Empowering GraphRAG with Knowledge Filtering and Integration

Kai Guo[1], **Harry Shomer**[2], **Shenglai Zeng**[1], **Haoyu Han**[1], **Yu Wang**[3], **Jiliang Tang**[1]
[1]Michigan State University [2] University of Texas at Arlington [3] University of Oregon
{guokai1, zengshe1, hanhaoy1, tangjili}@msu.edu,
harry.shomer@uta.edu,
yuwang@uoregon.edu

Abstract

In recent years, large language models (LLMs) have revolutionized the field of natural language processing. However, they often suffer from knowledge gaps and hallucinations. Graph retrieval-augmented generation (GraphRAG) enhances LLM reasoning by integrating structured knowledge from external graphs. However, we identify two key challenges that plague GraphRAG: (1) Retrieving noisy and irrelevant information can degrade performance and (2) Excessive reliance on external knowledge suppresses the model's intrinsic reasoning. To address these issues, we propose GraphRAG-FI (Filtering & Integration), consisting of GraphRAG-Filtering and GraphRAG-Integration. GraphRAG-Filtering employs a two-stage filtering mechanism to refine retrieved information. GraphRAG-Integration employs a logits-based selection strategy to balance external knowledge from GraphRAG with the LLM's intrinsic reasoning, reducing over-reliance on retrievals. Experiments on knowledge graph QA tasks demonstrate that GraphRAG-FI significantly improves reasoning performance across multiple backbone models, establishing a more reliable and effective GraphRAG framework.

1 Introduction

Large language models (LLMs) have achieved remarkable success in NLP tasks, particularly in tasks that require complex reasoning (Havrilla et al.; Wu et al., 2023; Hao et al., 2023; Zeng et al., 2025; Liu et al., 2025). However, despite their strengths, LLMs are prone to hallucinations, resulting in incorrect or poor reasoning (Ji et al., 2023; Huang et al., 2024; Sriramanan et al., 2025). GraphRAG techniques have emerged as a promising solution to this problem (Han et al., 2024; Zhang et al., 2025; He et al., 2025; Mavromatis and Karypis, 2024), by integrating relevant information from external graphs. Knowledge graphs, which store facts in the form of a graph, are commonly used for this problem. Specifically, relevant facts (i.e., triples) or paths are extracted from the knowledge graph and used to enrich the context of the LLMs with structured and reliable information (Luo et al., 2024; Li et al., 2025; Ma et al., 2024). This approach has shown ability to improve the reasoning capabilities and reduce the presence of hallucinations in LLMs (Sun et al.; Li et al., 2025; Dong et al., 2024).

To better assess the efficacy of GraphRAG, in Section 3 we conduct a preliminary study comparing its performance with an LLM-only model (i.e., LLM without GraphRAG). This comparison reveals both the advantages and limitations of GraphRAG. While GraphRAG improved reasoning accuracy by correcting some LLM errors, it also introduces some notable weaknesses. For example, incorporating external knowledge will sometimes cause questions that were originally answered correctly by the LLM to be misclassified. This highlights the dangers of retrieving irrelevant information. Furthermore, excessive retrieval compounds this issue by introducing both noise and redundant information, thus further hindering the reasoning process.

Meanwhile, we find that *LLM-only and GraphRAG can complement one another*. Specifically, GraphRAG can enhance reasoning for those questions LLMs lack knowledge of; while excessive reliance on external information may cause the model to overlook internally known correct answers. These findings highlight two key limitations of existing GraphRAG methods. **First**, GraphRAG is highly susceptible to retrieving irrelevant or misleading information. **Second**, GraphRAG struggles to balance external retrieval with the LLM's internal knowledge, often missing parts of the answer that the LLM-only model can provide using its own knowledge. These challenges differ from those encountered in standard RAG settings (Wang

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 25439–25453
November 4-9, 2025

et al., 2023b; Chang et al., 2024); they arise specifically within the graph context. Our focus is the GraphRAG setting, which involves structured retrieval formats such as paths, triples, and subgraphs—highly compact yet densely informative knowledge representations that are inherently more difficult for LLMs to interpret compared to unstructured text (Luo et al., 2024; Li et al., 2025). This leads to several unique limitations: 1) Structured Noise: Retrieved graph data is often multi-hop and entangled. Noise manifests not as irrelevant facts, but as misleading or incomplete reasoning chains that hinder accurate inference. 2) Multi-Answer Complexity: Many KGQA queries have multiple correct answers, requiring the retrieved context to support all of them. This increases the difficulty of filtering, as naive top-K selection tends to overlook less dominant yet still valid answers.

To address these complexities, we introduce a novel framework tailored to resolve the challenges inherent in graph-based retrieval. First, we aim to enhance the retrieval quality to better avoid retrieving irrelevant information. Second, we integrate GraphRAG with the intrinsic reasoning ability of the LLM, thereby leveraging complementary knowledge sources. In particular, to mitigate the issue of retrieving irrelevant information, we introduce a two-stage filtering process. Furthermore, to mitigate GraphRAG from over-relying on retrieved information while underutilizing the LLM's inherent reasoning ability, we introduce a logits-based selection mechanism that dynamically integrates LLMs' standalone answers with GraphRAG's outputs. This approach ensures that the final response effectively balances external knowledge with the model's internal reasoning. The main contributions of our work are summarized as follows:

- We identify two key challenges in GraphRAG: **(1)** It is susceptible to errors by retrieving irrelevant or misleading information. **(2)** It overemphasizes the externally retrieved knowledge, at the expense of the intrinsic reasoning capabilities of LLMs.
- We introduce a novel approach that enhances GraphRAG by incorporating a two-stage filtering mechanism to refine the retrieved knowledge and dynamically integrate this knowledge with a LLMs' standalone reasoning capabilities.
- Extensive experiments on knowledge graph QA demonstrate the effectiveness of our method across multiple backbone models.

2 Related work

GraphRAG. GraphRAG aims to address hallucinations and outdated knowledge in LLMs by incorporating additional information retrieved from external knowledge bases (Sun et al.; Li et al., 2025; Dong et al., 2024). G-Retriever (He et al., 2025) identifies relevant nodes and edges for a given query based on cosine similarity, and then constructs a subgraph to aid in the generation process. Similarly, RoG (Luo et al., 2024) introduces a planning-retrieval-reasoning framework, where it retrieves reasoning paths guided by a planning module and performs reasoning using these paths. On the other hand, GNN-RAG (Mavromatis and Karypis, 2024) leverages Graph Neural Networks (GNNs) (Kipf and Welling, 2016) to process the intricate graph structures within knowledge graphs, enabling effective retrieval.

Filter Methods. ChunkRAG (Singh et al., 2024) tries to improve RAG systems by assessing and filtering retrieved data at the chunk level, with each "chunk" representing a concise and coherent segment of a document. Zeng et al. (2024b) introduce Rep-PCA, which employs representation classifiers for knowledge filtering. RoK (Wang et al., 2024) refines the reasoning paths within the subgraph by computing the average PageRank score for each path. Similarly, He et al. (2024) use PageRank to identify the most relevant entities. More details are presented in Appendix A.1.

3 Preliminary studies

To evaluate the effectiveness of GraphRAG, we compare the performance with and without retrieved external knowledge. Furthermore, we analyze the attention scores of the LLM to assess its ability to discern both the relevance and importance of the retrieved information. Lastly, we evaluate the performance of internal knowledge filtering.

3.1 Experimental settings

In this section, we aim to study the importance of retrieving external information when using GraphRAG for knowledge graph QA. To do so, we report the QA performance when using: LLM *with* GraphRAG and LLM *w/o* GraphRAG (i.e., LLM-only). For GraphRAG, we use RoG (Luo et al., 2024) and GNN-RAG (Mavromatis and Karypis,

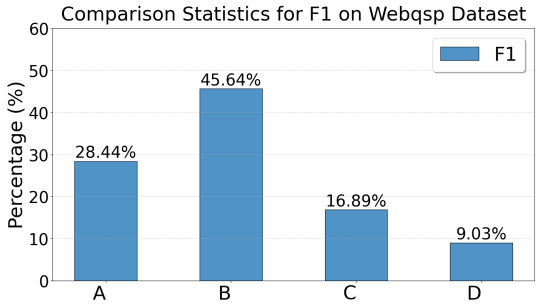

Figure 1: Category A includes cases where both GraphRAG and the LLM-only model are correct. Category B covers instances where GraphRAG outperforms the LLM-only model, while Category C includes cases where the LLM-only model performs better than GraphRAG. Category D represents cases where both models fail.

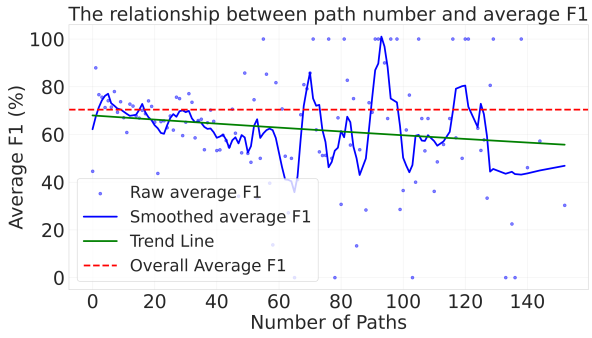

Figure 2: The relationship between path number and average F1

2024). For the LLM-only experiments, we use the fine-tuned LLaMA 2-7B model, which is the same LLM used by RoG. The experiments are conducted on two common datasets the WebQSP (Yih et al., 2016) and CWQ (Talmor and Berant, 2018) datasets. In this study, we mainly use the F1 score to evaluate the performance.

3.2 The Impact of GraphRAG

To understanding the effectiveness of GraphRAG, we compare prediction outcomes between LLM *with* GraphRAG and LLM *w/o* GraphRAG (i.e., LLM-only). We categorize the results into four groups based on F1 scores, as shown in the Figure 1. Category A includes cases where both GraphRAG and the LLM-only model provide correct answers. Category B consists of instances where GraphRAG produces a more accurate answer than the LLM-only model. Category C includes cases where the LLM-only model outperforms GraphRAG. Finally, Category D represents instances where both GraphRAG and the LLM-only model fail to generate the correct answer. Figure 1 illustrates the key observations from our experiments. While GraphRAG enhances certain predictions, it also introduces notable challenges that require further investigation.

Positive Impact of GraphRAG GraphRAG can enhance the LLM's reasoning capabilities by correcting errors that the standalone model would typically commit. Notably, in the category **B**, 45.64% of previously incorrect responses were successfully rectified with the integration of GraphRAG. This highlights the advantage of leveraging structured knowledge graphs to boost LLM performance.

Limited Impact of GraphRAG Category **A** contains those answers where both GraphRAG and LLM-only are correct. This show that GraphRAG can sometimes preserve the performance of a LLM when the LLM already possesses the correct knowledge. Conversely, category **D**, representing 9.03% of cases, corresponds to those cases where GraphRAG fails to enhance the model's accuracy. For this category, neither the standalone LLM nor GraphRAG are able to provide the correct answer. This pattern implies that GraphRAG does not always access or incorporate sufficiently informative or relevant knowledge.

Negative Impact of GraphRAG A notable drawback of GraphRAG is that will occasionally degrade the performance of a standalone LLM. That is, it will sometimes lead to wrong predictions for queries that the standalone LLM originally got right. These instances are represented by category **C** and accounts for 16.89% of samples when evaluating via the F1 score. In these cases, GraphRAG misleads the model rather than improving it. This suggests that some of the retrieved information may be incorrect, noisy, or irrelevant, ultimately leading to poorer predictions. Therefore, in some cases, LLMs without GraphRAG outperform those with GraphRAG, because existing works have shown that LLMs tend to over-rely on external information (Ren et al., 2023; Tan et al., 2024; Wang et al., 2023a; Ni et al., 2024; Zeng et al., 2024a). When retrieval is insufficient or the quality of retrieved knowledge is low, this reliance can degrade generation quality.

3.3 The Impact of the Number of Retrieved Paths

Due to the structure of knowledge graphs, nodes with high degrees and numerous relational edges have a greater likelihood of yielding a large number

of retrieved paths. In this subsection, we study the impact of the number of retrieved paths on performance. Figure 2 illustrates the relationship between the number of retrieved paths and the model's performance. To present the information more clearly and statistically, we include **interval statistics** in Table 10, located in Appendix A.10. Interestingly, as indicated by the smoothed line (blue), incorporating a moderate amount of retrieved information enhances performance. However, increasing the number of retrieved paths ultimately leads to a decline in performance. This trend (green line) suggests that retrieving too much information will introduce noise, making it harder for the model to use the correct and relevant knowledge for the task. This phenomenon thus highlights an important insight – **more information does not necessarily indicate better performance**. Instead, an overabundance of retrieved data can overwhelm the model with irrelevant details. This observation underscores the necessity for effective filtering mechanisms that can prioritize high-quality, relevant knowledge while discarding extraneous or misleading information.

3.4 Attention Reflects the Importance of Retrieved Information

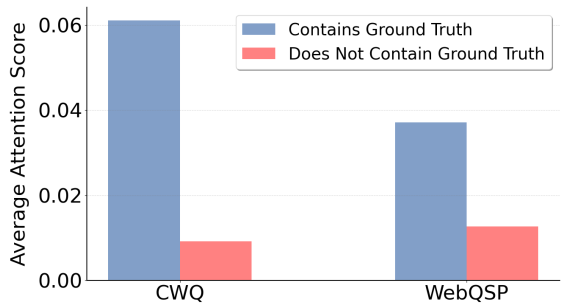

Figure 3: Attention Scores for Retrieved Information With/Without Ground Truth

In this subsection, we analyze the ability of the LLM to distinguish the importance of retrieved external knowledge. The attention scores of a LLM can provide a natural indicator of the relevance and significance of the retrieved knowledge (Yang et al., 2024; Ben-Artzy and Schwartz, 2024). The attention scores, derived from the model's internal mechanisms, effectively capture which pieces of information are most influential in reaching the final decision. Inspired by recent work (Chuang et al., 2023; Halawi et al., 2023), which suggests that attention scores in the middle layers are more effective. We examine the attention scores of the (middle + 2)-th layer in the LLM for each retrieved path. We obtain the attention scores for all retrieved paths and categorize them into two groups: (1) paths that contain the ground truth and (2) paths that do not. We then compute the average attention score for each group and present the results in Figure 3. As demonstrated in Figure 3, there is a clear alignment between the attention scores and the ground truth labels, suggesting that these scores can be used to assess the relevance of retrieved information.

This observation inspires a key insight: The attention scores highlight the most significant retrieved information, suggesting their potential use in filtering out noisy or irrelevant knowledge. Since retrieved information with lower attention scores contribute minimally to the final output, they can be pruned to streamline retrieval and enhance overall performance.

3.5 Internal Knowledge Filtering

Large language models (LLMs) generate responses that may contain both correct and incorrect information. To assess the reliability of these responses, we analyze the associated logits, which represent the model's confidence in its predictions. Typically, higher confidence correlates with correctness (Ma et al., 2025; Virk et al., 2024). Leveraging this property, we implement "Internal Knowledge Filtering", which uses the logits to help refine the answer selection.The logits of answer can be directly obtained from the LLM's output. Formally, let A_L denote the sets of answer candidates from the LLM model. Furthermore, let it's corresponding logits after softmax function be given by $\ell_L(a)$. The filtering step is given by the following:

$$A_L^{\text{filtered}} = \{a \in A_L \mid \ell_L(a) \geq \tau_L\}, \quad (1)$$

where $\tau_L = 1$, a represents a specific candidate answer and is an element of A_L. This allows us to filter out the responses that the LLM has low-confidence in. The experimental results are shown in Table 1. We can clearly see that that leveraging logits to filter out low-confidence responses has a large positive effect on performance. In this way, we can reconsider intrinsic knowledge and apply this approach to GraphRAG to better balance internal and external knowledge base on logits. Furthermore, we provide an experimental analysis comparing our method with naive answer merging without using logit threshold $\tau_L = 1$ in Appendix A.5.

Methods	WebQSP		CWQ	
	Hit	F1	Hit	F1
LLM	66.15	49.97	40.27	34.17
LLM with Logits	84.17	76.74	61.83	58.19

Table 1: Impact of logits on LLM performance

3.6 Discussions

In this subsection, we summarize the key findings and discussions from our preliminary study. The performance issues observed in GraphRAG primarily arise from two key factors. (1) **Noisy or Irrelevant Retrieval:** Some retrieved paths contain irrelevant or misleading information. This negatively impacts the model's ability to properly answer the query. Furthermore, this noise can introduce conflicting or unnecessary information that hinders the decision-making process rather than improving it. (2) **Lack of Consideration for LLM's Own Knowledge:** GraphRAG does not always take into account the inherent reasoning ability of the LLM itself. In some cases, the retrieved information overrides the LLM's correct predictions, leading to performance degradation rather than enhancement. A more adaptive approach is needed to balance external knowledge retrieval with the model's internal knowledge.

4 Method

Based on our analysis, we propose a new framework to address the identified challenges, guided by two key insights: **(1)** *Filtering retrieved information*: Given the tendency of GraphRAG to retrieve irrelevant or incorrect retrieved information, it is essential to refine the retrieved knowledge. **(2)** *Properly leveraging the LLMs standalone capabilities*: The LLM itself can often correctly answer some questions. It's thus necessary to effectively integrate and use the inherent reasoning ability of LLMs along with GraphRAG.

An overview of our framework **GraphRAG-FI** is given in Figure 4. It consists of two core components: **GraphRAG-Filtering** and **GraphRAG-Integration**. GraphRAG-Filtering first refines the retrieved information by removing irrelevant or misleading knowledge. GraphRAG-Integration module balances the retrieved knowledge with the LLM's inherent reasoning ability, thereby mitigating the overuse of retrieved information that can negatively impact performance. In the following subsections, we will introduce each component of our framework in detail.

4.1 GraphRAG-Filtering

Let $P = \{p_1, p_2, \ldots, p_N\}$ denote the set of N retrieved paths or triplets, where each path p_i is assigned an attention score $\hat{a}_i$. Then we design filtering via the following two stages.

Stage 1: Coarse Filtering using Attention: In the first stage, we perform a coarse filtering by retaining only those paths whose attention scores exceeds a threshold τ. This is given formally by:

$$P_{\text{coarse}} = \{p_i \in P \mid \hat{a}_i \geq \tau\}. \tag{2}$$

where $\hat{a}_i$ denotes the attention score of path p_i. The detailed procedure for computing attention scores is provided in Appendix A.6.

Stage 2: Fine Filtering via LLMs: After the initial coarse filtering, which significantly reduces the number of candidate paths, we perform a more precise evaluation with a LLM on the remaining subset. This two-stage filtering approach not only enhances the quality of the retrieved paths but also greatly reduces the overall cost by limiting the use of the LLM to only those paths deemed promising in the first stage. Let $f(p)$ represent the evaluation score provided by the LLM for a path p, and let τ' be the corresponding threshold. The final set of filtered paths is then given by:

$$P_{\text{final}} = \{p \in P_{\text{coarse}} \mid f(p) \geq \tau'\}, \tag{3}$$

where P_{coarse} is the set of paths that passed the coarse filtering stage, τ' is not predefined but is determined by the LLM itself. Specifically, we prompt the LLM to achieve this goal. The prompt is presented in Fig 6 in Appendix.

Prompt Construction for Question Answering: After the two filtering stages, we incorporate the selected paths and query into the prompt to further guide the model's reasoning. The prompt contains the following two types of retrieved paths:

- **High Priority Paths**: These are the final filtered paths given by P_{final}, which are considered the most reliable.
- **Additional Paths**: We also consider the the remaining paths included by the coarse filter but removed via the fine filter, $P_{\text{coarse}} - P_{\text{final}}$. We conjecture that while they may not be as important as those paths in P_{final}, they can still offer some useful supplementary context.

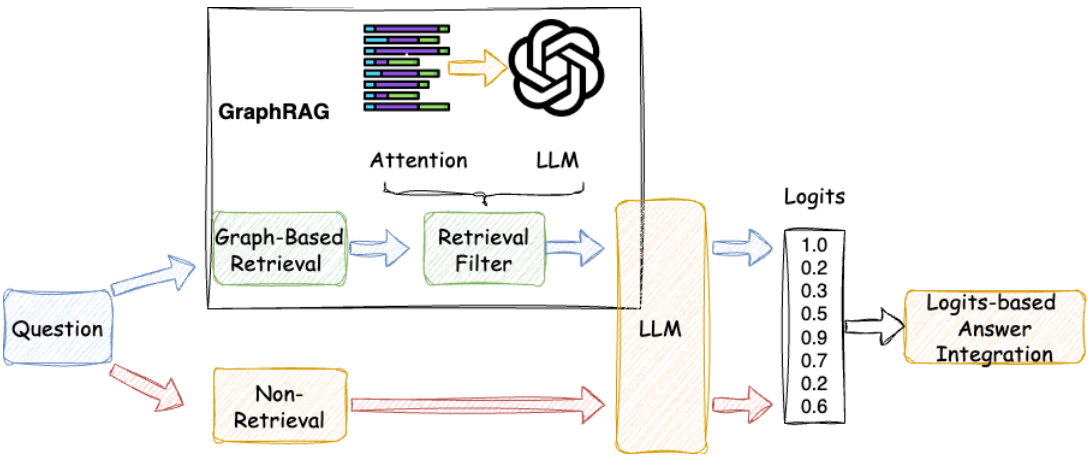

Figure 4: An overview of the GraphRAG-FI framework.

The new prompt is then constructed by first inserting a header for the high-priority paths, followed by each path on a separate line. The same process is repeated for the additional paths. By structuring the prompt in this way, we are able to clearly delineate the paths by their priority. This ensures that the most critical information (P_{final}) is emphasized and processed first, while still incorporating the supplementary context from the additional paths. An example prompt is given in Fig 5 in Appendix A.3. In graph-based retrieval settings, limitations such as structured noise and multi-answer complexity are commonly encountered. To enhance recall and ensure comprehensive answer coverage, it is crucial to incorporate additional paths beyond the top-ranked evidences. Accordingly, we conduct an experiment to assess the impact of these additional paths, as detailed in Appendix A.11.

4.2 Integration with LLMs' Internal Knowledge

As noted in Section 3.2, in addition to ensuring we only retrieve high-quality information, we also want to retain internal knowledge of the LLMs. As such, we want to also integrate the capabilities of just the LLM into our framework. However, a challenge is knowing when to defer to which method. *When do we trust the answers given by GraphRAG and when the standalone LLM*? Furthermore, *how do we fuse the answers given by both methods*?

To achieve this goal, we need a method to determine which answers produced by both LLM-only and GraphRAG are actually relevant. In Section 3.5, we found that the LLM's logits can provide a useful tool to refine the potential answers. That is, focusing only on those answers that are given a higher confidence is helpful. This naturally provides us with an easy way to focus on just the high-quality information. For both GraphRAG and the LLM-only model, we filter the answers based on their logits, ensuring that only high-confidence responses are retained. After this logits-based filtering, the refined answers from both sources are combined to produce the final answer, thereby enhancing robustness and accuracy.

Formally, let A_G and A_L denote the sets of answer candidates from GraphRAG and the LLM-only model, respectively. We further use a to indicate a single candidate answer in either set. Furthermore, let their corresponding logits after the softmax function be given by $\ell_G(a)$ and $\ell_L(a)$. The filtering step is given by the following:

$$A_G^{\text{filtered}} = \{a \in A_G \mid \ell_G(a) \geq \tau_G\}, \quad (4)$$

$$A_L^{\text{filtered}} = \{a \in A_L \mid \ell_L(a) \geq \tau_L\}, \quad (5)$$

where τ_G and τ_L are predefined thresholds, τ_L is set to 1. Subsequently, the final answer is determined by combining the filtered sets:

$$A_{\text{final}} = \text{Combine}\left(A_G^{\text{filtered}}, A_L^{\text{filtered}}\right), \quad (6)$$

where $\text{Combine}(\cdot)$ denotes the function that integrates the filtered answers into the final reliable output.

5 Experiment

In our experiments, we seek to address the following research questions: **RQ1:** How effective is the proposed method when applied to state-of-the-art GraphRAG retrievers in the knowledge graph QA task? **RQ2:** How does the proposed method compare to other filtering approaches? **RQ3:** How does

Type	Methods	WebQSP		CWQ	
		Hit	F1	Hit	F1
LLMs	Alpaca-7B(Taori et al., 2023)	51.8	-	27.4	-
	LLaMA2-Chat-7B(Touvron et al., 2023)	64.4	-	34.6	-
	ChatGPT	66.8	-	39.9	-
	ChatGPT+CoT	75.6	-	48.9	-
LLMs+KGs	ROG	86.73	70.75	61.91	54.95
	ROG + Similarity	85.50	69.38	61.62	54.38
	ROG + PageRank	85.44	69.60	61.34	54.41
	ROG + GraphRAG-Filtering	87.40	73.41	63.86	**57.25**
	ROG + GraphRAG-FI	**89.25**	**73.86**	**64.82**	55.12
	GNN-RAG	90.11	73.25	69.10	60.55
	GNN-RAG + Similarity	89.68	72.17	68.50	60.26
	GNN-RAG + PageRank	89.18	71.92	66.75	58.73
	GNN-RAG + GraphRAG-Filtering	91.28	74.74	69.70	**60.96**
	GNN-RAG + GraphRAG-FI	**91.89**	**75.98**	**71.12**	60.34
	SubgraphRAG	76.90	64.65	53.87	50.43
	SubgraphRAG + Similarity	72.72	59.98	52.05	48.27
	SubgraphRAG + PageRank	61.79	50.65	46.75	43.23
	SubgraphRAG + GraphRAG-Filtering	81.01	**68.40**	58.82	**54.71**
	SubgraphRAG + GraphRAG-FI	**81.08**	68.28	**58.96**	52.52

Table 2: Performance comparison with different baselines on the two KGQA datasets.

the performance change when more noisy information is introduced? and **RQ4:** What is the impact of the two modules on performance?

Our code is available at https://github.com/KaiGuo20/GraphRAG-FI.

5.1 Experiment Settings

Datasets. To assess the effectiveness of our method, we evaluate it on two widely recognized KGQA benchmark datasets: WebQSP (Yih et al., 2016) and CWQ (Talmor and Berant, 2018). WebQSP contains 4,737 natural language questions that require reasoning over paths of up to two hops. In contrast, CWQ includes 34,699 more complex questions that necessitate multi-hop reasoning over up to four hops. Both datasets are built upon Freebase , which consists of around 88 million entities, 20 thousand relations, and 126 million triples. Further details on the datasets are provided in Appendix A.2.

Retriever Backbones. Our framework adopts three existing retrieval methods as its backbone: path-based retrieval (ROG (Luo et al., 2024)), GNN-based retrieval (GNN-RAG (Mavromatis and Karypis, 2024)), and subgraph-based retrieval (SubgraphRAG (Li et al., 2025)). Path-based retrieval extracts relevant paths using heuristics or shortest-path algorithms, while GNN-based retrieval leverages a Graph Neural Network to learn and retrieve informative paths. In contrast, subgraph-based retrieval retrieves relevant subgraphs and encodes them as **triples**, enabling fine-grained relational reasoning. Therefore, both path-based and GNN-based methods generate paths as input for the LLM. Lastly, subgraph-based methods decompose the subgraph into triples—i.e., (h, r, t), which are then used as input to the LLM. Considering these allows us to test the framework on diverse retrieval methods.

Filter Baselines. The most commonly used filtering methods for RAG are similarity-based approaches used in (Gao et al., 2025). Similarity-based methods evaluate the relevance of retrieved information by measuring feature similarity. For retrieval over graphs, PageRank-based filtering is widely adopted (Wang et al., 2024). PageRank-based filtering leverages the graph structure to rank nodes based on their connectivity and importance. These methods provide a baseline filtering mechanism for refining the retrieved results. To evaluate the effectiveness of our method, we compare it with several filtering and reranking approaches within the traditional RAG framework, including FILCO (Wang et al., 2023b), Main-RAG (Chang et al., 2024), and BGE-Reranker (Chen et al., 2024). The results are presented in Table 8 in Appendix.

Implementation and Evaluation Metrics. We use LLaMA2-Chat-7B from ROG as the LLM backbone, which is instruction-finetuned on the training split of WebQSP and CWQ, as well as Freebase, for three epochs. For the similarity-based filter, we utilize SentenceTransformer ('all-MiniLM-L6-v2') to generate representations for retrieval. We evaluate our retrieval methods using both Hit Rate (Hit) and F1-score (F1). Hit Rate measures the proportion of relevant items successfully retrieved, reflecting retrieval effectiveness. F1-score balances precision and recall, providing a comprehensive assessment of retrieval quality. These metrics ensure a robust evaluation of retrieval performance. We adjust the thresholds τ and τ_G within the ranges [top 40, top 50] and [0.4, 0.5], respectively.

5.2 Main Results

In this section, we evaluate the performance of our method with various retrievers and compare it against baseline filter models.

RQ1: KGQA Performance Comparison. In this subsection, we apply our method to different retrievers, including the path-based retriever, GNN-based retriever, and subgraph-based retriever. The results presented in Table 2 demonstrate that our method consistently improves all retrievers, achieving an average improvement of 3.81% in Hit and 2.35% in F1 over ROG, 2.46% in Hit and 1.7% in F1 over GNN-RAG, and significant gains of 7.47% in Hit and 4.88% in F1 over SubgraphRAG across two datasets. These results demonstrate that our approach is effective across different retrieval paradigms, reinforcing its adaptability to various retrieval strategies in QA tasks. We also evaluate our method on additional models to verify its effectiveness, with detailed results provided in Appendix A.8. In addition, we provide some **case studies** in Appendix A.9.

RQ2: Comparison with other filter methods. We compare our method against other filtering baselines, with the results presented in Table 2. Our approach consistently outperforms competing methods across both datasets and retriever types. Specifically, for ROG, our method can achieve an average improvement of 4.78% in Hit and 3.95% in F1 compared to similarity-based filtering on both datasets. Furthermore, compared to the PageRank-based filtering method, our approach yields an average increase of 5.03% in Hit and 3.70% in F1 across both datasets. These results highlight the superiority of our method in enhancing retrieval effectiveness and overall performance.

Methods	WebQSP		CWQ	
	Hit	F1	Hit	F1
ROG-original	86.73	70.75	61.91	54.95
ROG*	85.87	68.81	60.49	53.72
ROG* + GraphRAG-Filtering	86.61	73.01	61.91	55.67

Table 3: Performance when adding more noise

5.3 Robustness to Noise

In this subsection, we evaluate robustness of different methods to noise. To evaluate the noise resistance of the backbone model and our filter method, we use GPT to generate 30 additional noise paths that contain both irrelevant and incorrect information. This information is then incorporated into the retrieved context. We then analyze the impact of this noise on performance. The experimental results presented in Table 3, ROG* represents the cases where noise is introduced. As the noise level increases, the Hit score decreases by 2.29%, and the F1 score drops by 2.23% on the CWQ dataset, highlighting the model's sensitivity to noise. However, when applying our method, we observe a 2.23% improvement in Hit and a 3.63% improvement in F1 over ROG* on CWQ. These results demonstrate the effectiveness of our approach in mitigating the negative impact of noisy retrieval.

5.4 Ablation Study

We conduct an ablation study to analyze the effectiveness of the filtering module and integrating module in GraphRAG-FI. From the results in Table 4, we can see that GraphRAG-Filtering is useful for the ROG retriever, as it improves both the F1 and Hit scores. For example, GraphRAG-Filtering increases the F1 score by 4.19% and the Hit score by 3.15% on CWQ dataset. We also see a boost in performance for GraphRAG-Integration, with a 1.60% and 2.62% increase in F1 and Hit score, respectively, on WebQSP. These results demonstrate the effectiveness of our two components. In addition, we conduct a **parameter study**, with the results presented in Appendix A.4.

Methods	WebQSP		CWQ	
	Hit	F1	Hit	F1
ROG-original	86.73	70.75	61.91	54.95
ROG + GraphRAG-Filtering	87.40	73.41	63.86	**57.25**
ROG + GraphRAG-Integration	89.00	71.88	64.25	55.19
ROG + GraphRAG-FI	**89.25**	**73.86**	**64.82**	55.12

Table 4: Ablation study.

6 Conclusion

In this work, we propose GraphRAG-FI (Filtering & Integration), an enhanced GraphRAG framework that addresses key challenges in graph retrieval-augmented generation. By incorporating GraphRAG-Filtering, which utilizes a two-stage filtering mechanism to refine retrieved information, and GraphRAG-Integration, which employs a logits-based selection strategy to balance retrieval and intrinsic reasoning, our approach mitigates the impact of noisy retrievals and excessive dependence on external knowledge.

Limitations

In this work, we identify two key challenges in GraphRAG: (1) it is prone to errors due to the retrieval of irrelevant or misleading information, and (2) it places excessive emphasis on externally retrieved knowledge, which can diminish the intrinsic reasoning capabilities of LLMs. Future research will first explore a broader range of large language models to evaluate their effectiveness within GraphRAG. Additionally, further investigation into diverse filtering methods could enhance the refinement of retrieved information and reduce noise. More sophisticated fusion strategies may also be explored to dynamically balance external knowledge with the intrinsic reasoning of LLMs, enabling more effective information integration.

Acknowledgements

Kai Guo, Shenglai Zeng, Haoyu Han and Jiliang Tang are supported by the National Science Foundation (NSF) under grant numbers CNS2321416, IIS2212032, IIS2212144, IIS 2504089, DUE2234015, CNS2246050, DRL2405483 and IOS2035472, the Michigan Department of Agriculture and Rural Development, US Dept of Commerce, Amazon Faculty Award, Meta, NVIDIA, Microsoft and SNAP.

References

Amit Ben-Artzy and Roy Schwartz. 2024. Attend first, consolidate later: On the importance of attention in different llm layers. *arXiv preprint arXiv:2409.03621*.

Chia-Yuan Chang, Zhimeng Jiang, Vineeth Rakesh, Menghai Pan, Chin-Chia Michael Yeh, Guanchu Wang, Mingzhi Hu, Zhichao Xu, Yan Zheng, Mahashweta Das, et al. 2024. Main-rag: Multi-agent filtering retrieval-augmented generation. *arXiv preprint arXiv:2501.00332*.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*.

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James Glass, and Pengcheng He. 2023. Dola: Decoding by contrasting layers improves factuality in large language models. *arXiv preprint arXiv:2309.03883*.

Jialin Dong, Bahare Fatemi, Bryan Perozzi, Lin F Yang, and Anton Tsitsulin. 2024. Don't forget to connect! improving rag with graph-based reranking. *arXiv preprint arXiv:2405.18414*.

Zengyi Gao, Yukun Cao, Hairu Wang, Ao Ke, Yuan Feng, Xike Xie, and S Kevin Zhou. 2025. Frag: A flexible modular framework for retrieval-augmented generation based on knowledge graphs. *arXiv preprint arXiv:2501.09957*.

Danny Halawi, Jean-Stanislas Denain, and Jacob Steinhardt. 2023. Overthinking the truth: Understanding how language models process false demonstrations. *arXiv preprint arXiv:2307.09476*.

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*.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173.

Alexander Havrilla, Sharath Chandra Raparthy, Christoforos Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, and Roberta Raileanu. Glore: When, where, and how to improve llm reasoning via global and local refinements. In *Forty-first International Conference on Machine Learning*.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2025. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *Advances in Neural Information Processing Systems*, 37:132876–132907.

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*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2024. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*.

Ziwei Ji, Tiezheng Yu, Yan Xu, Nayeon Lee, Etsuko Ishii, and Pascale Fung. 2023. Towards mitigating llm hallucination via self reflection. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1827–1843.

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

Mufei Li, Siqi Miao, and Pan Li. 2025. Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation. In *International Conference on Learning Representations*.

Zhining Liu, Rana Ali Amjad, Ravinarayana Adkathimar, Tianxin Wei, and Hanghang Tong. 2025. Self-elicit: Your language model secretly knows where is the relevant evidence. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 9153–9173. Association for Computational Linguistics.

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

Huan Ma, Jingdong Chen, Guangyu Wang, and Changqing Zhang. 2025. Estimating llm uncertainty with logits. *arXiv preprint arXiv:2502.00290*.

Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, and Jian Guo. 2024. Think-on-graph 2.0: Deep and interpretable large language model reasoning with knowledge graph-guided retrieval. *arXiv e-prints*, pages arXiv–2407.

Costas Mavromatis and George Karypis. 2024. Gnn-rag: Graph neural retrieval for large language model reasoning. *arXiv preprint arXiv:2405.20139*.

Shiyu Ni, Keping Bi, Jiafeng Guo, and Xueqi Cheng. 2024. When do llms need retrieval augmentation? mitigating llms' overconfidence helps retrieval augmentation. *arXiv preprint arXiv:2402.11457*.

Ruiyang Ren, Yuhao Wang, Yingqi Qu, Wayne Xin Zhao, Jing Liu, Hao Tian, Hua Wu, Ji-Rong Wen, and Haifeng Wang. 2023. Investigating the factual knowledge boundary of large language models with retrieval augmentation. *arXiv preprint arXiv:2307.11019*.

Ishneet Sukhvinder Singh, Ritvik Aggarwal, Ibrahim Allahverdiyev, Muhammad Taha, Aslihan Akalin, Kevin Zhu, and Sean O'Brien. 2024. Chunkrag: Novel llm-chunk filtering method for rag systems. *arXiv preprint arXiv:2410.19572*.

Gaurang Sriramanan, Siddhant Bharti, Vinu Sankar Sadasivan, Shoumik Saha, Priyatham Kattakinda, and Soheil Feizi. 2025. Llm-check: Investigating detection of hallucinations in large language models. *Advances in Neural Information Processing Systems*, 37:34188–34216.

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

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. *arXiv preprint arXiv:1803.06643*.

Hexiang Tan, Fei Sun, Wanli Yang, Yuanzhuo Wang, Qi Cao, and Xueqi Cheng. 2024. Blinded by generated contexts: How language models merge generated and retrieved contexts for open-domain qa? *arXiv preprint arXiv:2401.11911*.

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

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

Yuvraj Virk, Premkumar Devanbu, and Toufique Ahmed. 2024. Enhancing trust in llm-generated code summaries with calibrated confidence scores. *arXiv preprint arXiv:2404.19318*.

Yile Wang, Peng Li, Maosong Sun, and Yang Liu. 2023a. Self-knowledge guided retrieval augmentation for large language models. *arXiv preprint arXiv:2310.05002*.

Yuqi Wang, Boran Jiang, Yi Luo, Dawei He, Peng Cheng, and Liangcai Gao. 2024. Reasoning on efficient knowledge paths: Knowledge graph guides large language model for domain question answering. *arXiv preprint arXiv:2404.10384*.

Zhiruo Wang, Jun Araki, Zhengbao Jiang, Md Rizwan Parvez, and Graham Neubig. 2023b. Learning to filter context for retrieval-augmented generation. *arXiv preprint arXiv:2311.08377*.

Xiaoqian Wu, Yong-Lu Li, Jianhua Sun, and Cewu Lu. 2023. Symbol-llm: leverage language models for symbolic system in visual human activity reasoning. *Advances in Neural Information Processing Systems*, 36:29680–29691.

Lijie Yang, Zhihao Zhang, Zhuofu Chen, Zikun Li, and Zhihao Jia. 2024. Tidaldecode: Fast and accurate llm decoding with position persistent sparse attention. *arXiv preprint arXiv:2410.05076*.

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

Shenglai Zeng, Pengfei He, Kai Guo, Tianqi Zheng, Hanqing Lu, Yue Xing, and Hui Liu. 2025. Towards context-robust llms: A gated representation fine-tuning approach. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 10262–10276. Association for Computational Linguistics.

Shenglai Zeng, Jiankun Zhang, Pengfei He, Yue Xing, Yiding Liu, Han Xu, Jie Ren, Shuaiqiang Wang, Dawei Yin, Yi Chang, et al. 2024a. The good and the bad: Exploring privacy issues in retrieval-augmented generation (rag). *arXiv preprint arXiv:2402.16893*.

Shenglai Zeng, Jiankun Zhang, Bingheng Li, Yuping Lin, Tianqi Zheng, Dante Everaert, Hanqing Lu, Hui Liu, Yue Xing, Monica Xiao Cheng, et al. 2024b. Towards knowledge checking in retrieval-augmented generation: A representation perspective. *arXiv preprint arXiv:2411.14572*.

Qinggang Zhang, Shengyuan Chen, Yuanchen Bei, Zheng Yuan, Huachi Zhou, Zijin Hong, Junnan Dong, Hao Chen, Yi Chang, and Xiao Huang. 2025. A survey of graph retrieval-augmented generation for customized large language models. *arXiv preprint arXiv:2501.13958*.

A Appendix

A.1 Related work

GraphRAG. GraphRAG aims to address hallucinations and outdated knowledge in LLMs by incorporating additional information retrieved from external knowledge bases (Sun et al.; Li et al., 2025; Dong et al., 2024). G-Retriever (He et al., 2025) identifies relevant nodes and edges for a given query based on cosine similarity, and then constructs a subgraph to aid in the generation process. Similarly, RoG (Luo et al., 2024) introduces a planning-retrieval-reasoning framework, where it retrieves reasoning paths guided by a planning module and performs reasoning using these paths. On the other hand, GNN-RAG (Mavromatis and Karypis, 2024) leverages Graph Neural Networks (GNNs) (Kipf and Welling, 2016) to process the intricate graph structures within knowledge graphs, enabling effective retrieval. They also use retrieval augmentation techniques to enhance diversity. However, the effectiveness of these methods is heavily dependent on the quality of the retrieved information, and their performance significantly declines when the retrieved graph data is either noisy or unrelated to the query (He et al., 2025) .

Filter Methods. Filtering attempts to only keep those pieces of retrieved information that are relevant to the given query (Gao et al., 2025). ChunkRAG (Singh et al., 2024) tries to improve RAG systems by assessing and filtering retrieved data at the chunk level, with each "chunk" representing a concise and coherent segment of a document. This method first applies semantic chunking to partition documents into meaningful sections. It then leverages LLM-based relevance scoring to evaluate how well each chunk aligns with the user query. Zeng et al. (2024b) thoroughly investigate LLM representation behaviors in relation to RAG, uncovering distinct patterns between positive and negative samples in the representation space. This distinction enables representation-based methods to achieve significantly better performance for certain tasks. Building on these insights, they introduce Rep-PCA, which employs representation classifiers for knowledge filtering. RoK (Wang et al., 2024) refines the reasoning paths within the subgraph by computing the average PageRank score for each path. Similarly, He et al. (2024) use PageRank to identify the most relevant entities.

Datasets	#Train	#Test	Max #hop
WebQSP	2,826	1,628	2
CWQ	27,639	3,531	4

Table 5: Statistics of datasets.

A.2 Datasets

We utilize two benchmark KGQA datasets, WebQSP (Yih et al., 2016) and CWQ (Talmor and Berant, 2018), as proposed in previous studies. Following ROG, we maintain the same training and testing splits. The dataset statistics are provided in Table 5. Each query have one or multiple correct answers. Specifically, for the WebQSP dataset, there are 815 queries with a single answer and 813 queries with multiple answers, accounting for 49.94% of the total. For the CWQ dataset, 2676 queries have a single answer, while 855 queries have multiple answers, representing 24.21% of the total.

A.3 Prompt example

We provide an example prompt for question answering in Fig.5, and the prompt used for fine filtering is shown in Fig.6.

Prompts

Based on the reasoning paths, please answer the given question. Please keep the answer as simple as possible and return all the possible answers as a list.
Reasoning Paths:
High Priority Paths:
Northern Colorado Bears football → education.educational_institution.sports_teams → University of Northern Colorado
Additional Paths:
Northern Colorado Bears football → education.educational_institution.sports_teams → University of Northern Colorado
Greeley → location.location.containedby → United States of America
Greeley → location.location.containedby → Greeley Masonic Temple
Question: What educational institution has a football sports team named Northern Colorado Bears is in Greeley, Colorado?

Figure 5: An Example Prompt of Question Answering

A.4 Parameter study

We conducted an additional parameter study to evaluate the robustness of our method under different hyperparameter settings. We focus on two parameters: τ_G and τ. The performance (Hit and F1 scores) on the WebQSP dataset under varying settings is reported in Table 6: As shown, our model performs consistently well across different parameter settings, demonstrating that it is robust to moderate changes in both τ_G and τ.

Hit Score for Parameter Study		
	$\tau_G = 0.4$	$\tau_G = 0.5$
$\tau =$ top 40	89.07	89.06
$\tau =$ top 50	89.10	89.25
F1 Score for Parameter Study		
	$\tau_G = 0.4$	$\tau_G = 0.5$
$\tau =$ top 40	73.84	73.47
$\tau =$ top 50	73.73	73.86

Table 6: Performance Metrics for Parameter Study

A.5 Comparison with naively merging answers

To verify whether the performance gain comes from our proposed GraphRAG integration rather than simply increasing the number of candidate answers, we conducted an additional ablation where we naively merged all answers from the internal LLM and external retrieved results without any filtering. The results are shown in Table 7. Although the naive merging strategy collects more answers through over-coverage, it leads to lower precision. In contrast, our method achieves significantly higher F1 scores, indicating more accurate and faithful answer selection. This highlights the effectiveness of our fusion mechanism in balancing internal and external knowledge, filtering out noise, and enhancing the overall quality of predictions.

Method	**WebQSP**	**CWQ**
Naively merging	66.07	48.91
GraphRAG-FI	**73.86**	**55.12**

Table 7: F1 scores on WebQSP and CWQ datasets comparing naive merging and GraphRAG-FI.

A.6 Attention Score for Coarse Filtering.

To get attention scores for coarse filtering, we perform the following steps:

1. **Target Layer Selection:** We extract the attention weights from an intermediate layer of LLMs.

2. **Token Alignment Within Path:** For each path p_i, we identify its token span in the serialized prompt by locating the position of each path in the input string, and using the LLM tokenizer to obtain the corresponding token indices.

3. **Attention Score Computation:** We extract the attention weights from the *last decoding token* (usually corresponding to the generation of the first output token) to all tokens in path p_i, using the target attention layer. For each path, we compute an attention-based relevance score as:

$$\hat{a}_i = \frac{\max(\text{att}(p_i)) + \text{mean}(\text{att}(p_i))}{2} \quad (7)$$

where $\text{att}(p_i)$ refers to the attention weights from the decoding token to the token span of path p_i.

A.7 Comparison with more baselines

To evaluate the effectiveness of our method, we compare it with several filtering and reranking approaches within the traditional RAG framework, including FILCO (Wang et al., 2023b), Main-RAG (Chang et al., 2024), and BGE-Reranker (Chen et al., 2024). Specifically, FILCO is a mutual information-based filtering method, Main-RAG leverages prompt-based filtering, and BGE-Reranker utilizes a reranking model for candidate selection. The results, presented in Table 8, show that our method consistently outperforms these baselines across all metrics and datasets. This highlights the effectiveness of our framework, particularly in leveraging graph-structured knowledge.

Method	**WebQSP**		**CWQ**	
	Hit	F1	Hit	F1
Main-RAG	82.06	63.67	54.97	48.41
BGE-Reranker	85.93	69.23	61.56	54.95
FILCO	86.67	69.89	61.09	53.89
Our method	**89.25**	**73.86**	**64.82**	**55.12**

Table 8: Performance comparison between baselines and our method on WebQSP and CWQ datasets.

Prompt Template

```
messages = [
{"role": "system",
"content": "You are a reasoning assistant. Your task is to analyze numbered reasoning paths and
return only the indices of useful paths."},
{"role": "user",
"content": "Question:\n{question}\n\n
Numbered Reasoning Paths:\n{numbered_paths_text} \n\n
Giving question, return only the indices of useful paths as a comma-separated list"}
]
```

Figure 6: Prompt of Fine Filtering

A.8 Performance on other models

In addition to models like LLaMA2-Chat-7B, we also tested our method using Qwen2.5-7B and Llama3.1-70B for question answering. The results in Table 9 consistently show that our method outperforms the RoG across different model scales, demonstrating its effectiveness.

	Hit	F1
Qwen2.5-7B		
RoG	80.96	65.44
Our method	**82.50**	**66.40**
Llama3.1-70B		
RoG	84.58	71.20
Our method	**85.40**	**77.67**

Table 9: Results on WebQSP

A.9 Case study

We provide qualitative case studies to illustrate the effectiveness of our method. Below in Figure 7 are two representative examples from the WebQSP dataset: These examples demonstrate that our method filters out noisy or irrelevant candidates and successfully integrates internal and external knowledge to recover all correct answers with high precision.

A.10 Statistics of path count intervals

To make the information more clear and statistically, we provide interval statistics, including the average F1 scores within each interval and the p-values from t-tests, as shown in the Table 10. From the results, we observe an overall decreasing trend in F1 score as the number of paths increases. In 3 out of 5 cases, the p-values indicate a statistically significant decrease.

A.11 The importance of additional paths.

To demonstrate the necessity of the additional paths discussed, we conducted an ablation study by removing those paths and keeping only the top-ranked reasoning paths. As shown in Table 11, removing secondary but informative paths leads to a significant performance drop, confirming that these paths are crucial for ensuring complete reasoning coverage, recall and improving final prediction quality.

Example 1	
Question: Ground Truth:	*What countries does Greece share borders with?* Albania, Bulgaria, Republic of Macedonia, Turkey
Method	**Prediction**
LLM-only	Bulgaria, Turkey, Republic of Macedonia, Albania
LLM+GraphRAG	Turkey, Republic of Macedonia, Albania *(Misses Bulgaria)*
Our Method	Albania, Bulgaria, Turkey, Republic of Macedonia *(All correct)*
Example 2	
Question: Ground Truth:	*What colleges did Harper Lee attend?* University of Alabama, Huntingdon College, University of Oxford, University of Alabama School of Law
Method	**Prediction**
LLM-only	University of Alabama, Huntingdon College, University of Alabama School of Law, University of Tennessee, Cornell University, Shortridge High School, Butler University, University of Chicago, Carnegie Mellon University *(includes many irrelevant institutions)*
LLM+GraphRAG	Huntingdon College, University of Alabama School of Law, University of Oxford *(Misses University of Alabama)*
Our Method	Huntingdon College, University of Alabama, University of Alabama School of Law, University of Oxford *(All correct)*

Figure 7: Comparison of different methods on two example questions

Interval	Average F1 (%)	Comparison	P-value
0-10	74.69		
10-40	66.91	0-10 vs 10-40	0.0002
40-70	58.43	10-40 vs 40-70	0.0269
70-100	65.96	40-70 vs 70-100	0.1691
100-130	62.17	70-100 vs 100-130	0.5723
130-160	35.12	100-130 vs 130-160	0.0469

Table 10: Statistics Table of Path Count Intervals

Method	**WebQSP**		**CWQ**	
	Hit	F1	Hit	F1
Without additional paths	37.78	34.35	27.27	25.01
Our method	**89.25**	**73.86**	**64.82**	**55.12**

Table 11: Comparison of hit rates and F1 scores with and without additional paths on WebQSP and CWQ datasets.