Mixture of Structural-and-Textual Retrieval over Text-rich Graph Knowledge Bases

Anonymous ACL submission

Abstract

Text-rich Graph Knowledge Bases (TG-KBs) have become increasingly crucial for answering queries by providing textual and structural knowledge. However, current retrieval methods often retrieve these two types of knowledge in isolation without considering their mutual reinforcement and some hybrid methods even bypass structural retrieval entirely after neighboring aggregation. To fill in this gap, we propose a Mixture of Structural-and-Textual Retrieval (MoR) to retrieve these two types of knowledge via a Planning-Reasoning-Organizing framework. In the Planning stage, MoR generates textual planning graphs delineating the logic for answering queries. Following planning graphs, in the Reasoning stage, MoR interweaves structural traversal and textual matching to obtain candidates from TG-KBs. In the Organizing stage, MoR further reranks fetched candidates based on their structural trajectory. Extensive experiments demonstrate the superiority of MoR in harmonizing structural and textual retrieval with insights, including uneven retrieving performance across different query logics and the benefits of integrating structural trajectories for candidate reranking. Our code will be publically available upon acceptance.

1 Introduction

Text-rich Graph Knowledge Bases (TG-KBs), due to their structured representation of textual documents, are ubiquitously used for storing textual and structural knowledge (Chen et al., 2024). For example, scholars retrieve relevant research to advance scientific discoveries in academic paper management systems where nodes represent papers and edges denote references. With large language models (LLMs)-powered generators gradually approaching human intelligence in language comprehension and generation, retrieving supporting knowledge from TG-KBs to contextualize and ground generation has become increasingly crucial for correctly answering queries (Gao et al., 2023b).

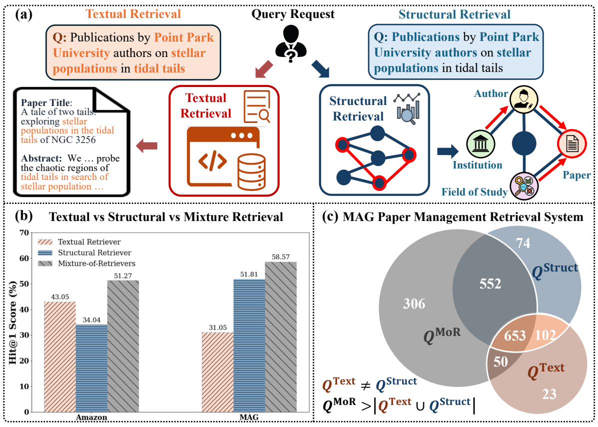

Figure 1: (a) Textual retrieval and structural retrieval. (b) The effectiveness of retrieval methods varies across different TG-KBs. (c) Within the same TG-KB, queries addressed by textual (i.e., $\mathcal{Q}^{\text{Text}}$) and structural retrieval (i.e., $\mathcal{Q}^{\text{Struct}}$) exhibit both overlaps and distinctiveness.

Since supporting knowledge in TG-KBs typically appears in both the textual and structural formats (Jin et al., 2024b; Kolomiyets and Moens, 2011), retrieval methods should be designed to leverage both formats effectively as Figure 1(a). Textual retrieval methods retrieve textual knowledge such as indexed documents (Mitra and Chaudhuri, 2000) based on its similarity to the given query and can be broadly categorized into lexical methods (e.g., BM25) and semantic methods (e.g., Contriever) (Karpukhin et al., 2020; Izacard et al., 2022). Structural retrieval methods retrieve structural knowledge such as neighboring entities (Edge et al., 2024; Jiang et al., 2023; Wang et al., 2024) by conducting graph traversal and applying graph machine learning models (Tian et al., 2024; Yasunaga et al., 2021a). Despite the advancements in both textual and structural retrieval, they are often applied independently and fail to mutually reinforce each other. As shown by Figure 1(b), neither structural retrieval by following the logical structure of the query nor textual retrieval by conducting Top-K BM25 matching can achieve better performance on both Amazon and MAG datasets simultaneously.

To retrieve hybrid knowledge from TG-KBs, recent works (Xia et al., 2024; Li et al., 2024) aggregate neighboring documents to fuse structural knowledge into textual narratives followed by textual retrieval, with Xia et al. (2024) filtering irrelevant neighbors by their relations and Li et al. (2024) weighted aggregating neighbors based on their fields. However, three challenges remain. First, rewording aggregated neighbors requires frequently invoking LLMs, resulting in prohibitive resources when facing long documents with exponentially growing neighbors. Second, structural signals that humans use to form logical plans are completely discarded after neighbor aggregation. Third, the rigid neighbor aggregation overlooks the varying desire for structural and textual knowledge across different queries and TG-KBs. In Figure 1(c), even within MAG, queries answered by textual retrieval (i.e., $\mathcal{Q}^{\text{Text}}$) are not the same as queries answered by structural retrieval (i.e., $\mathcal{Q}^{\text{Struct}}$).

To address the above three challenges, we infuse the mixture-of-expert philosophy into retrieval design and propose a Mixture of Structural-and-Textual Retrieval (MoR) in Figure 2. MoR begins with a planning module that generates planning graphs outlining query logics that preserve structural signals without rewording aggregated neighbors, overcoming the first and second challenges. Next, MoR interleaves structural traversal and textual matching in the reasoning module, enabling these two retrieval to reinforce each other. Finally, MoR devises a structure-aware Reranker in the organization module that adaptively adjusts the retrieved textual/structural knowledge, addressing the third challenge. Via Planning–Reasoning–Organizing, MoR intelligently retrieves structural and textual knowledge based on query patterns. Our key contributions are:

- **Planning via Textual Graph Generation**: We define retrieval planning as generating textual graphs that outline the logical structure, i.e., the plan, for identifying entities relevant to the query.
- **Reasoning via Mixture of Structural-and-Textual Traversal**: We devise a mixed traversal by interweaving textual matching and structural traversal to retrieve knowledge following the query logic depicted by the generated plan.
- **Organizing via Structure-aware Rerank**: With candidates obtained from mixed traversal, we propose a Structure-aware Rerank to select Top-K candidates via their traversal trajectory.

2 Preliminary

Notations: A Text-rich Graph Knowledge Base (TG-KB) $\mathcal{B}$ generally consists of a set of connected nodes $\mathcal{V}$ in the graph with each node $v \in \mathcal{V}$ associated with its corresponding document $\mathcal{D}_v \in \mathcal{D}$ and category $\mathcal{E}_v \in \mathcal{E}$. When retrieving nodes with supporting documents from $\mathcal{B}$ for answering a given query $Q \in \mathcal{Q}$, we typically follow certain rationale encapsulating the underlying logic of that query (Xu et al., 2024; Xue et al., 2024), which can be characterized by a text-attributed planning graph G. As utilized in many existing works (Jin et al., 2024a; Wu et al., 2024b), this planning graph can usually be decomposed into multiple reasoning paths $G = \{\mathcal{P}_i\}_{i=1}^{|G|}$ where the i^{th} reasoning path $\mathcal{P}_i = (p_{i1} \rightarrow p_{i2} \rightarrow, ..., \rightarrow p_{iL_i})$ is a distinctive reasoning chain of length L_i encoding a unique logic and the j^{th} node p_{ij} corresponds to an entity in $\mathcal{B}$ with its own category $\mathcal{E}_{p_{ij}}$ and textual restriction $\mathcal{T}_{p_{ij}}$ extracted from the query. For example, in Figure 1(a), the query *Publications by Point...* has a planning graph with two paths, i.e., $\mathcal{P}_1 =$ (Institution $\rightarrow$ Author $\rightarrow$ Paper) and $\mathcal{P}_2 =$ (Field-of-Study $\rightarrow$ Paper), where the category and textual retriection of the first node on $\mathcal{P}_1$ are $\mathcal{E}_{p_{11}} =$ Institution and $\mathcal{T}_{p_{11}} =<$ *Point Park Univeristy* $>$, respectively. Comprehensive notations are summarized in Table 4 in Appendix A.

Problem Setup: With the above notations, the investigated problem here is to retrieve entities $\mathcal{C} \subseteq \mathcal{V}$ supporting answering a given query Q.

Textual Retrieval retrieves candidates based on the textual signals of both the query and documents. One common strategy is to retrieve candidates $\widetilde{\mathcal{C}}$ from the whole documents $\mathcal{D}$ that have Top-K textual similarity to query Q measured by lexical or semantic similarity (Vijaymeena and Kavitha, 2016). The textual retrieval used in MoR retrieves documents for a given query by matching them with textual descriptions in the query, e.g., matching *stellar populations in tidal tails* shown in Figure 1.

Structural Retrieval retrieves candidates by applying prescribed rules to structured databases such as knowledge graphs and SQL (Guo et al., 2023). Common strategies include graph-based traversal (e.g., BFS, DFS) and rule fetching (Jiang et al., 2023). Specifically, MoR conducts structural retrieval by traversing neighbors of certain categories from the generated planning graph. For example, in Figure 1(a), only "Paper" typed neighbors of the Author can be traversed by our structural retrieval.

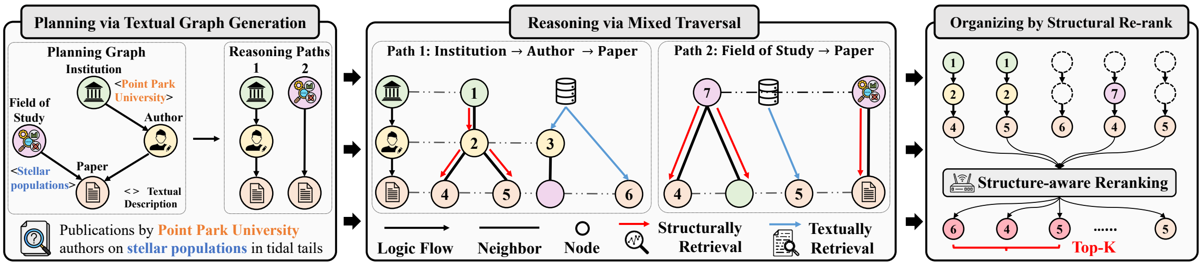

Figure 2: Our MoR framework consists of a planning module to generate a planning graph, a reasoning module to conduct mixed traversal, and an organizing module to rerank the retrieved candidates.

3 Framework

In a nutshell, we formulate our MoR as the conditional distribution $P_{\mathbf{\Theta}}(\mathcal{C}|Q,\mathcal{B})$ of retrieved candidates $\mathcal{C}$ given the user input query Q over TG-KB $\mathcal{B}$, which is further factorized into three distributions corresponding to our proposed three modules: planning via generating the text-attributed planning graph G, reasoning via conducting mixture of structural-and-textual traversal to obtain intermediate candidates $\widetilde{\mathcal{C}}$ following the generated planning graph G, and organizing via applying structure-aware reranking to the obtained candidates $\widetilde{\mathcal{C}}$, obtaining final candidates $\mathcal{C}$:

$$P_{\mathbf{\Theta}}(\mathcal{C}|Q,\mathcal{B}) = \sum_{G\in\mathbb{G}}\Big[\sum_{\widetilde{\mathcal{C}}\in\mathbb{C}} \underbrace{P_{\mathbf{\Theta}_3}(\mathcal{C}|\widetilde{\mathcal{C}},G,Q,\mathcal{B})}_{\text{Organizing}} \times \underbrace{P_{\mathbf{\Theta}_2}(\widetilde{\mathcal{C}}|G,Q,\mathcal{B})}_{\text{Reasoning}}\Big] \times \underbrace{P_{\mathbf{\Theta}_1}(G|Q,\mathcal{B})}_{\text{Planning}}$$

where $P_{\mathbf{\Theta}_1}(G|Q,\mathcal{B})$ is the probability distribution of generating the text-attributed planning graph G given the input query Q and TG-KB $\mathcal{B}$; $P_{\mathbf{\Theta}_2}(\widetilde{\mathcal{C}}|G,Q,\mathcal{B})$ is the probability distribution of retrieving intermediate candidates $\widetilde{\mathcal{C}}$ given the planning graph G and the query Q via our mixed traversal; $P_{\mathbf{\Theta}_3}(\mathcal{C}|\widetilde{\mathcal{C}},G,Q,\mathcal{B})$ is the probability distribution of reranking the intermediate candidates so that Top-K positions form the ground-truth entities $\mathcal{C}$. $\mathbb{G}/\mathbb{C}$ denotes the collection of all possible planning graphs and all possible configurations of size-K candidate nodes from all nodes $\mathcal{V}$ of TG-KB $\mathcal{B}$. The overall objective is to maximize the likelihood of retrieving ground-truth candidates $\mathcal{C}$ for each input query $Q \in \mathcal{Q}$:

$$\mathbf{\Theta}^* = \arg\max_{\mathbf{\Theta}} \prod_{Q\in\mathcal{Q}} P_{\mathbf{\Theta}}(\mathcal{C}|Q,\mathcal{B}) \tag{1}$$

Following the above paradigm, we next introduce the three components: Planning via textual graph generation in Section 3.1, Reasoning via mixed traversal in Section 3.2, and Organizing via structure-aware reranking in Section 3.3.

3.1 Planning via Textual Graph Generation

To effectively reason over the underlying logic of queries and answer them, we propose a planning module that constructs a planning graph to capture their underlying logical structures. Unlike conventional approaches relying on rigid heuristics, e.g., shortest-path retrieval in knowledge graphs (Luo et al., 2023; Delile et al., 2024), or step-by-step prompting of LLMs, which incurs high computational costs (Sun et al., 2023a; Wang et al., 2024), our method generates the entire planning graph in one shot, eliminating repeated LLM calls. More importantly, as planning graphs integrate entity restrictions encoding query-specific constraints and entity categories capturing broader logical structure, our MoR can generalize learned patterns and efficiently adapt to new queries with the same underlying logic. For example, any query with the form *Papers associated with <institution> and are in the field of <field>* shares the same patterns with the query in Figure 2. Below, we first formalize the planning graph and then optimize its generation.

3.1.1 Planning Graph Formulation

A planning graph G is a structured representation where nodes represent entities and edges denote their logical relations. Each entity is associated with both a category and query-specific restriction. For example, given the query *Can you give me publications by Point Park University authors on stellar populations in tidal tails*, the generated planning graph is: G = (Institution<*Point Park University*> $\rightarrow$ Author $\rightarrow$ Paper $\leftarrow$ Field-of-Study<*Stellar Population*>) with Institution, Author, Paper, Field-of-Study as categories and <*Point Park University*>, <*Stellar Populations*> as restrictions. Note that edges in our planning graph can also possess different categories. For example, in the biomedical TG-KBs, the relation between Disease and Drug entities could be Indication or Contra-indication (Wu et al., 2024b), adding a finer level of semantic distinction to the relation.

3.1.2 Planning Graph Optimization

To ensure that our generated planning graph captures the query logic, we train a textual graph generator to maximize the likelihood of generating ground-truth planning graphs given their queries. Formally, given the joint distribution of the training pairs between queries and planning graphs $P^{\text{Train}}_{\mathbb{Q}\times\mathbb{G}}$, we optimize the planning module $P_{\mathbf{\Theta}_1}$ by solving:

$$\arg\max_{\mathbf{\Theta}_1} \mathbb{E}_{(Q,G)\sim P^{\text{Train}}_{\mathbb{Q}\times\mathbb{G}}} \log P_{\mathbf{\Theta}_1}(G|Q,\mathcal{B}) \quad (2)$$

To avoid the combinatorial explosion of exponentially growing planning graph candidates (You et al., 2018), we decompose each planning graph into multiple reasoning paths $G = \{\mathcal{P}_i\}_{i=1}^{|G|}$. Each path $\mathcal{P}_i = (p_{i1} \rightarrow, ..., \rightarrow p_{iL_i})$ represents a distinct reasoning chain, where node p_{ij} denotes an entity in TG-KB sharing the same textual category $\mathcal{E}_{p_{ij}}$ and satisfying the restriction $\mathcal{T}_{p_{ij}}$ from the query. Given the sequential nature and textual formats of these decomposed reasoning paths, LLMs can be naturally employed here as the planning graph generator, which conducts next-token prediction by predicting j^{th} token t_j conditioned on preceding tokens $t_{<j}$, the query Q and the TG-KB $\mathcal{B}$:

$$P_{\mathbf{\Theta}_1}(G|Q) = \prod_{j=1}^{n} P_{\mathbf{\Theta}_1}(t_j|t_{<j}, Q, \mathcal{B}). \quad (3)$$

Note that our proposed planning graph generator is not limited to LLMs. Any graph generative model preserving both structural dependencies and textual associations can be employed (Zhu et al., 2022).

3.2 Reasoning via Mixed Traversal

Following the reasoning paths of the above planning graph $G = \{\mathcal{P}_i\}_{i=1}^{|G|}$, the reasoning module conducts a mixed traversal by interweaving neighbor fetching and textual matching to form intermediate candidates $\widetilde{\mathcal{C}}$, which are introduced next.

3.2.1 Structural Traversal

Following the definition in Section 2 that structural retrieval follows prescribed rules for knowledge retrieval, here we set these prescribed rules to be iteratively performing layer-wise breadth-first-search that traverses neighboring entities with categories aligning with those in the reasoning paths. Concretely, reasoning at the l^{th}-step of the planning path $\mathcal{P}_i$, we check for each node v in candidates set of last layer $\forall v \in \widetilde{\mathcal{C}}_i^{l-1}$ and fetch its neighbors $\forall u \in \mathcal{N}_v$ with the same category as the corresponding node p_{il} (i.e., $\mathcal{E}_u = \mathcal{E}_{p_{il}}$) in the reasoning path, which can be mathematically formulated as:

$$\widetilde{\mathcal{C}}_i^{l,\text{Struct}} = \cup_{v\in\widetilde{\mathcal{C}}_i^{l-1}}\{u|u \in \mathcal{N}_v, \mathcal{E}_u = \mathcal{E}_{p_{il}}\} \quad (4)$$

where $\widetilde{\mathcal{C}}_i^{l,\text{Struct}}$ denotes the set of structurally retrieved entities at the l^{th} reasoning step according to the path $\mathcal{P}_i$ and $\mathcal{E}_u = \mathcal{E}_{p_{il}}$ ensures that the category of the traversed neighbor u matches the corresponding entity category routine by the planning graph, resonating the nature of rule-based structural retrieval. Note that the seeding candidates $\widetilde{\mathcal{C}}_i^{1,\text{Struct}}$ at the very first layer are initialized by retrieving Top-K entities through textual matching, i.e., $\widetilde{\mathcal{C}}_i^{1,\text{Struct}} = \widetilde{\mathcal{C}}_i^{1,\text{Text}}$, which is introduced next.

3.2.2 Textual Matching

In addition to retrieving structural knowledge, our MoR also retrieves textual knowledge via Textual Matching, which retrieves candidates based on their textual similarity to queries. For each reasoning node p_{il} at l^{th} reasoning step along the reasoning path $\mathcal{P}_i$, we concatenate the query and the textual restriction of p_{il}, i.e., $Q' = [Q : \mathcal{T}_{p_{il}}]$, then compute its textual similarity to documents of nodes in TG-KB, i.e., $\phi(Q', \mathcal{D}_v), \forall v \in \mathcal{V}$, and finally retrieve the Top-K scored candidates:

$$\widetilde{\mathcal{C}}_i^{l,\text{Text}} = \text{TopK}(\{v \mid v \in \mathcal{V}, \mathcal{E}_v = \mathcal{E}_{p_{il}}\}, \phi(Q', \mathcal{D}_v)) \quad (5)$$

Integrating candidates from structural traversal and textual matching together, the final candidates at l^{th}-step of $\mathcal{P}_i$ are formed as:

$$\widetilde{\mathcal{C}}_i^{l} = \widetilde{\mathcal{C}}_i^{l,\text{Struct}} \cup \widetilde{\mathcal{C}}_i^{l,\text{Text}}, \forall l \in \{1, 2, ..., L_i\} \quad (6)$$

The integrated candidates $\widetilde{\mathcal{C}}_i^{l}$ serve as seeding nodes initializing the next round of planning graph-guided structural traversal and textual matching, which creates a mutual reinforcement between structural and textual knowledge since previously retrieved two knowledge can both inform next round of structural/textual knowledge retrieval.

We iteratively conduct mixed traversal for every reasoning path $\mathcal{P}_i \in G$ and integrate retrieved entities together by taking their intersection, i.e., $\widetilde{\mathcal{C}} = \cap_{\mathcal{P}_i\in G}\widetilde{\mathcal{C}}_i^{L_i}$, adhering to the fact that candidates should simultaneously satisfy the logic routine by all reasoning paths. Note that no training is involved in the mixed graph traversal, i.e., $P_{\mathbf{\Theta}_2}(\widetilde{\mathcal{C}}|G, Q, \mathcal{B}) = P(\widetilde{\mathcal{C}}|G, Q, \mathcal{B})$. Future works could explore optimizing graph traversal by rewards from agent-environment interactions (Nguyen et al., 2024).

3.3 Organizing via Structure-aware Rerank

Although the retrieved candidates from Section 3.2 strictly adhere to the prescribed rule given by the planning graph, the sheer volume of candidates misaligns with realistic constraints (e.g., Top-20 retrieval budget (Zeng et al., 2024)) and may even cause difficulty to downstream executors such as long-context challenges for LLMs. To better emulate human reasoning, where multiple clues are gathered, analyzed in relation to the query, and synthesized into a coherent answer, we propose a structure-aware reranker to organize and rerank the candidates $\widetilde{\mathcal{C}}$, and select Top-K ones as the final retrieved answers $\mathcal{C}$. Instead of relying only on textual features (Hu et al., 2019), our reranker assigns a ranking score based on features of structural trajectories obtained from the mixed traversal in Section 3.2, innovatively leveraging both structural and textual knowledge in reranking.

Previously, $\widetilde{\mathcal{C}}$ is defined as intermediate retrieved entities. To consider structural features in reranking, we pair each retrieved candidate in $\widetilde{\mathcal{C}}$ with its corresponding traversal trajectory obtained from the reasoning module. Specifically, each trajectory $\mathcal{P}_i$ of length L_i is featuring three types of attributes:

- **Textual Fingerprint (TF)**: Concatenation of similarity scores between the expanded query and each node on the path: $\|_{l=1}^{L_i}\phi(Q', \mathcal{D}_{p_{il}})$.
- **Structural Fingerprint (SF)**: Concatenation of node categories at each step on the path: $\|_{l=1}^{L_i}\mathcal{E}_{p_{il}}$
- **Traversal Identifier (TI)**: Concatenation of the indicator specifying whether each step uses a structural or textual retrieval: $\|_{l=1}^{L_i}\mathcal{I}_{p_{il}}$.

We then train a reranker on these trajectories using the cross-entropy loss. For a training query Q and its associated candidate trajectory $\mathcal{P}_i$, the loss is computed as follows:

$$\mathcal{L}_{\Theta_3} = -\sum_{(\mathcal{P}_i,Q)\in\widetilde{\mathcal{C}}}\sum_{j=1}^{2} y_j^i \log(\sigma(f(\underbrace{\|_{l=1}^{L_i}\phi(Q',\mathcal{D}_{p_{il}})}_{\text{Textual Fingerprint}} : \underbrace{\|_{l=1}^{L_i}\mathcal{E}_{p_{il}}}_{\text{Structural Fingerprint}} : \underbrace{\|_{l=1}^{L_i}\mathcal{I}_{p_{il}}}_{\text{Traversal Identifier}}))_j). \tag{7}$$

where $f(\cdot)$ is the reranker producing a score for each $(Q, \mathcal{P}_i)$ pair, $\sigma(\cdot)$ denotes the softmax function, and $y_j^i \in \{0, 1\}$ indicates whether the j-th candidate is a correct (positive) or incorrect (negative) match for Q. This formulation encourages the reranker to assign higher scores to positive trajectories, thereby improving ranking performance.

4 Experiment

4.1 Experimental Setup

We briefly introduce experimental settings to verify our proposed MoR, including Datasets & Baselines, Implementation Details, and Evaluation Metrics. More details are in Appendix B.

Datasets & Baselines: We use three TG-KBs from STaRK (Wu et al., 2024b) covering three different domains of knowledge, including E-commerce Products (Amazon), Academic Papers (MAG), and Biomedicine (Prime). We compare our MoR with baselines established by Wu et al. (2024b) and more recent state-of-the-art models such as KAR (Xia et al., 2024) and MFAR (Li et al., 2024), which are two most advanced hybrid knowledge retrieval approaches for TG-KBs.

Implementation Details: To enhance the planning capability of our planning module, we fine-tune the Llama 3.2 (3B) on 1000 sampled queries with their corresponding ground-truth planning graphs, serving as the textual graph generator. In the absence of ground-truths, we synthesize them using LLMs. For the Prime dataset, we empirically find that directly prompting LLMs can hardly generate accurate planning graphs due to the lack of biomedical domain knowledge (Shen et al., 2024). Therefore, we adopt an alternative approach. First, we instruct LLMs to extract triplets from each query and then construct the planning graphs by merging triplets with shared entities. During mixed traversal, textual matching can be implemented using any lexical or semantic methods. For this study, we employ BM25 for Amazon and MAG and fine-tune a contriever to complement the biomedical knowledge for Prime. To initialize the structural traversal, we employ textual matching to locate the top 5 nodes most relevant to the query as seeds. Additionally, at each layer, we incorporate the top 10 nodes retrieved via textual matching and append them to the current candidate set for the next round of traversal. Notably, due to the uncertainty of LLMs, the generated planning graphs can be invalid. In this case, we will directly conduct textual matching to retrieve candidates. For our ablations without reranker, we employ Ada-002 (Wu et al., 2024b) with cosine similarity as the scorer to rank candidates for evaluating performance.

Evaluation Metrics: We follow Wu et al. (2024b) for evaluation by reporting Hit@1 (H@1), Hit@5 (H@5), Recall@20 (R@20), and mean reciprocal rank MRR.

Category	Retrieval Baseline	AMAZON				MAG				PRIME				AVERAGE			
		H@1	H@5	R@20	MRR	H@1	H@5	R@20	MRR	H@1	H@5	R@20	MRR	H@1	H@5	R@20	MRR
Textual	BM25 (Wu et al., 2024b)	44.94	67.42	53.77	55.30	25.85	45.25	45.69	34.91	12.75	27.92	31.25	19.84	27.85	46.86	43.57	36.68
	Ada-002 (Wu et al., 2024b)	39.16	62.73	53.29	50.35	29.08	49.61	48.36	38.62	12.63	31.49	36.00	21.41	26.96	47.94	45.88	36.79
	Multi-ada-002 (Wu et al., 2024b)	40.07	64.98	55.12	51.55	25.92	50.43	50.80	36.94	15.10	33.56	38.05	23.49	27.03	49.66	47.99	37.33
	DPR (Karpukhin et al., 2020)	15.29	47.93	44.49	30.20	10.51	35.23	42.11	21.34	4.46	21.85	30.13	12.38	10.09	35.00	38.91	21.31
Structural (KG)	QAGNN (Yasunaga et al., 2021b)	26.56	50.01	52.05	37.75	12.88	39.01	46.97	29.12	8.85	21.35	29.63	14.73	16.10	36.79	42.88	27.20
	ToG (Sun et al., 2023b)	-	-	-	-	13.16	16.17	11.30	14.18	6.07	15.71	13.07	10.17	9.62	15.94	12.18	12.18
Hybrid	AvaTaR (Wu et al., 2024a)	49.87	69.16	**60.57**	58.70	44.36	59.66	50.63	51.15	18.44	36.73	39.31	26.73	37.56	55.18	50.17	45.53
	KAR (Xia et al., 2024)	**54.20**	68.70	57.24	61.29	50.47	69.57	60.28	58.65	30.35	49.30	50.81	39.22	45.01	62.52	56.11	53.05
	MFAR* (Li et al., 2024)	41.20	70.00	58.50	54.20	49.00	69.60	71.70	58.20	**40.90**	**62.80**	**68.30**	**51.20**	43.70	67.47	**66.17**	54.53
	MoR	52.19	**74.65**	59.92	**62.24**	**58.19**	**78.34**	**75.01**	**67.14**	36.41	60.01	63.48	46.92	**48.93**	**71.00**	66.14	**58.77**
Ablation	$\text{MoR}_{\text{w/o R}}$	44.21	68.87	56.50	55.28	34.33	62.55	67.55	47.40	31.59	53.48	60.74	41.81	31.07	57.04	57.73	43.03
	$\text{MoR}_{\text{w/o RT}}$	34.04	53.41	45.16	42.85	51.81	73.54	74.17	61.68	28.95	46.12	49.54	36.56	36.39	56.73	55.73	45.53
	$\text{MoR}_{\text{w/o RS}}$	43.05	69.36	57.38	54.69	31.05	51.84	50.56	40.64	22.27	38.45	39.21	29.41	28.95	51.28	48.02	38.98

Table 1: Comparing different retrieval methods with our proposed MoR and its ablations on Amazon, MAG, and Prime datasets. The best and runner-up results are in **bold** and underlined. Overall, MoR achieves the best performance. Note that MFAR* denotes the best model variant proposed in (Li et al., 2024)

4.2 Overall Retrieval Performance

We compare MoR with other baselines on three TG-KBs in Table 1. Generally, hybrid methods, AvaTAR, KAR, MFAR, and our MoR, achieve better performance than purely textual or structural methods owing to their ability to integrate both structural and textual knowledge. Among all baselines, our proposed MoR achieves the overall best performance with a substantial margin on average, with the first ranking on MAG and the second ranking on Amazon/Prime datasets. This demonstrates the effectiveness of our proposed mixture of structural and textual knowledge retrieval. Textual retrieval performs better on Amazon than on MAG, suggesting that Amazon queries rely more on textual knowledge. In contrast, its weaker performance on MAG is due to MAG's lower textual richness and stronger structural signals. This disparity aligns with the distribution analysis in (Wu et al., 2024b) and supports our hypothesis that queries in different TG-KB datasets require varying desires of textual and structural knowledge. Meanwhile, structural retrieval methods such as conventional knowledge graph-based ones perform poorly because they are designed for graphs with minimal textual information compared to TG-KBs. Different from Amazon and MAG, all existing methods without supervised tuning (e.g., Ada-002) exhibit significantly lower performance on Prime. This is due to the extreme domain expertise required in biology, where word-count-based, pre-trained textual similarity-based, and even more powerful LLMs are all not directly applicable here. Through fine-tuning, MFAR and our proposed MoR generally achieve better performance, demonstrating the necessity of domain-specific knowledge for answering queries in knowledge-intensive domains.

4.3 Ablation Study

After verifying the superiority of MoR, we conduct ablation studies to assess its different components, including module and feature ablation.

4.3.1 Module Ablation

To assess the contribution of each module in MoR, namely, Text Matching-based Retrieval, Neighborhood-Fetching-based Structural Retrieval, and Reranker, we conduct a series of ablation experiments. First, we remove the Reranker, resulting in the variant $\text{MoR}_{\text{w/o R}}$. On top of that, we further separately eliminate Text Retrieval and Structural Retrieval, yielding $\text{MoR}_{\text{w/o RT}}$ and $\text{MoR}_{\text{w/o RS}}$, respectively. As shown in Table 1, the complete MoR framework consistently achieves the highest performance across all datasets, demonstrating the synergistic effect of the Textual Retriever, Structural Retriever, and Reranker. After removing Reranker, $\text{MoR}_{\text{w/o R}}$ exhibits a consistent performance drop across all datasets and evaluation metrics. This underscores the importance of the Reranker in refining retrieval by suppressing noisy candidates from the intermediate reasoning stage. Eliminating Text Retrieval, i.e., $\text{MoR}_{\text{w/o RT}}$, leads to a notable performance drop on Amazon but an unexpected improvement on MAG. This suggests that while textual knowledge benefits Amazon, it introduces misleading hard negatives that compromise the ranking method (e.g., Ada-002) for MAG. Conversely, removing Structural Retrieval, $\text{MoR}_{\text{w/o RS}}$, results in a slight performance decrease on MAG , reinforcing the importance of structural knowledge in MAG-related queries. These results underscore the Reranker's crucial role in adaptively harmonizing, balancing, and selecting knowledge from both structural and textual retrieval experts.

Dataset	TF	SF	TI	H@1	H@5	R@20	MRR
MAG	✔	✘	✘	48.96	73.02	72.44	59.79
	✘	✔	✘	18.79	41.91	52.85	29.84
	✘	✘	✔	18.16	41.53	52.78	29.31
	✔	✔	✘	58.04	77.14	74.42	66.75
	✔	✘	✔	58.16	77.59	74.96	66.85
	✘	✔	✔	17.93	38.01	46.79	27.48
	✔	✔	✔	**58.19**	**78.34**	**75.01**	**67.14**
Amazon	✔	✘	✘	51.21	74.05	59.79	61.27
	✘	✔	✘	8.09	24.48	25.62	16.94
	✘	✘	✔	5.84	16.62	12.94	11.57
	✔	✔	✘	50.91	73.38	59.58	61.15
	✔	✘	✔	51.09	73.56	59.61	61.14
	✘	✔	✔	8.09	24.48	25.62	16.94
	✔	✔	✔	**52.19**	**74.65**	**59.92**	**62.24**

Table 2: Ablation study investigating the importance of three features, Textual Fingerprint (**TF**), Structural Fingerprint (**SF**), and Traversal Identifier (**TI**), of the traversal trajectories used in our Structure-aware Reranker.

4.3.2 Feature Ablation

The above ablation study highlights the crucial role of Structure-aware Reranker in adaptively integrating structural and textual knowledge. To further analyze the contributions of its three key features, **Textual Fingerprint (TF)**, **Structural Fingerprint (SF)**, and **Traversal Identifier (TI)** defined in Section 3.3, we conduct a feature ablation analysis and report retrieval performance across different feature configurations in Table 2. Overall, using three features together yields the best performance on both MAG and Amazon, highlighting their synergistic effect. Individually, TF contributes the most and outperforms SF and TI on both datasets. The reason is that based on the definition in Section 3.3, TF directly captures the relevance between the query and the retrieved nodes along the trajectory, whereas SF and TI primarily characterize the structural patterns and retrieval types, serving more as complementary factors. Therefore, equipping TF with these complementary factors (i.e., SF or TI) yields around 10% additional gains on MAG. This is because SF and TI help the reranker selectively emphasize the relevance scores given by TF for certain nodes along the path. However, this boost is not observed on Amazon. We hypothesize that the textual knowledge needed there is predominantly derived from the final node on each path, making the structural cues provided by SF and TI less beneficial and even prone to overfitting. A deeper analysis to further justify this hypothesis is in Section 4.4. Overall, these findings underscore the varying importance of structural features in ranking across datasets.

Feature	MAG			Amazon		
	H@1	R@20	MRR	H@1	R@20	MRR
Last Node	49.91	73.49	59.92	50.36	59.62	61.05
Trajectory	**58.19**	**75.01**	**67.14**	**52.19**	**59.92**	**62.24**

Table 3: Comparing reranking performance using last node in the retrieved trajectory and the whole trajectory.

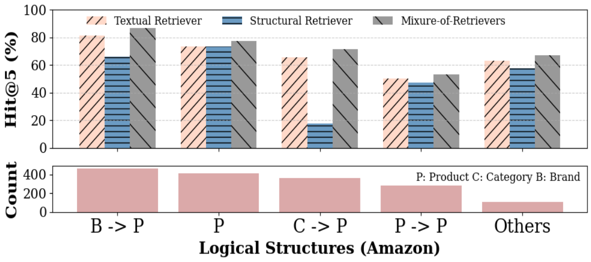

Figure 3: Imbalance number of queries and performance of different retrievers across different logic patterns.

4.4 Further Analysis

This section understands MoR's behavior by examining three questions, each of which enriches our insight into MoR's functionality and offers novel perspectives inspiring future research.

Does structure signals affect reranking? To assess the impact of trajectory information on the Reranker's decision-making, we introduce a node-based Reranker that constructs trajectory features using only TF/SF/TI of the last node. In Table 3, the path-based Reranker outperforms the node-based variant, especially on MAG. This highlights the critical role of trajectory features and structural knowledge in enhancing reranking accuracy.

How does MoR perform on different query structures? Figure 3 shows the average performance of MoR on each query group categorized by their logical structures (Wu et al., 2024b). "Others" refer to queries with unfounded logical structures. MoR consistently outperforms both structural and textual retrievers across different logical structures. Among all queries, MoR performs the worst on "P $\rightarrow$ P" queries due to the ambiguity of the repeated product entities. While such entity ambiguity is a well-known challenge, it uniquely arises from the multi-step reasoning desired for retrieving structural knowledge. The "Others" group includes reasoning paths not covered in (Wu et al., 2024b), and their high performance highlights the utility of diverse planning strategies for the same query. Lastly, the skewed query distribution and retrieval performance across planning patterns reflect the varying nature of real-world planning needs. We hope these insights inspire research on data-centric reasoning designs and error control of planning.

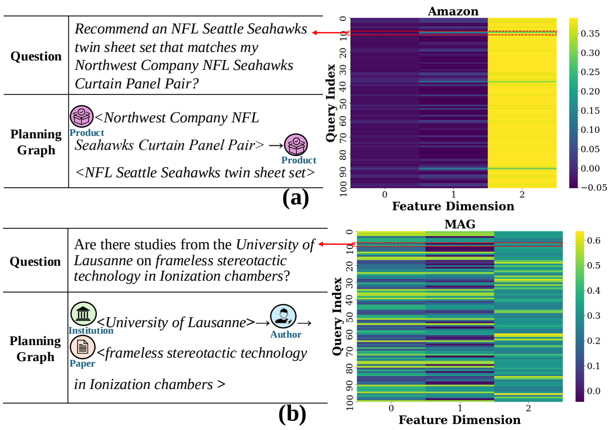

Figure 4: Saliency map visualization of query attention over three entities along the retrieved paths

Does MoR indeed adaptively leverage the trajectory knowledge? To understand how our proposed reranker prioritizes candidates in the Top-K results, we visualize the saliency map by computing the gradient of ranking scores with respect to the textual fingerprint (TF) of three nodes along the retrieved path, which quantifies their importance for answering a given query. Figure 4 illustrates this by analyzing trajectories for 100 ground-truth candidates across 100 queries on the Amazon and MAG datasets. Each dimension corresponds to a traversed node, with the final one representing the candidate itself. While the saliency score is concentrated in the last dimension for Amazon, MAG exhibits a more evenly distributed saliency pattern, where multiple nodes along the path contribute significantly to ranking score computation. This suggests that structural knowledge is more critical for answering queries in MAG, aligning with the previously observed lower performance of purely textual retrieval on MAG (Table 1). Further case studies explain why the reranker attends different nodes for different queries. In Figure 4(a), the reranker favors the last two dimensions as the rich textual restriction (i.e., "Northwest Company..." and "NFL Seattle...") aids in identifying the correct node at the corresponding reasoning step, as discussed in Section 3.2. The correct nodes, having higher similarity scores with the query, help guide the retrieval process toward the ground truth. Conversely, in Figure 4(b), since the first node ("University of Lausanne") helps narrow the search space and the last node ("frameless...") further filter candidates, both nodes have high saliency scores. Overall, our findings demonstrate that the reranker dynamically adapts its reliance on structural and textual knowledge depending on the dataset and query.

5 Related Work

Retrieval-augmented Generation (RAG): RAG enhances generative tasks by retrieving relevant information from external knowledge sources (He et al., 2024; Gao et al., 2023c) and has been widely used to improve question-answering (Liu et al., 2023). With LLMs, RAG has been used for mitigating hallucinations (Yao et al., 2023), enhancing interpretability (Gao et al., 2023a), and enabling dynamic knowledge updates (Wang et al., 2024). This work essentially leverages the idea of RAG to retrieve supporting entities from TG-KBs to contextualize answer generation. Depending on concrete types of knowledge being retrieved, existing retrievers can be categorized into structural and textual retrieval, which are reviewed next.

Textual and Structural Retrieval: Early textual retrieval models, such as TF-IDF and BM25 (Robertson et al., 2009), rely on lexical similarity and keyword matching (Chen et al., 2017; Yang et al., 2019; Mao et al., 2021). Modern approaches address this limitation by learning dense representations (Karpukhin et al., 2020). Beyond textual retrieval, structural retrieval leverages graph-based techniques to extract structured knowledge. Methods such as graph traversal (Wang et al., 2024; Jiang et al., 2023), community detection (Edge et al., 2024), and graph machine learning models, including graph neural networks (Yasunaga et al., 2021a; Mavromatis and Karypis, 2024), play a crucial role in structural retrieval. Our approach integrates the strengths of both textual and structural retrieval by infusing the mixture-of-expert philosophy into retrieval design.

Due to page limitation, a comprehensive version of the related work is attached in Appendix D.

6 Conclusion

In this work, we propose a mixture of structural and textual retrieval (MoR) to adaptively retrieve structural and textual knowledge based on query desire, which first utilizes a textual graph generator to generate the planning graph. Following the planning graph, we perform a mixed traversal and conduct organizing via a structure-aware reranker to obtain final candidates. Experiments demonstrate the advantages of our MoR in harmonizing the retrieval of both textual and structural knowledge with insightful discoveries, including balancing retrieval performance across queries with different patterns and query-adaptive knowledge desire for structural/textual knowledge.

7 Limitations

In this paper, we integrate a mixture of expert philosophy into retrieval design and propose a Mixture of structural-and-textual Retrieval (MoR) to adaptively retrieve textual and structural knowledge. The limitations of MoR can be categorized into two main areas:

Lack of Domain-Specific Knowledge: Our proposed MoR, similar to other baselines, does not exhibit significantly higher performance on PRIME than AMAZON and MAG. The reason is the lack of biomedical knowledge required to comprehend biomedical questions, extract key information, navigate relevant entities and relations, and rerank retrieved candidates. This suggests that current state-of-the-art retrieval models, even paired with LLMs' intelligence, still struggle to handle domain-specific knowledge effectively. Such limitations may extend to other specialized domains, such as finance and law. Future research could integrate domain-specific knowledge into retrieval.

Reranking at Every Traversal Layer: Our current MoR adaptively routes retrieved candidates into the Top-K positions at the final layer via reranking, effectively implementing a conventional Mixture of Experts (MoE) routing mechanism. Despite the state-of-the-art performance we have achieved in Table 1, this routing mechanism could also be applied to intermediate layers, where after each retrieval step, candidates are reranked, and only Top-K proceeds to the next round of traversal and retrieval. This enables every layer of mixed traversal to emulate the router design of the MoE.

References

Payal Chandak, Kexin Huang, and Marinka Zitnik. 2023. Building a knowledge graph to enable precision medicine. *Scientific Data*, 10(1):67.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Zhuo Chen, Yichi Zhang, Yin Fang, Yuxia Geng, Lingbing Guo, Xiang Chen, Qian Li, Wen Zhang, Jiaoyan Chen, Yushan Zhu, Jiaqi Li, Xiaoze Liu, Jeff Z. Pan, Ningyu Zhang, and Hua zeng Chen. 2024. Knowledge graphs meet multi-modal learning: A comprehensive survey. *ArXiv*, abs/2402.05391.

Julien Delile, Srayanta Mukherjee, Anton Van Pamel, and Leonid Zhukov. 2024. Graph-based retriever captures the long tail of biomedical knowledge. *ArXiv*, abs/2402.12352.

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*, abs/2404.16130.

Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, Haofen Wang, and Jiawei Zhang. 2023a. Chat-rec: Towards interactive and explainable llms-augmented recommender system. *arXiv preprint arXiv:2303.14524*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2023b. Retrieval-augmented generation for large language models: A survey. *ArXiv*, abs/2312.10997.

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

Chunxi Guo, Zhiliang Tian, Jintao Tang, Shasha Li, Zhihua Wen, Kaixuan Wang, and Ting Wang. 2023. Retrieval-augmented gpt-3.5-based text-to-sql framework with sample-aware prompting and dynamic revision chain. In *International Conference on Neural Information Processing*, pages 341–356. Springer.

Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A Rossi, Subhabrata Mukherjee, Xianfeng Tang, et al. 2024. Retrieval-augmented generation with graphs (graphrag). *arXiv preprint arXiv:2501.00309*.

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*.

Minghao Hu, Yuxing Peng, Zhen Huang, and Dongsheng Li. 2019. Retrieve, read, rerank: Towards end-to-end multi-document reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2285–2295.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*.

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 *Conference on Empirical Methods in Natural Language Processing*.

Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Suhang Wang, Yu Meng, and Jiawei Han. 2024a. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. In *Annual Meeting of the Association for Computational Linguistics*.

Bowen Jin, Yu Zhang, Sha Li, and Jiawei Han. 2024b. Bridging text data and graph data: Towards semantics and structure-aware knowledge discovery. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 1122–1125.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781.

Oleksandr Kolomiyets and Marie-Francine Moens. 2011. A survey on question answering technology from an information retrieval perspective. *Information Sciences*, 181(24):5412–5434.

Meng-Chieh Lee, Qi Zhu, Costas Mavromatis, Zhen Han, Soji Adeshina, Vassilis N. Ioannidis, Huzefa Rangwala, and Christos Faloutsos. 2024. Hybgrag: Hybrid retrieval-augmented generation on textual and relational knowledge bases. *ArXiv*, abs/2412.16311.

Millicent Li, Tongfei Chen, Benjamin Van Durme, and Patrick Xia. 2024. Multi-field adaptive retrieval. *arXiv preprint arXiv:2410.20056*.

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*, pages 2477–2486.

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

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2021. Generation-augmented retrieval for open-domain question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, volume 1, pages 4089–4100.

Costas Mavromatis and George Karypis. 2024. Gnn-rag: Graph neural retrieval for large language model reasoning. *ArXiv*, abs/2405.20139.

Mandar Mitra and BB Chaudhuri. 2000. Information retrieval from documents: A survey. *Information retrieval*, 2:141–163.

Dang Nguyen, Viet Dac Lai, Seunghyun Yoon, Ryan A Rossi, Handong Zhao, Ruiyi Zhang, Puneet Mathur, Nedim Lipka, Yu Wang, Trung Bui, et al. 2024. Dynasaur: Large language agents beyond predefined actions. *arXiv preprint arXiv:2411.01747*.

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

Junhong Shen, Neil Tenenholtz, James Hall, David Alvarez-Melis, and Nicoló Fusi. 2024. Tag-llm: Repurposing general-purpose llms for specialized domains. *ArXiv*, abs/2402.05140.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Sai Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, Heung yeung Shum, and Jian Guo. 2023a. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *International Conference on Learning Representations*.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Heung-Yeung Shum, and Jian Guo. 2023b. Think-on-graph: Deep and responsible reasoning of large language model with knowledge graph. *arXiv preprint arXiv:2307.07697*.

Dhaval Taunk, Lakshya Khanna, Pavan Kandru, Vasudeva Varma, Charu Sharma, and Makarand Tapaswi. 2023. Grapeqa: Graph augmentation and pruning to enhance question-answering. *Companion Proceedings of the ACM Web Conference 2023*.

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.

MK Vijaymeena and K Kavitha. 2016. A survey on similarity measures in text mining. *Machine Learning and Applications: An International Journal*, 3(2):19–28.

Yu Wang, Nedim Lipka, Ryan A Rossi, Alexa Siu, Ruiyi Zhang, and Tyler Derr. 2024. Knowledge graph prompting for multi-document question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19206–19214.

Shirley Wu, Shiyu Zhao, Qian Huang, Kexin Huang, Michihiro Yasunaga, Kaidi Cao, Vassilis N Ioannidis, Karthik Subbian, Jure Leskovec, and James Zou. 2024a. Avatar: Optimizing llm agents for tool-assisted knowledge retrieval. *arXiv preprint arXiv:2406.11200*.

Shirley Wu, Shiyu Zhao, Michihiro Yasunaga, Kexin Huang, Kaidi Cao, Qian Huang, Vassilis N Ioannidis, Karthik Subbian, James Zou, and Jure Leskovec. 2024b. Stark: Benchmarking llm retrieval on textual and relational knowledge bases. *arXiv preprint arXiv:2404.13207*.

Yu Xia, Junda Wu, Sungchul Kim, Tong Yu, Ryan A Rossi, Haoliang Wang, and Julian McAuley. 2024. Knowledge-aware query expansion with large language models for textual and relational retrieval. *arXiv preprint arXiv:2410.13765*.

Mayi Xu, Yongqi Li, Ke Sun, and Tieyun Qian. 2024. Adaption-of-thought: Learning question difficulty improves large language models for reasoning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 5468–5495, Miami, Florida, USA. Association for Computational Linguistics.

Shangzi Xue, Zhenya Huang, Xin Lin, Jiayu Liu, Longhu Qin, Tianhuang Su, Haifeng Liu, and Qi Liu. 2024. Enhancing the completeness of rationales for multi-step question answering. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 2753–2763.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with bertserini. *arXiv preprint arXiv:1902.01718*.

Jia-Yu Yao, Kun-Peng Ning, Zhen-Hui Liu, Mu-Nan Ning, and Li Yuan. 2023. Llm lies: Hallucinations are not bugs, but features as adversarial examples. *arXiv preprint arXiv:2310.01469*.

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

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021b. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. *arXiv preprint arXiv:2104.06378*.

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. 2018. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pages 5708–5717. PMLR.

Huimin Zeng, Zhenrui Yue, Qian Jiang, and Dong Wang. 2024. Federated recommendation via hybrid retrieval augmented generation. *2024 IEEE International Conference on Big Data (BigData)*, pages 8078–8087.

Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D. Manning, and Jure Leskovec. 2022. Greaselm: Graph reasoning enhanced language models for question answering. *ArXiv*, abs/2201.08860.

Yanqiao Zhu, Yuanqi Du, Yinkai Wang, Yichen Xu, Jieyu Zhang, Qiang Liu, and Shu Wu. 2022. A survey on deep graph generation: Methods and applications. In *Learning on Graphs Conference*, pages 47–1. PMLR.

A Summary of Notations

Table 4: Notations and the corresponding descriptions.

Notations	Definitions or Descriptions
$\mathcal{B}$	Text-rich Graph Knowledge Base (TG-KB)
$\mathcal{V}, \mathcal{E}, \mathcal{D}$	Set of Nodes, Categories and Documents of TG-KB
$\mathcal{D}_v, \mathcal{E}_v$	Document and Category of Node v
$Q \in \mathcal{Q}$	Query Q from Query set $\mathcal{Q}$
$\mathcal{Q}^{\text{Struct}}, \mathcal{Q}^{\text{Text}}$	Query targeted by structural and textual retrieval
$G = \{\mathcal{P}_i\}_{i=1}^{\lvert G \rvert}$	Planning Graph consisting of multiple reasoning paths
$\mathcal{P}_i = (p_{i1} \rightarrow \ldots \rightarrow p_{iL_i})$	Reasoning path consisting of L_i sequential entities
$\mathcal{E}_{p_{ij}}, \mathcal{T}_{p_{ij}}$	Textual category and restriction of path entity p_{ij}
$\widetilde{\mathcal{C}}$	Retrieved candidates after reasoning module.
$\widetilde{\mathcal{C}}_i^l = \widetilde{\mathcal{C}}_i^{l,\text{Struct}} \cup \widetilde{\mathcal{C}}_i^{l,\text{Text}}$	Retrieved candidates at l^{th} layer for i^{th} path including structurally retrieved one and textually retrieved one.
$\mathcal{C}$	Final retrieved candidates after organizing module.
$P_{\mathbb{Q}\times\mathbb{G}}$	Joint distribution of query and planning graph.
$\mathcal{N}_v$	Neighborhood of entity v
$\mathcal{I}_{p_{il}}$	Traversal Identifier of Structural and Textual Retrieval
$P_{\boldsymbol{\Theta}_1}$	Planning module with its parameters $\boldsymbol{\Theta}_1$
$P_{\boldsymbol{\Theta}_2}$	Reasoning module with its parameters $\boldsymbol{\Theta}_2$
$P_{\boldsymbol{\Theta}_3}$	Organizing module with its parameters $\boldsymbol{\Theta}_3$

Dataset	# Entities	# Text Tokens	# Relations	Avg. Degree
AMAZON	1,035,542	592,067,882	9,443,802	18.2
MAG	1,872,968	212,602,571	39,802,116	43.5
PRIME	129,375	31,844,769	8,100,498	125.2

Table 5: Statistics of text-rich graph knowledge bases in STaRK benchmark (Wu et al., 2024b).

B Experimental Details

B.1 Datasets

To evaluate the effectiveness of our proposed framework, we conduct experiments using three Text-rich Graph Knowledge Bases (TG-KBs) from STaRK (Wu et al., 2024b). These TG-KBs cover a wide range of domains, including product reviews (Amazon), academic papers (MAG), and biomedical knowledge (Prime). Each TG-KB comprises a textual graph and an associated corpus, with the corpus containing documents linked to the nodes in the graph. Queries are meticulously crafted for each TG-KB and encompass varying levels of complexity, which desire different levels of textual and structural knowledge to answer.

Amazon: a dataset provides a realistic simulation of product search and recommendation. Its textual graph consists of four categories of nodes: *product*, *category*, *color*, and *brand*. Nodes are interconnected through relations such as *has_brand* and *has_category*. Textual documents encapsulate properties of corresponding nodes, such as product descriptions and customer reviews.

MAG: a comprehensive resource for academic paper retrieval. In the textual graph, *papers* can be connected to other nodes, such as *field_of_study* via the *paper_has_topic_field_of_study* relation and *institution* through a combination of relations like *author_affiliated_with_institution* and *author_writes_paper*. Each *paper* document includes the title, abstract, and metadata, such as the publication date and venue, providing rich contextual knowledge for retrieval and analysis.

Prime: a highly domain-specific dataset. It focuses on medical inquiries and is sourced from the PrimeKG knowledge graph (Chandak et al., 2023), which comprises ten entity types and eighteen relation types, offering multiple target node categories, such as *disease*, *gene/protein*, and *drug*. The associated documents are aggregated from various databases, providing a rich and diverse source of medical knowledge.

Detailed dataset statistics are in Table 5.

B.2 Implementation Details

Prompt for Planning Graph Generation: For planning graph generation in Section 3.1, we follow previous works (Luo et al., 2023; Wu et al., 2024b) to linearize the planning process by decomposing the planning graph into sequential reasoning paths, which can be generated by LLMs via next token prediction. Given the lack of ground-truth planning graphs for training, we prompt LLMs to synthesize these ground-truth planning graphs due to their superior reasoning capability. The prompt for generating the ground-truth planning graph is:

Prompt 1: Planning Graph Generation

System Message: You are a planning graph finder agent. Your role is to:
1. Identify the underlying **meta-path** from a given question, which consists of the **entity types** at each reasoning step.
2. Extract the **content restriction** for each **entity type** based on the question. If there is no restriction for an entity type, leave its value empty.
You will be provided with a predefined **Entity Type List**. Only use the entity types from this list when constructing the meta-path and restrictions. Your response must be concise and strictly adhere to the specified **output format**.

Entity Type List: Provide the entity type list.
Demonstrations: Examples for in-context learning.
Output Fromat: *Metapath: "", Restriction: {}.*

Trajectory Collection: As mentioned in Section 3.3, our reranker reorders the intermediate retrieved candidates based on their trajectory. To achieve this, we collect three key features: **Textual Fingerprint (TF)**, **Structural Fingerprint (SF)**, and **Traversal Identifier (TI)**.

Textual Fingerprint (TF): We record the BM25 similarity scores between the query and the traversed nodes computed. Since empirical observations indicate that the length of reasoning paths is typically less than three, we fix the textual fingerprint to the length of three by padding additional 0 similarity scores for those reasoning paths whose length is less than three, allowing for batch-wise training. Additionally, we append the initial semantic ranking score of the candidate computed using cosine similarity coupled with Ada-002 embedding to the end of three BM25-based similarity scores to complement the lexical perspective. This vector is then passed through a linear layer to be transformed into an embedding of size 128. Note that this initial ranking score is also used to select the intermediate retrieved candidates used for reranking.

Structural Fingerprint (SF): We concatenate the categories of all nodes in the corresponding reasoning path as a text sequence. If the reasoning path is shorter than three nodes, we prepend the sequence with "padding" tokens to ensure a fixed length. The structural fingerprint is then processed using a transformer model, which converts the sequence into an embedding of size 768, followed by a linear layer that projects it down to size 128.

Traversal Identifier (TI): We track whether each node is retrieved via textual matching or structural traversal and encoding them with distinct values by initializing a learnable embedding matrix mapping each traversal identifier encoding to a 3x128-dimensional embedding vector.

After obtaining all the above three trajectory features, we concatenate their obtained vectors into a unified vector (128 + 128 + 128x3 = 640) and apply a fully connected layer to transform the combined representation into a reranking score for each candidate. This score determines the final ranking.

C Additional Results

C.1 Query Pattern Analysis

Figure 5 illustrates the analysis of query patterns in the MAG dataset. With richer relational information, queries in this dataset form a wider variety of patterns, including longer and more diverse structures. Similar to the Amazon dataset, we observe a general trend where the performance of MoR declines as the query count decreases across different patterns. Beyond this overall trend, certain query patterns in the MAG dataset stand out, such as "P $\rightarrow$ A $\rightarrow$ P" (Product-to-Author-to-Product) and

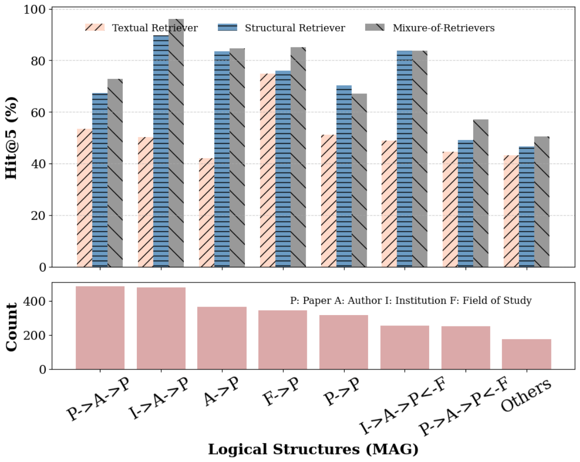

Figure 5: Imbalance number of queries and performance of different retrievers across different logic patterns.

"P → P" (Paper-to-Paper). Despite their relatively high occurrence, MoR still performs worse on these patterns. This is similar to low performance on the "Product → Product" pattern observed in the Amazon dataset, where repeated entity types appear within a single query. Such repetition causes the textual retriever to shift focus from the target to the repeated entities, leading to lower performance.

D Comprehensive Related Work

D.1 Retrieval-augmented Generation (RAG)

With the unprecedented success of recent LLMs in approaching human-level intelligence, retrieving relevant knowledge to support downstream generation has become increasingly crucial. Retrieval-augmented generation enhances generative tasks by integrating relevant information from external knowledge sources (He et al., 2024; Gao et al., 2023c; Han et al., 2024) and has been widely adopted to improve question-answering (Liu et al., 2023). In the context of LLMs, RAG has been utilized to mitigate hallucinations (Yao et al., 2023), enhance interpretability (Gao et al., 2023a), and enable dynamic knowledge updates (Wang et al., 2024). This work leverages RAG to retrieve supporting entities from TG-KBs, providing contextual grounding for answer generation. Depending on the type of knowledge retrieved, existing retrievers can be classified into structural and textual retrieval approaches, which are reviewed next.

D.2 Textual and Structural Retrieval

Since real-world knowledge is commonly stored in both textual and structural formats (Kolomiyets and Moens, 2011), such as indexed texts and knowledge graphs, each requires a retrieval method tailored to its unique representation. Textual retriever retrieves knowledge based on its similarity to the given query and can be categorized into: lexical methods (e.g., TF-IDF and BM25 (Robertson et al., 2009)) and semantic methods (e.g., DPR and Contriever (Karpukhin et al., 2020; Izacard et al., 2022)). Despite their broad applicability, the predefined linguistic rules and embedding-based semantics may struggle to capture the structural knowledge stored in graph-structured knowledge bases such as knowledge graphs and text-rich networks. To address this challenge, structural retrieval has been proposed by using graph analysis techniques (e.g., graph traversal (Wang et al., 2024; Jiang et al., 2023; Zhang et al., 2022; Edge et al., 2024)) and graph machine learning models (e.g., graph neural networks (Yasunaga et al., 2021a; Mavromatis and Karypis, 2024)). Early methods extract local subgraphs of seeding nodes (Yasunaga et al., 2021a; Taunk et al., 2023) or pre-define paths approaching answers (e.g., shortest paths (Luo et al., 2023; Delile et al., 2024)). To avoid exponentially expanding neighbors in the local subgraphs and break the rigid logic routined by pre-defined paths, recent advancements integrated LLMs to dynamically adjust graph traversal (Sun et al., 2023a; Wang et al., 2024; Jin et al., 2024a). While promising, frequently invoking LLMs introduces prohibitive resource overhead. Despite the above advancements in both textual and structural retrieval, they are often applied independently and fail to mutually reinforce each other. This motivates the recent research trend of developing hybrid retrieval, which is reviewed next.

D.3 Hybrid Retrieval

Recently, several works have explored hybrid knowledge retrieval from TG-KBs. One approach (Xia et al., 2024; Li et al., 2024) aggregate documents from neighboring nodes, with Xia et al. (2024) applying relational filtering to remove irrelevant neighbors and Li et al. (2024) weighting neighbors based on field importance. Another approach (Lee et al., 2024) uses LLMs to choose either structural or textual retrieval. In contrast, our proposed MoR fully leverages the graph structure and rich texts by integrating textual matching and graph traversal into a unified framework, enabling a more seamless and interpretable interaction between structural and textual knowledge