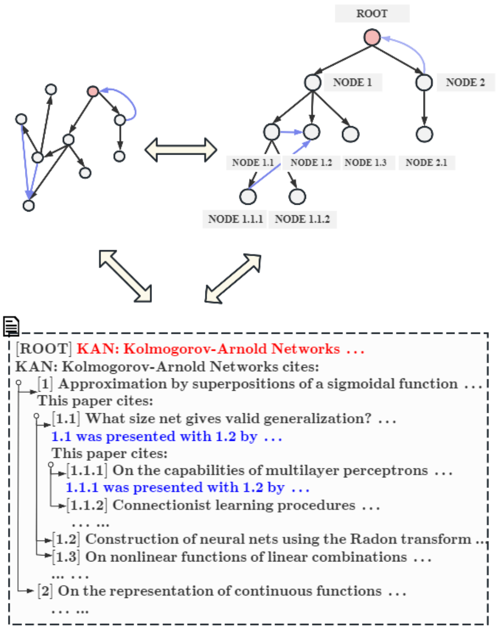
Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. 2022. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348.

A Appendix

A.1 Hierarchical Description

As shown in Figure 4, a 2-gop ego-graph is transformed into a nested, indented list, which mirrors the graph's structure. Each level in the hierarchy corresponds to a level in the graph, representing connections between nodes. For example, "NODE 1" contains sub-nodes (NODE 1.1, NODE 1.2, etc.), which are further divided into lower levels, reflecting the original graph's branching structure.



```
[ROOT] KAN: Kolmogorov-Arnold Networks ...
KAN: Kolmogorov-Arnold Networks cites:
  [1] Approximation by superpositions of a sigmoidal function ...
  This paper cites:
    [1.1] What size net gives valid generalization? ...
    1.1 was presented with 1.2 by ...
    This paper cites:
      [1.1.1] On the capabilities of multilayer perceptrons ...
      1.1.1 was presented with 1.2 by ...
      [1.1.2] Connectionist learning procedures ...
       ... ...
    [1.2] Construction of neural nets using the Radon transform ...
    [1.3] On nonlinear functions of linear combinations ...
    ... ...
  [2] On the representation of continuous functions ...
    ... ...
```

Figure 4: An Example hierarchical description for a 2-hop ego-graph from a citation network.

Each node in the hierarchy is accompanied by descriptive text or titles, conveying the content or subject matter associated with that node. This setup preserves the textual information within the graph, such as titles of cited papers or key phrases. The hierarchical formatting maintains the topological relationships between nodes, with indentation and nested levels reflecting their connections. This structure mirrors the links between the root node and its sub-nodes, capturing the connectivity of the original graph. By presenting the graph as a sequential, hierarchical text format, the order and relationships among nodes become clear, as each node's connection to its parent and child nodes is preserved through indentation and citation references.

A.2 Comparison Retrievers

BM25 (Robertson et al., 2009), which is a statistical model, scores documents based on term frequency, inverse document frequency, and document length, using probabilistic principles to estimate the relevance of documents to a query; MiniLM-L12-v2, which is a SentenceTransformer model (Reimers and Gurevych, 2019) widely used in clustering and semantic search; LaBSE (Feng et al., 2022), a BERT-based model that performs retrieval

Table 5: Performance of RAG-based retrievers: Hit@1 on WebQSP and Acc on ExplaGraphs.

Model	WebQSP					ExplaGraphs		
	top-3	top-5	top-10	top-15	top-20	top-3	top-5	top-10
BM25	0.3722	0.3821	0.4109	0.4165	0.4287	0.5704	0.5921	0.6011
MiniLM-L12-v2	0.4251	0.4251	0.4539	0.4625	0.4730	0.5848	0.5939	0.6011
LaBSE	0.4091	0.4171	0.4294	0.4527	0.4496	0.6011	0.6011	0.6011
mContriever-Base	0.4183	0.4158	0.4349	0.4459	0.4453	0.5866	0.5866	0.5866
E5-Base	0.4404	0.4558	0.4662	0.4650	0.4705	0.5921	0.5939	0.6011

by using a dual-encoder framework to learn cross-lingual sentence embeddings; mContriever (Izacard et al., 2021), which utilizes a contrastive learning approach with a bi-encoder architecture to independently encode documents and queries; E5 (Wang et al., 2022), that employs a contrastive pre-training strategy using a bi-encoder architecture, optimizing similarity between relevant pairs while distinguishing from irrelevant ones using in-batch negatives; G-Retriever (He et al., 2024), which retrieves relevant nodes and edges, and then constructs a relevant subgraph using a Prize-Collecting Steiner Tree method.

A.3 Implementation

The data splits for training, validation, and test sets are 60%/20%/20% for ExplaGraphs and 60%/5%/35% for WebQSP. All experiments are performed on a Linux-based server with 4 NVIDIA A10G GPUs. We use SentenceBert (Reimers and Gurevych, 2019) to encode the question and text attributes to obtain vectors for the retrieval process. The graph encoder, i.e. GAT (Veličković et al., 2018), has 4 layers with 4 heads per layer and a hidden dimension size of 1024.

The LLM backbone is Llama-2-7b-hf, while the model used is in the setting of LLM only is Llama-2-7b-chat-hf. We employ Low-rank Adaptation (LoRA) (Hu et al., 2021) for fine-tuning, configuring the LoRA parameters as follows: the dimension of the low-rank matrices is set to 8; the scaling factor is 16; and the dropout rate is 0.05. For the optimization, AdamW optimizer (Loshchilov and Hutter, 2018) is used. The initial learning rate is set to 1e-5 and the weight decay is 0.05. Each experiment runs for up to 10 epochs, and the batch size is 2. For compared retrievers, each experiment on ExplaGraphs is replicated three times, utilizing different retrieval settings for each run, i.e., top-3, top-5 and top-10; Each experiment on WebQSP is replicated five times, utilizing different retrieval settings for each run, i.e., top-3, top-5, top-10, top-15 and top-20, where top-k denotes that the k most relevant nodes and k edges are retrieved and used for generation. In our GRAG approach, since the graphs in ExplaGraphs are constructed from several triples, each graph is actually a chain consisting of only a few nodes. Therefore, we feed the entire graph to the LLM.

A.4 Experiment

Evaluation Metrics. Hit@1 assesses whether the top retrieved result is correct. It is particularly useful for understanding the accuracy of the first retrieval hit in graph-based question answering tasks. F_1 **Score** is the harmonic mean of precision and recall, providing a single metric that balances both false positives and false negatives. **Recall** measures the proportion of relevant entities that are successfully retrieved. High recall indicates that the retrieval system captures most of the relevant information. **Accuracy (Acc)** measures the proportion of correctly answered questions. It is particularly useful for tasks like ExplaGraphs, where the focus is on commonsense reasoning.

Effects of the Number of Retrieved Entities. Top-k indicates k nodes and k edges are retrieved. The performance of various RAG retrievers on the WebQSP and ExplaGraphs datasets, with different numbers of retrieved entities, is summarized in Table 5. As shown in Figure 5, GRAG replaces hard prompts with texts of retrieved entities, while the soft prompt is represented by tokens generated from the retrieved K-hop ego-graphs. Increasing the number of retrieved entities generally improves performance up to a certain point. For example, BM25's Hit@1 score on WebQSP rises from 0.3722 with top-3 retrievals to 0.4287 with top-20 retrievals, and MiniLM-L12-v2 shows improvement from 0.4251 to 0.4730 over the same range. However, this trend does not continue indefinitely; for some models, performance plateaus or even slightly decreases beyond a certain number of entities. For instance, LaBSE's performance peaks at top-15 and then slightly declines at top-20 on WebQSP. This suggests that retrieving too many entities can introduce irrelevant information, potentially impairing final generation quality. On the ExplaGraphs dataset, the trend is less pronounced due to smaller graph sizes, with most models showing minimal performance changes beyond top-5 retrievals. When the graph size is small, indicating limited information, all RAG-based retrievers encounter a performance bottleneck. In contrast, our GRAG approach leverages topological information effectively, enabling it to overcome this limitation.

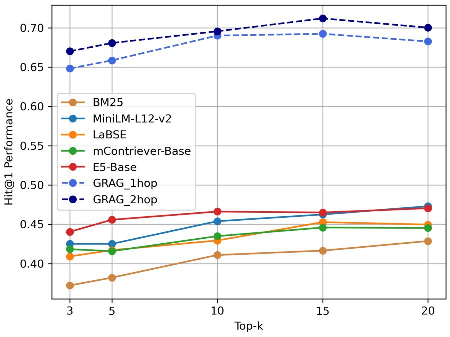


Figure 5: Effects of the number of retrieved entities on the WebQSP dataset.