Align-GRAG: Anchor and Rationale Guided Dual Alignment for Graph Retrieval-Augmented Generation

Anonymous ACL submission

Abstract

Large language models (LLMs) have achieved strong performance, yet they remain vulnerable to hallucinations and outdated knowledge, which limits their reliability in knowledge-intensive tasks. Graph-based retrieval-augmented generation (GRAG) enriches LLMs with knowledge by retrieving graphs leveraging relational evidence, but it faces two challenges: structure-coupled irrelevant knowledge introduced by neighbor expansion and structure-reasoning discrepancy between graph embeddings and LLM semantics. We propose **`Align-GRAG`**, an anchor-and-rationale guided refinement framework to address these challenges. It prompts an LLM to extract anchors and rationale chains, which provide intermediate supervision for **(1) node-level alignment** that identifies critical nodes and prunes noisy evidence, and **(2) graph-level alignment** that bridges graph and language semantic spaces via contrastive learning. Extensive experiments on commonsense reasoning, scene graph understanding, and knowledge graph reasoning demonstrate consistent gains over 18 strong baselines, validating the effectiveness of **`Align-GRAG`** for improving graph-grounded generation. The code can be found in `https://anonymous.4open.science/r/Align-GRAG-F3D8/`.

1 Introduction

Recent progress in large language models (LLMs) has highlighted their strong abilities in understanding, reasoning (Zhao et al., 2023; Xu et al., 2024a), and knowledge-intensive tasks (Zhu et al., 2023; AlKhamissi et al., 2022; Sui et al., 2025). Despite these advances, LLMs still raise reliability concerns, including hallucinations (producing false or misleading information) (Huang et al., 2023; Xu et al., 2024c) and reliance on outdated data (Fang et al., 2025). These issues are especially concerning in high-stakes fields such as healthcare (He et al., 2023) and law (Lai et al., 2024).

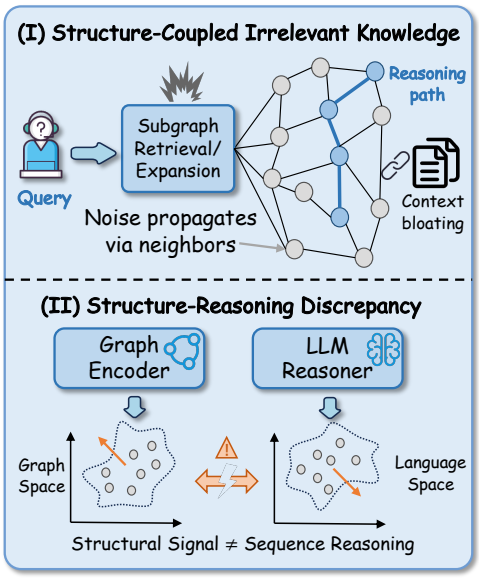


Figure 1: The challenges in graph RAG scenarios.

Retrieval-augmented generation (RAG) addresses these concerns by retrieving query-relevant evidence from external databases and grounding generation in a verifiable context, improving factuality and robustness (Fan et al., 2024; Zhou et al., 2025). Nevertheless, in real-world scenarios, RAG systems often divide long context into independent chunks, overlooking the deeper connections between fragments and lacking a global perspective (Guo et al., 2024). Moreover, many knowledge sources are inherently graph-structured, including recommendation systems (Wang et al., 2025), the Web, and knowledge graphs (Pan et al., 2024). Motivated by this, graph-based RAG (GRAG) (Edge et al., 2024; He et al., 2024; Tian et al., 2024; Sun et al.) extends RAG by retrieving subgraphs rather than isolated passages (He et al., 2024), preserving correlations among multiple text units.

However, integrating RAG with graphs is non-trivial, raising two GRAG-specific challenges, as shown in Figure 1. **(I) Structure-Coupled Ir-**

relevant Knowledge. To ensure multi-hop coverage, GRAG typically does expansions with retrieved interconnected neighbors. While necessary, this process introduces "structure-coupled" noise–irrelevant nodes that are topologically close to the target but semantically unhelpful for the specific reasoning chain. Crucially, determining the utility of these nodes requires a holistic view of the reasoning path rather than assessing local query similarity. Existing post-retrieval rerankers (Xiao et al., 2024; Li et al., 2023; Jia et al., 2024), designed for independent text segments, fail in this context: by treating nodes or triples as isolated units, they sever the structural dependencies essential for judging reasoning-dependent relevance. As a result, current GRAG systems (Guo et al., 2024; Edge et al., 2024) lack a structure-aware post-retrieval refinement mechanism with sufficient semantic modeling. **(II) Structure-Reasoning Discrepancy.** Incorporating structure embeddings (e.g., from GNNs) is hindered by a representation gap: graph embeddings primarily capture structural signals, whereas LLMs operate over sequence semantics. This mismatch makes it difficult for LLM reasoning to reliably exploit graph structure, limiting graph-grounded generation (Liu et al., 2023; Zhao et al., 2022). Existing GRAG methods either linearize graphs as text (Sun et al.; LUO et al.; Xu et al., 2025; Hu et al., 2022; Chen et al., 2024; Jiang et al., 2024) or incorporate graph encoders with simple concatenation/projection (Tian et al., 2024; Perozzi et al., 2024; He et al., 2024); however, they typically lack an explicit objective that aligns structural representations with the semantics required for LLM reasoning. These research gaps present a question:

> *How can we construct a compact yet structure-preserving subgraph and learn aligned graph representations for better LLM reasoning?*

In this work, we propose **Align-GRAG**, a rationale-guided dual alignment framework tailored for GRAG. To tackle the two challenges above, we introduce a graph aligner as a pre-refinement module. Specifically, we leverage LLM with well-crafted prompts to extract rationale chains and anchors that highlight key intermediate concepts. We then use these signals as supervision to optimize dual alignment: ♣ **Node-level Alignment**, which identifies and prioritizes reasoning-critical nodes/edges to prune structure-coupled irrelevant evidence; and ♠ **Graph-level Alignment**, which learns an aligned semantic space between graph and language representations via contrastive learning. Together, these components yield a compact yet structure-preserving subgraph, enabling the generator to produce more accurate and context-aware responses grounded in relational evidence. Extensive experiments are conducted on GraphQA benchmark (He et al., 2024), covering commonsense reasoning, scene graph understanding, and knowledge graph reasoning tasks. The results consistently outperform 18 strong baselines and highlight the effectiveness of **Align-GRAG**.

2 Preliminaries

Textual Graph. A textual graph is a graph where nodes and edges are enriched with textual information, capturing structural and semantic details. A textual graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \{t_n\}_{n\in\mathcal{V}}, \{t_e\}_{e\in\mathcal{E}})$, where $\mathcal{V}$ and $\mathcal{E}$ are the sets of nodes and edges. $t_n \in \mathcal{D}^{L_n}$ represents the text associated with a node $n \in \mathcal{V}$, where $\mathcal{D}$ is the vocabulary, and L_n is the length. Similarly, $t_e \in \mathcal{D}^{L_e}$ is the text for edge $e \in \mathcal{E}$ with length L_e.

Task Formulation. This work addresses RAG with textual graph. The goal is to retrieve relevant information from textual graphs and generate accurate responses. Given a query t_q, a sequence of tokens from $\mathcal{D}$, the model retrieves a subgraph $\mathcal{G}_r = (\mathcal{V}_r, \mathcal{E}_r)$ from $\mathcal{G}$ based on the semantic similarity of t_n, t_e with t_q, where $\mathcal{V}_r \subseteq \mathcal{V}$ and $\mathcal{E}_r \subseteq \mathcal{E}$. The retrieved nodes, edges, and texts are refined to improve the input quality for the LLM. During generation, the retrieved subgraph, query t_q, and prompt $\mathcal{P}$ are provided as input to the LLM, which produces the final output Y, grounded in the retrieved knowledge and graph context.

3 Methodology

This section presents **Align-GRAG** (Figure 2), an anchor-and-rationale guided refinement framework for GRAG. **Align-GRAG** first extracts anchors concepts and compact rationale chains with LLM, and then uses them to guide a graph aligner that (i) performs structure-aware pruning to remove irrelevant evidence while preserving relational dependencies and (ii) reduces the gap between graph and language representations.

3.1 Anchor and Rationale Extraction

To optimize the Aligner module, we leverage an explicit reasoning signal extracted for each query,

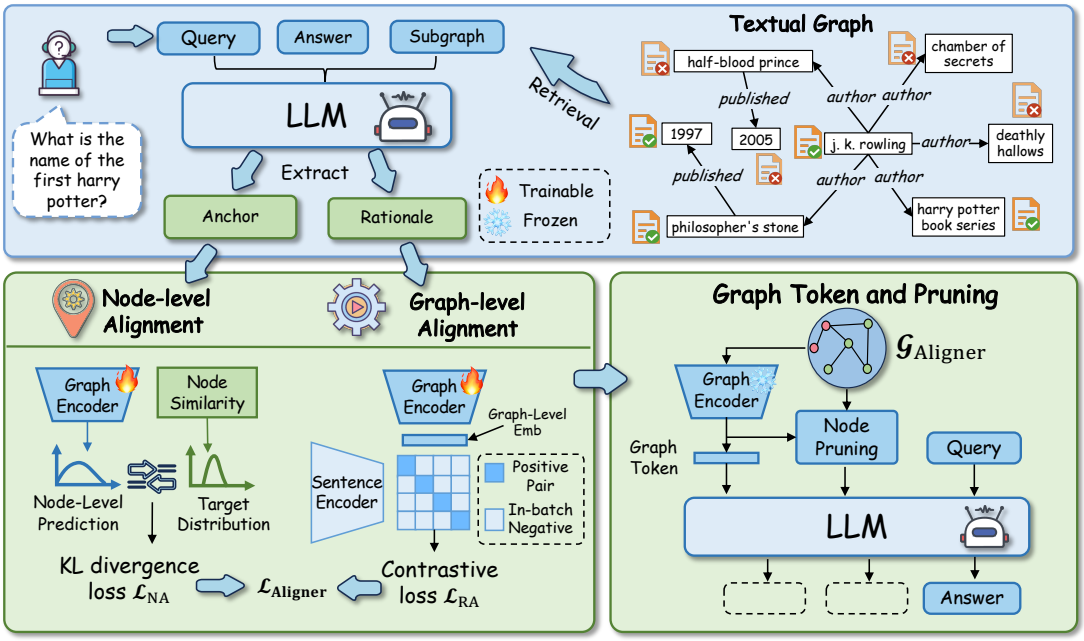


Figure 2: Overview of **Align-GRAG**. We first extract a rationale chain and anchors with LLM. The rationale guides graph-level alignment, while anchors supervise node-level alignment.

which is especially important for multi-hop questions requiring intermediate steps. Inspired by Chain-of-Thought (Wei et al., 2022) and O1-style reasoning (Zhong et al., 2024), we propose an LLM-based extraction module that produces (i) a compact rationale chain and (ii) a small set of anchors.

Specifically, given a question q and its answer, we first retrieve an initial subgraph (as elaborated in Appendix A). We then prompt a strong LLM to extract (i) a concise rationale chain that connects q to the supporting evidence and (ii) anchors, i.e., key intermediate entities/relations that can be grounded to graph nodes, from the retrieved subgraph. For example, in Figure 10, for the query *"What is the name of the first Harry Potter novel?"*, the extracted rationale highlights `J.K. Rowling` as a critical intermediate concept, which serves as an anchor. These outputs support dual alignment: the anchors **node-level alignment** for structure-aware pruning, while rationale guides supervise **graph-level alignment** to better ground language representations to graph.

3.2 Guided Dual Alignment

Node-level Alignment. We propose an Aligner module that utilizes the extracted anchors to identify and align relevant nodes within the graph, effectively pruning redundant nodes and edges by filtering out unrelated information at the node level. Specifically, we employ a GNN (e.g., GraphTransformer (Shi et al., 2020) or GAT (Veličković et al., 2018)), to encode the structural information of the graph. The GNN produces node-level embeddings $\boldsymbol{n}_g$ based on the input graph:

$$\boldsymbol{n}_g = \text{GNN}(\mathcal{G}) \in \mathbb{R}^{|\mathcal{V}| \times d}, \tag{1}$$

where $|\mathcal{V}|$ is the number of node and d is feature dim. For the anchors, we employ SBERT to encode the textual description into a embedding $\boldsymbol{r}_s$ that captures its semantic meaning:

$$\boldsymbol{r}_s = \text{SBERT}(t_{\text{anchor}}) \in \mathbb{R}^{d_s}. \tag{2}$$

We concatenate the embedding of each node in the subgraph with the query embedding. The concatenated embeddings are then passed through an MLP module, which generates a predicted importance score for each node $\boldsymbol{p}_{\text{prediction}}$. These scores are transformed into probability distributions using the softmax function, which produces importance scores for both the prediction and the anchor.

$$\boldsymbol{p}_{\text{prediction}} = \text{Softmax}(\text{MLP}([\boldsymbol{n}_g, \boldsymbol{q}^{\text{expand}}])) \in \mathbb{R}^{|\mathcal{V}|}, \tag{3}$$

$$\boldsymbol{p}_{\text{anchor}} = \text{Softmax}(\cos(\boldsymbol{n}_t, \boldsymbol{r}_s)) \in \mathbb{R}^{|\mathcal{V}|}, \tag{4}$$

where $[,]$ means concat operation, cos() means cosine similarity, $\boldsymbol{n}_t$ is text embedding of node, $\boldsymbol{q}^{\text{expand}}$ is the query embedding broadcasted across

all nodes. To align the node distribution, we minimize the Kullback–Leibler (KL) divergence (Kullback and Leibler, 1951) between predicted probabilities $\boldsymbol{p}_{\text{prediction}}$ and the probabilities $\boldsymbol{p}_{\text{anchor}}$. The KL divergence loss for a subgraph is given by:

$$\mathcal{L}_{\text{NA}} = \frac{1}{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{V}|} \boldsymbol{p}_{\text{anchor}}(i) \log \frac{\boldsymbol{p}_{\text{anchor}}(i)}{\boldsymbol{p}_{\text{prediction}}(i)}. \quad (5)$$

Optimizing $\mathcal{L}_{\text{NA}}$ enables effective alignment of relevant knowledge.

Graph-level Alignment. To bridge the representation gap between graph structures and textual descriptions, our Aligner module treats the text representation derived from rationale chains as the target label, aligning the graph and text embeddings to encourage semantic consistency. We apply a mean pooling operation across the node embeddings $\boldsymbol{n}_g$ to obtain a unified graph-level token $\boldsymbol{r}_g$:

$$\boldsymbol{r}_g = \text{POOL}(\boldsymbol{n}_g) = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \boldsymbol{n}_g(v) \in \mathbb{R}^d. \quad (6)$$

To unify the graph and text representations in a shared semantic space, we apply a contrastive loss with in-batch negative sampling. This loss encourages positive pairs (i.e., graph and text embeddings) to have higher similarity, while pushing apart non-matching pairs. Then, a shared-weight MLP layer is further designed to map $\boldsymbol{r}_g$ and $\boldsymbol{r}_s$ to dimension d_t of LLM token embeddings:

$$\hat{\boldsymbol{r}}_s = \text{MLP}(\boldsymbol{r}_s), \quad \hat{\boldsymbol{r}}_g = \text{MLP}(\boldsymbol{r}_g). \quad (7)$$

The contrastive loss for graph-level alignment from $\hat{\boldsymbol{r}}_g$ to $\hat{\boldsymbol{r}}_s$ is defined as:

$$\mathcal{L}_{GA(\hat{\boldsymbol{r}}_g \to \hat{\boldsymbol{r}}_s)} = -\frac{1}{N} \sum_{i=1}^{N} \left[\log \frac{\exp\left(\cos(\hat{\boldsymbol{r}}_g^i, \hat{\boldsymbol{r}}_s^i)/\tau\right)}{\sum_{j=1}^{N} \exp\left(\cos(\hat{\boldsymbol{r}}_g^i, \hat{\boldsymbol{r}}_s^j)/\tau\right)} \right], \quad (8)$$

where, N is the batch size, $(\hat{\boldsymbol{r}}_g^i, \hat{\boldsymbol{r}}_s^i)$ is the i-th positive (graph-text) pair in the batch, τ is a temperature parameter to control the sharpness. Similarly, we can obtains the loss $\mathcal{L}_{GA(\hat{\boldsymbol{r}}_s \to \hat{\boldsymbol{r}}_g)}$ from $\hat{\boldsymbol{r}}_s$ to $\hat{\boldsymbol{r}}_g$. The final representation alignment loss is obtained by:

$$\mathcal{L}_{GA} = \frac{1}{2}\Big(\mathcal{L}_{GA(\hat{\boldsymbol{r}}_s \to \hat{\boldsymbol{r}}_g)} + \mathcal{L}_{GA(\hat{\boldsymbol{r}}_g \to \hat{\boldsymbol{r}}_s)}\Big). \quad (9)$$

To achieve an optimized Graph Aligner, we perform joint optimization of node and graph Alignment. The total loss for the Graph Aligner is defined as: $\mathcal{L}_{\text{Aligner}} = \mathcal{L}_{\text{GA}} + \mathcal{L}_{\text{NA}}$. The parameters of GNN encoder are jointly optimized using the loss $\mathcal{L}_{\text{Aligner}}$ with a specified training step, which reflects degree of alignment. We evaluate the impact of the alignment degree in Figure 3 and the effectiveness in Section 4.5.

3.2.1 Graph Token and Pruning

We apply the trained Graph Aligner to obtain an aligned graph representation token and a pruning subgraph for generation. Note that gold answers are only used to train the Aligner.

Encoding graph token. Given the retrieved subgraph, we leverage the Aligner to produce node embeddings and a pooled graph token $\boldsymbol{r}_g$, which is optimized by $\mathcal{L}_{\text{Aligner}}$ to be consistent with the rationale-derived text space.

Structure-aware pruning. We then use the node-level prediction scores to remove irrelevant evidence. Specifically, we select the top n_{seed} nodes as seed nodes and expand them with their first-order neighbors to form the pruned subgraph $\mathcal{G}_{\text{Aligner}} = (\mathcal{V}_{\text{Aligner}}, \mathcal{E}_{\text{Aligner}})$.

Finally, we concatenate the linearized text of $\mathcal{G}_{\text{Aligner}}$ with the query tokens t_q and feed them into the LLM embedding layer to obtain token embeddings $\boldsymbol{r}_t$. We fuse $\boldsymbol{r}_t$ with $\boldsymbol{r}_g$ to condition answer generation:

$$p_\theta(Y \mid t_q, \mathcal{G}_{\text{Aligner}}) = \prod_{i=1}^{m} p_\theta(y_i \mid [\boldsymbol{r}_g, \boldsymbol{r}_t], y_{<i}). \quad (10)$$

For efficiency, we fine-tune the generator with parameter-efficient methods such as LoRA (Hu et al., 2021).

4 Experiments

4.1 Experimental Settings

Datasets. Following G-Retriever (He et al., 2024), we conducted experiments on the GraphQA benchmark (He et al., 2024), which includes ExplaGraphs (commonsense reasoning), SceneGraphs (scene graph understanding), and WebQSP (knowledge graph reasoning). The dataset statistics are summarized in Appendix B.1.

Baselines. We group baselines into four categories. (1) **Inference-only** methods leverage frozen LLMs by taking textualized graph and query as input, including Zero-shot, Zero-CoT (Wei et al., 2022), CoT-BAG (Wang et al., 2024), KAPING (Baek et al., 2023), and ToG (Sun et al.).

Method	ExplaGraphs	SceneGraphs	WebQSP		
	Accuracy↑	Accuracy↑	F1↑	Hit@1↑	Accuracy↑
Inference-only					
Zero-shot‡	0.5650	0.3974	-	0.4106	-
Zero-CoT‡	0.5704	0.5260	-	0.5130	-
CoT-BAG‡	0.5794	0.5680	-	0.3960	-
KAPING‡	0.6227	0.4375	-	0.5264	-
ToG	0.7664± 0.0348	0.6194±0.0460	0.3014± 0.0245	0.5871± 0.0258	0.3719±0.0183
Raw Fine-tuning					
Prompt tuning	0.5763±0.0243	0.6341±0.0024	0.2652±0.0049	0.4807±0.0055	0.2827±0.0073
LoRA	0.8538±0.0353	0.7862±0.0031	0.4445±0.0058	0.6505±0.0068	0.4479±0.0091
Reranker-based					
GTE-base	0.8557±0.0144	0.8556±0.0095	0.5378±0.0044	0.7373±0.0064	0.5251±0.0052
GTE-large	0.8776±0.0095	0.8592±0.0074	<u>0.5392±0.0013</u>	0.7340±0.0044	0.5374±0.0038
BGE-reranker-base	0.8534±0.0159	0.8577±0.0029	0.5323±0.0052	0.7397±0.0012	0.5254±0.0010
BGE-reranker-large	0.8612±0.0184	0.8644±0.0060	0.5366±0.0045	0.7391±0.0093	0.5401±0.0077
G-RAG	0.8484±0.0174	0.8474±0.0147	0.5181±0.0023	0.7114±0.0113	0.5080±0.0041
G-RAG-RL	0.8478±0.0112	0.8509±0.0142	0.5291±0.0066	0.7167±0.0039	0.5185±0.0026
GNN-based					
GraphToken‡	0.8508± 0.0551	0.4903± 0.0105	-	0.5705±0.0074	-
GNP	0.8704±0.0034	0.8616±0.0096	0.5369±0.0049	0.7391±0.0100	0.5441±0.0046
G-Retriever$^{\dagger}_{PT}$	0.8516±0.0092	0.8131±0.0162	0.4740±0.0049	0.6921±0.0099	0.4740±0.0033
G-Retriever$^{\dagger}_{LoRA}$	0.8705±0.0329	<u>0.8683±0.0072</u>	0.5366±0.0031	0.7366±0.0049	0.5405±0.0031
GRAG$^{\dagger}_{PCST}$	<u>0.8805±0.0050</u>	0.8561±0.0052	0.5355±0.0049	<u>0.7485±0.0104</u>	<u>0.5503±0.0035</u>
`Align-GRAG` (Ours)	**0.8992±0.0124**	**0.8804±0.0106**	**0.5445±0.0041**	**0.7626±0.0063**	**0.5700±0.0039**
Improvement (△)	+2.12%*	+1.39%	+1.68%	+1.88%*	+3.58%*

Table 1: Performance comparison using the same retrieval settings and generator for all methods. The table reports the mean and standard deviation results across three random seeds. For methods marked with '†', we reproduce the results on WebQSP to report F1 and Accuracy. Results for methods marked with '‡' are taken directly from He et al. (2024). The best results are highlighted in **bold**, and the second-best results are <u>underlined</u>. 'Improvement (△)' represents the gain over second-best baseline. '*' indicates statistically significant improvements (i.e., two-sided t-test with $p < 0.05$) over second-best baseline. ↑: higher is better.

(2) **Raw Fine-tuning** adapts LLM via parameter-efficient tuning, covering Prompt Tuning (Lester et al., 2021) and LoRA (Hu et al., 2021). (3) **Reranker-based** baselines rerank retrieved documents before feeding them into LLM, including GTE (Li et al., 2023), BGE (Xiao et al., 2024), G-RAG, and G-RAG-RL (Dong et al., 2024). (4) **GNN-based** approaches integrate GNN encoders with LLM embeddings, including GraphToken (Perozzi et al., 2024), GNP (Tian et al., 2024), G-Retriever (He et al., 2024), and GRAG (Hu et al., 2025).

Implementation Details. We use GraphTransformer (Shi et al., 2020), GAT (Veličković et al., 2018), and GCN (Kipf and Welling, 2016) as GNN encoders, and Llama-2-7b-hf, Llama-2-13b-hf and Llama-3.1-8b (Touvron et al., 2023) as generators. For the retrieval process, we use the same retrieval results across all baselines to ensure fair comparisons. Both reranker-based and GNN-based methods apply LoRA for fine-tuning. All methods are compared using the same training hyperparameters where applicable. In our module, we explore two hyperparameters: alignment degree and the number of seed nodes (n_{seed}), with the analysis shown in Figure 9. To extract the anchor and rationale, we employ Qwen3-8B and Llama-3.1-70B-Instruct (Dubey et al., 2024). More details can be found in Appendix B. More detailed empirical analyses, including reasoning-hop analysis, cost analysis, relevance and faithfulness evaluation, sensitivity studies, and generality results, can be found in Appendix C.

4.2 Main Results

We conduct extensive experiments against 18 baselines, and Table 1 shows that **`Align-GRAG`** consistently improves all metrics across three datasets, with a particularly large Accuracy gain on WebQSP over the second-best method due to its alignment-based graph pruning and optimization.

LLM Backbones		Llama-2-7b-hf		Llama-2-13b-hf		Llama-3.1-8B		Qwen3-8B	
GNN	**Method**	*ExplaGraphs* Accuracy↑	*WebQSP* Hit@1↑	*ExplaGraphs* Accuracy↑	*WebQSP* Hit@1↑	*ExplaGraphs* Accuracy↑	*WebQSP* Hit@1↑	*ExplaGraphs* Accuracy↑	*WebQSP* Hit@1↑
GT	GNP	0.8704	0.7391	0.8880	0.7696	0.8892	0.7557	0.8915	0.7602
	G-Retriever	0.8705	0.7366	0.9115	0.7739	0.9014	0.7482	0.9088	0.7535
	Align-GRAG	**0.8992**	**0.7626**	**0.9241**	**0.7789**	**0.9321**	**0.7712**	**0.9365**	**0.7794**
GAT	GNP	0.9061	0.7291	0.8989	0.7676	0.9062	0.7521	0.9104	0.7588
	G-Retriever	0.7960	**0.7414**	0.8953	**0.7737**	0.9111	0.7719	0.9150	0.7760
	Align-GRAG	**0.9151**	0.7309	**0.9151**	0.7573	**0.9214**	**0.7742**	**0.9282**	**0.7815**
GCN	GNP	0.7545	0.7298	0.8682	0.7564	0.8572	0.7495	0.8615	0.7530
	G-Retriever	**0.8592**	0.7352	0.9007	0.7521	0.8512	0.7527	0.8588	0.7592
	Align-GRAG	0.8574	**0.7377**	**0.9152**	**0.7592**	**0.8596**	**0.7573**	**0.8695**	**0.7645**

Table 2: Generalization Analysis on different LLM and GNN backbones.

Compared with **inference-only methods**, which rely purely on LLM reasoning (e.g., Zero-shot) or heuristic KG search (e.g., ToG), **Align-GRAG** benefits from task-aware structural optimization and achieves clearly stronger results. Compared with **raw fine-tuning methods** such as Prompt Tuning and LoRA, **Align-GRAG** further demonstrates that parameter-efficient adaptation alone is insufficient to fully exploit graph structure. Compared with **reranker-based methods** (e.g., gte-large and related variants), which mainly rerank nodes/triples and thus miss richer structural dependencies, **Align-GRAG** achieves superior performance by aligning graph reasoning with LLM-derived reasoning chains as the optimization target. Compared with **GNN-based methods** (e.g., G-Retriever and GNP), which inject graph embeddings via projection or pooling but lack explicit graph–language alignment, **Align-GRAG** more effectively bridges the two representation spaces and yields state-of-the-art performance.

4.3 Generalization Analysis

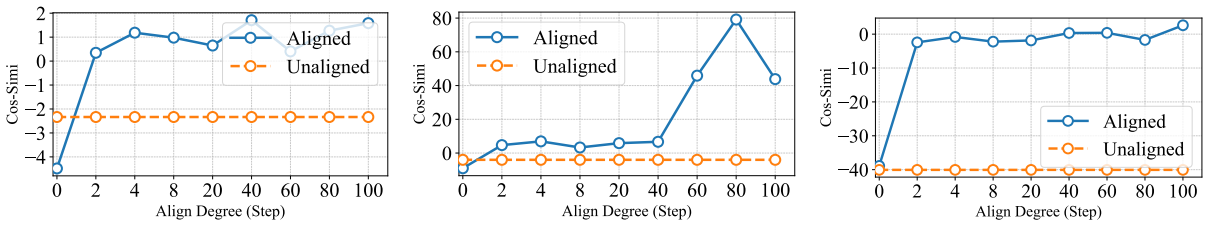
In this section, we evaluate **whether Align-GRAG generalizes across both GNN architectures and LLM scales** on ExplaGraphs (Accuracy) and WebQSP (Hit@1). As summarized in Table 2, across three GNN backbones (GT, GAT, GCN) and multiple LLM families/sizes, **Align-GRAG** outperforms GNP and G-Retriever in the vast majority of configurations, indicating strong robustness to different encoder–reasoner pairings. The largest gains are observed with the GraphTransformer backbone, suggesting that its ability to capture long-range dependencies synergizes with **Align-GRAG** 's alignment objectives to improve retrieval and reasoning, while consistent improvements with GAT further show that alignment complements attention-based aggregation. Moreover, scaling up the LLM generally boosts results—most notably with the weaker GCN encoder—implying that larger LLMs can better leverage aligned graph signals to compensate for limited graph expressivity. The consistent gains on both datasets confirm that **Align-GRAG** transfers well across different settings.

Method	WebQSP		
	F1↑	Hit@1↑	Accuracy↑
Align-GRAG	**0.5445**	**0.7626**	**0.5700**
w/o Graph Alignment	0.5423	0.7578	0.5673
w/o Node Alignment	0.5344	0.7371	0.5339
w/o Both	0.5348	0.7328	0.5216
Random Alignment	0.4617	0.6861	0.4865

Table 3: Ablation study on different alignment strategy.

4.4 Ablation Study

We conduct ablation study on WebQSP with three metrics to assess the contribution of each module. As shown in Table 3, we consider four variants: **(1) w/o Node Alignment** (removing the KL loss and pruning), **(2) w/o Graph Alignment** (removing the contrastive loss and graph token), **(3) w/o Both**, and **(4) Random Alignment** (training the aligner with random labels). Removing Node Alignment causes the largest degradation across all metrics, confirming the importance of node-level alignment and pruning. Disabling Graph Alignment also reduces performance, and removing both modules further amplifies the drop, underscoring the necessity of dual alignment for suppressing irrelevant knowledge and mitigating the graph language representation gap. Finally, Random Alignment performs worst, indicating that effective alignment requires meaningful supervision; our LLM-extracted anchor and rationale provide such a useful signal.



(a) Align with Query (b) Align with Rationale (c) Align with Textualized Graph

Figure 3: Graph-level Alignment Analysis: The cosine similarity score between graph embeddings and language embeddings (aligned using the aligner module vs. the unaligned setting).

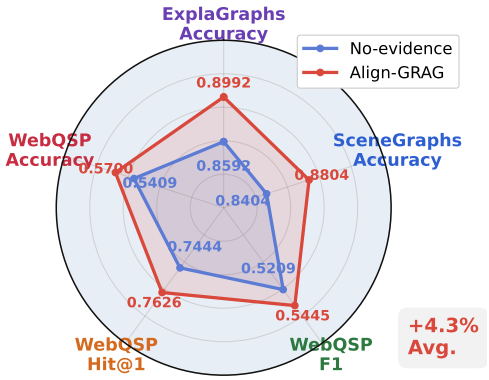


Figure 4: Quantitative Comparison of Different anchor and rationale Quality.

Method	# Tokens	Infer Time	Hit@1
No Pruning			
Non-Retriever	100626.28 (387.02%)	OOM	-
BM25	2569.57 (99.72%)	17:24 min	0.4394
Contriever	2593.28 (100.01%)	17:37 min	0.4429
G-Retriever	2576.66 (100.00%)	17:28 min	0.4502
Node Pruning with Different Degree			
`Align-GRAG`			
$n_{\text{seed}} = 4$	496.54 (19.27%)	3:31 min	0.4299↓
$n_{\text{seed}} = 6$	698.54 (27.11%)	4:45 min	**0.4527**↑
$n_{\text{seed}} = 8$	905.52 (35.14%)	6:55 min	**0.4699**↑
$n_{\text{seed}} = 10$	1120.75 (43.50%)	8:27 min	**0.4785**↑
$n_{\text{seed}} = 15$	1546.90 (60.04%)	11:50 min	**0.4914**↑

Table 4: Evaluation of Efficiency with and without node pruning for inference time. ↑ indicates results better than G-Retriever, while ↓ indicates worse results. (xx%) represents the percentage of tokens relative to G-Retriever. OOM means out of memory.

4.5 Evaluation of Graph-level Alignment

In this section, we empirically verify whether the aligner can bridge the representation gap. Specifically, we measure cosine similarity between graph embeddings (unaligned vs. aligned via our contrastive loss) and language embeddings of the query, rationale, and textualized graph on the test set; Figure 3 shows a clear upward trend in similarity as alignment proceeds, with aligned embeddings consistently surpassing unaligned ones after sufficient training, indicating that the aligner effectively narrows the graph–language representation gap. Meanwhile, as discussed in Section D.2, the benefit saturates beyond a certain point, since overly strong alignment can distort original graph information and hurt accuracy.

4.6 Quality Analysis

We investigate **whether the quality of extracted anchors and rationale directly affects QA performance.** We compare `Align-GRAG` with `No-evidence Extraction`, where the LLM generates anchors and a rationale chain using only the question (without accessing the retrieved subgraph evidence). As shown in Figure 4, `Align-GRAG` consistently performs better across datasets and metrics. This gap indicates that evidence-grounded extraction produces more relevant and faithful supervision signals for both node- and graph-level alignment. In contrast, when extraction is detached from graph evidence, the generated chains are more prone to plausible-but-ungrounded reasoning, which weakens pruning and representation alignment, ultimately harming QA.

4.7 Evaluating the Efficiency of Node Pruning

In this section, we evaluate **the efficiency improvements brought by node pruning in Aligner module**. In an inference-only setting on WebQSP, we compare retrieval variants (w/ Aligner with different seed-node choices). To fully utilize resources and memory, we adjusted the batch size to be as large as possible under different average token conditions. For comparison, the BM25 and Contriever methods retrieve triples and ensure that the token consumption is roughly similar to that of G-Retriever. Table 4 shows that with a

Top-K	3	5	10	15	20
G-Retriever	0.6942	0.6996	0.7366	0.7165	0.7145
Align-GRAG	**0.7229**	**0.7285**	**0.7506**	**0.7285**	**0.7248**

Table 5: Hit@1 on WebQSP under different retrieval top-K settings.

moderate number of seed nodes, our pruning preserves core evidence while removing redundant context—**achieves performance comparable to G-Retriever while only utilizing 27.11% tokens**, whereas too few seeds can over-prune and hurt performance, and more seeds retain more useful knowledge to outperform G-Retriever in longer-context regimes, indicating that selective pruning improves both efficiency and effectiveness. Finally, under similar token budgets, BM25 and Contriever lag behind, suggesting that graph-aware retrieval better exploits relational connectivity.

4.8 Comparison on Different Top-K

To further examine the sensitivity to retrieval top-K, we conduct an additional comparison on WebQSP by evaluating Align-GRAG and G-Retriever. As shown in Table 5, Align-GRAG consistently outperforms G-Retriever across all tested values of K, demonstrating that the advantage of our method is stable rather than tied to a particular retrieval budget. Both methods perform best at $K = 10$, suggesting that a moderate number of retrieved subgraphs achieves a better trade-off between evidence coverage and noise. When K is too small, relevant evidence may be missing, while larger K values can introduce more irrelevant context. Nevertheless, the performance margin of Align-GRAG remains clear throughout, indicating that it is robust to retrieval depth and can more effectively exploit retrieved graph evidence.

5 Related Work

Retrieval-Augmented Generation (RAG) combines information retrieval to address issues like hallucination and outdated information (Gao et al., 2023; Zhou et al., 2025; Xu et al., 2024c). By integrating pre-retrieval (e.g., query rewriting), retrieval, post-retrieval (e.g., reranking), and generation (Fan et al., 2024; Fang et al., 2025), RAG (Sui et al., 2025; Zamani and Bendersky, 2024; Yang et al., 2024; Sudhi et al., 2024; Dong et al., 2024) improves the relevance of retrieved documents and enhances response reliability.

Large Language Models on Graph. Graphs model relational data in various domains, and GNNs (Shi et al., 2020; Veličković et al., 2018; Kipf and Welling, 2016) are effective for encoding graph structures. Recent works integrate LLMs with graph learning to enhance graph-based ACL tasks (Huang et al., 2024a; Jin et al., 2024; Jiang et al., 2025; Ren et al., 2024). Approaches range from feeding graph tokens into LLMs (Tang et al., 2024; Perozzi et al., 2024; Chai et al., 2023) to embedding graph neural layers within transformer architectures (Qin et al., 2023; Zhu et al., 2024; Huang et al., 2024b; Wang et al., 2025).

Graph RAG. Graph RAG incorporates graph-structured knowledge to enhance retrieval and reasoning (Baek et al., 2023; Wang et al., 2024; Han et al., 2024; Xu et al., 2024b). It utilizes graph databases (Bollacker et al., 2008; Vrandečić and Krötzsch, 2014) to retrieve triples or subgraphs (Edge et al., 2024; Mavromatis and Karypis, 2024; Hu et al., 2025; Wu et al., 2024). Techniques like graph construction from text (Guo et al., 2024; Fan et al., 2025) and GNN-based encoding (Liu et al., 2025; Jiang et al., 2022; He et al., 2024; Tian et al., 2024) enhance reasoning. Recent approaches leverage LLMs for iterative reasoning over graphs (Jiang et al., 2024; LUO et al.; Sun et al.; Chen et al., 2024). While Graph RAG emphasizes retrieval, it often lacks robust post-retrieval strategies. Our **Align-GRAG** introduces a dual alignment mechanism to better link retriever and generator, improving subgraph-query alignment.

6 Conclusion

In this paper, we study GRAG and identify two key challenges: (i) structure-coupled irrelevant evidence introduced by subgraph expansion and (ii) the structure-reasoning discrepancy between graph encoders and LLM semantics. To address them, we propose **Align-GRAG**, an anchor-and-rationale guided refinement framework. Experiments on three challenging benchmarks demonstrate consistent gains over 18 strong baselines. Moreover, **Align-GRAG** generalizes well across diverse GNN architectures and LLM backbones, showing robust transferability across encoder–reasoner pairings. Notably, its structure-aware pruning reduces input tokens and inference time while maintaining or improving accuracy, making **Align-GRAG** an effective and practical approach for efficient graph-grounded generation.

Limitations

Align-GRAG demonstrates strong empirical performance, robust generalization across encoder–reasoner pairings, and favorable efficiency–accuracy trade-offs across diverse benchmarks. One limitation is that the current framework introduces additional offline training complexity due to anchor/rationale extraction and aligner optimization.

Ethical considerations

Our method reduces hallucination by grounding generation in retrieved graph evidence, but it can still produce misleading answers if retrieval returns incomplete/noisy subgraphs or if extracted anchors/rationales are weakly grounded. Since the pipeline relies on LLMs for extraction and assessment, model biases and errors may propagate to both supervision and evaluation; therefore, outputs should be verified before use in high-stakes settings. What is more, we only use LLMs to polish the writing in this work.

References

Badr AlKhamissi, Millicent Li, Asli Celikyilmaz, Mona Diab, and Marjan Ghazvininejad. 2022. A review on language models as knowledge bases. *arXiv preprint arXiv:2204.06031*.

Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. *arXiv preprint arXiv:2306.04136*.

Daniel Bienstock, Michel X Goemans, David Simchi-Levi, and David Williamson. 1993. A note on the prize collecting traveling salesman problem. *Mathematical programming*, 59(1):413–420.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Ziwei Chai, Tianjie Zhang, Liang Wu, Kaiqiao Han, Xiaohai Hu, Xuanwen Huang, and Yang Yang. 2023. Graphllm: Boosting graph reasoning ability of large language model. *arXiv preprint arXiv:2310.05845*.

Liyi Chen, Panrong Tong, Zhongming Jin, Ying Sun, Jieping Ye, and Hui Xiong. 2024. Plan-on-graph: Self-correcting adaptive planning of large language model on knowledge graphs. *arXiv preprint arXiv:2410.23875*.

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*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Tianyu Fan, Jingyuan Wang, Xubin Ren, and Chao Huang. 2025. Minirag: Towards extremely simple retrieval-augmented generation. *arXiv preprint arXiv:2501.06713*.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6491–6501.

Siyuan Fang, Kaijing Ma, Tianyu Zheng, Xeron Du, Ningxuan Lu, Ge Zhang, and Qingkun Tang. 2025. KARPA: A training-free method of adapting knowledge graph as references for large language model's reasoning path aggregation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24724–24746, Vienna, Austria. Association for Computational Linguistics.

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

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, and 1 others. 2024. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594*.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. Lightrag: Simple and fast retrieval-augmented generation. *arXiv preprint arXiv:2410.05779*.

Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A Rossi, Subhabrata Mukherjee, Xianfeng Tang, and 1 others. 2024. Retrieval-augmented generation with graphs (graphrag). *arXiv preprint arXiv:2501.00309*.

Kai He, Rui Mao, Qika Lin, Yucheng Ruan, Xiang Lan, Mengling Feng, and Erik Cambria. 2023. A survey of large language models for healthcare: from data, technology, and applications to accountability and ethics. *arXiv preprint arXiv:2310.05694*.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *arXiv preprint arXiv:2402.07630*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Xixin Hu, Xuan Wu, Yiheng Shu, and Yuzhong Qu. 2022. Logical form generation via multi-task learning for complex question answering over knowledge bases. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1687–1696.

Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. 2025. GRAG: Graph retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4145–4157, Albuquerque, New Mexico. Association for Computational Linguistics.

Chao Huang, Xubin Ren, Jiabin Tang, Dawei Yin, and Nitesh Chawla. 2024a. Large language models for graphs: Progresses and directions. In *Companion Proceedings of the ACM Web Conference 2024*, WWW '24, page 1284–1287, New York, NY, USA. Association for Computing Machinery.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and 1 others. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*.

Xuanwen Huang, Kaiqiao Han, Yang Yang, Dezheng Bao, Quanjin Tao, Ziwei Chai, and Qi Zhu. 2024b. Can gnn be good adapter for llms? In *Proceedings of the ACM on Web Conference 2024*, pages 893–904.

Pengyue Jia, Derong Xu, Xiaopeng Li, Zhaocheng Du, Xiangyang Li, Xiangyu Zhao, Yichao Wang, Yuhao Wang, Huifeng Guo, and Ruiming Tang. 2024. Bridging relevance and reasoning: Rationale distillation in retrieval-augmented generation. *arXiv preprint arXiv:2412.08519*.

Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, Yang Song, Chen Zhu, Hengshu Zhu, and Ji-Rong Wen. 2024. Kg-agent: An efficient autonomous agent framework for complex reasoning over knowledge graph. *arXiv preprint arXiv:2402.11163*.

Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Unikgqa: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. *arXiv preprint arXiv:2212.00959*.

Xinke Jiang, Ruizhe Zhang, Yongxin Xu, Rihong Qiu, Yue Fang, Zhiyuan Wang, Jinyi Tang, Hongxin Ding, Xu Chu, Junfeng Zhao, and Yasha Wang. 2025. HyKGE: A hypothesis knowledge graph enhanced RAG framework for accurate and reliable medical LLMs responses. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11836–11856, Vienna, Austria. Association for Computational Linguistics.

Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. 2024. Large language models on graphs: A comprehensive survey. *IEEE Transactions on Knowledge and Data Engineering*.

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

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Jinqi Lai, Wensheng Gan, Jiayang Wu, Zhenlian Qi, and S Yu Philip. 2024. Large language models in law: A survey. *AI Open*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*.

Runxuan Liu, Bei Luo, Jiaqi Li, Baoxin Wang, Ming Liu, Dayong Wu, Shijin Wang, and Bing Qin. 2025. Ontology-guided reverse thinking makes large language models stronger on knowledge graph question answering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15269–15284, Vienna, Austria. Association for Computational Linguistics.

Shengchao Liu, Weili Nie, Chengpeng Wang, Jiarui Lu, Zhuoran Qiao, Ling Liu, Jian Tang, Chaowei Xiao, and Animashree Anandkumar. 2023. Multimodal molecule structure–text model for text-based retrieval and editing. *Nature Machine Intelligence*, 5(12):1447–1457.

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

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

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap.

IEEE Transactions on Knowledge and Data Engineering.

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

Yijian Qin, Xin Wang, Ziwei Zhang, and Wenwu Zhu. 2023. Disentangled representation learning with large language models for text-attributed graphs. *arXiv preprint arXiv:2310.18152*.

Nils Reimers and Iryna Gurevych. 2020. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Xubin Ren, Jiabin Tang, Dawei Yin, Nitesh Chawla, and Chao Huang. 2024. A survey of large language models for graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6616–6626.

Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. 2020. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*.

Viju Sudhi, Sinchana Ramakanth Bhat, Max Rudat, and Roman Teucher. 2024. Rag-ex: A generic framework for explaining retrieval augmented generation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 2776–2780, New York, NY, USA. Association for Computing Machinery.

Yuan Sui, Yufei He, Nian Liu, Xiaoxin He, Kun Wang, and Bryan Hooi. 2025. FiDeLiS: Faithful reasoning in large language models for knowledge graph question answering. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 8315–8330, Vienna, Austria. Association for Computational Linguistics.

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*.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2024. Graphgpt: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 491–500.

Yijun Tian, Huan Song, Zichen Wang, Haozhu Wang, Ziqing Hu, Fang Wang, Nitesh V Chawla, and Panpan Xu. 2024. Graph neural prompting with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19080–19088.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

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

Shijie Wang, Wenqi Fan, Yue Feng, Lin Shanru, Xinyu Ma, Shuaiqiang Wang, and Dawei Yin. 2025. Knowledge graph retrieval-augmented generation for LLM-based recommendation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27152–27168, Vienna, Austria. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

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

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 641–649.

Derong Xu, Wei Chen, Wenjun Peng, Chao Zhang, Tong Xu, Xiangyu Zhao, Xian Wu, Yefeng Zheng, Yang Wang, and Enhong Chen. 2024a. Large language models for generative information extraction: A survey. *Frontiers of Computer Science*, 18(6):186357.

Derong Xu, Xinhang Li, Ziheng Zhang, Zhenxi Lin, Zhihong Zhu, Zhi Zheng, Xian Wu, Xiangyu Zhao, Tong Xu, and Enhong Chen. 2025. Harnessing large language models for knowledge graph question answering via adaptive multi-aspect retrieval-augmentation. *Preprint*, arXiv:2412.18537.

Zhentao Xu, Mark Jerome Cruz, Matthew Guevara, Tie Wang, Manasi Deshpande, Xiaofeng Wang, and Zheng Li. 2024b. Retrieval-augmented generation with knowledge graphs for customer service question answering. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2905–2909.

Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. 2024c. Hallucination is inevitable: An innate limitation of large language models. *arXiv preprint arXiv:2401.11817*.

Diji Yang, Jinmeng Rao, Kezhen Chen, Xiaoyuan Guo, Yawen Zhang, Jie Yang, and Yi Zhang. 2024. Im-rag: Multi-round retrieval-augmented generation through learning inner monologues. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 730–740.

Hamed Zamani and Michael Bendersky. 2024. Stochastic rag: End-to-end retrieval-augmented generation through expected utility maximization. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2641–2646.

Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. 2022. Learning on large-scale text-attributed graphs via variational inference. *arXiv preprint arXiv:2210.14709*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, and 1 others. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Tianyang Zhong, Zhengliang Liu, Yi Pan, Yutong Zhang, Yifan Zhou, Shizhe Liang, Zihao Wu, Yanjun Lyu, Peng Shu, Xiaowei Yu, and 1 others. 2024. Evaluation of openai o1: Opportunities and challenges of agi. *arXiv preprint arXiv:2409.18486*.

Yigeng Zhou, Wu Li, Yifan Lu, Jing Li, Fangming Liu, Meishan Zhang, Yequan Wang, Daojing He, Honghai Liu, and Min Zhang. 2025. Reflection on knowledge graph for large language models reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 23840–23857, Vienna, Austria. Association for Computational Linguistics.

Yun Zhu, Yaoke Wang, Haizhou Shi, and Siliang Tang. 2024. Efficient tuning and inference for large language models on textual graphs. *arXiv preprint arXiv:2401.15569*.

Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Haonan Chen, Zheng Liu, Zhicheng Dou, and Ji-Rong Wen. 2023. Large language models for information retrieval: A survey. *arXiv preprint arXiv:2308.07107*.

A Graph Retrieval

Existing RAG methodologies are mainly designed for plain text documents or triplets, where information retrieval occurs independently of the graph structure (Fan et al., 2024; Zhu et al., 2023; Pan et al., 2024). In the retrieval stage, we first utilize an encoder-only language model (e.g., Sentence-BERT (Reimers and Gurevych, 2020)) to encode textual information, including the query t_q, text of each node t_n and edge t_e in the graph, respectively:

$$\boldsymbol{q} = \text{SBERT}(t_q) \in \mathbb{R}^d, \tag{11}$$

$$\boldsymbol{n} = \text{SBERT}(t_n) \in \mathbb{R}^d, \tag{12}$$

$$\boldsymbol{e} = \text{SBERT}(t_e) \in \mathbb{R}^d \tag{13}$$

Then we compute cosine similarity $\text{sim}(\cdot)$ between query embeddings $\boldsymbol{q}$ and embeddings of nodes $\boldsymbol{n}$ and edges $\boldsymbol{e}$. The top-k nodes and edges are selected as the most relevant entities and relations.

$$\mathcal{V}_k = \underset{\boldsymbol{n} \in \mathcal{V}}{\arg\text{topk}} \left(\text{sim}(\boldsymbol{q}, \boldsymbol{n})\right), \tag{14}$$

$$\mathcal{E}_k = \underset{\boldsymbol{e} \in \mathcal{E}}{\arg\text{topk}} \left(\text{sim}(\boldsymbol{q}, \boldsymbol{e})\right) \tag{15}$$

This forms top-k entities and relation sets, denoted as $\mathcal{G}_{\text{retriever}}$. Inspired by He et al. (2024), we further leverage the Prize-Collecting Steiner Tree algorithm (PCST) (Bienstock et al., 1993) to maintain a controlled graph size.

To achieve the PCST algorithm, we assign "prize" to nodes and edges based on their similarity to a given query. Relevance is determined through the ranked sets of cosine similarity, $\mathcal{V}_k$ and $\mathcal{E}_k$, as follows:

$$\text{prize}(n) = \begin{cases} k - i, & \text{if } n \in \mathcal{V}_k \text{ and } n \\ & \text{is the } i\text{-th ranked node} \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

where i is the rank of n in set $\mathcal{V}_k$. Nodes that are not among top k rankings are assigned a prize of zero. The objective of the PCST algorithm is to identify a subgraph that maximizes the total prize

of nodes and edges while minimizing the cost:

$$\mathcal{G}_{\text{retriever}} = \underset{\substack{S \subseteq \mathcal{G}, \\ S \text{ is connected}}}{\arg\max} \left(\sum_{n \in \mathcal{V}_S} \text{prize}(n) + \sum_{e \in \mathcal{E}_S} \text{prize}(e) - \text{cost}(S) \right). \tag{17}$$

where $\mathcal{V}_S$ and $\mathcal{E}_S$ are the sets of nodes and edges in the subgraph S, respectively. The cost of constructing the subgraph is defined as $\text{cost}(S) = |\mathcal{E}| \cdot C_e$, where $|\mathcal{E}|$ is the number of edges, and C_e is a predefined per-edge cost that serves as a regularization parameter to control the subgraph's size. In this way, we can obtain a preliminary retrieved subgraph.

B Experimental Details

B.1 Datasets and Metrics.

Following G-Retriever (He et al., 2024), we conducted experiments on the GraphQA benchmark (He et al., 2024), which includes ExplaGraphs (commonsense reasoning), SceneGraphs (scene graph understanding), and WebQSP (knowledge graph reasoning). The dataset statistics are summarized in Table 6, with a data split of train:validation:test = 6:2:2.

- **ExplaGraphs** is a dataset designed for commonsense reasoning and focuses on generating explanation graphs to predict stances in debates. It provides detailed, explicit, and commonsense-enriched graphs to evaluate arguments that either support or refute a given stance. The primary task is to determine whether an argument supports or opposes a stance.

- **SceneGraphs** is a visual question answering dataset. Each graph provides detailed descriptions of objects, attributes, and relationships within an image. It is designed for tasks that involve spatial understanding and multi-step reasoning. The goal is to answer open-ended questions based on textual descriptions of scene graphs, with Accuracy used as the evaluation standard.

- **WebQSP** is a large-scale, multi-hop knowledge graph question-answering dataset built on a subset of Freebase. It focuses on questions requiring multi-hop reasoning and includes facts within a two-hop neighborhood of the entities mentioned in the question. Since a single question may have multiple answers, the model's performance is evaluated using F1, Hit@1, and Accuracy following Hu et al. (2025).

B.2 Baselines.

To assess the effectiveness of our proposed method, we compare it against four categories of baselines:
Inference-only. This category includes approaches that employ frozen LLMs for question answering, using textual graph descriptions and queries as input. Different prompt strategies are considered, including:

- **Zero-shot**: directly answering questions based on retrieved information,

- **Zero-CoT** (Wei et al., 2022): enhances zero-shot reasoning by appending the phrase "Let's think step by step.", which encourages the model to explicitly generate intermediate reasoning steps.

- **CoT-BAG** (Wang et al., 2024): adds "Let's construct a graph with the nodes and edges first." after providing the textual graph description,

- **KAPING** (Baek et al., 2023): augments LLMs with knowledge graph facts at the prompt level, enabling more accurate QA without requiring model fine-tuning retrieving relevant graph triples.

- **ToG** (Sun et al.): integrates LLMs with KGs to support deep and reliable reasoning through interactive exploration of entities and relations.

Raw Fine-tuning. In this setting, following (He et al., 2024), we fine-tune the LLM via parameter-efficient methods, without applying advanced reranking techniques. Two widely used approaches are included:

- **Prompt Tuning** (Lester et al., 2021): learns a set of continuous prompt embeddings that guide the frozen LLM during inference, enabling lightweight adaptation to specific tasks.

- **LoRA** (Hu et al., 2021): introduces low-rank adaptation layers into transformer weights, achieving efficient fine-tuning with significantly reduced computational and memory costs.

Reranker-based. These baselines employ reranking models to refine the ranking of candidate documents before passing them into the LLM. Representative methods include:

Dataset	#Training	#Validation	#Test	Task Type
ExplaGraphs	1,659	553	554	Commonsense reasoning
SceneGraphs	59,978	19,997	20,025	Visual scene understanding
WebQSP	2,826	245	1,628	Knowledge-based QA

Table 6: Statistics of GraphQA Benchmark (He et al., 2024).

- **GTE** (General Textual Embedding) (Li et al., 2023): developed by Alibaba DAMO Academy, with two variants (gte-base, 109M parameters; gte-large, 335M parameters), trained on large-scale relevance pairs to improve retrieval accuracy,
- **BGE** (BAAI General Embedding) (Xiao et al., 2024): cross-encoder models with two variants (bge-reranker-base, 278M parameters; bge-reranker-large, 560M parameters), optimized for retrieval-augmented generation, offering high accuracy at the cost of efficiency,
- **G-RAG** and **G-RAG-RL** (Dong et al., 2024): reranker models that leverage GNNs to incorporate document connections and semantic cues from abstract meaning representation graphs for context-aware ranking.

GNN-based. This category integrates GNN encoders with LLM embeddings to enhance graph reasoning, including:

- **GraphToken** (Perozzi et al., 2024): encodes graph structures as explicit prompts for LLMs, improving performance on graph reasoning tasks,
- **GNP** (Tian et al., 2024): a plug-and-play framework that applies a GNN encoder with cross-modality pooling to enrich LLMs with knowledge graph information,
- **G-Retriever** (He et al., 2024): introduces the GraphQA benchmark and employs retrieval-augmented generation with soft prompting, improving graph-based question answering, comprehension, and reducing hallucinations,
- **GRAG** (Hu et al., 2025): proposes a divide-and-conquer strategy for efficient textual subgraph retrieval and combines text and graph views within LLMs for graph context-aware generation.

B.3 Implementation Details

In this section, we further present the Implementation Details. To implement baselines, for GNP (Tian et al., 2024), we implemented the Graph Neural Prompting module within our framework, including components such as the GNN encoder, cross-modality pooling, and domain projector. For G-RAG and G-RAG-RL (Dong et al., 2024), we adopted their ranking approach, combining cross-entropy loss with pairwise ranking loss. However, since treating documents as nodes was infeasible in our case, we instead treated entities as nodes and employed the same GNN encoder as our method. We implemented GTE (Li et al., 2023) and BGE-reranker (Xiao et al., 2024) using their official open-source models. Nodes and triples are ranked by query similarity and, as in our method, fed into the LLM for generation. For the GRAG$_{PCST}$ model (Hu et al., 2025), we reproduced its experiments. However, to ensure a fair evaluation, we standardized the retrieval process by using the PCST method for retrieval. This allowed us to directly compare it with their graph encoder approach. Table 7 summarizes the hyperparameter settings for all datasets used in our experiments.

Hyperparameter	Value
Number of Seed Nodes	25
Align Degree (Step)	60
Top-K Retrieval	10
Batch Size	8
Max Epochs	20
Learning Rate	1×10^{-5}
GNN Layers	4
GNN Hidden Dim	1024
LoRA Rank	8
LoRA Alpha	16
LoRA Target Modules	`q_proj, v_proj`

Table 7: Hyperparameter settings used in our experiments.

C Additional Empirical Analysis

C.1 Analysis by Reasoning Hop

To directly examine the effect of reasoning depth, we analyze hop-count distributions where hop an-

Dataset	1-hop	2-hop	≥3-hop
WebQSP	65.49%	34.51%	0.00%
CWQ	40.91%	38.34%	20.75%

(a) Hop-count distribution for datasets with available or derivable hop annotations.

Method	**WebQSP**		**CWQ**		
	1-hop	2-hop	1-hop	2-hop	≥3-hop
G-Retriever	0.7665	0.6799	0.6125	0.5583	0.3491
Align-GRAG	**0.7926**	**0.7058**	**0.6374**	**0.5785**	**0.3750**

(b) Performance comparison across different hop-count buckets.

Table 8: Hop-count distribution and performance comparison across different reasoning-hop buckets.

notations are available and further report performance by hop bucket. Specifically, WebQSP provides derivable hop information from its annotated reasoning structure, while SceneGraphs and ExplaGraphs do not provide reliable hop labels or canonical reasoning paths, making such analysis infeasible on those datasets; therefore, we report hop statistics on WebQSP and additionally introduce ComplexWebQuestions (CWQ), which contains questions with reasoning chains of up to four hops, to evaluate performance in more challenging multi-hop settings. As shown in Table 8a, WebQSP mainly consists of 1-hop and 2-hop questions, whereas CWQ contains a substantial portion of ≥3-hop cases, making it a more suitable benchmark for testing deeper reasoning ability. The results in Table 8b show that Align-GRAG consistently outperforms G-Retriever across all hop buckets on both datasets, improving Hit@1 from 0.7665 to 0.7926 and from 0.6799 to 0.7058 on 1-hop and 2-hop WebQSP questions, respectively, while also yielding clear gains on CWQ, including the ≥3-hop subset (0.3750 vs. 0.3491). These findings indicate that the advantage of Align-GRAG is not limited to relatively simple reasoning cases, but generalizes to more challenging multi-hop questions, supporting the effectiveness and robustness of our alignment-and-pruning design.

C.2 Evaluation of Relevance and Faithfulness

We evaluate the extracted anchors and rationale chains for (i) relevance to the question answer context and (ii) faithfulness to the provided evidence, following an LLM-as-a-judge protocol (Gu et al., 2024). We employ GPT-4o, Gemini-2.5-Flash, and human annotators, and randomly sample 100 instances. The evaluation prompt is shown in Figure 12, and the results are reported in Figure 5. Overall, the summaries demonstrate strong quality. Relevance reaches 90%/92%/92% and faithfulness reaches 87%/89%/90% under GPT-4o, Gemini, and human evaluation, respectively, indicating that the generated chains are largely aligned with the QA context while remaining grounded in the given evidence. The close agreement between LLM-based and human judgments further supports the reliability of the proposed evaluation.

C.3 Cost Analysis

To make the computational cost of Align-GRAG explicit, we report a wall-clock breakdown on the WebQSP test set, where all numbers are measured as average wall-clock time per query (total time divided by the number of queries) on 2×A100 GPUs, using LLaMA2-7B as the generator and Qwen3-8B as the extractor. As shown in Table 9, anchor and rationale extraction incurs 0.616 s/query but is performed only once offline during training, while graph retrieval remains lightweight at 0.156 s/query. The offline training stage, including aligner training and generator fine-tuning with LoRA, takes 7.9 minutes per epoch. More importantly, at deployment, the pipeline reduces to retrieval and QA inference, where Align-GRAG achieves a substantially lower end-to-end inference cost than G-Retriever, requiring only 0.311 s/query compared with 0.644 s/query under the same generator and hardware setting. These results indicate that although Align-GRAG introduces an additional one-time offline supervision step, it yields recurring inference-time efficiency gains together with improved QA effectiveness, making the overall framework practically attractive in real-world deployment scenarios where repeated online queries dominate the total cost.

C.4 Effect of Explicit Node-Level Alignment

We conduct an additional experiment on WebQSP by integrating our node-level alignment module into G-Retriever while keeping the rest of its pipeline unchanged. Unlike G-Retriever, which primarily encodes the retrieved subgraph into graph embeddings for injection into the generator, our node-level alignment explicitly optimizes fine-grained node representations using anchor-guided supervision derived from evidence-bounded an-

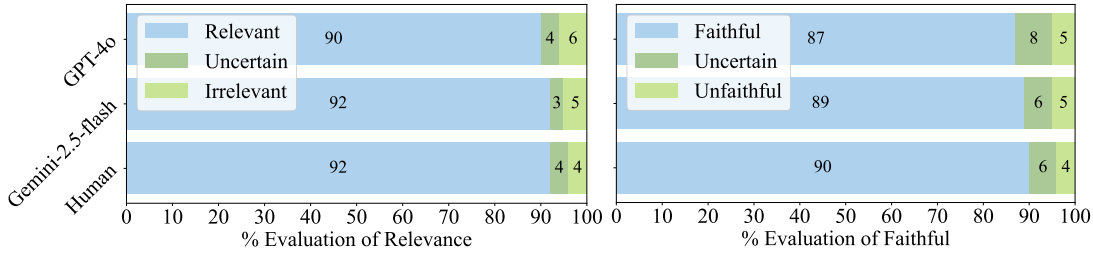


Figure 5: Evaluating Relevance and Faithfulness of Reasoning Chain.

Stage	**When used**	**Model**	**Time**
Anchor/Rationale Extraction	Offline	Qwen3-8B	0.616 s/query
Graph Retrieval	Online + Offline	SBERT + PCST	0.156 s/query
Aligner Training + Generator Fine-tuning (LoRA)	Offline (training)	Graph Aligner + LLaMA2-7B	7.9 min/epoch
Align-GRAG (Aligner + pruning + QA generation)	Online (inference)	LLaMA2-7B	**0.311 s/query**
G-Retriever (baseline)	Online (inference)	LLaMA2-7B	0.644 s/query

Table 9: Cost breakdown on the WebQSP.

Method	Hit@1	Accuracy	F1
G-Retriever	0.7366	0.5405	0.5366
G-Retriever w/ Node Alignment	**0.7521**	**0.5618**	**0.5394**

Table 10: Effect of incorporating the proposed node-level alignment module into G-Retriever on WebQSP.

chors and rationales, thereby reducing the representation gap at the node granularity and naturally enabling structure-coupled pruning of irrelevant nodes. As shown in Table 10, adding this module consistently improves all evaluation metrics. These results demonstrate that explicit node-level alignment contributes additional modeling power beyond G-Retriever's original design, and that the proposed anchor-guided alignment and pruning mechanism is a substantive improvement rather than a reformulation of existing graph-augmented retrieval methods.

C.5 Sensitivity to the Extraction Model

Method	F1	Hit@1	Accuracy
G-Retriever	0.5366	0.7366	0.5405
Extraction model for Align-GRAG			
Llama-3.1-70B-Instruct	**0.5445**	**0.7626**	**0.5700**
Qwen3-8B	0.5418	0.7589	0.5677

Table 11: Sensitivity analysis of the extraction model on WebQSP.

To evaluate whether Align-GRAG depends on a strong extraction model, we conduct a controlled study on WebQSP by replacing the extractor used for anchor and rationale extraction from Llama-3.1-70B-Instruct with the much smaller Qwen3-8B, while keeping the retrieval results, aligner training pipeline, generator, and all other settings unchanged. As shown in Table 11, using Qwen3-8B yields an F1 of 0.5418, a Hit@1 of 0.7589, and an Accuracy of 0.5677, which remains clearly better than the G-Retriever baseline and only slightly below the version using Llama-3.1-70B-Instruct. This small performance gap indicates that the effectiveness of Align-GRAG does not primarily rely on a very large extractor; instead, the gains are largely preserved because the proposed alignment mechanism can still leverage reasonably good anchors and rationales produced by a compact model. Moreover, since extraction is performed only offline during aligner training, adopting a smaller extractor can reduce training-time cost without introducing any additional inference overhead at test time.

C.6 Generality with Closed-Source LLMs via Node Pruning

Although graph-token injection in Align-GRAG requires access to the LLM embedding layer, we further investigate a practical black-box setting in which only the Node-level Alignment module is used before generation. Specifically, on WebQSP, we apply the proposed structure-aware node pruning module to obtain a compact textualized subgraph, and then feed this pruned context into different LLMs used purely as generators, including both open-source and closed-source models, with-

Training-Free Settings	# Tokens	Llama2-7B	gpt-4o-mini	gpt-5-mini	gemini-2.5-flash
Contriever	2593.28	0.4429	0.5124	0.6037	0.6189
G-Retriever	2576.66	0.4502	0.5248	0.6315	0.6494
Align-GRAG (Node Pruning)	**1546.90**	**0.4914**	**0.5639**	**0.6731**	**0.6867**

Table 12: Node-pruning results on WebQSP with different LLMs used purely as generators.

out any embedding access or graph-token fusion; all other settings are kept unchanged. As shown in Table 12, Align-GRAG with node pruning consistently achieves the best Hit@1 across all generators while substantially reducing the input length from over 2.5K tokens to 1546.90 tokens, corresponding to an approximately 40% reduction in context length. More importantly, the performance gains remain stable when replacing the generator with closed-source models such as gpt-4o-mini, gpt-5-mini, and gemini-2.5-flash, demonstrating that even without embedding-level graph integration, the pruning module of Align-GRAG remains drop-in compatible with black-box LLMs and provides simultaneous benefits in both efficiency and answer accuracy.

C.7 Sensitivity to Graph-Level Readout Functions

Method	Align-GRAG (Accuracy)
Mean pooling	0.8992
Max pooling	0.8963
Attention pooling	**0.9057**

Table 13: Effect of different graph-level readout functions in the contrastive alignment component on ExplaGraphs.

To further examine the sensitivity of our contrastive alignment module to the graph-level readout function, we conduct an additional experiment on ExplaGraphs and replace the graph-level pooling operation in the alignment component with three alternative strategies, namely mean pooling, max pooling, and attention pooling, while keeping all other training and inference settings unchanged. As shown in Table 13, Align-GRAG achieves consistently strong performance across all three variants, with accuracies of 0.8992, 0.8963, and 0.9057 for mean, max, and attention pooling, respectively. These results suggest that the proposed alignment mechanism is generally robust to the specific choice of readout function and does not rely on a particular pooling design to remain effective. At the same time, attention pooling yields the best result, indicating that a more expressive readout function can better preserve informative graph signals and produce a higher-quality graph representation for graph-level alignment.

D Impact of Hyperparameters

In this section, we analyze **how hyperparameters affect performance**, including: the Number of Seed Nodes n_{seed}, Align Degree (training step) and Top K retrieval.

D.1 Impact of Seed Nodes

We analyze the impact of the **number of seed nodes** (n_{seed}) on model performance. The experiments are conducted on the WebQSP dataset, where we evaluate the Hit@1, F1 and Accuracy metrics. From the experiments in Figure 6, we observe that the Hit@1, F1 and Accuracy performance peaks when the number of seed nodes is set to 25. Beyond this point (from 25 to 30), the performance starts to decline. This indicates that including too many nodes introduces a significant amount of irrelevant knowledge, which negatively impacts the model. Our pruning strategy effectively eliminates irrelevant knowledge to enhance model performance. On the other hand, when the number of seed nodes is as low as 5, the performance is considerably poor. This suggests that excessive pruning removes crucial knowledge, which is detrimental to performance. This highlights a trade-off: pruning reduces noise and improves performance, but over-pruning leads to the loss of essential knowledge.

D.2 Impact of Align Degree

This section examines how the **Align Degree** (number of training steps for Aligner module) influences model performance. As shown in Figure 7, we evaluate the Hit@1, F1 and Accuracy metrics on the WebQSP dataset. From the experimental curve of Align Degree, we observe that the Hit@1, F1 and Accuracy metrics peak at around 60 epochs be-

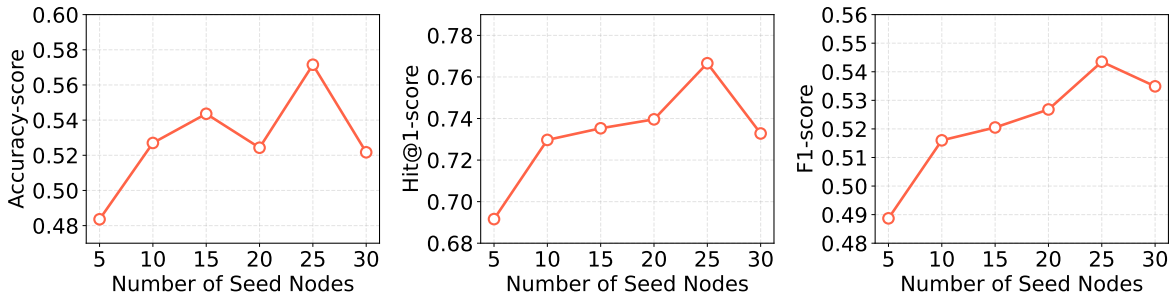


Figure 6: Hyperparameters Analysis of the Number of seed nodes.

fore declining. This indicates that, as training progresses, bridging the representation gap between the graph and language helps the LLM better understand graph data. However, excessive training may lead to overfitting, which disrupts graph information and ultimately causes a drop in performance. This suggests that there is an optimal point in training where the graph alignment is most effective, and training beyond this point can be detrimental.

D.3 Joint Impact of Seed Nodes and Align Degree

To further investigate the interaction between the **number of seed nodes** (n_{seed}) and the **Align Degree** (aligner training steps), we report a two-dimensional performance landscape in Figure 8. Each heatmap cell corresponds to one hyperparameter pair (n_{seed}, Align Degree), evaluated on WebQSP using Accuracy, Hit@1, and F1. Overall, the three heatmaps exhibit a highly consistent trend: performance is maximized in a moderate regime, centered around $n_{\text{seed}} \approx 25$ and Align Degree ≈ 60. When n_{seed} is small (e.g., 5), all metrics remain low across nearly all Align Degrees, indicating that **insufficient retrieved evidence cannot be compensated** by longer aligner training. In contrast, when n_{seed} becomes large (e.g., 30), the performance degrades, and the degradation is more pronounced under larger Align Degrees, suggesting that excessive retrieved noise may be amplified by over-training the aligner and can eventually harm the graph-language alignment quality. Similarly, for a fixed n_{seed} in the mid-to-high range, increasing Align Degree improves results initially but starts to decline after the peak region, matching the **overfitting** pattern observed in the 1D analysis. These results confirm that the best performance requires balancing evidence coverage and noise control, where moderate n_{seed} ensures adequate reasoning support while the aligner is trained sufficiently (but not excessively) to bridge the representation gap.

D.4 Impact of Top K retrieval

Figure 9 illustrates the impact of varying the **Top K retrieval** of entities and relations on model performance across Hit@1, F1 and Accuracy. By analyzing the trends in the graphs, we can derive insights into the effect of Top K on model performance. All three metrics show a similar trend: performance improves as K increases, peaks at K = 10, and then declines. This suggests that K = 10 strikes the optimal balance between retrieving relevant results and avoiding noise. Smaller K values may miss relevant information, while larger K values dilute relevance, reducing precision and retrieval quality. These findings highlight the importance of selecting an appropriate K value to maximize performance in retrieval-based systems.

E Case study

In this case study, we investigate how summarization enhances the discovery of critical intermediate nodes within anchors and rationales, as illustrated in Figure 10. Using well-crafted prompts in conjunction with a graph database, we show that concise, targeted summaries can surface pivotal entities—such as 'J.K. Rowling'—that bridge the semantic gap between an initial query and the final answer. Our analysis evaluates the extent to which summarization reliably extracts these key connectors, thereby enabling coherent multi-hop reasoning that would otherwise be obscured within the raw graph structure. We further demonstrate that, without the interpretive layer provided by summarization, the underlying graph database lacks the necessary contextual signals to assemble the correct reasoning path end-to-end, underscoring the value of summarization as a catalyst for interpretable and effective knowledge retrieval.

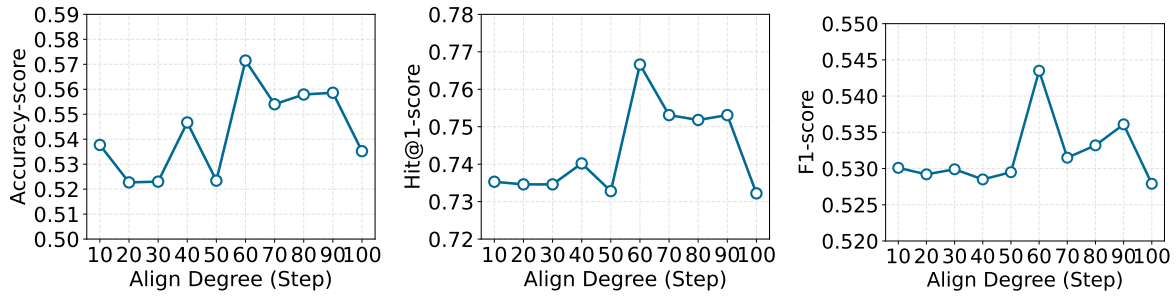


Figure 7: Hyperparameters Analysis of Align degree.

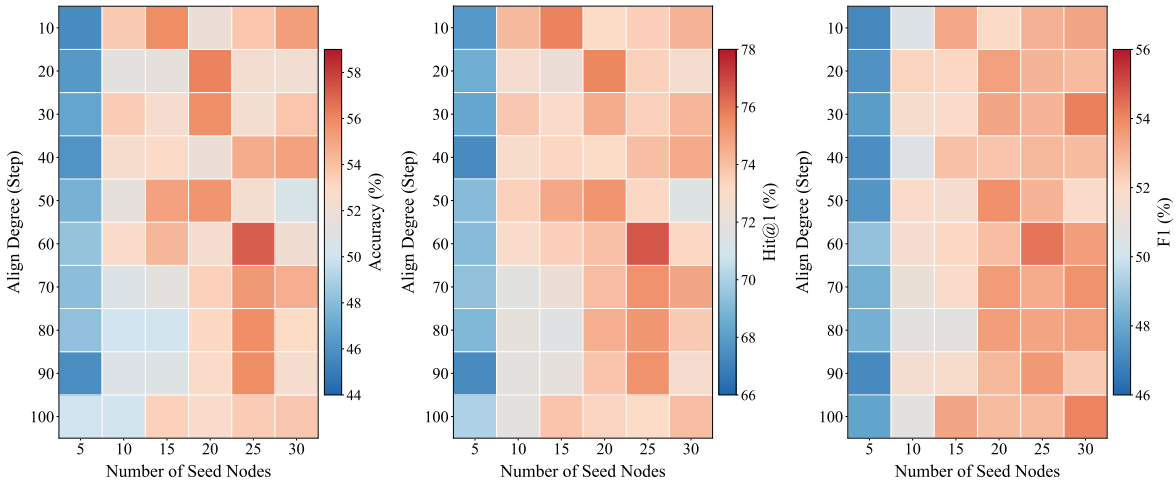


Figure 8: Hyperparameter interaction heatmaps on WebQSP.

F LLM Prompts

We introduce three prompts: the Extraction Prompt, the Generator Prompt, and the Relevance and Faithfulness Prompt. Each serves a specific function in enhancing the performance and evaluation of LLMs. Below is a detailed explanation of their purposes and applications, as illustrated in Figure 11, Figure 13, and Figure 12.

The Extraction Prompt is specifically designed to generate a comprehensive and cohesive anchor and rationale by synthesizing three key inputs: a given question, its corresponding answer, and the relevant textualized graph data. The goal of this prompt is to create a logical connection between the question and the answer, while also integrating information extracted from the provided graph data. This ensures the output not only answers the question but also demonstrates an understanding of the contextual relationships within the graph. By combining these elements, the prompt facilitates the generation of a rich, informative summary that serves as a bridge between structured and unstructured data.

The Relevance and Faithfulness Prompt is an evaluation-focused tool that serves two distinct but complementary purposes:

- Relevance Evaluation: This component assesses whether the anchor and rationale provided is relevant to the given question. A relevant anchor and rationale must contain information that contributes meaningfully to answering the question, even if the contribution is partial. The model is required to return one of two labels: Relevant or Irrelevant, with no additional commentary.

- Faithfulness Evaluation: This component measures the faithfulness of anchor and rationale in responding to the question, which should accurately represent the information in the context, avoid fabrications, and not contradict the provided data. Similarly, the model outputs one of two labels: Faithful or Not Faithful. This dual evaluation ensures the generated responses are both accurate and contextually grounded.

The Generator Prompt is tailored to the dataset being used, with its structure varying depending

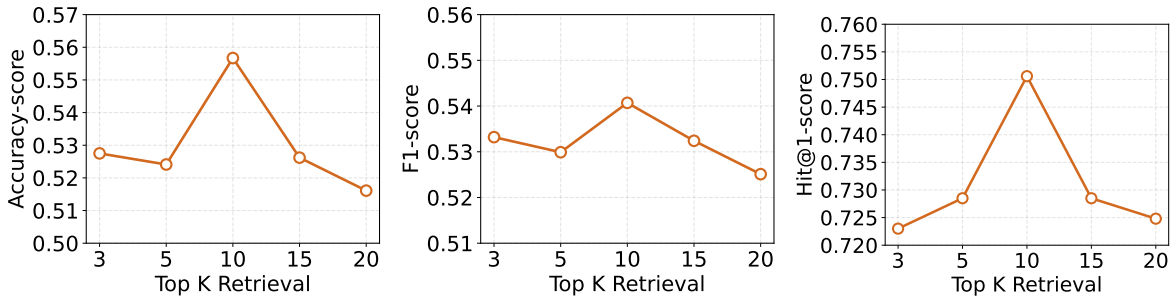


Figure 9: Effect of Top K retrieval.

on the specific requirements of the dataset. For WebQSP and SceneGraphs, the prompt focuses on answering a question by leveraging the provided textualized graph data. This encourages the model to extract and synthesize relevant information from graph representations to deliver a concise and accurate response. For ExplaGraphs, the prompt shifts its focus to reasoning. It requires the model to evaluate the relationship between two arguments (e.g., whether they support or counter each other), based on the provided textualized graph data. The model outputs a single-word response, either support or counter, reflecting the nature of the relationship.

Case Study of Extracted Rationale & Anchors

Extraction_Prompt
Question: what is the name of the first harry potter novel?
Answer: harry potter and the philosopher's stone

Graph DataBase:

```
node_id,node_attr
0,harry potter and the chamber of secrets
1,harry potter and the philosopher's stone
3,j. k. rowling
7,harry potter and the half-blood prince
9,harry potter and the prisoner of azkaban
11,harry potter and the goblet of fire
16,harry potter
24,harry potter and the deathly hallows
57,fiction
59,harry potter literary series
76,professor severus snape
98,fantasy
......
src,edge_attr,dst
16,freebase.equivalent_topic.equivalent_domain,91
91,freebase.domain_profile.featured_views,806
199,book.written_work.subjects,455
59,book.book_subject.works,199
3,book.author.works_written,670
24,media_common.adapted_work.adaptations,46
59,book.book_subject.works,371
0,book.book.characters,325
24,book.book.genre,57
24,book.book_edition.book,24
9,book.book.genre,98
190,book.book_edition.book,24
1,book.book.genre,224
24,book.book_edition.book,24
7,book.book.genre,98
59,book.literary_series.fictional_universe,16
478,book.book_edition.book,24
......
```

Extracted Rationale:

- The question asks for the name of the **first harry potter novel**, and the graph includes a specific book node labeled **harry potter and the philosopher's stone**.
- The graph's provided summary states that **j. k. rowling** wrote the first Harry Potter novel, explicitly naming it as **harry potter and the philosopher's stone**.
- Therefore, using the graph evidence, the first Harry Potter novel is **harry potter and the philosopher's stone**, matching the given answer.

Extracted Anchors:

- **entity**: **harry potter and the philosopher's stone**
- **entity**: **j. k. rowling**
- **relation**: **book.author.works_written**

Figure 10: The case for extracting the anchors and rationales. The extracted rationale chain identifies the critical intermediate node `j. k. rowling` and links it to the answer `harry potter and the philosopher's stone`.

Anchor and Rationale Extraction Prompt

You are a helpful assistant for **anchor and rationale extraction** from graph evidence.
Inputs: a question, its answer, and a textualized graph database (nodes/edges described in text).
Goal: extract (1) a *rationale chain* that links the question to the answer using graph evidence, and (2) a small set of *anchors* (key intermediate entities/relations) that can be grounded to graph nodes/edges.
Definitions:

- **Rationale chain**: an ordered list of short reasoning steps. Each step should mention the evidence used and move closer to the answer.
- **Anchors**: reasoning-critical intermediate entities/relations that appear in the graph database and are necessary for the reasoning.

Constraints:

- Use only information supported by the provided graph database and the given answer.
- Anchors must be **verbatim spans** from the graph database (copy exact surface forms).
- Keep each rationale step concise.

Output format (follow exactly):

1. **RationaleChain**: a numbered list of steps (3–6 steps).
2. **Anchors**: a bullet list of anchors. For each anchor, provide its type (`entity` or `relation`) and the copied span.

Question: {question}
Answer: {Answer}
Graph DataBase: {Textualized Graph}
Now produce the output.

Figure 11: Prompt for anchor and rationale extraction.

Relevance and Faithfulness Prompt

Evaluation of Relevance:
Evaluate the relevance of the anchor and rationale in answering the QUESTION. The relevant anchor and rationale contain information that helps answer the question, even if partially. Return one of the following labels: 'Relevant', or 'Irrelevant' without any additional response.

Evaluation of Faithfulness:
Evaluate the following anchor and rationale for faithfulness in answering the QUESTION. A faithful response should include information that helps answer the question, even if partially, avoid inventing new details, and not contradict the context. Return one of the following labels: 'Faithful' or 'Not Faithful' without any additional response.

Figure 12: Evaluation of Relevance and Faithfulness Prompt.

Generator Prompt

Prompt for WebQSP and SceneGraphs datasets:
Textualized Graph: {Textualized Graph}.
Please answer the given question. Question: {question}
Answer:

Prompt for ExplaGraphs dataset:
Textualized Graph: {Textualized Graph}.
Argument 1: {arg1}
Argument 2: {arg1}
Question: Do argument 1 and argument 2 support or counter each other? Answer in one word in the form of 'support' or 'counter'.
Answer:

Figure 13: Prompt for Generator.