

MixRAG : Mixture-of-Experts Retrieval-Augmented Generation for Textual Graph Understanding and Question Answering

Lihui Liu *
hw6926@wayne.edu
Wayne State University
Detroit, Michigan, USA

Subhabrata Mukherjee
subho@hippocraticai.com
Hippocratic AI
Palo Alto, California, USA

Jiayuan Ding
jiayuan@hippocraticai.com
Hippocratic AI
Palo Alto, California, USA

Carl Yang
j.carlyang@emory.edu
Emory University
Atlanta, Georgia, USA

Abstract

Large Language Models have achieved impressive performance across a wide range of applications. However, they often suffer from hallucinations in knowledge-intensive domains due to their reliance on static pretraining corpora. To address this limitation, Retrieval-Augmented Generation (RAG) enhances LLMs by incorporating external knowledge sources during inference. Among these sources, Textual Graphs offer structured and semantically rich information that supports more precise and interpretable reasoning. This has led to growing interest in Graph-based RAG systems. Despite their potential, most existing approaches rely on a single retriever to identify relevant subgraphs, which limits their ability to capture the diverse aspects of complex queries. Moreover, these systems often struggle to accurately judge the relevance of retrieved content, making them prone to distraction by irrelevant noise. To address these challenges, in this paper, we propose MixRAG, a Mixture-of-Experts Graph-RAG framework that introduces multiple specialized graph retrievers and a dynamic routing controller to better handle diverse query intents. Each retriever is trained to focus on a specific aspect of graph semantics, such as entities, relations, or subgraph topology. A Mixture-of-Experts module adaptively selects and fuses relevant retrievers based on the input query. To reduce noise in the retrieved information, we introduce a query-aware GraphEncoder that carefully analyzes relationships within the retrieved subgraphs, helping to highlight the most relevant parts while down-weighting unnecessary noise. Empirical results show that our method achieves state-of-the-art performance and consistently outperforms various baselines. The code can be found from <https://github.com/lihuiullh/MixRAG>.

CCS Concepts

• Information systems → Language models.

*Lihui Liu is the first author and the corresponding author. Jiayuan Ding, Subhabrata Mukherjee and Carl Yang are collaborators.



This work is licensed under a Creative Commons Attribution 4.0 International License. *WWW '26, Dubai, United Arab Emirates*
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2307-0/2026/04
<https://doi.org/10.1145/3774904.3792680>

Keywords

Textural graph question answering

ACM Reference Format:

Lihui Liu, Jiayuan Ding, Subhabrata Mukherjee, and Carl Yang. 2026. MixRAG : Mixture-of-Experts Retrieval-Augmented Generation for Textual Graph Understanding and Question Answering. In *Proceedings of the ACM Web Conference 2026 (WWW '26)*, April 13–17, 2026, Dubai, United Arab Emirates. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3774904.3792680>

1 Introduction

Large Language Models have greatly advanced natural language processing, showing strong performance on many reasoning and text generation tasks [42, 47]. However, they often struggle in knowledge-heavy areas, where accurate answers depend on up-to-date and detailed information. This is mainly because LLMs are trained on fixed datasets, which may be outdated, incomplete, or lack the specific knowledge needed for complex reasoning. To address this shortcoming, Retrieval-Augmented Generation (RAG) [4, 19, 45] has emerged as a prominent paradigm. By retrieving relevant context from external sources and conditioning LLM responses on the retrieved content, RAG enhances factuality, reduces hallucination, and supports knowledge-intensive tasks. While early RAG methods [19, 45] focused on retrieving unstructured documents (e.g., Wikipedia passages), recent efforts have explored retrieval over structured sources like textual graphs [4, 9]. These structured RAG systems aim to leverage the explicit semantics and relational structure of textual graphs to enable more grounded reasoning.

Despite promising progress, existing graph-based RAG methods face two key challenges that limit their effectiveness in real-world applications. First, most systems rely on a single retriever model to handle diverse query intents. However, textual graphs inherently encode multi-aspect information. For example, nodes often represent entities with domain-specific semantics (e.g., “Vitamin D,” “Mushroom”), while edges express relational dependencies (e.g., “part_of,” “requires”). Using a single retriever trained on shallow lexical matching or embedding similarity often fails to capture this heterogeneity. Queries seeking complex relational inference may require different retrieval patterns than those involving single hop factual recall of entities. A one-size-fits-all retriever lacks the specialization to disentangle these nuances.

Second, retrieved subgraphs frequently contain irrelevant or noisy information, which can mislead the LLM during generation.

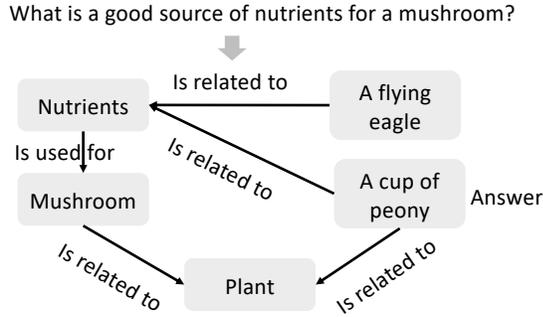


Figure 1: An example of a retrieved subgraph for question answering. The subgraph contains both correct answer and noise data.

Figure 3 shows a motivating case: when asked “What is a good source of nutrients for a mushroom?”, a standard retriever surfaces a subgraph containing entities such as “a flying eagle” and “a cut peony,” which are semantically disconnected from the query. Injecting such noisy knowledge into the LLM can degrade factual precision. This highlights a fundamental challenge: *not all retrieved knowledge is equally useful*. Therefore, it raises an important question: *How can we identify which retrieved knowledge is valuable and which is not?* This question becomes especially crucial when a significant portion of the retrieved subgraph lacks direct relevance or even introduces contradictions.

To address these limitations, we propose MixRAG, a Mixture-of-Experts [12] Graph-RAG framework. Instead of relying on a single retriever, our method introduces multiple expert retrievers, each trained to capture a specific aspect of graph semantics (e.g., entity names, relational paths, and local graph neighborhoods). A Mixture-of-Experts (MoE) controller is then used to dynamically combine suitable expert(s), based on query intent and expert specialization. This design enables flexible and query-adaptive retrieval that better aligns with the semantic demands of different question types. Furthermore, to mitigate the noise in the retrieved subgraph, we design a query-aware *GraphEncoder* module. This encoder performs fine-grained relational modeling over the retrieved graph fragments, learning to amplify useful signals and suppress distracting ones. The resulting graph-aware embeddings are used to augment the prompt space of the LLM, enabling more grounded and context-sensitive reasoning. We conduct extensive experiments on the GraphQA benchmark and show that our method significantly outperforms strong baselines and achieves the state-of-the-art performance. Ablation studies further validate the contribution of expert-based retrieval and fine-grained subgraph embedding to downstream performance. In summary, we make the following contributions.

- We introduce **Mixture-of-Experts Graph-RAG**, a novel RAG architecture that uses multiple specialized graph retrievers and a learned routing controller to merge relevant experts dynamically.

Table 1: Main notations used in this paper.

Symbol	Definition
$\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{T})$	A textual graph
v_i	The i^{th} text node (entity, sentence, or paragraph) in \mathcal{G}
r_i	The i^{th} relation/edge in \mathcal{G}
q	The natural language query
s_i	The i -th subgraph retrieved
A_q	Answer set of q
a_q	A candidate answer of q
h_i	Dense embedding of node v_i or relation r_i in \mathbb{R}^d
h_q	Dense embedding of query q in \mathbb{R}^d

- We propose a query-specific *GraphEncoder* that encodes retrieved subgraphs based on structural and relational relevance, producing more informative embeddings for generation.
- We empirically demonstrate state-of-the-art performance on the GraphQA benchmark, outperforming existing approaches. These results underscore the effectiveness of our proposed MOE-GraphRAG framework in capturing and leveraging multi-aspect knowledge for graph-based question answering.

The remainder of this paper is structured as follows. Section 2 defines the problem and outlines the notations adopted throughout the paper. Section 3 presents the proposed framework along with its key technical components. Experimental results are discussed in Section 4, while Section 5 reviews relevant literature. Finally, Section 6 summarizes the main findings and concludes the paper.

2 Problem Definition

Table 1 gives the main notations used throughout this paper. A textual graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{T})$, where \mathcal{V} is the set of text nodes, \mathcal{R} the set of relations, and \mathcal{T} the set of factual triples. Unlike conventional knowledge graphs, nodes in textual graphs are free-form text units such as entities, sentences, or short paragraphs. Each triple is represented as (h, r, t) , where $h \in \mathcal{V}$ is the head node, $t \in \mathcal{V}$ the tail node, and $r \in \mathcal{R}$ the relation linking them. Each node $v_i \in \mathcal{V}$ and relation $r_i \in \mathcal{R}$ can be encoded into a dense vector $h_i \in \mathbb{R}^d$. Similarly, a query q is encoded into an embedding $h_q \in \mathbb{R}^d$.

Textual Graph Question Answering. Textual Graph Question Answering (TGQA) aims to answer natural language questions by using information stored in a textual graph. The main challenge is to map an input question into a reasoning process over the graph to identify the correct answer node(s). Various approaches have been proposed, including few-shot prompting [53] and instruction tuning [42]. Recent work leverages Retrieval-Augmented Generation (RAG) to improve TGQA. Instead of directly predicting the answer, RAG first retrieves a relevant subgraph from the textual graph and then uses a large language model (LLM) to generate the answer based on this subgraph. Specifically, given a question q , the retriever selects a subgraph $S \subseteq \mathcal{G}$ that contains relevant nodes and edges. The retrieved subgraph is then converted into a textual format—such as a sequence of triples or natural language descriptions—which is combined with the original question to form

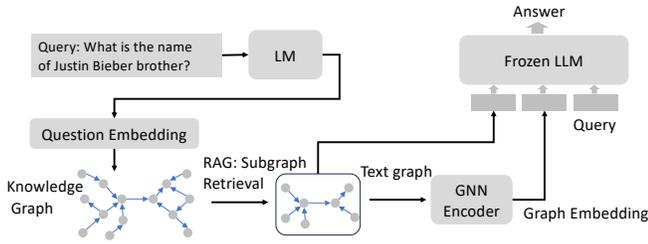


Figure 2: Traditional retrieval-augmented generation (RAG) pipeline over textual graphs.

the input to the LLM. The LLM uses this augmented input to generate the answer a . Alternatively, the retrieved subgraph can be encoded with a graph neural network (GNN) [8, 15] to produce embeddings that capture both semantic and structural information. The LLM can attend to both the textualized subgraph and the GNN embeddings, resulting in more accurate and grounded reasoning (Figure 2).

Mixture-of-Experts (MoE) was originally proposed in [44] as a strategy to improve model expressiveness by combining multiple specialized “expert” networks, with a gating mechanism dynamically selecting which experts to use for each input. This approach is more powerful than a single monolithic model, as it allows the system to adaptively leverage the most relevant expertise for each instance, enabling better representation, higher capacity, and improved generalization across diverse inputs.

Limitations and Motivation for Mixture-of-Experts. Standard RAG-based methods typically rely on a single retrieval perspective (e.g., entity-level or relation-level), which may overlook complementary evidence. To address this, we propose a Mixture-of-Experts (MoE) retrieval framework. Each expert focuses on a distinct retrieval signal—such as entity-centric, relation-centric, or subgraph-based retrieval—and the system dynamically selects among experts. This enables the model to gather multi-perspective evidence from the textual graph, which is then used to guide the LLM in generating faithful and accurate answers.

Formally, the problem this paper studies is defined as follows.

PROBLEM DEFINITION. *Answering Query TGQA:*

Given: (1) A textual graph \mathcal{G} , (2) an natural language question;

Output: The answer of the question.

3 Proposed Method

In this section, we introduce the details of the proposal method, we begin with multiaspect knowledge retrieval. We then delve into two specific modules: semantic reasoning and subgraph reasoning, providing detailed explanations of their functionalities, and finally, we introduce how to fuse them together.

The key idea behind our model is that real-world questions vary in complexity. Some questions are simple and can be easily answered, in which case a naive entity retriever is sufficient to find the correct answer. Using a more complex subgraph retriever for such questions may introduce unnecessary noise and even degrade performance. On the other hand, more complex questions require multi-step reasoning, where a sophisticated retriever—such as a

subgraph-based approach—can provide richer context and significantly improve the model’s reasoning capabilities. By adopting different retrievers for different types of questions and then combining their outputs, the system can adapt to the complexity of each question and achieve better overall performance.

3.1 Retrieval of Multi-Aspect Knowledge

Previous TGQA methods typically focus on retrieving either entities and relations [7] or multiple triples to construct subgraphs [9]. In contrast, we propose to simultaneously leverage three complementary types of knowledge: entities, relations, and subgraphs. Each captures a different retrieval mechanism that is suitable for different types of queries. By jointly utilizing different knowledge types, we enable more robust and accurate alignment between questions and the corresponding knowledge components. Here, we describe each retrieval in detail.

Entity Retrieval. Entity retrieval selects a small set of candidate entities from the graph that are most relevant to the input question. It directly predicts which entities are likely to be the answer or closely related to the question based on their similarity in the embedding space. The question is encoded into a dense vector using a language model and compared against entity embeddings. The top- k most similar entities are selected during the reasoning process. The motivation behind this retriever is that some questions can be answered with minimal reasoning; in such cases, a simple entity retriever may outperform more complex subgraph-based retrieval methods, which can introduce unnecessary noise.

Relation Retrieval. Complementing entity retrieval, relation retrieval identifies the most relevant triples that reflect the semantic intent of the question. It predicts which triples are likely to be the answer or closely related to the question. Like entities, relations are encoded as embeddings using their names and descriptions. The top- k matching triples are then selected based on their similarity to the question embedding. The idea behind a relation retriever is similar to that of an entity retriever, but it leverages richer semantic information. Instead of focusing solely on individual entities, it also considers the relations between pairs of entities, capturing more contextual information and providing a more comprehensive understanding compared with a simple entity retriever.

Subgraph Retrieval. Subgraph retrieval aims to extract a compact yet comprehensive subgraph that is most relevant to the question. It is designed to capture richer information needed for answering complex queries. It serves two main goals: filtering out irrelevant information that could distract the language model from the essential context, and maintaining a manageable graph size that can be effectively serialized into text for LLM processing. The subgraph is typically constructed by expanding from the retrieved seed entities along top-ranked relations, within a limited number of hops. This process ensures that the resulting subgraph preserves both the structural and semantic connections most relevant to the query. Unlike entity and relation retrievers, a subgraph retriever extracts richer information from the underlying data graph, enabling support for more complex reasoning tasks.

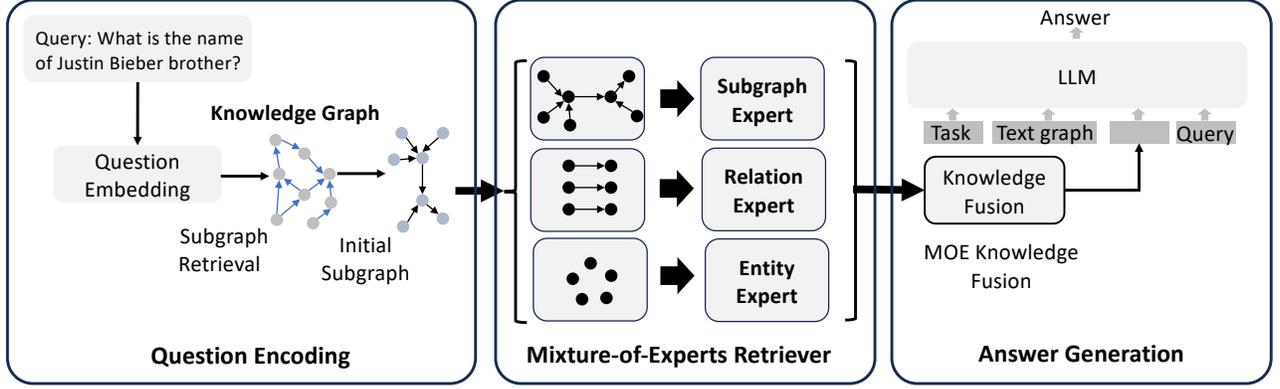


Figure 3: The framework of MixRAG.

3.2 Semantic Reasoning with Entity and Relation Retrieval

In the previous subsection, we briefly introduced the three retrievers and their respective roles. Here, we focus on the details of Entity Retrieval and Relation Retrieval, which together form the basis of Semantic Reasoning. Semantic reasoning aims to identify the components of a textual graph that are most semantically aligned with the input query. Within this framework, entity and relation retrieval act as complementary processes: entity retrieval predicts the most likely answer entities, while relation retrieval identifies the most relevant triples.

Formally, we represent the initial subgraph retrieved by subgraph retriever as $G = (E, R, T)$, where E denotes the set of entities, R the set of relations, and $T \subseteq E \times R \times E$ the set of textualized triples. Each entity $e_i \in E$ is encoded into a dense vector $h_i \in \mathbb{R}^d$. For each triple $(h, r, t) \in T$, we construct an initial representation by concatenating the head, relation, and tail embeddings into a vector $[h_h; h_r; h_t] \in \mathbb{R}^{3d}$, which is subsequently projected through a learned transformation $W_t \in \mathbb{R}^{d \times 3d}$. This ensures that both entity-level and triple-level representations reside in a unified semantic space. Given a natural-language query q , its embedding $h_q \in \mathbb{R}^d$ is obtained using an instruction-tuned language model. To measure the relevance of each graph element to the query, we concatenate its representation with the query embedding, $z_i = [h_i \| h_q]$, and compute a relevance logit using a learnable scoring function $f(\cdot)$, $\phi_i = f(z_i)$. This parameterized compatibility function enables the model to learn query-specific semantic matching over entities or triples.

To perform differentiable selection over graph elements, we adopt the Gumbel-Softmax continuous relaxation. Using the reparameterization trick, we perturb the logits with Gumbel noise and compute

$$p_i = \text{softmax}\left(\frac{\phi_i + g_i}{\tau}\right), \quad g_i \sim \text{Gumbel}(0, 1),$$

where τ is the temperature controlling the sharpness of the distribution and is gradually annealed during training. The resulting distribution $p \in \mathbb{R}^{|E|}$ (for entities) or $p \in \mathbb{R}^{|T|}$ (for triples) represents attention weights over the textual graph, highlighting the most relevant components with respect to the query.

Finally, we compute a query-specific graph representation by first selecting the top nodes or edges using the differentiable top- k sampler, and then encoding the resulting masked subgraph with a GNN. Using these GNN-refined embeddings, we can optionally pool over the selected nodes to obtain a graph-level representation

$$\tilde{h}_G = \text{POOL}(\{h_i \mid m_i = 1\}),$$

which yields a compact, semantically aligned summary of the most relevant entities or triples with respect to the input query. This query-specific representation is subsequently used by downstream reasoning modules to generate contextually grounded answers.

3.3 Subgraph Retriever and Reasoning

While entity and relation retrieval focus on pinpointing specific knowledge components, subgraph retrieval aims to extract a structured and compact set of interconnected facts that collectively support more complex reasoning. In this paper, we follow the idea of GRetriever framework [9] to identify a relevant subgraph conditioned on the input query q . Let $G = (V, E)$ denote the textual graph, where both nodes and edges are associated with textual descriptions. We encode the query and the graph components using a pretrained language model, Sentence-BERT [43].

$$z_q = \text{LM}(q), \quad z_{v_i} = \text{LM}(v_i), \quad z_{e_{i,j}} = \text{LM}(e_{i,j}) \quad (1)$$

We then compute cosine similarities between the query embedding z_q and all node and edge embeddings, and retrieve the top- k nodes V_k and edges E_k with the highest similarity scores. To ensure the subgraph is connected and informative, we apply the Prize-Collecting Steiner Tree (PCST) algorithm [9]. Each retrieved node or edge is assigned a prize based on its similarity rank. The final subgraph $S \subseteq G$ is selected to maximize the total prize while minimizing edge costs:

$$S = \underset{S \subseteq G}{\text{argmax}} \left(\sum_{v_i \in V_k} \text{prize}(v_i) + \sum_{e_{i,j} \in E_k} \text{prize}(e_{i,j}) - c \cdot |E_S| \right)$$

This subgraph serves as both a textual context for the LLM and an input to graph encoders for downstream reasoning.

After retrieving the subgraph, the next step is to perform reasoning over it. Prior methods such as G-Retriever employ GCNs [16] or GATs [48] to encode subgraphs. However, these architectures

often suffer from the over-smoothing problem [3], where node embeddings become indistinguishable after several layers of message passing. This issue is particularly problematic in our setting, where the retrieved subgraphs may include a mix of relevant and irrelevant (noisy) nodes. Therefore, it is crucial to generate *query-aware* representations that emphasize the most relevant nodes and edges with respect to the input query. For example, consider a query involving a cut peony. The encoder should highlight the node corresponding to a cut peony while downweighting unrelated nodes such as a flying eagle, even if they are structurally nearby in the graph. To achieve this, we design a query-conditioned GNN in which both message passing and node interactions are dynamically modulated by the input query q .

Concretely, we redefine the attention weights over edges using a query-aware mechanism. At each GNN layer l , the attention weight $\zeta_{e_{i,j}}^{(l)}$ for edge $e_{i,j}$ is computed as:

$$\alpha_{v_i}^{(l)} = \text{LINEAR} \left(\text{CONCAT}(z_{v_i}^{(l)}, q) \right) \quad (2)$$

$$\beta_{v_j}^{(l)} = \text{LINEAR} \left(\text{CONCAT}(z_{v_j}^{(l)}, q) \right) \quad (3)$$

$$\gamma_{e_{i,j}} = \text{LINEAR} \left(\text{CONCAT}(z_{e_{i,j}}, q) \right) \quad (4)$$

$$\zeta_{e_{i,j}}^{(l)} = \tanh \left(\alpha_{v_i}^{(l)} + \gamma_{e_{i,j}} - \beta_{v_j}^{(l)} \right) \quad (5)$$

Here, $\alpha_{v_i}^{(l)}$, $\beta_{v_j}^{(l)}$, and $\gamma_{e_{i,j}}$ are intermediate representations that encode query-conditioned information for the source node, target node, and edge respectively. The resulting attention weight $\zeta_{e_{i,j}}^{(l)}$ modulates how much influence the message passed along edge $e_{i,j}$ should have.

The message from node v_i to node v_j at layer l is generated as:

$$\text{msg}_{e_{i,j}}^{(l)} = \text{LINEAR} \left(\text{CONCAT}(z_{v_i}^{(l)}, z_{v_j}^{(l)}, z_{e_{i,j}}, q) \right) \quad (6)$$

Each node then updates its representation using the attention-weighted aggregation of incoming messages:

$$z_{v_j}^{(l+1)} = \frac{1}{d_{v_j}} \sum_{v_i \in \mathcal{N}(v_j)} \zeta_{e_{i,j}}^{(l)} \cdot \text{msg}_{e_{i,j}}^{(l)} \quad (7)$$

Unlike standard GCNs that apply fixed, query-independent filters, our query-conditioned GNN dynamically adapts both message generation and propagation to the specific information needs encoded in the query q . This enables the model to attend to semantically aligned regions of the subgraph while filtering out unrelated content. After L layers of message passing, we obtain the final node representations $z_{v_j}^{(L)}$ for each node v_j in the subgraph S .

3.4 Mixture of Experts Gating

In the previous subsection, we introduced various types of information retrievers. Here, we describe how to effectively fuse the retrieved signals. To accommodate the diverse reasoning requirements in textual graphs, we adopt a Mixture-of-Experts (MoE) framework that dynamically integrates knowledge from multiple specialized, query-aware reasoning modules: the entity (node) expert f_e , the relation (edge) expert f_r , and the subgraph expert f_s . This design allows the model to adaptively leverage the most relevant processing pathway for each node in the graph.

Let $o_v^{(i)} \in \mathbb{R}^d$ denote the node-level output produced by expert $f_i \in \{f_e, f_r, f_s\}$ for node v . Instead of computing a single global gating score, we use a node-wise, graph-aware gating network. For each node, we construct a gate input by combining its original node feature with features aggregated from its local graph context with gnn. This input is fed into a small MLP to produce per-expert logits,

$$\phi_v = \text{MLP}(\text{node feature} + \text{learned contextual features}) \in \mathbb{R}^3.$$

These logits are converted into a node-wise probability distribution over experts using a softmax function:

$$\alpha_{v,i} = \frac{\exp(\phi_{v,i})}{\sum_{j \in \{e,r,s\}} \exp(\phi_{v,j})}, \quad i \in \{e, r, s\},$$

so that $\sum_i \alpha_{v,i} = 1$ for each node v .

The final fused representation at node v is the convex combination of expert outputs weighted by the node-wise gate,

$$\tilde{\mathbf{h}}_v = \sum_{i \in \{e,r,s\}} \alpha_{v,i} o_v^{(i)}.$$

When a graph-level soft prompt is required, the node-wise fused embeddings are pooled over the selected nodes to form a compact graph representation:

$$\mathbf{p}_{\text{soft}} = \text{MeanPool}(\{\tilde{\mathbf{h}}_v\}),$$

which serves as a query-conditioned graph prompt for downstream modules.

3.5 Answer Generation

After obtaining the fused information from multiple retrievers, we prepare the input for response generation using a large language model (LLM). The goal is to generate an answer that is grounded in both the user’s query q and the retrieved supporting evidence. To incorporate task-specific control signals, we prepend a trainable soft prompt p_{soft} to the input. This soft prompt is produced by a Mixture-of-Experts (MoE) controller that dynamically selects and combines expert embeddings based on the current input context as described in the last subsection. It serves as a continuous prefix that guides the LLM toward producing task-aware and knowledge-grounded responses. The retrieved subgraph G_{sub} is verbalized into natural language using predefined templates that describe entities and relations in sentence form.

The complete input to the LLM consists of four parts: the soft prompt p_{soft} , the tokenized task instruction p_{task} , the textualized subgraph p_{text} , and the user query q .

$$a_{\text{gen}} = \text{LLM}([p_{\text{task}}; p_{\text{soft}}; p_{\text{text}}; q]), \quad (8)$$

Here, a_{gen} is the final response generated by the LLM. By integrating multi-aspect information, the model is guided to produce coherent, grounded, and contextually relevant responses.

3.6 Theoretical Analysis

In this section, we provide a theoretical justification that our proposed framework subsumes existing graph-based retrieval-augmented generation (RAG) methods as special cases. The central idea is that our model integrates three complementary retrievers—entity-, relation-, and subgraph-based—within a Mixture-of-Experts (MoE) architecture, whereas most prior work relies on a single retriever, typically subgraph-based.

Let $\mathcal{F}_{\text{ours}}$ denote the hypothesis class induced by our framework. Given a query q , each expert $f_i \in \{f_e, f_r, f_s\}$ produces node-level, query-conditioned representations over the graph. A node-wise MoE controller computes routing weights $\alpha_{v,i}(q)$ for each node v and expert i , satisfying

$$\sum_{i \in \{e,r,s\}} \alpha_{v,i}(q) = 1, \quad \alpha_{v,i}(q) \geq 0.$$

Here, the gating weights are produced by a learnable controller that depends on the query and aggregated graph context.

Now consider an existing graph RAG method with hypothesis class $\mathcal{F}_{\text{base}}$ that employs only a single retriever, such as a subgraph retriever f_s . Such a method can be realized as a special case of our framework by choosing the node-wise routing weights as

$$\alpha_{v,s}(q) = 1, \quad \alpha_{v,e}(q) = \alpha_{v,r}(q) = 0, \quad \forall v \in \mathcal{V}.$$

Under this setting, the node-level fusion reduces to

$$\tilde{\mathbf{h}}_v(q) = f_s(q)_v,$$

which exactly matches the graph representation used in prior subgraph-based RAG approaches [9]. Analogously, enforcing $\alpha_{v,e}(q) = 1$ or $\alpha_{v,r}(q) = 1$ for all nodes recovers pure entity-based or relation-based retrieval methods, respectively. Thus, existing methods correspond to specific points in the space of routing weights defined by our MoE controller.

Implication. This analysis establishes that our framework is strictly more general than existing graph RAG approaches. Since our MoE controller can always collapse to a single expert, we can guarantee that our method performs at least as well as existing baselines in the worst case. More importantly, by dynamically combining multiple retrieval perspectives, our framework has the potential to improve performance on complex queries where complementary evidence from entities, relations, and subgraphs is required.

4 Experiments

In this section, we present the experimental results of MixRAG, demonstrating its effectiveness and flexibility in graph-based reasoning tasks. Our experiments aim to validate the benefits of combining multiple retrievers in a Mixture-of-Experts framework, including entity, relation, and subgraph retrievers, and to analyze how each component contributes to overall performance across different datasets and query types. We also examine hyperparameters such as the number of GraphEncoder layers and visualize how the model distributes attention among experts to adapt to varying reasoning requirements.

4.1 Experimental Setup

Datasets. We evaluate on the GraphQA benchmark [9], which includes three datasets: ExplaGraphs, SceneGraphs, and WebQSP. These datasets span a diverse range of reasoning requirements over textual graphs. ExplaGraphs is a dataset for generative commonsense reasoning, SceneGraphs targets spatial reasoning over visual scene graphs, and WebQSP involves complex natural language questions over Freebase-derived subgraphs. The statistics for each dataset are summarized in Table 2.

Metrics. Following GraphQA [9], we use accuracy (ACC) for both ExplaGraphs and SceneGraphs, as they are treated as single-answer

classification tasks. For WebQSP, which often has multiple valid answers per query, we report Hit@1, where a prediction is considered correct if it matches any ground truth answer. This allows for flexible evaluation under one-to-many supervision.

Baselines. We compare our method against two main categories of baselines. (1) Inference-only methods: these include Zero-shot prompting, Zero-CoT [17], CoT-BAG [49], and KAPING [1], which do not incorporate graph-structured knowledge or prompt adaptation. (2) Prompt-tuning methods: these include Prompt Tuning, GraphToken [40], and the state-of-the-art G-Retriever [9], which leverage graph-derived prompts or neural retrievers to inject external knowledge into LLMs. For MixRAG, we set k as 20.

4.2 Effectiveness of MixRAG

The results are summarized in Table 4, which compares MixRAG against all baseline methods. We report performance under varying model settings: inference-only, frozen LLM with prompt tuning, and tuned LLM. As we can see, among inference-only baselines, KAPING achieves the strongest results on datasets ExplaGraphs and WebQSP, outperforming Zero-CoT and CoT-BAG. While CoT-BAG achieves the best performance on SceneGraphs. However, once prompt tuning is enabled over a frozen LLM, performance improves substantially. For example, G-Retriever shows significant gains across all datasets, especially on SceneGraphs (0.8131) and WebQSP (70.49), outperforming GraphToken and basic prompt tuning by large margins. This highlights the advantage of retrieval-aware soft prompting in handling textual-graph reasoning tasks.

We see further performance improvements when allowing light-weight finetuning of the LLM. Using LoRA-based tuning [10] leads to better results than keeping the LLM frozen. For example, it achieves 66.03 on WebQSP, compared to 57.05 from GraphToken. Interestingly, G-Retriever combined with LoRA consistently performs better than using LoRA alone, suggesting that structured retrieval and efficient tuning work well together. Finally, our method, MixRAG with LoRA, achieves new state-of-the-art results on all three datasets: 0.8863 on ExplaGraphs, 0.8712 on SceneGraphs, and 75.31 on WebQSP. These results show that combining multi-aspect graph retrieval with a mixture-of-experts design helps the model generalize well across different reasoning tasks.

4.3 Ablation Study

Retrieval Fusion: We begin by evaluating the performance of different retrieval strategies. Specifically, we compare the effectiveness of using the entity retriever, the relation retriever, and the subgraph retriever. The results are summarized in Table 4. As shown, using only the subgraph retriever leads to better performance than using either the entity or relation retriever alone, highlighting the importance of structural context. While combining entity and relation retrieval brings some improvement, combinations that include subgraph retrieval consistently perform better. Notably, our full model (MixRAG) achieves the best results on both datasets, showing that integrating entity, relation, and subgraph information enables more complete and semantically grounded reasoning for textual graph question answering.

Number of GNN layers: In the hyperparameter study, we evaluate how the number of GraphEncoder layers influences model

Table 2: Statistics of datasets.

Dataset	ExplaGraphs	SceneGraphs	WebQSP
#Graphs	2,766	100,000	4,737
Average #Nodes	5.17	19.13	1370.89
Average #Edges	4.25	68.44	4252.37
Node Attribute	Commonsense concepts	Object attributes	Entities in Freebase
Edge Attribute	Commonsense relations	Spatial relations	Relations in Freebase
Task	Commonsense reasoning	Scene graph QA	KGQA

Table 3: Performance comparison across ExplaGraphs, SceneGraphs, and WebQSP datasets for different configurations, including Inference-only, Frozen LLM with prompt tuning (PT), and Tuned LLM settings. Mean scores and standard deviations (mean ± std) are presented. The best result for each task is highlighted in bold, and the second best result is underlined.

Setting	Method	ExplaGraphs	SceneGraphs	WebQSP
Inference-only	Zero-shot	0.5650	0.3974	41.06
	Zero-CoT [17]	0.5704	0.5260	51.30
	CoT-BAG [49]	0.5794	0.5680	39.60
	KAPING [1]	0.6227	0.4375	52.64
Frozen LLM w/ PT	Prompt tuning	0.5763 ± 0.0243	0.6341 ± 0.0024	48.34 ± 0.64
	GraphToken [40]	0.8508 ± 0.0551	0.4903 ± 0.0105	57.05 ± 0.74
	G-Retriever	0.8516 ± 0.0092	0.8131 ± 0.0162	70.49 ± 1.21
Tuned LLM	LoRA	0.8538 ± 0.0353	0.7862 ± 0.0031	66.03 ± 0.47
	G-Retriever w/ LoRA	<u>0.8705 ± 0.0329</u>	<u>0.8683 ± 0.0072</u>	<u>73.79 ± 0.70</u>
	MixRAG w/ LoRA	0.8863 ± 0.0288	0.8712 ± 0.0064	75.31 ± 0.81

Table 4: Accuracy of Different Expert Combinations. Results on SceneGraphs are omitted due to high computational cost.

Expert Combination	ExplaGraphs	WebQSP
Only Entity	0.8247	67.76
Only Relation	0.8466	69.89
Only Subgraph	0.8765	73.99
Entity + Relation	0.8574	71.92
Entity + Subgraph	0.8646	72.91
Relation + Subgraph	0.8682	73.03
MixRAG (All)	0.8863	75.31

performance. As shown in Figure 4, accuracy improves consistently when increasing the number of layers from one to three, with three layers yielding the highest accuracy. However, adding a fourth layer results in performance degradation, likely due to over-smoothing. These results highlight the critical role of encoder depth in balancing expressive power and over-smoothing risk. They also suggest that deeper encoders do not necessarily translate to better performance in textual graph reasoning. Overall, using two or three layers is sufficient to capture the necessary structural and semantic information for effective question answering over textual graphs in MixRAG.

Distribution of Expert Weights Across Different Datasets: Figure 5 shows how the model distributes attention across different experts for each dataset (we select a subset of data points). For

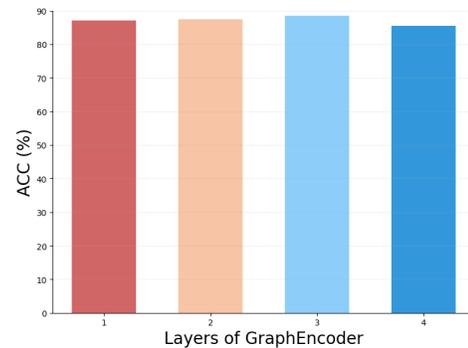


Figure 4: Accuracy of MixRAG on ExplaGraphs with respect to different numbers of GraphEncoder layers.

ExplaGraphs, the relation retriever receives the highest weight, which makes sense because the task involves comparing relations between concepts to determine whether one claim supports or contradicts another. In contrast, for SceneGraphs and WebQSP, the subgraph retriever plays the most important role, followed by the relation retriever, while the entity retriever contributes the least. These patterns reflect how different datasets require different types of reasoning, and MixRAG adapts its retrieval strategy to meet those needs.

Distribution of Expert Weights Across Different Query Types in WebQSP: Figure 6 shows how the model assigns expert weights

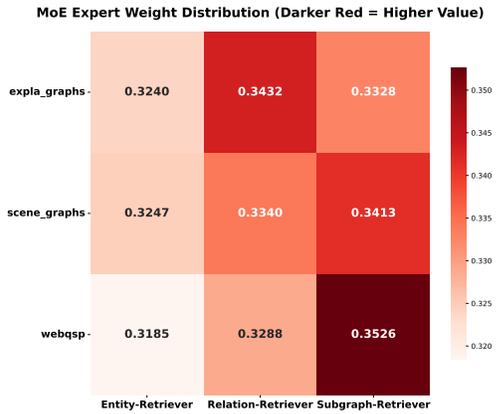


Figure 5: Distribution of Expert Weights Across Different Datasets (Darker Red = Higher Value).

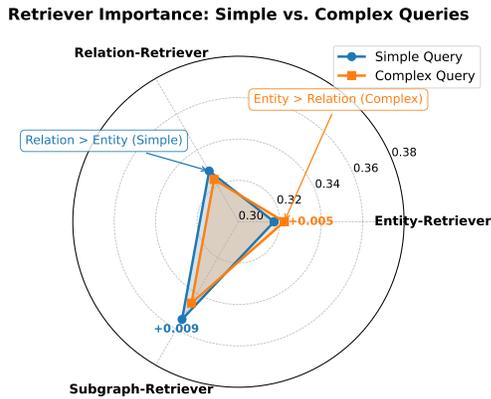


Figure 6: Distribution of Expert Weights Across Different Query Types in WebQSP.

to different retrievers for different types of queries in WebQSP. We divide the queries into two types: simple queries, which are 1-hop questions, and complex queries, which require multiple reasoning steps. From the figure, we can see that the subgraph retriever always gets the highest weight, no matter if the query is simple or complex. The difference lies in how the relation and entity retrievers are used. For simple queries, the relation retriever is more important than the entity retriever. This makes sense because a 1-hop query often needs just one relation to find the answer. But for complex queries, the entity retriever becomes more important. This is likely because the relation retriever may struggle to connect multiple relations and can introduce noise. In such cases, directly retrieving relevant entities works better and improves accuracy.

5 Related work

Retrieval-Augmented Generation (RAG). RAG has emerged as a powerful paradigm for mitigating key limitations of large language models (LLMs), particularly their tendency to hallucinate or produce factually inaccurate responses in knowledge-intensive

settings [5, 6, 22, 32]. By conditioning generation on retrieved external knowledge, RAG improves factual grounding and task-specific adaptability. Existing RAG approaches can be broadly categorized into three main paradigms. (1) Naive RAG adopts a straightforward pipeline consisting of indexing, retrieval, and generation, typically relying on embedding-based retrievers to identify relevant information [38]. (2) Advanced RAG enhances retrieval effectiveness through techniques applied before and after retrieval. Pre-retrieval methods include query rewriting, expansion, or transformation [39, 57], while post-retrieval methods involve reranking the retrieved candidates based on their relevance [41]. (3) Modular RAG provides greater flexibility by incorporating multiple types of data, such as unstructured text, structured tables, and knowledge graphs. It also utilizes large language models (LLMs) to generate or refine retrieval queries [56]. These methods support more robust and adaptable retrieval strategies, making them especially suitable for handling complex reasoning tasks.

LLMs and Knowledge Graphs. Knowledge graph reasoning has been studied for a long time. [20–36, 54]. Combining structured knowledge with large language models (LLMs) has been shown to improve factual accuracy, enhance interpretability, and boost reasoning performance. Broadly, there are three main strategies for integrating LLMs with knowledge graphs (KGs). First, KG-enhanced LLMs incorporate information from KGs either during pretraining [37, 46] or at inference time through retrieval-based conditioning [18, 50]. Second, LLM-augmented KGs use LLMs to support tasks such as KG construction [2], completion [14], and representation learning [51]. Third, synergistic integration refers to a co-evolution process in which the KG guides LLM inference [34], while the LLM simultaneously enriches or restructures the KG. This mutual reinforcement leads to more powerful and flexible reasoning capabilities [55].

Mixture-of-Experts. The Mixture of Experts framework [11] has established itself as a fundamental paradigm in machine learning for developing adaptive systems. Initial work focused on traditional machine learning implementations [13], with subsequent breakthroughs emerging through its integration with deep neural networks [16]. More recently, researchers have explored applying MoE approaches to in-context learning scenarios [52], demonstrating their potential to enhance large language model performance.

6 Conclusion

In this paper, we propose MixRAG, a Mixture of Expert Retrieval-Augmented Generation framework that combines multiple specialized graph retrievers to better match query intent. Each retriever focuses on a different aspect of graph semantics, enabling more accurate and flexible retrieval. To reduce noise in the retrieved subgraphs, we introduce a query-aware GraphEncoder that highlights relevant information and filters out distractions. Experiments demonstrate that our approach outperforms strong baselines, achieving state-of-the-art performance.

6.1 Acknowledge

This research was partially supported by the the internal funds and GPU servers provided by the Computer Science Department of

Emory University, the US National Science Foundation under Award Numbers 2442172, 2312502, 2319449, and the US National Institutes of Health under Award Numbers K25DK135913, RF1NS139325, R01DK143456 and U18DP006922.

References

- [1] Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023. Knowledge-Augmented Language Model Prompting for Zero-Shot Knowledge Graph Question Answering. In *Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE)*, Bhavana Dalvi Mishra, Greg Durrett, Peter Jansen, Danilo Neves Ribeiro, and Jason Wei (Eds.). Association for Computational Linguistics, Toronto, Canada, 78–106. doi:10.18653/v1/2023.nlrse-1.7
- [2] Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: Commonsense Transformers for Automatic Knowledge Graph Construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Anna Korhonen, David Traum, and Lluís Màrquez (Eds.). Association for Computational Linguistics, Florence, Italy, 4762–4779. doi:10.18653/v1/P19-1470
- [3] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. 2020. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 3438–3445.
- [4] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitanitsky, Robert Osazuwa Ness, and Jonathan Larson. 2025. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. arXiv:2404.16130 [cs.CL] <https://arxiv.org/abs/2404.16130>
- [5] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise Zero-Shot Dense Retrieval without Relevance Labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 1762–1777. doi:10.18653/v1/2023.acl-long.99
- [6] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv:2312.10997 [cs.CL] <https://arxiv.org/abs/2312.10997>
- [7] Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond I.I.D.: Three Levels of Generalization for Question Answering on Knowledge Bases. In *Proceedings of the Web Conference 2021 (WWW '21)*. ACM, 3477–3488. doi:10.1145/3442381.3449992
- [8] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*. 1024–1034.
- [9] 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. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. <https://openreview.net/forum?id=MPJ3oXtZl>
- [10] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685 [cs.CL] <https://arxiv.org/abs/2106.09685>
- [11] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. Adaptive Mixtures of Local Experts. *Neural Computation* 3, 1 (1991), 79–87. doi:10.1162/neco.1991.3.1.79
- [12] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Léo Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of Experts. arXiv:2401.04088 [cs.LG] <https://arxiv.org/abs/2401.04088>
- [13] Michael Jordan, Zoubin Ghahramani, and Lawrence Saul. 1996. Hidden Markov Decision Trees. In *Advances in Neural Information Processing Systems*, M.C. Mozer, M. Jordan, and T. Petsche (Eds.), Vol. 9. MIT Press. https://proceedings.neurips.cc/paper_files/paper/1996/file/6c8dba7d0df1c4a79dd07646be9a26c8-Paper.pdf
- [14] Bosung Kim, Taesuk Hong, Youngjoong Ko, and Jungyun Seo. 2020. Multi-Task Learning for Knowledge Graph Completion with Pre-trained Language Models. In *Proceedings of the 28th International Conference on Computational Linguistics*, Donia Scott, Nuria Bel, and Chengqing Zong (Eds.). International Committee on Computational Linguistics, Barcelona, Spain (Online), 1737–1743. doi:10.18653/v1/2020.coling-main.153
- [15] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
- [16] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. arXiv:1609.02907 [cs.LG] <https://arxiv.org/abs/1609.02907>
- [17] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large Language Models are Zero-Shot Reasoners. arXiv:2205.11916 [cs.CL] <https://arxiv.org/abs/2205.11916>
- [18] 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 (2020), 9459–9474.
- [19] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (Vancouver, BC, Canada) (NIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, Article 793, 16 pages.
- [20] Lihui Liu. 2025. Graph-O1: Monte Carlo Tree Search with Reinforcement Learning for Text-Attributed Graph Reasoning. *arXiv preprint arXiv:2512.17912* (2025).
- [21] Lihui Liu. 2025. HyperKGR: Knowledge Graph Reasoning in Hyperbolic Space with Graph Neural Network Encoding Symbolic Path. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Suzhou, China.
- [22] Lihui Liu. 2025. Monte Carlo Tree Search for Graph Reasoning in Large Language Model Agents. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (Seoul, Republic of Korea) (CIKM '25)*. Association for Computing Machinery, New York, NY, USA.
- [23] Lihui Liu. 2025. A Survey of LLM Post-Training: Approaches to Training, Reasoning and Data Creation. *Authoria Preprints* (2025).
- [24] Lihui Liu, Yuzhong Chen, Mahashweta Das, Hao Yang, and Hanghang Tong. 2023. Knowledge Graph Question Answering with Ambiguous Query. In *Proceedings of the ACM Web Conference 2023*.
- [25] Lihui Liu, Boxin Du, Yi Ren Fung, Heng Ji, Jiejun Xu, and Hanghang Tong. 2021. Kompare: A knowledge graph comparative reasoning system. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3308–3318.
- [26] Lihui Liu, Boxin Du, Heng Ji, ChengXiang Zhai, and Hanghang Tong. 2021. Neural-answering logical queries on knowledge graphs. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 1087–1097.
- [27] Lihui Liu, Boxin Du, Jiejun xu, and Hanghang Tong. 2019. G-Finder: Approximate Attributed Subgraph Matching. In *2019 IEEE International Conference on Big Data (Big Data)*. 513–522. doi:10.1109/BigData47090.2019.9006525
- [28] Lihui Liu, Boxin Du, Jiejun Xu, Yinglong Xia, and Hanghang Tong. 2022. Joint Knowledge Graph Completion and Question Answering. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (Washington DC, USA) (KDD '22)*. Association for Computing Machinery, New York, NY, USA, 1098–1108.
- [29] Lihui Liu, Blaine Hill, Boxin Du, Fei Wang, and Hanghang Tong. 2024. Conversational Question Answering with Language Models Generated Reformulations over Knowledge Graph. In *Findings of the Association for Computational Linguistics: ACL 2024*. Association for Computational Linguistics, Bangkok, Thailand.
- [30] Lihui Liu, Houxiang Ji, Jiejun Xu, and Hanghang Tong. 2022. Comparative reasoning for knowledge graph fact checking. In *2022 IEEE International Conference on Big Data (Big Data)*. IEEE, 2309–2312.
- [31] Lihui Liu, Jinha Kim, and Vidit Bansal. 2024. Can Contrastive Learning Refine Embeddings. arXiv:2404.08701 [cs.LG] <https://arxiv.org/abs/2404.08701>
- [32] Lihui Liu and Kai Shu. 2025. *Unifying Knowledge in Agentic LLMs: Concepts, Methods, and Recent Advancements*. Technical Report. TechRxiv.
- [33] Lihui Liu and Hanghang Tong. 2023. Knowledge graph reasoning and its applications. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5813–5814.
- [34] Lihui Liu, Zihao Wang, Ruizhong Qiu, Yikun Ban, Eunice Chan, Yangqiu Song, Jingrui He, and Hanghang Tong. 2024. Logic query of thoughts: Guiding large language models to answer complex logic queries with knowledge graphs. *arXiv preprint arXiv:2404.04264* (2024).
- [35] Lihui Liu, Zihao Wang, and Hanghang Tong. 2025. Neural-Symbolic Reasoning over Knowledge Graphs: A Survey from a Query Perspective. *SIGKDD Explor. NewsL*. 27, 1 (July 2025), 124–136. doi:10.1145/3748239.3748249
- [36] Lihui Liu, Ruining Zhao, Boxin Du, Yi Ren Fung, Heng Ji, Jiejun Xu, and Hanghang Tong. 2022. Knowledge Graph Comparative Reasoning for Fact Checking: Problem Definition and Algorithms. *IEEE Data Eng. Bull.* 45, 4 (2022), 19–38.
- [37] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-bert: Enabling language representation with knowledge graph. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 2901–2908.
- [38] Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query Rewriting in Retrieval-Augmented Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore.
- [39] Wenjun Peng, Guiyang Li, Yue Jiang, Zilong Wang, Dan Ou, Xiaoyi Zeng, Derong Xu, Tong Xu, and Enhong Chen. 2024. Large Language Model based Long-tail

- Query Rewriting in Taobao Search. arXiv:2311.03758 [cs.IR] <https://arxiv.org/abs/2311.03758>
- [40] 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:2402.05862 [cs.LG] <https://arxiv.org/abs/2402.05862>
- [41] Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. 2024. Large Language Models are Effective Text Rankers with Pairwise Ranking Prompting. In *Findings of the Association for Computational Linguistics: NAACL 2024*, Kevin Duh, Helena Gomez, and Steven Bethard (Eds.). Association for Computational Linguistics, Mexico City, Mexico.
- [42] A Radford, J Wu, and R Child. 2018. Language Models are Unsupervised Multitask Learners. (2018).
- [43] 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)*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, Hong Kong, China.
- [44] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. arXiv:1701.06538 [cs.LG] <https://arxiv.org/abs/1701.06538>
- [45] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023. Replug: Retrieval-augmented black-box language models. *arXiv preprint arXiv:2301.12652* (2023).
- [46] Tianxiang Sun, Yunfan Shao, Xipeng Qiu, Qipeng Guo, Yaru Hu, Xuanjing Huang, and Zheng Zhang. 2020. CoLAKE: Contextualized Language and Knowledge Embedding. In *Proceedings of the 28th International Conference on Computational Linguistics*, Donia Scott, Nuria Bel, and Chengqing Zong (Eds.). International Committee on Computational Linguistics, Barcelona, Spain (Online), 3660–3670. doi:10.18653/v1/2020.coling-main.327
- [47] Romal Thoppilan, Daniel De Freitas, and Jamie Hall. 2022. LaMDA: Language Models for Dialog Applications. arXiv. doi:10.48550/ARXIV.2201.08239
- [48] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. arXiv:1710.10903 [stat.ML] <https://arxiv.org/abs/1710.10903>
- [49] Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2024. Can Language Models Solve Graph Problems in Natural Language? arXiv:2305.10037 [cs.CL] <https://arxiv.org/abs/2305.10037>
- [50] Jianing Wang, Qiushi Sun, Xiang Li, and Ming Gao. 2024. Boosting Language Models Reasoning with Chain-of-Knowledge Prompting. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 4958–4981. doi:10.18653/v1/2024.acl-long.271
- [51] Peng Wang, Xin Xie, Xiaohan Wang, and Ningyu Zhang. 2023. Reasoning Through Memorization: Nearest Neighbor Knowledge Graph Embeddings. arXiv:2201.05575 [cs.CL] <https://arxiv.org/abs/2201.05575>
- [52] Ruo Chen Wang, Sohyun An, Minhao Cheng, Tianyi Zhou, Sung Ju Hwang, and Cho-Jui Hsieh. 2024. One Prompt is not Enough: Automated Construction of a Mixture-of-Expert Prompts. In *International Conference on Machine Learning*.
- [53] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903 [cs.CL] <https://arxiv.org/abs/2201.11903>
- [54] Yuchen Yan, Lihui Liu, Yikun Ban, Baoyu Jing, and Hanghang Tong. 2021. Dynamic knowledge graph alignment. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 4564–4572.
- [55] Michihiro Yasunaga, Antoine Bosselut, Hongyu Ren, Xikun Zhang, Christopher D. Manning, Percy Liang, and Jure Leskovec. 2022. Deep Bidirectional Language-Knowledge Graph Pretraining. In *Neural Information Processing Systems (NeurIPS)*.
- [56] Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2023. Generate rather than retrieve: Large language models are strong context generators. In *International Conference for Learning Representation (ICLR)*.
- [57] Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V Le, and Denny Zhou. 2024. Take a Step Back: Evoking Reasoning via Abstraction in Large Language Models. arXiv:2310.06117 [cs.LG] <https://arxiv.org/abs/2310.06117>