B TECHNICAL REPORT

B.1 Experimental Settings

All experiments are running on Ubuntu 20.04.6 LTS (Intel(R) Xeon(R) Platinum 8358 CPU@2.60GHz Processor, 4 A100-80G, 400GB memory). The detailed experimental settings are as follows.

Hyperparameters. Table 1 presents the hyperparameters applied across the various GraphRAG instances in this study. The configurations are organized according to the specific combinations of subgraph-extraction (SBE or SAE) and path-retrieval (SBR or I/OSAR) modules used. Key parameters, including the maximum number of retained nodes or entities (max_ent), maximum edges (max_edge), and additional constraints such as filtering metrics (top_k), beam width (beam_width), and maximum path length (max_hop), are outlined for each group. This structured overview highlights the distinct configurations and the parameter tuning strategies employed to ensure consistent and fair comparisons across different instances.

Model Card. This study utilizes a diverse set of models categorized as EEMs and LLMs. Table 2 summarizes the models, providing detailed information about their size and sources.

- **EEMs:** The embedding-focused models include *all-MiniLM-L6-v2* [120], a compact embedding model with 22.7M parameters, and *bge-reranker-v2-m3* [15], a larger reranking model with 568M parameters. These models emphasize efficiency in retrieval and ranking tasks.
- **LLMs:** The large language models demonstrate significant diversity in scale and architecture, with parameters ranging from 7.62B to 72.7B.

Table 1: The Hyperparameters Used in Different Instances

Instance	Subgraph-Extraction Module	Path-Filtering Module
Group(I) SBE & SBR		
No.1 SBE-PPR+SBR-SPR	max_ent=1000	-
Group(II) SAE & I/OSAR		
No.2 SAE-EEMs+OSAR-EEMs	max_edge=64	top_k=32
No.3 SAE-EEMs+OSAR-LLMs	max_edge=64	top_k=32
No.4 SAE-LLMs+OSAR-EEMs	max_edge=64	top_k=32
No.5 SAE-LLMs+OSAR-LLMs	max_edge=64	top_k=32
No.6 SAE-EEMs+ISAR-EEMs	max_edge=64	beam_width=8, max_hop=4
No.7 SAE-EEMs+ISAR-LLMs	max_edge=64	beam_width=8, max_hop=4
No.8 SAE-LLMs+ISAR-EEMs	max_edge=64	beam_width=8, max_hop=4
No.9 SAE-LLMs+ISAR-LLMs	max_edge=64	beam_width=8, max_hop=4
Group(III) SAE & SBR		
No.10 SAE-EEMs+SBR-SPR	max_edge=64	-
No.11 SAE-LLMs+SBR-SPR	max_edge=64	-
Group(IV) SBE & I/OSAR		
No.12 SBE-PPR+OSAR-EEMs	max_ent=1000	top_k=32
No.13 SBE-PPR+OSAR-LLMs	max_ent=1000	top_k=32
Group(V) SBR & I/OSAR		
No.14 SBE-PPR+ISAR-EEMs	max_ent=1000	beam_width=8, max_hop=4
No.15 SBE-PPR+ISAR-LLMs	max_ent=1000	beam_width=8, max_hop=4

Table 2: Information About the Models Used in This Paper

Model Type	Model Name	Model Size
EEMs	all-MiniLM-L6-v2 [120]	22.7M params
	bge-reranker-v2-m3 [15]	568M params
LLMs	Qwen2-7B-Instruct [127]	7.62B params
	glm-4-9b-chat[33]	9.4B params
	Llama-3.3-70B-Instruct [111]	70.6B params
	Qwen2-72B-Instruct [127]	72.7B params

B.2 Datasets description

Freebase, as a comprehensive knowledge base, includes a wide range of domain-specific graphs (such as social relations, finance, etc.) along with their associated queries. The data is represented in a triple format, which forms the foundation for complex and comprehensive graph-based reasoning and question answering tasks for GraphRAG. We utilize four GraphRAG query datasets based on the Freebase within the GraphRAG community, WebQSP [131], CWQ [108], GrailQA [34], and WebQuestions [6]. The domain distribution of these datasets is summarized in Figure 1.

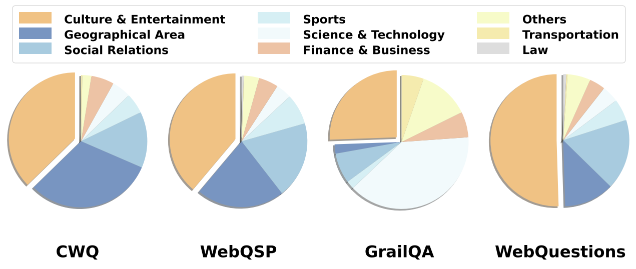


Figure 1: Domain distribution of four datasets

we conducted additional experiments on the MetaQA dataset, which is built on Wiki-Movies—a knowledge base focused on the movie domain—and can be viewed as a single-domain social network. Detailed statistics of Wiki-Movies are provided in Table 3. As MetaQA is independent of Freebase, it provides an additional evaluation to verify the transferability of the LEGO-GraphRAG framework. We process the datasets into json format files through a unified preprocessing step, which includes the question, central entities, answers and hop information corresponding to each query.

Table 3: Statistics of constructed knowledge graphs.

KG	#Entities	#Relations	#Triples
Freebase [8]	100176641	658641	298458255
Wiki-Movie [86]	43234	9	133582

To address the substantial scale of Freebase, we leverage well-established methodologies [42, 55, 106] to construct dataset-specific knowledge graphs (KGs). Our approach involves filtering triples based on domain-specific and relation-level constraints. Specifically, we exclude relations such as *type.object.type* and *type.object.name*, as well as those with prefixes like *common.* and *freebase.*. Additionally, relations containing keywords such as *sameAs* or *sameas* are removed to eliminate redundant or non-informative connections. To further refine the dataset, we discard triples associated with pre-defined domains, including *music.release* and *book.isbn*. This filtering process ensures that only meaningful and relevant triples are retained in the dataset-specific KGs. By significantly reducing noise and focusing on informative triples, the resulting graphs are more efficient for downstream GraphRAG tasks. Detailed preprocessing steps can be found in our code.

The selection of datasets in our benchmark is guided by the following criteria:

- **Coverage of Diverse Domains:** The datasets are sampled from knowledge graphs covering a wide range of domains. This ensures that our evaluation captures a broad spectrum of real-world topics.
- **Diversity in Query Types:** The selected datasets include queries with varying reasoning characteristics—such as the number of reasoning paths, the number of answer entities, and the number of query entities—covering both simple and complex reasoning tasks.
- **Consistency with Prior Work:** These datasets have been widely used in existing GraphRAG studies. In contrast to most prior works, which evaluate only one or two of them, we include all four datasets from Freebase and one dataset from Wiki-Movies to enable more comprehensive benchmarking.

B.3 Additional Experiment on MetaQA dataset

We conduct a new full-scale experiment on the *MetaQA* dataset, which is based on the *Wiki-Movies* knowledge graph. MetaQA features a different schema and a relatively smaller graph structure. This dataset can be viewed as a movie-related social network and helps validate the adaptability of our framework to different graph structures and domains.The corresponding results are shown in Figure 2. The results show that the relative performance of different GraphRAG instances on MetaQA aligns well with the trends observed in other datasets, further supporting the generality of our findings.

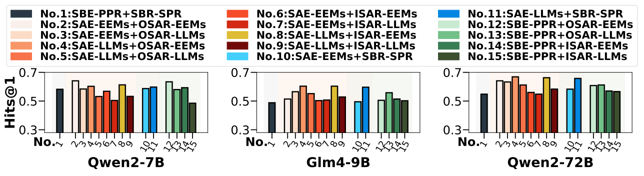


Figure 2: Results of LEGO-GraphRAG Instances on MetaQA Dataset

B.4 Different Evaluation Methods

In our main experiments, we adopt the widely accepted metric-based evaluation (Hits@1). We also evaluate the F1 performance of each Instance on four data sets based on FreeBase. As shown in Figure 3, the results of F1 score are similar to Hits@1 and do not affect the relative comparisons among different GraphRAG instances.

Additionally, we additionally conduct LLM-based evaluation (Qwen2-72B). As shown in Figure 4, we compare Hits@1 with Judge@1 (LLM-based evaluation). The trends of the two metrics are closely aligned and do not affect the comparative results or findings in our manuscript. We also observe that Judge@1 generally yields slightly higher scores, indicating that LLM-based evaluation is more tolerant of semantically correct but non-identical answers, whereas Hits@1 applies a stricter matching criterion and is more likely to classify such cases as incorrect.

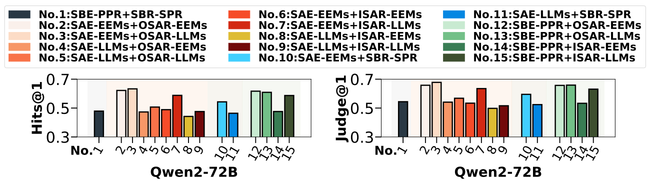


Figure 4: Evaluate the Instance with Hits@1 and Judge@1 on CWQ dataset

B.5 The trade-off between the F1 Score and Recall

We further discuss the trade-off between F1 Score and Recall and provide several practical guidelines for adjusting this balance in different scenarios. However, we first clarify that, **in most cases, a fixed threshold is sufficient, as F1 Score is the primary metric for evaluating subgraph-extraction**. We elaborate on this point in the following discussion, which has also been briefly addressed in our manuscript.

F1 Score balances Precision and Recall equally, reflecting overall performance. We additionally consider Recall because subgraph-extraction is an early module in the GraphRAG pipeline, and it affects the search space for downstream components. If the subgraph is too small due to a strong focus on Precision, the F1 Score may still be high, but there is a risk that some critical graph components may be omitted, which could potentially affect downstream reasoning. To provide more comprehensive coverage, we introduce Recall—not as a conflicting metric, but as a constraint—to ensure a minimum level of information coverage. In practice, setting a moderate Recall threshold (e.g., around 60%) is usually sufficient. Raising this threshold excessively may enlarge the subgraph, increase computational costs in both modules, and introduce redundant information.

This design choice is supported by our results in **Figures 8, 9, and 10 of the manuscript**, where Recall across most extraction methods (particularly SAE methods) and subgraph sizes tends to fall within the [0.55, 0.75] range, with a majority concentrated around 0.6. This is further supported by results in Figure 5, where we study the impact of subgraph Recall on downstream reasoning by extracting subgraphs with varying Recall using the SBE method (PPR) on the CWQ dataset. As the Recall threshold increases from 0.3 to around 0.6, Hits@1 improves steadily; beyond 0.6, the gains plateau.

Next, we provide guidelines for dynamically adjusting the minimum Recall threshold.

(1). By query complexity

Queries involving multi-hop reasoning, multiple answer targets, or ambiguous semantics are more likely to be affected by low Recall. In such cases, the threshold should be raised to ensure broader coverage of potentially relevant nodes and edges. For simpler single-hop queries with clear semantics, the threshold can be lowered to reduce subgraph size, improve efficiency, and avoid redundant information

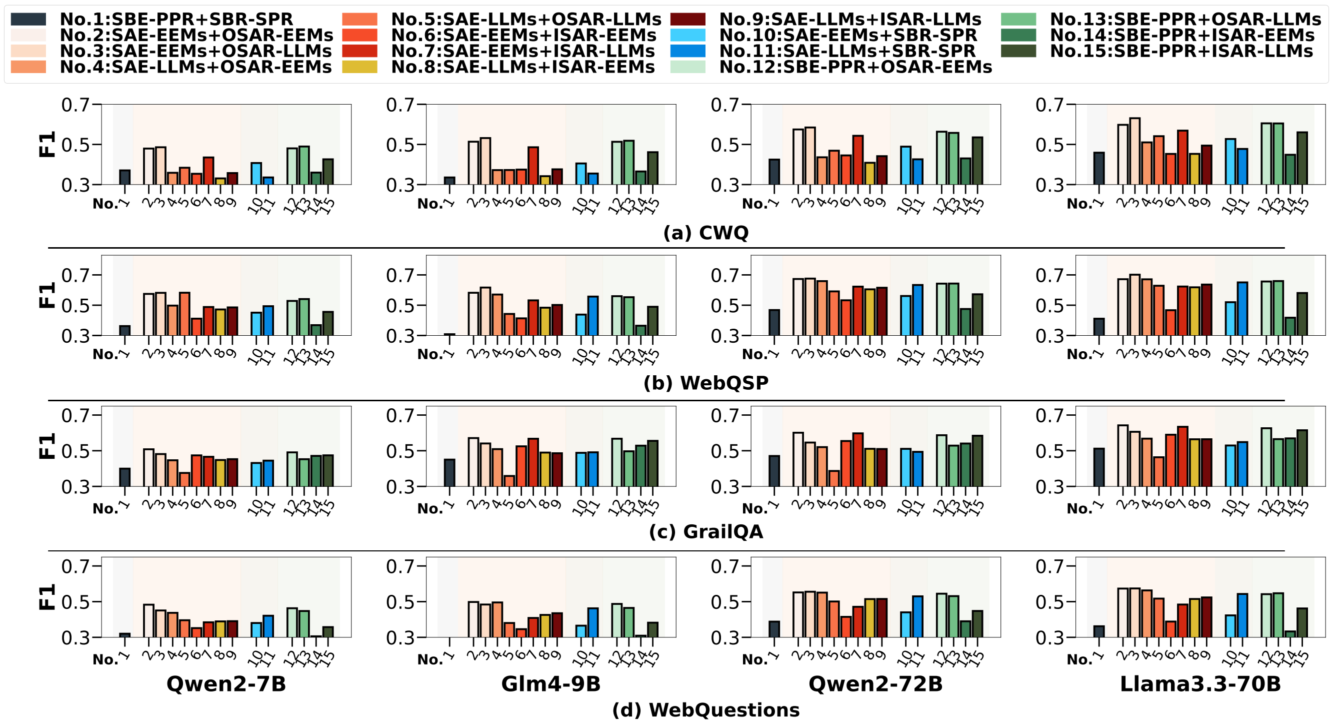


Figure 3: Reasoning Results of LEGO-GraphRAG Instances Across Four Datasets with F1

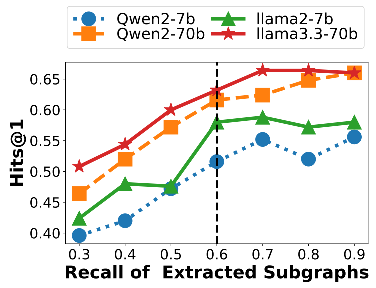


Figure 5: Influence of different Recall of Extracted Subgraphs on LLM reasoning results (SBE+OSAR-EEMs, CWQ)

that may hinder LLM reasoning. In practice, we recommend setting Recall thresholds based on the structural and semantic indicators of the query:

- **Structural indicators:** Queries with $\geq$ 2 answer targets or nested structures can be classified as high-complexity and assigned a higher threshold (e.g., 60%-75%). Others can use lower values (e.g., 50-60%).
- **Semantic indicators:** Queries with complex or vague semantics should also use a higher Recall threshold. We recommend leveraging LLMs to evaluate the semantic clarity of the query or predict the query's possible reasoning hops [78, 79, 106].

(2). By downstream applications

Recall thresholds should be adjusted according to the tolerance of downstream applications for missing or redundant information. We identify two typical categories:

- **Critical tasks**: Domains like healthcare, law, or finance often require relatively high Recall (e.g., 60%-75%) to avoid missing important information. However, the threshold should not be set too high (e.g., above 80%), as excessive Recall can introduce redundancy that negatively affects downstream modules and LLM reasoning.
- **Lightweight tasks**: Scenarios such as real-time QA, chat assistants, or interactive retrieval systems prioritize fast responses and concise outputs. In these cases, some information loss is acceptable, and lower Recall thresholds (e.g., 50%-60%) help reduce subgraph size and improve overall efficiency.

(3). By local graph density

The local graph structure associated with the query entities and relations also affects the minimum Recall threshold. In dense regions with many connected entities and paths, a high Recall threshold may introduce redundancy, leading to larger subgraphs and higher computational costs. In sparse regions with few relevant nodes and edges, a low threshold may miss key components needed for downstream reasoning.

- **Dense regions**: Use a lower Recall threshold (e.g., 50%-60%) to limit redundancy and keep the subgraph compact and efficient.

- **Sparse regions**: Use a higher threshold (e.g., 60%-75%) to ensure sufficient coverage of key entities and relations.

Local density can be estimated using indicators such as the number of neighbors or the clustering coefficient around the query entities and relations.

(4). By feedback logs

In addition to the aforementioned factors, Recall thresholds can also be dynamically optimized using historical query feedback. The logs can record the Recall thresholds used for different queries, along with the resulting reasoning accuracy. These data can be used to train predictive models, such as neural networks, to estimate the optimal Recall threshold for new queries based on similar historical patterns.

B.6 Fine-Tuning and Multiple LLM Calls

Our implementation integrates fine-tuning for both EEMs (i.e., embedding models and re-rankers) and LLMs, as well as the use of multiple LLM calls in the OSAR-LLM method to select reasoning paths. The test results for these two aspects on sample datasets are presented in Figures 6 and 7, and Table 4.

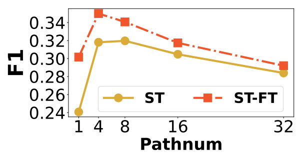


Figure 6: Fine-tuning for EEMs (ST, the Average of Four Datasets)

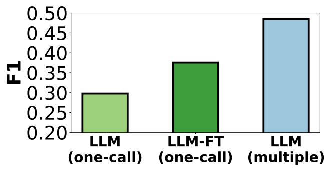


Figure 7: Fine-tuning & multiple Calls for LLMs (Qwen2-72B, WebQSP)

Table 4: Comparison of Fine-tuning Time and Average LLM Calls (WebQSP)

Method	Fine-tuning Time (s)	Ave. LLM Calls
ST-FT	28.7	-
LLM-FT	3748	1
Multiple LLM Calls	-	12.3

B.7 Efficiency of the Subgraph-Extraction

To address the efficiency bottleneck of the subgraph-extraction module, we explore three promising directions: (1) *computational acceleration*, (2) *precomputation-based acceleration*, and (3) *acceleration via vector databases* . The details are as follows:

(1).Computational acceleration for the SBE Method

This direction aims to directly improve the computational efficiency of the SBE method on large-scale graphs. Taking the commonly used PPR algorithm in SBE as an example, two typical strategies are considered:

- *Approximate PPR*: Reduces computational complexity by leveraging approximation techniques, leading to faster convergence.
- *Distributed PPR*: Employs distributed frameworks to compute PPR in parallel on large-scale graphs, thereby improving performance.

Table 5: Acceleration for PPR Algorithm

Approximate PPR			Distributed PPR	
Method	**Recall**	**Ave. Time (s)**	**Method**	**Speedup**
RBS [116]	0.31	**0.51**	HGPA [35]	3.4–4.1×
Fora [119]	0.18	7.61	PAFO [118]	58.7×
TopPPR [122]	0.44	42.08	Delta-Push [43]	**123–162×**
Standard PPR	**0.96**	75.29	**Standard PPR**	1× (baseline)

For approximate PPR, as shown in Table 5, we implement and compare three representative methods [1]—Fora, TopPPR, and RBS—on a large graph (Freebase with 100M nodes, 300M edges). We use a fixed subgraph size and measure both the average query time and Recall for each method. The results indicate the following:

(Effectiveness) Approximate PPR can significantly improve the efficiency of the subgraph-extraction module. For example, RBS completes the computation in 0.51s, compared to 75.29s required by the original PPR algorithm, an improvement of two orders of magnitude.

(Limitations) Although approximate methods improve efficiency, they inevitably introduce an approximation error, which primarily affects the Recall of the extracted subgraph. In general, approximate PPR methods offer different trade-offs between efficiency and Recall, depending on how their internal parameters are configured. Some configurations prioritize speed with lower Recall, while others target higher Recall at the cost of increased computational overhead.

(Research Opportunities) There is room to explore improved approximation strategies that reduce approximation error without significantly increasing computational cost. In addition, the choice of method or configuration can be adapted based on the Recall requirements of the extracted subgraph. As discussed in B.5, Recall requirements may vary with query complexity, downstream application needs, and local graph density. Taking these factors into account may help achieve a better balance between efficiency and accuracy in practice.

For distributed PPR, Table 5 reports the runtime speedups of three representative distributed PPR methods — HGPA, PAFO, and Delta-Push — compared to the standard PPR. Our observations are summarized as follows:

(Effectiveness) Distributed PPR can also substantially improve the efficiency of the subgraph-extraction module by leveraging distributed parallel computation. For instance, Delta-Push achieves the highest improvement, with a 123× to 162× speedup.

(Limitations) However, distributed PPR typically requires considerable computational resources and specific hardware configurations. Many methods are tightly coupled with specific distributed platforms, which limits their applicability in resource-constrained

[1]We use the default parameter settings provided in the official implementations of each method.

or heterogeneous environments. In addition, these methods often involve many system- and algorithm-level parameters that require careful tuning, posting challenges when applied to real-world graphs.

(Research Opportunities) Future work could explore more flexible distributed PPR frameworks that support plug-and-play integration in GraphRAG applications. Additionally, automated strategies for parameter tuning and system configuration optimization would improve the robustness and portability of these methods across diverse graphs. Combining approximate computation with distributed execution for subgraph-extraction also presents a promising hybrid optimization direction, potentially leveraging the strengths of both strategies.

(2).Acceleration via precomputation for the SBE Method

The core idea of this direction is to reduce the computational cost of subgraph-extraction through precomputation on large-scale graphs. Specifically, structure-, semantic-, or domain-aware strategies (e.g., graph clustering, community detection) can be used to construct smaller subgraphs in advance and generate semantic summaries. During extraction, the structural and semantic information of the query entity can be used to directly match relevant precomputed subgraphs (e.g., by computing the similarity between the query and the semantic summaries of the subgraphs). SBE methods are then applied within the matched subgraphs to improve overall efficiency.

Table 6: Performance and Cost Analysis of the SE Acceleration Methods on Freebase (About 100M Nodes, 300M Edges)

Method	Pre-time	Online-time	Recall	F1	WCC
Precomputation	19373s	19.02s	0.52	0.0021	**1**
Vector Database	14570s	**0.85s**	0.65	0.0023	12.86
Standard PPR	**0s**	75.29s	**0.96**	**0.0038**	**1**

To explore this direction, we implemented a prototype using the Leiden algorithm to cluster the graph and pre-partition it into subgraphs. During subgraph-extraction, we directly return the subgraph containing the query entity and apply PPR within that subgraph. The observations are presented in Table 6.

(Effectiveness) Precomputation effectively alleviates the efficiency bottleneck of subgraph-extraction. Our prototype completes extraction in 19.02 seconds with a Recall of 52%. This Recall is primarily constrained by the simplicity of the current prototype and can be further improved through more refined precomputation strategies.

(Limitations) Precomputation introduces additional overhead. In our simple prototype, it takes approximately 5 hours and requires additional storage for pre-partitioned subgraphs, their semantic summaries, and indexing structures. Moreover, predefined subgraph partitioning may cause information loss and limit support for multi-hop queries, as cross-subgraph relationships may not be effectively captured.

(Research Opportunities) Precomputation strategies can be further optimized to reduce overhead and improve subgraph partitioning quality. For example, adaptive partitioning methods can be explored to dynamically adjust subgraph granularity based on query distribution and graph structural properties, thereby mitigating information loss and reducing the limitations of cross-subgraph queries. In addition, incorporating multi-level indexing structures or incremental update mechanisms may further improve query efficiency while lowering both precomputation and storage costs.

(3).Acceleration via vector databases for the SAE Method

The core idea of this direction is to store semantic embedding vectors of graph components (e.g., nodes, edges, or triples) in a vector database, and retrieve relevant components based on their semantic similarity to the query, which are then used to construct subgraphs. Essentially, this idea leverages the efficient retrieval capabilities of vector databases to improve the efficiency of SAE methods.

We implemented a simple prototype to explore this direction. Specifically, we use a semantic model to encode the nodes of a large graph into semantic embedding vectors and store them in a vector database. For each query, we compute its embedding and retrieve the most similar nodes from the database. This retrieval process is highly efficient, enabled by the optimized similarity search capabilities of the vector database. To construct the subgraph, we include the one-hop neighbors of each retrieved node and merge them into a final subgraph. This ensures that the subgraph captures both query-relevant nodes and meaningful local structural information. The experimental results are shown in Table 6, and our observations are as follows:

(Effectiveness) Our prototype completes subgraph-extraction within 0.85 seconds, achieving a two-order-of-magnitude speedup compared to PPR. Moreover, the subgraphs achieve a Recall of 0.65, demonstrating superior preservation of query-relevant semantic information compared to other promising directions.

(Limitations) Using a vector database incurs non-trivial preprocessing overhead, including embedding generation and storage (approximately 4 hours in our prototype). Moreover, as graph components are embedded and stored independently, structural connectivity is not preserved, making it infeasible to construct fully connected subgraphs from retrieved nodes.

In our prototype, we construct subgraphs by merging the one-hop neighbors of the retrieved nodes. Despite this, the resulting subgraphs contain an average of 12 weakly connected components (WCC), whereas SBE methods typically yield a single connected component. While this approach enables efficient retrieval of semantically relevant nodes, the lack of structural constraints may limit its effectiveness in complex multi-hop reasoning tasks. Connectivity can be improved through expansion strategies such as incorporating two-hop or multi-hop neighborhoods, though at the cost of increased computation and reduced efficiency.

(Research Opportunities) To mitigate preprocessing overhead, more efficient embedding and storage strategies can be explored, such as using lightweight semantic models or applying vector quantization to reduce storage costs.

To improve subgraph connectivity, one direction is to embed and store graph triples. Although this incurs higher preprocessing costs compared to storing individual nodes or edges, it provides richer structural and semantic information. During subgraph extraction, the expansion range of retrieved triples can be dynamically determined based on the complexity of the query. This enables better control over the amount of structural context incorporated

Algorithm 1 ISAR Using Beam Search

```
Require: Graph $G$, Query $q$, start entity $v$, Beam width $B$, Max
    path length $L$, Semantic model $\mathcal{M}$ (EEMs or LLMs)
1:  Initialize $\mathcal{P}_q^{ISAR} \leftarrow \emptyset$, $\mathcal{B} \leftarrow \{([v], 0)\}$
2:  for each step $l = 1$ to $L$ do
3:      Initialize next beams $\mathcal{B}_{\text{next}} \leftarrow \emptyset$
4:      for each binary triple $(p, s) \in \mathcal{B}$ do
5:          $v_{\text{last}} \leftarrow$ The last node in path $p$
6:          for each edge $(v_{last}, v')$ where $v' \in$ Neighbors($v_{\text{last}}$) do
7:              $p' \leftarrow p + [v']$; $S(p') \leftarrow \mathcal{M}(p', q)$
8:              Add $(p', S(p'))$ to $\mathcal{B}_{\text{next}}$
9:          end for
10:     end for
11:     $\mathcal{B} \leftarrow$ Top $B$ binary triples $\in \mathcal{B}_{\text{next}}$
12:     Add paths in $\mathcal{B}$ to $\mathcal{P}_q^{ISAR}$
13: end for
14: return $\mathcal{P}_q^{ISAR}$
```

into the subgraph, achieving a balance between connectivity and computational efficiency.

B.8 Optimization of ISAR

Table 7: Results of ISAR-EEMs Optimization Strategies

Method	CWQ	WebQSP
SBE+ISAR-EEMs (Origin)	0.183	0.157
SBE+ISAR-EEMs (Triple-level scoring)	**0.224**	0.146
SBE+ISAR-EEMs (Dynamic pruning)	0.133	**0.23**

(1). Optimization of ISAR-EEMs

EEMs perform poorly under ISAR—a key finding of our work. A direct way to optimize ISAR-EEMs is to use stronger EEMs, such as customized models or fine-tuned ones for specific domains or graphs. In addition, our results on the supplementary MetaQA dataset show that ISAR-EEMs can achieve competitive performance in small-scale, single-domain graphs.

We also explore two strategies to improve ISAR-EEMs from the GraphRAG pipeline perspective.

- **Triple-level scoring:** In the standard ISAR process (e.g., Beam Search), the detailed algorithm flow is shown in Algorithm 1. Each path extension is typically evaluated based on the semantic representation of the full path. However, compared to LLMs, EEMs are more sensitive to accumulated noise from longer paths, which can lead to less reliable scores as the path grows. Thus, we propose scoring only the newly added triple at each step—that is, the *current node*, *edge*, *candidate next node*—using EEMs.
- **Dynamic pruning:** In standard ISAR, a fixed number of top-B paths are retained at each step (e.g., beam width B in Beam Search), assuming the semantic scores are well-separated and reliable. However, EEMs often produce flat score distributions with similar values, leading to premature pruning of good paths or retention of poor ones. Thus, we suggest normalizing the candidate scores at each step and applying a dynamic threshold based on the score distribution—e.g., retaining paths with scores above the current mean or a given percentile—so that the number of expanded paths adapts automatically during search.

We empirically evaluate both strategies on two benchmark datasets: CWQ focuses on multi-hop queries, while WebQSP covers mostly single-hop queries. As shown in Table 7, the two strategies offer complementary improvements to ISAR-EEMs.

The triple-level scoring strategy significantly improves performance on CWQ (from 0.183 to 0.224), but slightly decreases on WebQSP (from 0.157 to 0.146), as it helps reduce noise from full-path representations in longer reasoning chains. For single-hop queries, however, full-path scoring is generally sufficient, and localized scoring offers limited advantage. The dynamic pruning strategy improves performance on WebQSP (from 0.157 to 0.230) but degrades on CWQ (from 0.183 to 0.133). This is consistent with its design: dynamic beam adjustment is effective when score distributions are flat and decisions can be made early. In multi-hop queries, however, useful paths often emerge gradually, and pruning based on early scores may discard them before their relevance becomes clear.

Therefore, the two strategies can be combined. A lightweight classifier (e.g., FRAG [32]) can predict the number of hops in a query's reasoning path and inform whether to apply triple-level scoring, dynamic pruning, or both, to improve the stability of ISAR-EEMs performance.

(2) hybrid ISAR with EEMs and LLMs

Combining EEMs and LLMs in ISAR is a promising direction. We examine hybrid methods from two directions: ISAR-EEMs with LLMs, and ISAR-LLMs with EEMs. We also conduct empirical evaluation of these designs.

ISAR-EEMs with LLMs: As mentioned in the first part of this response, we propose two optimization strategies for ISAR-EEMs that do not involve LLMs, mainly aimed at reducing the impact of redundant semantic information from long paths in ISAR. Introducing LLMs also can help mitigate EEMs' limitations in handling such redundancy. We explore two methods:

- **LLMs refinement (LLMs-R):** After the ISAR-EEMs search process, LLMs are used to refine the retrieved paths by reducing redundancy and improving semantic coherence.
- **LLM early stopping (LLMs-ES):** During the ISAR-EEMs search process, LLMs determine whether the current path is sufficient, allowing the search to stop early when appropriate.

ISAR-LLMs with EEMs: While ISAR-LLMs outperform ISAR-EEMs in accuracy, they can be less efficient and less robust. To address this, we consider integrating EEMs into ISAR-LLMs. We explore two methods:

- **EEMs pre-filter (EEMs-PF):** At each step of the LLM-driven search, EEMs are used to pre-rank and filter candidate paths before passing them to the LLMs. This improves search efficiency and helps avoid exceeding the input length limit of the LLMs.

- **EEM supplement (EEMs-S):** At each step of path expansion, if the LLMs generates too few or low-quality candidates (e.g., incomplete or off-prompt outputs), EEMs are used to supplement the expansion with top-ranked relevant paths.

Table 8 reports the performance of hybrid ISAR methods combining EEMs and LLMs. For ISAR-EEMs, applying LLMs refinement (LLMs-R) significantly improves performance on all datasets, while LLMs early stopping (LLMs-ES) consistently leads to performance degradation. This suggests that relying on LLMs to make early stopping decisions tends to disrupt the EEMs-guided search process, preventing the exploration of complete reasoning paths.

For ISAR-LLMs, both EEMs pre-filtering (EEMs-PF) and EEMs supplementation (EEMs-S) yield further gains. Among them, EEMs-S achieves the best results, showing that EEMs can effectively complement LLMs when their outputs are insufficient or of low quality.

Table 8: Results of Hybrid ISAR Methods Combining EEMs and LLMs

Method	CWQ	WebQSP	GrailQA	WebQuestion
SBE+ISAR-EEMs	0.183	0.157	0.395	0.16
SBE+ISAR-EEMs (LLMs-ES)	0.135	0.117	0.304	0.120
SBE+ISAR-EEMs (LLMs-R)	**0.271**	**0.219**	**0.379**	**0.175**
SBE+ISAR-LLMs	0.249	0.264	0.371	0.249
SBE+ISAR-LLMs (EEMs-PF)	0.271	0.316	0.396	0.245
SBE+ISAR-LLMs (EEMs-S)	**0.310**	**0.373**	**0.435**	**0.276**

B.9 Improve Generation Quality of LLMs

Rather than enhancing the LLMs themselves, we focus on optimizing their integration within the GraphRAG workflow. From this perspective, we explore several strategies in the path-retrieval module to address two issues: the generation of fewer and lower-quality paths by LLMs.

(1) The issue of low-quality paths

We agree that reranking and filtering LLM-generated paths are necessary to improve their quality. To explore this direction, we implemented and evaluated two strategies:

- **EEMs reranking (EEMs-R):** After generating paths with the LLMs, we apply EEMs to rescore and filter the outputs, retaining only the top-ranked paths.
- **Multi-round LLMs (M-LLMs):** Instead of relying on a single LLM call, we invoke the LLM multiple times to iteratively refine the generated paths and improve their quality.

As shown in Table 9 (using SBE+OSAR-LLMs as the base instance), EEM-based reranking has a negligible effect on performance, while the multi-round LLM strategy yields consistent improvements across all datasets.

(2) The issue of fewer generated paths

To address the issue that LLMs may generate fewer reasoning paths than expected, we explore two optimization strategies:

- **EEMs supplementation (EEMs-S):** When the number of generated paths of LLMs is insufficient, we supplement them with top-ranked reasoning paths selected by EEMs.
- **LLMs embeddings (LLMs-E):** We treat LLMs similarly to EEMs by using them to generate semantic embeddings for candidate paths, and then perform similarity-based ranking instead of prompt-based generation.

As shown in Table 9, path supplementation improves performance, particularly on datasets like CWQ and GrailQA. In contrast, using LLMs purely as embedding models provides limited benefits and underperforms prompt-based generation, indicating that this approach does not fully leverage the reasoning capabilities of LLMs.

Table 9: Performance of strategies designed to mitigate the limitations of LLM-generated outputs

Method	CWQ	WebQSP	GrailQA	WebQuestions
SBE+OSAR-LLMs	0.304	0.401	0.401	0.328
SBE+OSAR-LLMs (EEMs-R)	0.303	0.397	0.395	0.327
SBE+OSAR-LLMs (M-LLMs)	0.349	**0.438**	0.381	0.348
SBE+OSAR-LLMs (LLMs-E)	0.303	0.277	0.414	0.233
SBE+OSAR-LLMs (EEMs-S)	**0.381**	0.427	**0.476**	**0.353**

B.10 Comparison between GraphRAG and text-based RAG

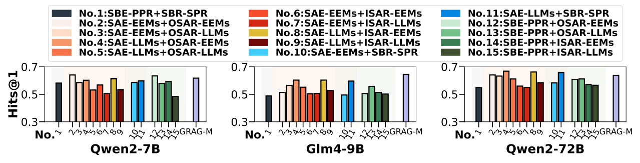


Figure 8: Results of LEGO-GraphRAG and GraphRAG-M Instance on MetaQA Dataset

We integrate graph construction as an optional phase in our framework to improve its generality. We also add a Microsoft GraphRAG instance and compare it with our 15 existing GraphRAG instances. Based on existing empirical studies and analysis, we summarize the differences between text-based RAG and GraphRAG, along with their respective strengths.

It is important to clarify that, while graph construction is relevant to the broader GraphRAG workflow, it is not the focus of our study. In many practical scenarios, the graph is either pre-constructed using domain knowledge or directly derived from structured knowledge resources. Moreover, constructing graphs from text has been widely studied, including recent methods based on LLMs and extraction techniques from the knowledge graph literature. These aspects, while important, fall outside the scope of our contribution. *Our work centers on the core process that follows graph construction—retrieving reasoning paths from the graph based on a query—which we consider the essence of GraphRAG.*

(1). Integration of Graph Construction Phase and Microsoft GraphRAG

Following Microsoft GraphRAG [25], we introduce the following definition of graph construction into our framework:

Definition 1 (**Graph Construction (***Optional***)**). *In GraphRAG, graph construction is an optional preprocessing phase that builds a text-attributed graph from large-scale textual data. It may include a step that precomputes auxiliary subgraphs and textual information from the graph to support LLM reasoning. Specifically:*

- ***Construction:*** *Identify entities and relationships from text and represent them as nodes and edges in the graph.*
- ***Precomputation (****Optional****):*** *Compute subgraphs and relevant textual summaries from the constructed graph, such as via graph clustering or community detection.*

Notably, precomputation can be seen as a way to integrate text-based RAG enhancements into GraphRAG. In addition to improving GraphRAG's performance, it can also boost the efficiency of subgraph extraction by precomputing candidate subgraphs. In Microsoft GraphRAG, hierarchical clustering is applied to generate communities (i.e., subgraphs), and a semantic summary is produced for each. These communities and summaries can then be directly matched with queries.

To demonstrate compatibility with the Microsoft GraphRAG workflow, we implement a corresponding instance within our framework. This instance performs community detection on the graph and generates community reports through precomputation. Given a query, it uses both the reasoning paths [2] and the community report associated with the query entity as additional input to the LLMs.

Figure 8 presents the performance of the Microsoft GraphRAG instance (GRAG-M) compared to the other 15 instances in our framework on the MetaQA dataset, as its rich semantic content makes it particularly suitable for performing precomputation. The results show that instance GRAG-M achieves competitive and sometimes superior performance across multiple reasoning LLMs, particularly Qwen2-7B and Glm4-9B. Interestingly, on Qwen2-72B, GRAG-M is not the best-performing instance, suggesting that larger LLMs may already capture sufficient contextual information, reducing the marginal benefit of precomputed community summaries.

Table 10: Experiment Results Comparison between RAG and GraphRAG

Dataset	NQ† [62]	News Articles* [109]	Podcast* [101]
text-based RAG	**64.78**	25.23	26.50
GraphRAG [24]	63.01	**34.18**	**32.21**

† Performance metric: F1 score. Data source: [38].
* Performance metric: Average number of extracted claims. Data source: [24].

(2). Text-based RAG v.s. GraphRAG

Building on the analysis of recent work already discussed [24, 36, 38, 72, 83, 135], we summarize the key differences between text-based RAG and GraphRAG across three aspects: (1) knowledge representation, (2) retrieval process, and (3) applicable scenarios.

- **Knowledge representation.** Text-based RAG represents knowledge as unstructured text chunks, without modeling the semantic structure or relational patterns underlying the knowledge [83]. In contrast, GraphRAG represents knowledge as structured graphs with explicitly defined entities and relations [24], enabling multi-hop, fine-grained reasoning.
- **Retrieval process.** Text-based RAG retrieves top-k text chunks based on semantic similarity, making it effective for simple factual queries [72], as demonstrated on the NQ dataset [62] in empirical study [38], as shown in Table 10. However, it struggles with multi-hop queries that require fact chaining or implicit links [38], especially when relevant information spans multiple text chunks, making it difficult to recover the entity links needed for multi-hop reasoning [24, 83]. This issue is illustrated by the performance of Microsoft GraphRAG on datasets such as News Articles [109] and Podcast [101]. In contrast, GraphRAG conducts retrieval over graphs by traversing paths between entities, naturally supporting multi-hop reasoning with clearer reasoning paths [24]. Its compact graph representation reduces token usage by 26%–97% during LLMs reasoning [135] and offers better interpretability than ranked text chunks, making it well suited for reasoning-oriented queries.
- **Applicable scenarios.** Text-based RAG is suitable for scenarios requiring fast response and retrieval with relaxed accuracy demands, such as online customer service, dialogue agents, or general-purpose search engines [38]. These applications typically handle large volumes of simple factual queries and can tolerate a certain degree of imprecision. In contrast, GraphRAG is more appropriate for scenarios that demand precise reasoning over structured knowledge, such as decision support in government, healthcare, or finance [72]. It is particularly useful in settings where traceable multi-hop reasoning or long-term knowledge tracking is critical [115].

B.11 Comparison between LEGO-GraphRAG and Modular RAG

Modular RAG contributes to the modularization of general RAG. However, Modular RAG does not address the specific workflow and requirements of GraphRAG. Its general framework is not directly applicable in this setting. In contrast, LEGO-GraphRAG is designed specifically for GraphRAG, with dedicated module definitions, technical summarization, and empirical analysis. It extends and goes beyond Modular RAG in several key aspects. Specifically:

(1). The core retrieval steps of GraphRAG are not modularized in Modular RAG. GraphRAG's retrieval phase fundamentally differs from that of traditional RAG, as it focuses on retrieving reasoning paths from graphs rather than text chunks. Modular RAG abstracts general RAG workflows and emphasizes mechanisms such as routing, scheduling, and fusion, but it does not offer fine-grained modularization of the key retrieval steps specific to GraphRAG. In contrast, LEGO-GraphRAG explicitly decomposes the retrieval phase into two core modules: subgraph-extraction and path-retrieval.

[2] The reasoning paths are provided by Instance No.12 (SBE-PPR + OSAR-EEMs), which showed strong performance in our experiments, to ensure their quality.

(2). Key techniques in GraphRAG are not systematically analyzed in Modular RAG. Modular RAG does not provide detailed analysis of core techniques widely used in GraphRAG, such as structure-based vs. semantics-based, or EEMs-based vs. LLMs-based. It gives high-level mentions of broad technical aspects, which are more relevant to traditional RAG. In contrast, LEGO-GraphRAG organizes these techniques into modules, offering a clearer understanding of their roles and trade-offs across different scenarios.

(3). GraphRAG is mentioned only as an orchestration module in Modular RAG, not as a complete and well-defined workflow. Modular RAG briefly refers to GraphRAG through the "Knowledge Guide" module (labeled as orchestration in the original paper), but does not model GraphRAG as a complete workflow. It also does not clarify which modules and methods are not applicable to GraphRAG, such as those designed specifically for text chunking. In contrast, LEGO-GraphRAG constructs a modular framework tailored to the specific needs of GraphRAG, supporting both the reproduction of existing work and the composition of new instances, with GraphRAG as the central focus.

(4). Modular RAG lacks empirical studies and key findings. LEGO-GraphRAG not only proposes a framework but also implements multiple GraphRAG instances and conducts comprehensive experiments on real-world graphs and diverse query sets. These studies reveal key findings, such as the impact of different module combinations on performance and resource usage. In contrast, Modular RAG focuses more on workflow abstraction and survey-style analysis, without empirical implementation or evaluation of GraphRAG.

B.12 Case Study of different Instances

To better illustrate effectiveness differences across GraphRAG instances, we conduct a case study comparing the retrieved reasoning paths and answers of seven selected instances. Based on overall performance across datasets and LLMs, we selected one relatively strong and one relatively weak instance from each group (except for Group I, which includes only one). The selected instances are: No.1 (SBE-PPR + SBR-SPR), No.2 (SAE-EEMs + OSAR-EEMs), No.8 (SAE-LLMs + ISAR-EEMs), No.10 (SAE-EEMs + SBR-SPR), No.11 (SAE-LLMs + SBR-SPR), No.12 (SBE-PPR + OSAR-EEMs), and No.14 (SBE-PPR + ISAR-EEMs).The results of this comparison are shown in Figure 9.

B.13 Prompts for LLMs

Subgraph-Extraction Prompt Example

You are an expert filterer with a deep understanding of relevant information. Your task is to analyze the given question and entities, and identify the potentially useful reasoning paths from the provided corpus. For each question, you will select the most relevant paths that can help answer the question effectively. Your selection should be based on the entities mentioned in the question and their relationships with other entities in the corpus. Make sure to consider the relevance, specificity, and accuracy of each path in relation to the given question and entities.

Example
<Example>
Question:
<Question>

Path-Retrieval Prompt Example

You are an expert at retrieving the most relevant paths from a given input to answer specific questions.
Example:
<Example>
Reasoning Paths:
<Reasoning Path>
Question:
<Question>

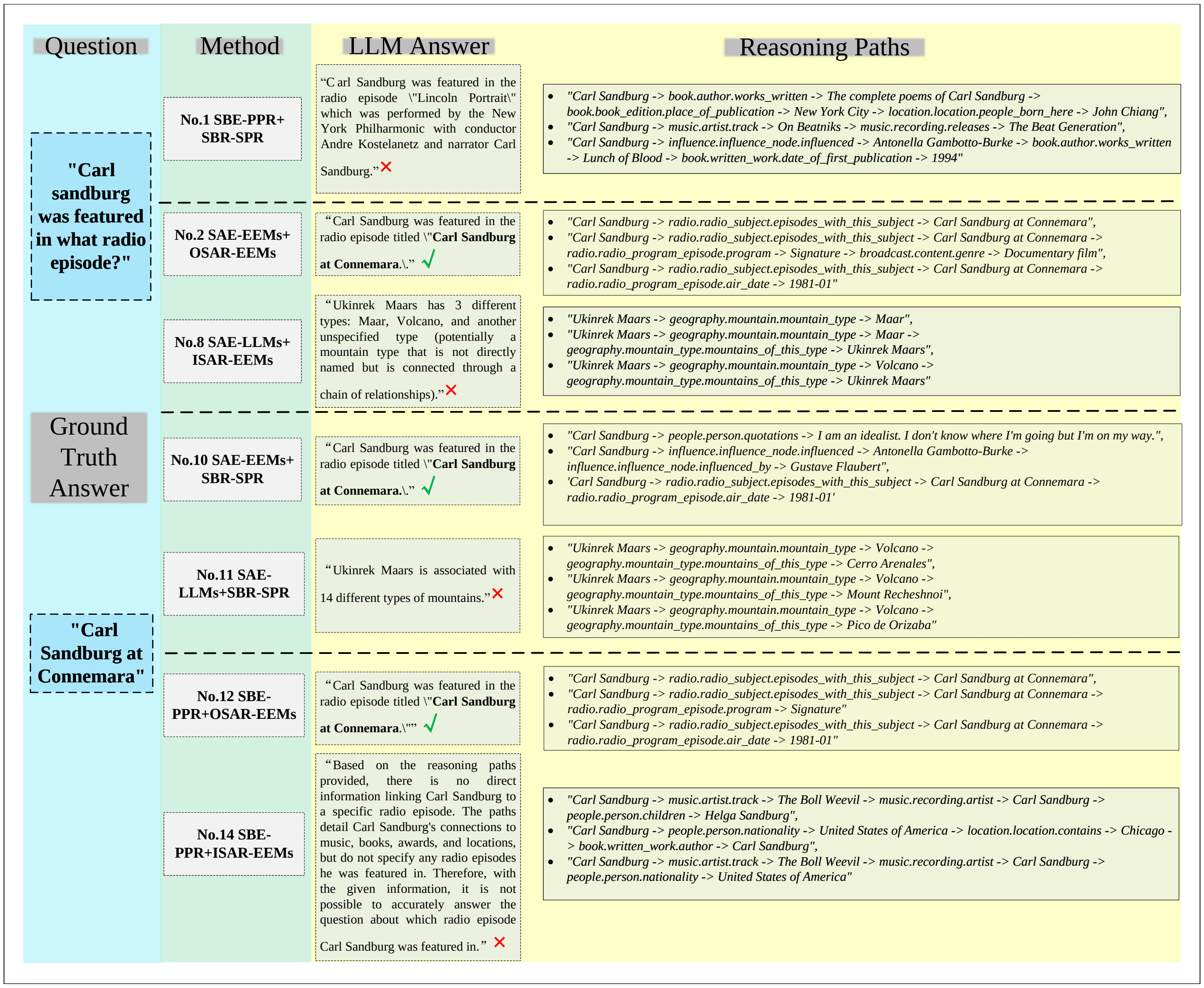
Zero-Shot Reasoning Prompt Example

You are an expert reasoner with a deep understanding of logical connections and relationships. Your task is to analyze the given reasoning paths and provide clear and accurate answers to the questions based on these paths. Based on the reasoning paths, please answer the given question.
Reasoning Path:
<Reasoning Paths>
Question:
<Question>

Here, <Example> refers to a set of human-annotated few-shot examples used to demonstrate the explanation process, <Question> indicates the question and <Reasoning Paths> refers to the retrieved reasoning paths, which are presented as a sequence of structured sentences.

Node Pruning with LLMs Prompt Example

You are an expert in identifying and selecting the most relevant entities from a given corpus to answer specific questions. Your role involves analyzing the provided question and related entities, and then determining the most useful entities that can help address the question effectively. Your selection should be guided by the relevance, specificity, and accuracy of each entity in relation to the entities mentioned in the question. Ensure that the entities you select are concise and directly related to the question.
Example:
<Example>
Entities:
<Entities>
Question:
<Question>



Question: "Carl sandburg was featured in what radio episode?"

Ground Truth Answer: "Carl Sandburg at Connemara"

Method	LLM Answer	Reasoning Paths
No.1 SBE-PPR+ SBR-SPR	"C arl Sandburg was featured in the radio episode \"Lincoln Portrait\" which was performed by the New York Philharmonic with conductor Andre Kostelanetz and narrator Carl Sandburg." ✗	• "Carl Sandburg -> book.author.works_written -> The complete poems of Carl Sandburg -> book.book_edition.place_of_publication -> New York City -> location.location.people_born_here -> John Chiang", • "Carl Sandburg -> music.artist.track -> On Beatniks -> music.recording.releases -> The Beat Generation", • "Carl Sandburg -> influence.influence_node.influenced -> Antonella Gambotto-Burke -> book.author.works_written -> Lunch of Blood -> book.written_work.date_of_first_publication -> 1994"
No.2 SAE-EEMs+ OSAR-EEMs	"Carl Sandburg was featured in the radio episode titled \"**Carl Sandburg at Connemara**.\"" ✓	• "Carl Sandburg -> radio.radio_subject.episodes_with_this_subject -> Carl Sandburg at Connemara", • "Carl Sandburg -> radio.radio_subject.episodes_with_this_subject -> Carl Sandburg at Connemara -> radio.radio_program_episode.program -> Signature -> broadcast.content.genre -> Documentary film", • "Carl Sandburg -> radio.radio_subject.episodes_with_this_subject -> Carl Sandburg at Connemara -> radio.radio_program_episode.air_date -> 1981-01"
No.8 SAE-LLMs+ ISAR-EEMs	"Ukinrek Maars has 3 different types: Maar, Volcano, and another unspecified type (potentially a mountain type that is not directly named but is connected through a chain of relationships)." ✗	• "Ukinrek Maars -> geography.mountain.mountain_type -> Maar", • "Ukinrek Maars -> geography.mountain.mountain_type -> Maar -> geography.mountain_type.mountains_of_this_type -> Ukinrek Maars", • "Ukinrek Maars -> geography.mountain.mountain_type -> Volcano -> geography.mountain_type.mountains_of_this_type -> Ukinrek Maars"
No.10 SAE-EEMs+ SBR-SPR	"Carl Sandburg was featured in the radio episode titled \"**Carl Sandburg at Connemara**.\"" ✓	• "Carl Sandburg -> people.person.quotations -> I am an idealist. I don't know where I'm going but I'm on my way.", • "Carl Sandburg -> influence.influence_node.influenced -> Antonella Gambotto-Burke -> influence.influence_node.influenced_by -> Gustave Flaubert", • 'Carl Sandburg -> radio.radio_subject.episodes_with_this_subject -> Carl Sandburg at Connemara -> radio.radio_program_episode.air_date -> 1981-01'
No.11 SAE-LLMs+SBR-SPR	"Ukinrek Maars is associated with 14 different types of mountains." ✗	• "Ukinrek Maars -> geography.mountain.mountain_type -> Volcano -> geography.mountain_type.mountains_of_this_type -> Cerro Arenales", • "Ukinrek Maars -> geography.mountain.mountain_type -> Volcano -> geography.mountain_type.mountains_of_this_type -> Mount Recheshnoi", • "Ukinrek Maars -> geography.mountain.mountain_type -> Volcano -> geography.mountain_type.mountains_of_this_type -> Pico de Orizaba"
No.12 SBE-PPR+OSAR-EEMs	"Carl Sandburg was featured in the radio episode titled \"**Carl Sandburg at Connemara**.\"" ✓	• "Carl Sandburg -> radio.radio_subject.episodes_with_this_subject -> Carl Sandburg at Connemara", • "Carl Sandburg -> radio.radio_subject.episodes_with_this_subject -> Carl Sandburg at Connemara -> radio.radio_program_episode.program -> Signature" • "Carl Sandburg -> radio.radio_subject.episodes_with_this_subject -> Carl Sandburg at Connemara -> radio.radio_program_episode.air_date -> 1981-01"
No.14 SBE-PPR+ISAR-EEMs	"Based on the reasoning paths provided, there is no direct information linking Carl Sandburg to a specific radio episode. The paths detail Carl Sandburg's connections to music, books, awards, and locations, but do not specify any radio episodes he was featured in. Therefore, with the given information, it is not possible to accurately answer the question about which radio episode Carl Sandburg was featured in." ✗	• "Carl Sandburg -> music.artist.track -> The Boll Weevil -> music.recording.artist -> Carl Sandburg -> people.person.children -> Helga Sandburg", • "Carl Sandburg -> people.person.nationality -> United States of America -> location.location.contains -> Chicago -> book.written_work.author -> Carl Sandburg", • "Carl Sandburg -> music.artist.track -> The Boll Weevil -> music.recording.artist -> Carl Sandburg -> people.person.nationality -> United States of America"

Figure 9: Case Study for LEGO-GraphRAG

Edge Pruning with LLMs Prompt Example

You are an expert in identifying and selecting the most relevant relational paths from a given corpus to answer specific questions. Your role involves analyzing the provided question and related entities, and then determining the most useful relational paths that can help address the question effectively. Your selection should be guided by the relevance, specificity, and accuracy of each path in relation to the entities mentioned in the question. Ensure that the paths you select are concise and directly related to the question.
Example:
<Example>
Relations:
<Relations>

Question:
<Question>

Triple Pruning with LLMs Prompt Example

You are an expert in identifying and selecting the most relevant triples from a given corpus to answer specific questions. Your role involves analyzing the provided question and related entities, and then determining the most useful triples that can help address the question effectively. Your selection should be guided by the relevance, specificity, and accuracy of each triple in relation to the entities mentioned in the question. Ensure that the triples you select are concise and directly related to the question.

Example:
<Example>
Triples:
<Triples>
Question:
<Question>

Here, <Entities>, <Relations> and <Triples> refers to the retrieved entities/relations/triples from KGs, which are presented as a sequence of structured sentences.