# SDRAG: Sub-question Dependency-Aware Retrieval-Augmented Generation for Multi-hop Question Answering

**Anonymous ACL submission**

## Abstract

Multi-hop question answering (QA) requires integrating evidence across multiple documents and reasoning over interdependent intermediate steps, posing significant challenges for retrieval-augmented generation (RAG) systems. While recent RAG-based approaches interleave retrieval and reasoning to improve factual grounding, they often struggle with complex multi-hop questions due to *retrieval–reason mismatch*, *ignored dependencies among sub-questions*, and *error propagation during decomposition*. In this work, we propose **SDRAG**, a sub-question dependency-aware RAG framework that explicitly models the intrinsic logic of multi-hop questions as a dependency graph. SDRAG first decomposes the question into fine-grained sub-questions and explicitly models their dependency relations, then refines the graph by pruning unreliable or redundant nodes, and finally traverses the pruned graph to coordinate retrieval and reasoning. To improve graph construction quality, we further introduce a supervised fine-tuning strategy that independently trains task-specific LoRA adapters for decomposition, dependency prediction, and pruning. Experiments on three common multi-hop QA benchmarks show that SDRAG consistently outperforms strong RAG-based baselines, while additional analyses demonstrate the effectiveness of dependency modeling, graph pruning, and fine-tuning. Code is available at `https://anonymous.4open.science/r/SubDepRAG-72C6`.

## 1 Introduction

Multi-hop Question Answering (MHQA) requires models to retrieve evidence across multiple documents and synthesize it through multi-step reasoning (Yang et al., 2018; Mavi et al., 2024), serving as a rigorous benchmark for evaluating the reasoning capabilities of AI systems.

Large Language Models (LLMs) have demonstrated remarkable reasoning capabilities (Achiam et al., 2023; Ke et al., 2025), but they remain prone to hallucination and stale knowledge when answering knowledge-intensive questions. This motivates the development of Retrieval-Augmented Generation (RAG), which grounds the generation process in external corpora to enhance factual accuracy (Lewis et al., 2020). To further support complex reasoning, recent studies have explored the integration of RAG with Chain-of-Thought reasoning, interleaving retrieval with intermediate reasoning steps to progressively refine the answer (Trivedi et al., 2023). However, existing RAG-based methods still struggle with complex multi-hop questions due to three critical issues:

(1) **Retrieval–Reason Mismatch.** Multi-hop questions often require fine-grained intermediate evidence that is not explicit in the original query. As shown in Figure 1, increasing Top-$K$ enlarges the retrieved context but yields saturated gains, indicating that simply retrieving more documents cannot resolve this mismatch.

(2) **Dependency Ignorance.** Planning RAG effectively reduces reasoning complexity by breaking down complex queries into sub-questions, which are commonly executed in either a sequential or parallel manner. However, both paradigms may suffer from dependency ignorance. As shown in Figure 2, subsequent sub-questions rely on answers from preceding steps to resolve ambiguity. Ignoring such dependencies leads to missing critical entities across retrieval stages, causing cascading retrieval and reasoning failures.

(3) **Error Propagation.** LLM-generated decompositions may contain noisy or misaligned sub-questions. Without verification, such nodes are executed as part of the reasoning plan and can propagate errors to downstream retrieval and inference steps.

To systematically address these challenges, we propose **SDRAG**, a dependency-aware RAG framework that reformulates MHQA as a dynamic graph

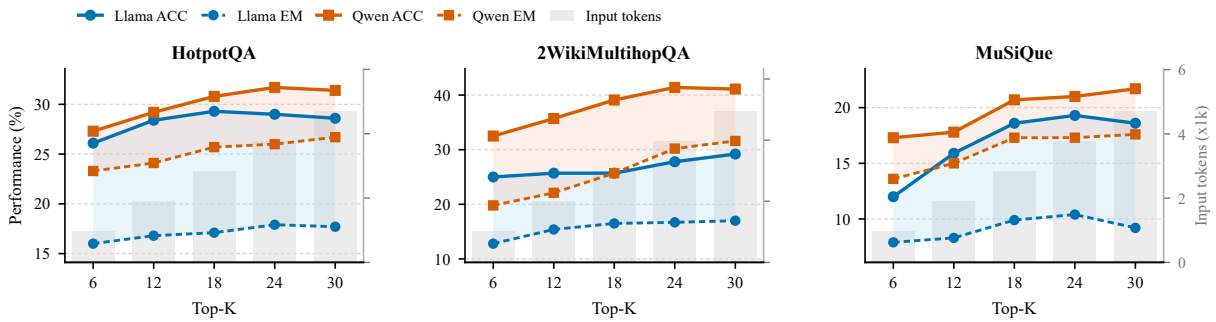



Figure 1: Standard RAG performance (ACC and EM) and input token length across three benchmarks as Top-$K$ increases from 6 to 30.

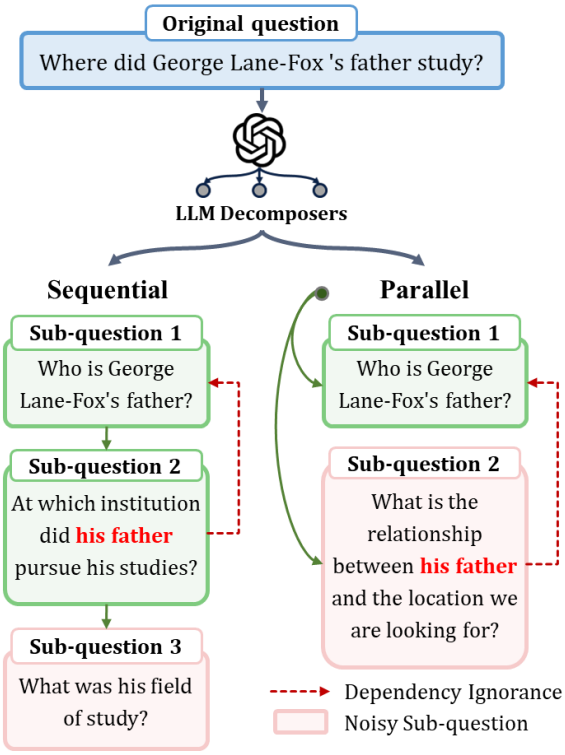



Figure 2: Sequential and parallel decomposition outputs for an example multi-hop question, illustrating two common failure modes: dependency ignorance (**red text** and red dashed arrows) and noisy sub-questions (**pink nodes**).

traversal process. For (1) and (2), SDRAG first employs a **Dependency-Aware Graph Construction** module, which maps sub-questions to graph nodes and models their logical dependencies as directed edges. This graph exposes the latent reasoning structure of the original question and provides step-specific retrieval targets. SDRAG then performs **Graph Topological Traversal**, dynamically coordinating retrieval and reasoning according to the dependency structure so that evidence from predecessor nodes can be propagated to downstream steps. For (3), SDRAG introduces a **Pruning-driven Graph Refinement** mechanism that filters low-quality or redundant nodes before traversal, reducing error propagation and producing a more robust reasoning chain. However, lightweight LLMs tend to struggle with graph construction, exhibiting issues such as over-decomposition or aggressive pruning. We introduce task-specific fine-tuning for decomposition, dependency analysis, and pruning.

Our contributions are threefold: (1) We propose **SDRAG**, a dependency-aware RAG framework that models multi-hop QA with a question-side sub-question dependency graph, enabling retrieval and reasoning to follow the intrinsic dependency structure of the question. (2) We introduce a task-specific fine-tuning strategy that trains separate LoRA adapters for decomposition, dependency analysis, and pruning, improving the reliability of graph construction for lightweight models. (3) Experiments on three multi-hop QA benchmarks show that SDRAG outperforms strong RAG-based baselines, improving over DualRAG by 5.1 EM points on average; task-specific fine-tuning further improves Llama-3.1-8B performance by 2.2 EM points on average.

## 2 Related Work

**MHQA** MHQA requires models to retrieve evidence across multiple documents and synthesize it through logical reasoning (Mavi et al., 2024), making it a key benchmark for evaluating retrieval and reasoning ability.

LLMs have become a dominant approach for MHQA owing to their strong pre-trained knowledge and reasoning capabilities. Existing LLM-based methods fall into two paradigms. The first paradigm is prompting strategies (Marvin et al., 2023; Kojima et al., 2022; Bandyopadhyay et al.,

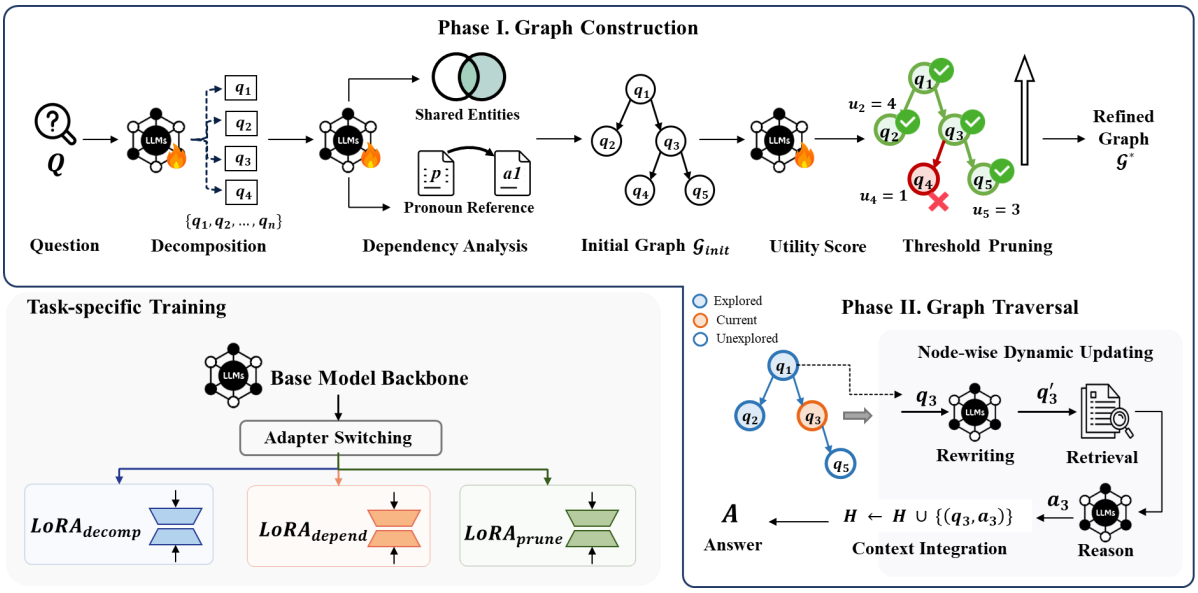



Figure 3: Overview of SDRAG. The framework consists of two stages: (1) **Graph Construction**, including query decomposition, dependency analysis and pruning judgment; and (2) **Graph Traversal**, which performs context-aware rewriting, retrieval, and reasoning. Task-specific LoRA adapters are trained for graph construction.

2025), which elicit the intrinsic reasoning ability of LLMs, represented by CoT (Wei et al., 2022), Complexity-based Prompting (Fu et al., 2023), and Self-Consistency (Wang et al., 2023b). Planning methods further decompose complex queries into manageable steps, such as Least-to-Most (Zhou et al., 2023), and Plan-and-Solve (Wang et al., 2023a). However, these methods still rely heavily on static parametric knowledge and are prone to hallucination (Lewis et al., 2020). The second paradigm augments LLMs with retrieval (Izacard and Grave, 2021; Borgeaud et al., 2022): IR-CoT (Trivedi et al., 2023) interleaves retrieval with intermediate reasoning, Self-Ask (Press et al., 2023) identifies missing knowledge through self-questioning, and FLARE (Jiang et al., 2023) triggers retrieval based on uncertainty.

Despite their progress, these methods often model reasoning as a linear or independent process, overlooking dependency structures among intermediate steps. In contrast, *SDRAG* reformulates MHQA as graph traversal guided by topological dependencies.

**Retrieval-Augmented Generation** RAG (Arslan et al., 2024) improves LLMs on knowledge-intensive tasks by grounding generation in external corpus. Classic RAG follows the retrieve-then-generate paradigm (Yu et al., 2023), using sparse retrievers such as BM25 (Lù, 2024) or dense retrievers such as DPR (Karpukhin et al., 2020) and Contriever (Izacard et al., 2022) to retrieve context.

For multi-hop reasoning, conventional RAG is limited by shallow semantic matching and one-shot retrieval. Iterative RAG methods (Trivedi et al., 2023) address this by performing retrieval and generation in multiple rounds, such as Iter-RetGen (Shao et al., 2023) and Self-RAG (Asai et al., 2024). However, most iterative methods still follow fixed linear pipelines and do not explicitly use the logical structure of the question.

Another line of work explores graph-based RAG, where documents are organized as graph structures. GraphRAG (Edge et al., 2024), HippoRAG (Gutiérrez et al., 2024), and LightRAG (Guo et al., 2025) typically construct corpus-side knowledge graphs and retrieve evidence through community search or personalized PageRank. While these methods construct corpus-side graphs and demonstrate the value of graph structures for organizing external knowledge, SDRAG constructs a question-side dependency graph to model the reasoning structure of each specific question.

## 3 Methodology

We propose **SDRAG**, a dependency-aware RAG framework for multi-hop question answering. Given a question $Q$ and corpus $\mathcal{C}$, SDRAG constructs a sub-question dependency graph, prunes unreliable nodes, and traverses the refined graph to coordinate retrieval and reasoning.

### 3.1 Overview

As shown in Figure 3, SDRAG consists of two stages. In **Graph Construction**, it decomposes the input question into atomic sub-questions, predicts their logical dependencies, and prunes redundant or noisy nodes to obtain a refined dependency graph. In **Graph Traversal**, it rewrites each sub-question with predecessor context, retrieves targeted evidence, and generates intermediate answers for final synthesis. To improve graph construction, SDRAG uses task-specific LoRA adapters for decomposition, dependency analysis, and pruning.

### 3.2 Graph Construction

Multi-hop questions are typically composed of multiple interdependent reasoning steps. To capture such dependencies, we transform the original question $Q$ into an initial sub-question dependency graph $G_{\text{init}} = (V, E)$, where nodes represent sub-questions and directed edges represent logical dependencies.

#### 3.2.1 Query Decomposition

Given the original question $Q$, we employ an LLM-based decomposer $M_{\text{decomp}}$ to generate a set of atomic sub-questions:

$$\mathcal{Q} = \{q_1, q_2, \ldots, q_n\}.$$

Each sub-question is expected to correspond to a single reasoning step, and the generated sub-questions form the node set $V$ of the dependency graph.

#### 3.2.2 Dependency Analysis

Traditional decomposition RAG paradigms often treat sub-questions as either independent or linear sequences, overlooking their logical correlations. We design a dependency analysis module to analyze all ordered sub-question pairs $\{q_i, q_j\}$ (where $i < j$) in $Q$, based on two core criteria:

The first is **Explicit Entity Entailment**. Shared entities indicate a context dependency between sub-questions. Let $\mathcal{E}(q)$ denote the set of named entities extracted from sub-question $q$. We define the explicit dependency relation $\mathcal{R}_{\text{exp}}$ as follows: if the intersection of their entity sets is non-empty, the relation holds.

$$\mathcal{R}_{\text{exp}}(q_i, q_j) \iff \mathcal{E}(q_i) \cap \mathcal{E}(q_j) \neq \emptyset.$$

In this case, we add a directed edge $q_i \to q_j$ to preserve the entity-level information flow.

The second is **Implicit Reference Dependency**. If $q_j$ contains ambiguous pronouns that semantically point to the potential answer of $q_i$, a dependency exists. Let $P(q_j)$ be the set of pronouns in $q_j$, and $\text{Ans}(q_i)$ denote the answer of $q_i$. We define the implicit dependency relation $\mathcal{R}_{\text{imp}}$ as follows:

$$\mathcal{R}_{\text{imp}}(q_i, q_j) \iff \exists p \in P(q_j), \text{Ref}(p) = \text{Ans}(q_i),$$

where $\text{Ref}(p)$ represents the semantic reference of the pronoun $p$. In this case, a directed edge $q_i \to q_j$ is added to ensure reference resolution during inference.

Based on these criteria, we construct a binary dependency matrix $M \in \{0, 1\}^{n \times n}$. To ensure that the graph is acyclic, we strictly constrain dependencies to satisfy the topological order $i < j$:

$$M_{ij} = \mathbb{I}\left((i < j) \wedge (\mathcal{R}_{\text{exp}}(q_i, q_j) \vee \mathcal{R}_{\text{imp}}(q_i, q_j))\right),$$

and thus obtain a DAG that explicitly encodes the implicit multi-hop reasoning structure.

#### 3.2.3 Graph Pruning

The initial graph may contain noisy or redundant sub-questions introduced during decomposition. Executing such nodes can retrieve irrelevant evidence and propagate errors to downstream reasoning. To mitigate this problem, SDRAG refines the initial graph by pruning low-quality nodes.

Given the current graph $G^{(t)} = (V^{(t)}, E^{(t)})$, we consider leaf nodes as pruning candidates:

$$L^{(t)} = \{v_i \in V^{(t)} \mid \mathcal{N}_{\text{out}}(v_i) = \emptyset\},$$

where $\mathcal{N}_{\text{out}}(v_i)$ denotes the outgoing neighbors of $v_i$. We focus on leaf nodes because removing them does not invalidate unresolved downstream dependencies.

For each candidate node $v_i$, the pruning model first estimates a utility score with respect to the original question:

$$s_i = M_{\text{score}}(v_i, Q), \quad s_i \in \{1, 2, 3, 4, 5\}.$$

The score provides a fine-grained assessment of whether the sub-question is necessary to answer $Q$. Conditioned on this score and the question context, the model then predicts a binary pruning label:

$$y_i = M_{\text{label}}(v_i, Q, s_i), \quad y_i \in \{\text{KEEP}, \text{PRUNE}\}.$$

The final pruning action is determined by the predicted label:

$$\text{Action}(v_i) = y_i.$$

After a node is removed, its parent may become a new leaf node. We therefore apply pruning iteratively until the graph no longer changes. The final refined graph is denoted as $G^\star$.

This bottom-up refinement prevents unreliable, redundant, or intent-shifted sub-questions from being executed, while preserving the dependency structure needed for downstream traversal.

### 3.3 Graph Traversal

The pruned graph $\mathcal{G}^*$ serves as a structured reasoning blueprint. Instead of treating sub-questions independently, SDRAG performs dynamic graph traversal, where prior reasoning results are used to guide subsequent retrieval. For a node $v_i$, the following operations are performed:

#### 3.3.1 Context-aware Rewriting

To resolve retrieval ambiguity, we aggregate contextual information $\mathcal{H}_i$ from its predecessor nodes

$$\mathcal{H}_i = \{(q_u, a_u) \mid u \in \mathcal{N}_{\text{in}}(v_i)\}.$$

Then, we employ a rewriting model $\mathcal{M}_{\text{rewrite}}$ to generate a semantically complete query $q_i'$:

$$q_i' = \mathcal{M}_{\text{rewrite}}(q_i, \mathcal{H}_i).$$

#### 3.3.2 Node-wise Retrieval and Reasoning

Using $q_i'$, we retrieve the top-$K$ documents $D_i$ and generate the intermediate answer $a_i$.

$$\begin{aligned} D_i &= \text{Retrieve}(q_i', \mathcal{C}), \\ a_i &= \mathcal{M}_{\text{gen}}(q_i', D_i). \end{aligned}$$

#### 3.3.3 Final Answer Synthesis

After all nodes in $G^\star$ are processed, SDRAG synthesizes the final answer from the complete traversal trajectory:

$$\begin{aligned} \mathcal{T}_G &= \{(q_v', D_v, a_v)\}_{v \in V^\star}, \\ A &= M_{\text{gen}}(Q, \mathcal{T}_G). \end{aligned}$$

Thus, SDRAG controls the flow of intermediate evidence and answers through the dependency graph, coordinating retrieval and generation according to the question structure.

### 3.4 Task-Specific Fine-tuning

To improve the reliability of graph construction with lightweight models (e.g., 8B parameters), we introduce a supervised fine-tuning method for three modules in SDRAG: question decomposition, dependency analysis and graph pruning.

**Training Data Construction.** We utilize GPT-4o to generate supervision signals based on the mixed multi-hop QA dataset from PERQARAG (Liu et al., 2025), covering 10 reasoning types. For each seed question, the teacher model produces task-specific supervision: (1) atomic sub-questions; (2) a dependency adjacency matrix; and (3) pruning decisions with utility scores and binary keep/prune labels. We apply two-stage filtering to remove malformed outputs and instances with incorrect reasoning traces. Full data construction details are provided in Appendix C.

**Negative Construction for Pruning.** The supervision of pruning task is highly imbalanced: most naturally generated sub-questions are useful and should be kept. As a result, the trained model tends to under-detect noisy or redundant sub-questions. To strengthen the pruning signal, we construct synthetic negatives that simulate common failure modes in decomposition.

We construct synthetic negatives in three ways: *intent-shift*, *entity-replace*, and *over-decompose* (see Appendix C for details). These negatives are paired with pruning supervision so that the model learns not only whether a node should be removed, but also why it is less useful for solving the original question. We generate 2,838 intent-shift, 1,761 entity-replace, and 1,401 over-decompose examples.

**Task-specific LoRA Training.** Three tasks have different output spaces and decision criteria. Training a single adapter for all tasks may therefore introduce task interference. Instead, we train three task-specific LoRA adapters on the same base model, denoted as $A_{\text{dec}}$, $A_{\text{dep}}$, and $A_{\text{prune}}$.

For each task $t \in \{\text{dec}, \text{dep}, \text{prune}\}$, and the corresponding adapter $A_t$ is optimized with the standard autoregressive negative log-likelihood objective:

$$\mathcal{L}_t = -\frac{1}{|y_t|} \sum_{i=1}^{|y_t|} \log p_{\theta, A_t}(y_{t,i} \mid y_{t,<i}, x_t),$$

where $\theta$ denotes the frozen base model parameters and $A_t$ denotes the trainable LoRA parameters for task $t$. We further balance the training data, resulting in 6,457 samples for each task. During inference, SDRAG loads the single base model and dynamically switches the active LoRA adapter according to the current graph-construction stage.

Table 1: Main results comparison including Non-Retrieval, RAG & Iterative RAG, Planning RAG, and Graph-based RAG. Bold and underline denote the best and second-best performance within each setting, respectively.

| Categories | Methods | HotpotQA | | | 2WikiMultihopQA | | | MuSiQue | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | EM | Acc | F1 | EM | Acc | F1 | EM | Acc | F1 |
| *Base LLM: Qwen2.5-32B-Instruct* | | | | | | | | | | |
| **Non-Retrieval** | Direct LLM | 15.2 | 22.4 | 16.8 | 11.5 | 22.0 | 13.8 | 7.9 | 10.5 | 16.5 |
| | CoT | 20.3 | 36.8 | 17.4 | 14.0 | 40.6 | 20.5 | 6.0 | 17.4 | 10.1 |
| **RAG & Iterative RAG** | Standard RAG | 23.3 | 27.4 | 32.3 | 19.8 | 32.5 | 23.2 | 14.2 | 18.8 | 21.3 |
| | IRCoT | 23.6 | 44.1 | 35.2 | 23.1 | 35.7 | 23.2 | 12.4 | 25.1 | 19.8 |
| | DualRAG | 37.0 | 34.8 | 38.0 | 38.6 | 49.9 | 45.6 | 22.0 | 34.5 | 32.8 |
| | GenGround | 29.0 | 45.5 | 35.0 | 39.2 | 42.2 | 38.9 | 28.0 | 36.0 | 38.9 |
| **Planning RAG** | DecomposedRAG | 29.9 | 47.5 | 31.2 | 22.9 | 54.3 | 25.0 | 10.3 | 30.0 | 19.1 |
| | PERQARAG | 36.6 | 30.9 | 33.1 | 37.3 | 44.5 | 43.4 | 23.6 | 30.4 | 31.9 |
| | **SDRAG** | **40.6** | **51.7** | **52.2** | **44.5** | **56.6** | **53.4** | **28.5** | **40.2** | **39.8** |
| *Base LLM: Llama-3.1-8B-Instruct* | | | | | | | | | | |
| **Non-Retrieval** | Direct LLM | 10.1 | 20.3 | 9.7 | 10.7 | 24.5 | 21.5 | 4.5 | 8.4 | 11.1 |
| | CoT | 10.5 | 22.2 | 13.9 | 14.0 | 22.1 | 19.4 | 2.9 | 11.5 | 5.8 |
| **RAG & Iterative RAG** | Standard RAG | 16.0 | 26.1 | 28.4 | 12.8 | 29.0 | 24.2 | 7.9 | 12.0 | 15.6 |
| | IRCoT | 16.4 | 31.4 | 13.6 | 13.5 | 30.3 | 16.2 | 11.7 | 13.1 | 14.2 |
| | DualRAG | 29.9 | 37.5 | 41.6 | 25.9 | 35.4 | 34.9 | 12.8 | 17.3 | 21.9 |
| | GenGround | 21.7 | 27.4 | 30.3 | 19.3 | 28.1 | 26.3 | 10.5 | 14.0 | 18.8 |
| **Planning RAG** | DecomposedRAG | 22.7 | 32.0 | 33.0 | 16.8 | 31.3 | 18.2 | 10.2 | 15.6 | 18.5 |
| | PERQARAG | 26.7 | 29.7 | 35.0 | 26.0 | 29.6 | 31.6 | 23.1 | 27.3 | 30.3 |
| | **SDRAG** | 32.0 | 37.1 | 42.9 | 27.4 | 34.6 | 33.7 | 24.0 | 31.2 | **33.7** |
| | **SDRAG-FT** | **35.0** | **44.9** | **45.7** | **30.0** | **37.9** | **36.3** | **24.9** | **33.4** | 33.3 |
| **Graph-based RAG** | HippoRAG | 41.8 | 51.8 | 53.4 | 40.0 | **50.2** | 45.4 | 17.6 | 24.3 | 25.5 |
| | **SDRAG-KC** | **44.3** | **54.6** | **56.9** | **41.2** | 49.2 | **45.8** | **22.1** | **31.5** | **30.0** |

## 4 Experiments

We evaluates SDRAG through extensive experiments on multiple multi-hop question answering benchmarks.

### 4.1 Datasets and Metrics

We evaluate SDRAG on three representative multi-hop QA benchmarks: **HotpotQA**, **2WikiMultihopQA**, and **MuSiQue**. Due to computational constraints, we randomly sample 1,000 questions from the validation or test set of each dataset. We report Exact Match (EM), Accuracy (Acc/Cover-EM), and token-level F1.

### 4.2 Baselines

We compare SDRAG with representative methods across different paradigms. **(1) Non-Retrieval Methods:** Direct LLM and CoT (Wei et al., 2022), which generate answers relying solely on parametric knowledge. **(2) RAG & Iterative RAG:** Standard RAG and iterative frameworks including IRCoT (Trivedi et al., 2023), GenGround (Shi et al., 2024), and DualRAG (Cheng et al., 2025). **(3) Planning RAG:** DecomposedRAG (Arslan et al., 2024), which solves sub-questions independently, and PERQARAG (Liu et al., 2025), which executes retrieval over a static plan graph. **(4) Graph-based RAG:** HippoRAG (Gutiérrez et al., 2024). We also report SDRAG-KC, SDRAG under the same retrieval corpus setting with HippoRAG.

### 4.3 Implementation Details

We use **Qwen2.5-32B-Instruct**[1] and **Llama-3.1-8B-Instruct**[2] as backbone models. For retrieval, we use the Wikipedia 2018 corpus as the general retrieval corpus for all datasets. We encode passages with **e5-base-v2** (Wang et al., 2022), build FAISS indexes, and retrieve the top-6 passages for each query as contextual evidence. All RAG methods are implemented based on FlashRAG (Jin et al., 2025) under the same corpus and retriever settings. For task-specific graph-construction training, we use Ms-Swift (Zhao et al., 2025) with LoRA for parameter-efficient fine-tuning. Detailed fine-tuning settings are provided in Appendix D.

[1] https://huggingface.co/Qwen/Qwen2.5-32B-Instruct

[2] https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct

### 4.4 Main Results

As shown in Table 1, SDRAG achieves the strongest overall performance among RAG-based and planning-based methods across both backbone models and all three datasets. Averaged over six settings, SDRAG improves over DualRAG by 5.1 EM points and over PERQARAG by 4.0 EM points. This consistent advantage suggests that explicitly modeling dependencies among sub-questions enables the model to retrieve evidence in a more structured and reasoning-aware manner, rather than relying on linear iterative retrieval or static planning.

Fine-tuning offers additional benefits. SDRAG-FT improves over SDRAG by 2.2 EM points on average with Llama-3.1-8B, showing that task-specific LoRA fine-tuning effectively enhances graph construction quality for smaller models. Under HippoRAG corpus setting, SDRAG-KC outperforms HippoRAG by 2.7 EM points on average with Llama-3.1-8B, demonstrating that question-side dependency graphs complement corpus-side knowledge graphs for multi-hop reasoning.

Table 2: Ablation study using Qwen2.5-32B-Instruct.

| Models | Dataset | EM | Acc | F1 |
|---|---|---|---|---|
| SDRAG | HotpotQA | **40.6** | **51.7** | 52.2 |
| | 2Wiki | **44.5** | **56.6** | **53.4** |
| | MuSiQue | **28.5** | **40.2** | **39.8** |
| w/o Pruning | HotpotQA | 39.6 | 50.4 | **53.6** |
| | 2Wiki | 42.5 | 53.5 | 51.8 |
| | MuSiQue | 26.9 | 35.1 | 33.5 |
| w/o Updating | HotpotQA | 32.5 | 39.7 | 44.5 |
| | 2Wiki | 27.5 | 35.6 | 35.6 |
| | MuSiQue | 18.3 | 25.3 | 29.0 |
| w/o Graph | HotpotQA | 29.9 | 47.5 | 31.2 |
| | 2Wiki | 22.9 | 54.3 | 25.0 |
| | MuSiQue | 10.3 | 30.0 | 19.1 |

### 4.5 Ablation Study

To analyze each component's contribution, we conduct ablation studies using Qwen2.5-32B-Instruct on three datasets. We compare the full model against three variants: (1) **w/o Pruning**: removes the graph pruning module; (2) **w/o Updating**: disables dynamic query rewriting during traversal; (3) **w/o Graph**: removes dependency modeling, reducing to the DecomposedRAG paradigm.

Results in Table 2 indicate: (1) **w/o Pruning** causes an average EM drop of 1.53, indicating that decomposition introduces noisy or redundant sub-questions that must be filtered to maintain reasoning quality. (2) **w/o Updating** leads to a larger drop of 11.8 points, as without context-aware rewriting, the retriever fails to resolve cross-hop dependencies. (3) **w/o Graph** yields the worst degradation (16.8 points), demonstrating that linear decomposition cannot capture complex multi-hop reasoning structures.

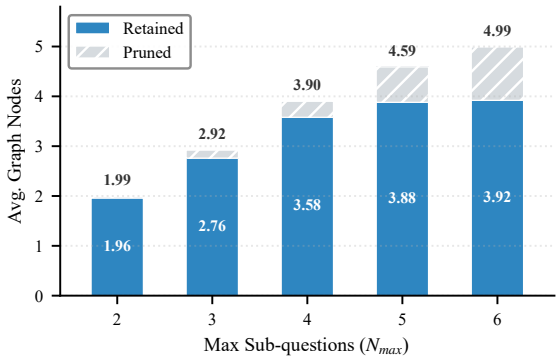



Figure 4: Graph node statistics under different decomposition scales on 2Wiki. The number of retained nodes saturates around 3.9 regardless of $N_{max}$.

### 4.6 Sensitivity Analysis

We further analyze the robustness of SDRAG with respect to two key hyperparameters:

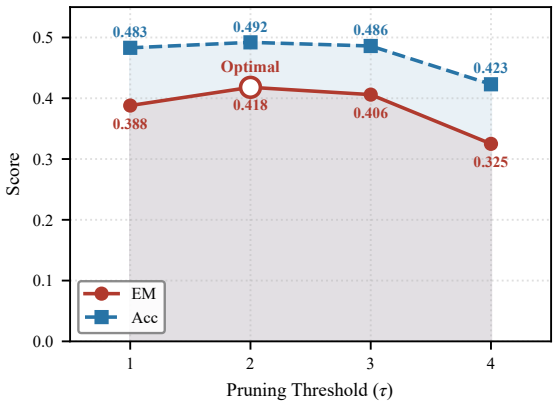
**Impact of Decomposition Scale** We analyze the impact of the maximum number of decomposed questions ($N_{max}$) on model performance using Llama-3.1-8B-Instruct on 2WikiMultihopQA. As shown in Table 3, both EM and Acc improve consistently as $N_{max}$ increases from 2 to 5, indicating that finer-grained decomposition better captures the intermediate reasoning steps and critical entities required for MHQA.

However, performance drops slightly at $N_{max}$ = 6. As illustrated in Figure 4, the pruned graph size remains stable despite the increased number of sub-questions, suggesting that LLMs tend toward over-decomposition by generating redundant or irrelevant sub-questions beyond a certain scale.

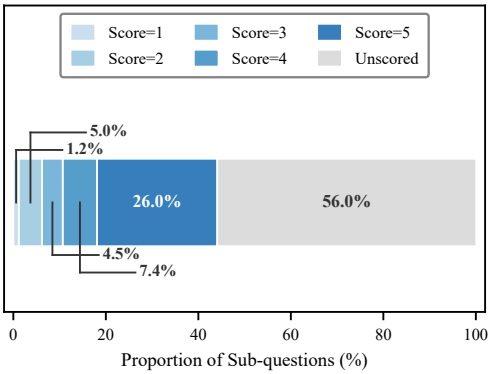
**Impact of Pruning Threshold.** We further analyze the effect of the pruning threshold $\tau$ using

Table 3: Performance under different decomposition scales ($N_{max}$) on 2WikiMultihopQA. The model achieves optimal performance at $N_{max} = 5$.

| $N_{max}$ | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| EM (%) | 22.9 | 25.0 | 26.8 | **27.4** | 26.2 |
| Acc (%) | 30.6 | 32.8 | 34.1 | **34.6** | 32.8 |




(a) Performance vs. Pruning Threshold




(b) Node Score Distribution

Figure 5: Sensitivity analysis on 2WikiMultihopQA: (a) Model performance under different pruning thresholds $\tau$. (b) Distribution of utility scores assigned to sub-question nodes.

Qwen2.5-32B-Instruct on 2WikiMultihopQA. As shown in Figure 5(a), performance follows an inverted U-shape, peaking at $\tau = 2$, indicating that filtering low-quality nodes (Score $\leq 2$) effectively reduces noise, while raising $\tau = 4$ removes essential reasoning steps and causes a notable accuracy drop. The score distribution in Figure 5(b) confirms this. Therefore, $\tau = 2$ strikes the optimal balance between noise reduction and information preservation.

### 4.7 Efficiency Analysis

We conduct a holistic efficiency evaluation on HotpotQA using five metrics: EM, F1, Latency, Cost, and Retrieval Count. Specifically, for performance metrics (EM, F1) where higher is better, we map the values directly to the $[0, 1]$ interval. For cost-related metrics (Latency, LLM Calls, Retrieval Count) where lower is better, we apply inverted normalization. As shown in Figure 6, SDRAG achieves the best trade-off between model performance and cost. Compared to DualRAG, which relies on extensive iterative retrieval steps to accumulate evidence, SDRAG significantly reduces the number of both retrieval and reasoning calls while achieving state-of-the-art performance (32.0% EM). This demonstrates that SDRAG's dependency graph modeling and pruning effectively streamline the retrieval process without sacrificing answer quality.

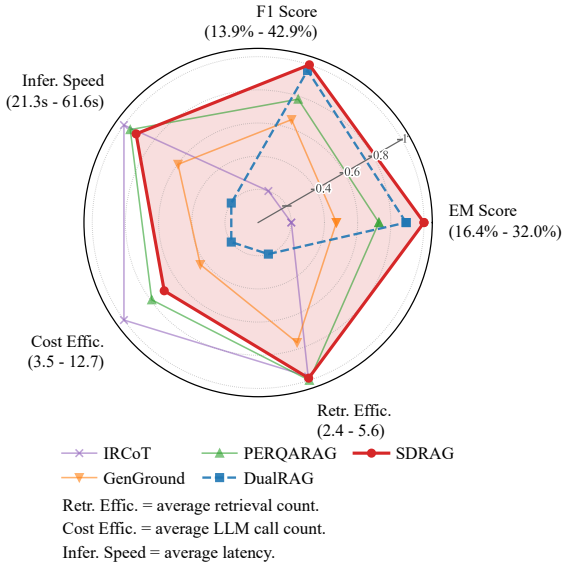



Figure 6: Holistic performance comparison on HotpotQA. SDRAG (red solid line) achieves the best trade-off across five metrics.

Additional analyses, including graph structural properties and case studies, are provided in Appendix E and G.

## 5 Conclusion

In this paper, we propose SDRAG, a dependency-aware RAG framework that reformulates multi-hop QA as traversal over a sub-question dependency graph. By modeling sub-question dependencies, pruning noisy nodes, and performing context-aware retrieval, SDRAG addresses key limitations of planning-based RAG methods. We further train task-specific LoRA adapters to improve graph construction for lightweight models. Experiments on three multi-hop QA datasets show that SDRAG consistently outperforms strong baselines. Fine-tuning further improves Llama-3.1-8B by 2.2 points on average.

## Limitations

Although SDRAG improves multi-hop reasoning by explicitly modeling sub-question dependencies, it still has several limitations. First, dynamic graph construction introduces additional computational latency compared with standard retrieve-then-generate pipelines, since the model needs to perform decomposition, dependency prediction, pruning, and graph traversal before final answer synthesis. Although pruning helps reduce redundant reasoning steps, the overall inference cost remains higher than that of simpler RAG methods. Second, our current dependency graph uses binary directed edges to represent whether one sub-question depends on another. This formulation is simple and effective, but may oversimplify more complex reasoning relations, such as soft dependencies, conditional dependencies, or multiple relation types among intermediate reasoning steps.

## Ethical considerations

This work uses publicly available multi-hop QA datasets (HotpotQA, 2WikiMultihopQA, MuSiQue) and open-source models for research purposes. We do not collect private user data or introduce new human-subject annotations. We use GPT-4o as a teacher model to generate supervision signals for fine-tuning; all generated data is used solely for research and does not contain personally identifiable information. Since SDRAG relies on LLM-based generation and external retrieval, it may produce incorrect or unsupported answers when retrieved evidence is incomplete or biased. We recommend human verification before applying the system in high-stakes domains.

## References


Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *ArXiv preprint*, abs/2303.08774.

Muhammad Arslan, Hussam Ghanem, Saba Munawar, and Christophe Cruz. 2024. A survey on rag with llms. *Procedia computer science*, 246:3781–3790.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*.

Dibyanayan Bandyopadhyay, Soham Bhattacharjee, and Asif Ekbal. 2025. Thinking machines: A survey of llm based reasoning strategies. *arXiv preprint arXiv:2503.10814*.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, and 1 others. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

Rong Cheng, Jinyi Liu, Yan Zheng, Fei Ni, Jiazhen Du, Hangyu Mao, Fuzheng Zhang, Bo Wang, and Jianye Hao. 2025. DualRAG: A dual-process approach to integrate reasoning and retrieval for multi-hop question answering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 31877–31899. Association for Computational Linguistics.

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

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2023. Complexity-based prompting for multi-step reasoning. In *The Eleventh International Conference on Learning Representations*.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2025. LightRAG: Simple and fast retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 10746–10761, Suzhou, China. Association for Computational Linguistics.

Bernal J Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in neural information processing systems*, 37:59532–59569.

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

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880. Association for Computational Linguistics.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992.

Jiajie Jin, Yutao Zhu, Zhicheng Dou, Guanting Dong, Xinyu Yang, Chenghao Zhang, Tong Zhao, Zhao Yang, and Ji-Rong Wen. 2025. Flashrag: A modular toolkit for efficient retrieval-augmented generation research. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 737–740.

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

Zixuan Ke, Fangkai Jiao, Yifei Ming, Xuan-Phi Nguyen, Austin Xu, Do Xuan Long, Minzhi Li, Chengwei Qin, PeiFeng Wang, silvio savarese, Caiming Xiong, and Shafiq Joty. 2025. A survey of frontiers in LLM reasoning: Inference scaling, learning to reason, and agentic systems. *Transactions on Machine Learning Research*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 9459–9474.

Qichuan Liu, Chentao Zhang, Chenfeng Zheng, Guosheng Hu, Xiaodong Li, and Zhihong Zhang. 2025. Beyond the answer: Advancing multi-hop qa with fine-grained graph reasoning and evaluation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23433–23456.

Xing Han Lù. 2024. Bm25s: Orders of magnitude faster lexical search via eager sparse scoring. *ArXiv preprint*, abs/2407.03618.

Ggaliwango Marvin, Nakayiza Hellen, Daudi Jjingo, and Joyce Nakatumba-Nabende. 2023. Prompt engineering in large language models. In *International conference on data intelligence and cognitive informatics*, pages 387–402. Springer.

Vaibhav Mavi, Anubhav Jangra, Adam Jatowt, and 1 others. 2024. Multi-hop question answering. *Foundations and Trends in Information Retrieval*, 17(5):457–586.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9248–9274.

Zhengliang Shi, Shuo Zhang, Weiwei Sun, Shen Gao, Pengjie Ren, Zhumin Chen, and Zhaochun Ren. 2024. Generate-then-ground in retrieval-augmented generation for multi-hop question answering. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7339–7353.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 10014–10037.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023a. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2634.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

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.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2023. Generate rather than retrieve: Large language models are strong context generators. In *The Eleventh International Conference on Learning Representations*.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, Wenmeng Zhou, and Yingda Chen. 2025. SWIFT: A scalable lightweight infrastructure for fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 29733–29735.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

## A Dataset Details

We evaluate SDRAG on three representative multi-hop question answering benchmarks: HotpotQA, 2WikiMultihopQA, and MuSiQue. All three datasets require models to aggregate evidence from multiple documents and perform multi-step reason.

**HotpotQA** is a widely used multi-hop QA benchmark constructed from Wikipedia. It is designed to evaluate whether models can retrieve supporting evidence and perform reasoning across multiple documents. The full dataset contains 90,447 training examples and 7,405 validation examples, and provides supporting fact annotations for evidence supervision and analysis.

**2WikiMultihopQA** is a Wikipedia-based multi-hop QA benchmark designed to cover diverse reasoning patterns, including comparison, inference, composition, and bridge-comparison. Compared with HotpotQA, it contains more structured multi-hop reasoning chains and requires models to capture temporal, causal, and logical relations across documents. The dataset contains 15,000 training examples and 12,576 validation examples.

**MuSiQue** is a challenging multi-hop QA benchmark constructed with strict compositional constraints. Its questions are designed to be answerable only by combining evidence from multiple paragraphs, which reduces shortcut reasoning based on a single document. The dataset focuses on 2–4 hop questions and contains 19,938 training examples and 2,417 validation examples.

## B Evaluation Metrics

Before evaluation, we apply standard answer normalization to both the predicted answer $\hat{a}$ and the gold answer $a^*$, including lowercasing and removing articles, punctuation, and extra whitespace. We report three metrics: Exact Match (EM), Accuracy (Acc), and token-level F1.

**Exact Match (EM).** EM measures whether the normalized prediction exactly matches the normalized gold answer:

$$\mathrm{EM}(\hat{a}, a^*) = \begin{cases} 1, & \text{if } \mathrm{Norm}(\hat{a}) = \mathrm{Norm}(a^*) \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The final EM score is averaged over all evaluation examples.

**Accuracy (Acc).** Following prior work, we also report Accuracy, also known as Cover-EM. It measures whether the normalized gold answer appears in the normalized prediction:

$$\mathrm{Acc}(\hat{a}, a^*) = \begin{cases} 1, & \text{if } \mathrm{Norm}(a^*) \subseteq \mathrm{Norm}(\hat{a}) \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

This metric is useful for generative QA because a model may produce a complete sentence that contains the correct answer but does not exactly match the gold span.

**Token-level F1.** Token-level F1 measures the overlap between the predicted and gold answers after normalization. Let $T(\hat{a})$ and $T(a^*)$ denote the token multisets of the prediction and the gold answer. Precision and recall are computed as:

$$P = \frac{|T(\hat{a}) \cap T(a^*)|}{|T(\hat{a})|}, \quad R = \frac{|T(\hat{a}) \cap T(a^*)|}{|T(a^*)|}. \tag{3}$$

The F1 score is then:

$$\mathrm{F1} = \begin{cases} \frac{2PR}{P+R}, & \text{if } P + R > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Token-level F1 provides a finer-grained measure of answer overlap and is less sensitive to minor wording differences than EM.

## C Fine-tuning Data Construction

We construct a supervised fine-tuning dataset for the graph-construction modules of SDRAG, including question decomposition, dependency analysis, and graph pruning. The seed questions are taken from the mixed multi-hop QA dataset released by PERQARAG (Liu et al., 2025), which contains 8,934 high-quality examples covering 10 typical reasoning types. We use GPT-4o as the

teacher model to generate reasoning traces and task-specific supervision signals.

For each seed question, the teacher model is prompted to solve the question step by step and produce supervision for three tasks. For decomposition, the output is a sequence of atomic sub-questions. For dependency analysis, the output is an adjacency matrix that represents logical dependencies among sub-questions. For pruning, the output contains a utility score for each candidate sub-question, followed by a binary keep/prune label. In this way, the pruning supervision captures both fine-grained usefulness estimation and the final pruning decision.

To ensure the quality of teacher-generated supervision, we apply two-stage filtering. First, we remove malformed instances whose sub-question list, adjacency matrix, utility score, or binary pruning label cannot be parsed into the required format. Second, we conduct end-to-end correctness filtering: only instances whose teacher-generated reasoning process leads to the correct final answer are retained. This step reduces the risk of learning from spurious decompositions or incorrect dependency structures. After filtering, we obtain 6,457 decomposition examples, 6,457 dependency-analysis examples, and 18,417 pruning examples.

The pruning data is naturally imbalanced because most decomposed sub-questions are useful for answering the original question. To strengthen the pruning signal, we further construct synthetic negative sub-questions that simulate common decomposition errors. Specifically, we consider three types of negatives: *intent-shift* negatives, which remain topically relevant but deviate from the original question intent; *entity-replace* negatives, which replace key entities with plausible but incorrect alternatives; and *over-decompose* negatives, which introduce unnecessary intermediate steps. We filter out invalid or ambiguous negatives and retain 6,000 examples in total, including 2,838 intent-shift, 1,761 entity-replace, and 1,401 over-decompose examples.

Finally, to avoid task imbalance during fine-tuning, we balance the training data across the three tasks. Each task contains 6,457 training examples in the final fine-tuning set. The resulting dataset provides aligned supervision for decomposition, dependency modeling, and pruning, enabling the lightweight backbone model to perform reliable question-side graph construction.

## D Fine-tuning Training Details

We fine-tune the graph-construction modules of SDRAG with Llama-3.1-8B-Instruct as the backbone model. Instead of training a single shared adapter for all graph-construction tasks, we train three independent LoRA adapters for question decomposition, dependency analysis, and graph pruning. This design reflects the different output spaces and decision criteria of the three tasks: decomposition generates a sequence of atomic sub-questions, dependency analysis predicts graph structure, and pruning estimates whether a candidate node should be retained.

For each task, we reserve 5% of the data for validation. We train all adapters using the Ms-Swift framework with parameter-efficient LoRA fine-tuning.

For each task, we search over several LoRA configurations with different ranks, target modules, learning rates, warmup ratios, and maximum sequence lengths. The best checkpoint is selected according to the lowest validation loss. The selected configurations are summarized in Table 4. During inference, SDRAG loads a single Llama-3.1-8B-Instruct backbone and dynamically switches the active LoRA adapter according to the current graph-construction stage. After graph construction and refinement, adapters are disabled, and the subsequent rewriting, retrieval, intermediate answering, and final synthesis steps are performed with the base model.

## E Structural Properties of Pruned Dependency Graphs

We further analyze the structural properties of the question-side dependency graphs produced by SDRAG after pruning. This analysis examines the hop distribution, the proportion of linear and non-linear graph structures, and the percentage of questions affected by pruning.

As shown in Figure 7(a), most questions are represented as three-hop dependency graphs after pruning, accounting for 67.4%, 58.9%, and 59.4% on HotpotQA, 2Wiki, and MuSiQue, respectively. At the same time, a non-negligible portion of questions requires four or more hops, especially on MuSiQue, where 35.3% of the questions fall into the 4-hop or $\geq$5-hop categories. This suggests that SDRAG captures compact but still multi-step reasoning structures.

Table 4: Selected task-specific LoRA adapters. The best configuration for each task is chosen according to validation loss.

| Task | Target Modules | Rank | Alpha | LR |
|---|---|---|---|---|
| Decomposition | all-linear | 32 | 64 | $6\times10^{-5}$ |
| Dependency Analysis | all-linear | 16 | 32 | $6\times10^{-5}$ |
| Graph Pruning | gate/up/down proj. | 16 | 32 | $8\times10^{-5}$ |

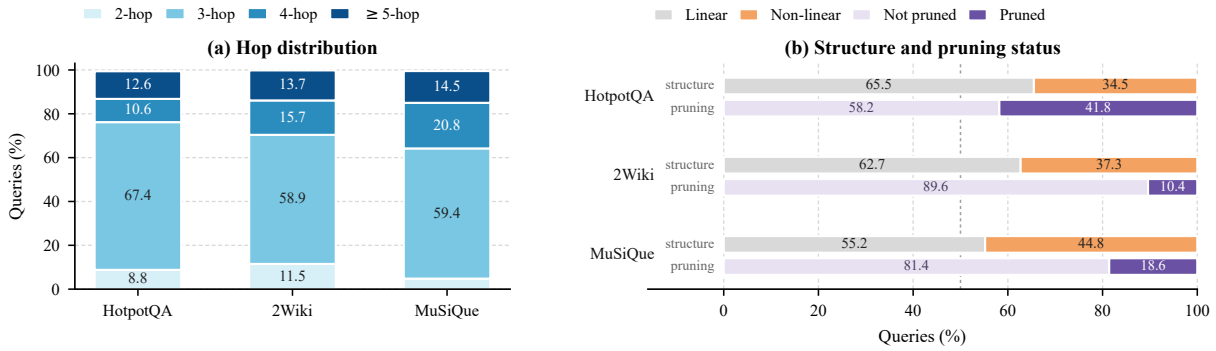



Figure 7: Structural properties of pruned dependency graphs. (a) shows the hop distribution after graph construction and pruning. (b) shows the proportion of linear/non-linear structures and the pruning status across datasets.

Figure 7(b) further shows that non-linear dependency structures are common across datasets, accounting for 34.5%, 37.3%, and 44.8% on HotpotQA, 2Wiki, and MuSiQue, respectively. This indicates that many multi-hop questions cannot be adequately represented as simple linear chains, supporting the need for explicit dependency modeling among sub-questions. The pruning statistics also show that SDRAG removes redundant nodes in a dataset-dependent manner, with pruning ratios of 41.8%, 10.4%, and 18.6%. Overall, these results suggest that the proposed pruning mechanism helps produce compact dependency graphs while preserving non-linear reasoning structures.

## F Prompts for SDRAG

This section provides the detailed prompts used in our SDRAG framework. We present the instructions and constraints for each core module, including query decomposition, dependency analysis, graph pruning, context-aware rewriting, and final answer synthesis.

## G Case Study

In this section, we provide a concrete case study to demonstrate the mechanism of SDRAG in handling complex multi-hop questions. We compare the reasoning and retrieval processes of Naive RAG, Decomposed RAG, and our proposed SDRAG to highlight how explicit graph structural modeling helps mitigate common failures in traditional pipelines. Detailed case study examples are illustrated in Figure 12 and 13.

## H Licenses

2WikiMultihopQA, Qwen2.5-32B-Instruct, and Ms-Swift are released under Apache License 2.0. HotpotQA is released under CC BY-SA 4.0, and MusiQue is released under Creative Commons Attribution 4.0 International. FlashRAG, FAISS, e5-base-v2, and PER-PSE are released under MIT License. Llama-3.1-8B-Instruct is licensed under the Llama 3.1 Community License. They are all for research purposes, and our experiments are consistent with their intended usage.

**Task Description**
You are a Search Query Optimizer. Your goal is to rewrite a sub-question to be fully self-contained and specific.
**CRITICAL RULES**
1. Replace ALL pronouns with specific entities. 2. Replace vague references with actual names. 3. Use EXACT entities from previous answers, not generic descriptions. 4. Keep the rewritten question focused and concise. 5. The output should be a clear question suitable for a retriever.
**Previous sub-questions and answers:** {previous_qa}
**Sub-question to rewrite**: {sub_question}
Rewritten sub-question:

Figure 8: Prompt for Context-Aware Rewriting

**Task Description**
You are a precise QA assistant. Synthesize a final answer to the original question.
Your task is to output the final answer based on the intermediate sub-questions, their answers, AND their retrieved documents. Use them, especially the retrieved documents, for factual grounding.
**Original Question:** {original_question}
**Intermediate Reasoning Steps:** {all_steps}
**CONSTRAINT:**
Output ONLY the answer.

Figure 9: Prompt for Final Synthesis

**Task Description**
You are an expert at analyzing logic dependencies between questions.
Given a list of sub-questions, identify the DIRECT dependencies for each question based on two core logical principles:
**1. Reference:** Does the current question contain pronouns that refer to the answer of a specific previous question?
**2. Entailment:** Does the current question explicitly require entities or information found ONLY in previous questions to be answered?
**Constraints**
- Only identify the IMMEDIATE dependencies (Direct Predecessors). - Do not list indirect dependencies. - If a question can be answered independently, it depends on []. - Do not output any explanation or other text. Just the JSON.
**Sub-questions:** {sub_questions}
**Output Format**
Return the result strictly in this JSON format (Key is the question index starting from 1): { "1": [], "2": [1], ... }

Figure 10: Prompt for Dependency Analysis

**Task Description**
You are an expert at evaluating the utility of sub-questions in a multi-hop QA reasoning chain. Given the original question and a leaf sub-question, rate the NECESSITY of this sub-question on a scale from 1 to 5.
**Score Meaning**
1 (Useless): Completely irrelevant, redundant, or asks for trivial details not needed for the final answer. 2 (Weak): Provides minor background context but is likely not essential. 3 (Neutral): Borderline relevance. Could be useful but might introduce noise. 4 (Relevant): Contains useful information that helps bridge the reasoning gap. 5 (Crucial): Directly asks for key entities or facts without which the original question CANNOT be answered.
**Original question:** {original_question}
**Sub-question:** {sub_question}
**Output Format**
Output strictly in this JSON format:
{"score": <int>, "reason": "<short explanation>"}

Figure 11: Prompt for Graph Pruning

**Case Study 1**
**Question:** What is the date of death of Gero II, Margrave of the Saxon Ostmark's father?
**Standard RAG:** Fail
Based on the provided documents, there is no specific information about the date of death of Gero II, Margrave of the Saxon Ostmark's father.
**QDRAG:** Fail
*Subq1:* Who was Gero II's father? → Answer 1: Gero II, Margrave of the Saxon Ostmark's father was Thietmar, Margrave of Meissen.
*Subq2:* What is the death date for Gero II, Margrave of the Saxon Ostmark's father? → Answer 2: The date of death ... is not provided in the retrieved information.
**Ours (SDRAG):** ✓
**Graph Construction:**

- $q_1$: Who was Gero II's father?
- $q_2$: What is the death date for Gero II, Margrave of the Saxon Ostmark's father?
- $q_3$: What was his life span?

**Graph Pruning:** Node 2 (Redundant) → Pruned Structure: $0 \rightarrow 1$
**Graph Traversal:**

- *Node 0 (Subq):* Who was Gero II's father? → *Answer:* Thietmar, Margrave of Meissen.
- *Node 1 (Rewritten):* What is the specific date of death for **Thietmar, Margrave of Meissen**? → *Answer:* Thietmar died on 3 August 979.

**Final Answer:** 3 August 979

Figure 12: Case Study I.

**Case Study 2**
**Question:** Where did George Lane-Fox (Mp)'s father study?
**Standard RAG:** Fail
Based on the provided documents, there is no specific information about where George Lane-Fox (MP)'s father studied.
**QDRAG:** Fail
*Subq1:* Who is George Lane-Fox (Mp)'s father? → Answer 1: James Fox-Lane.
*Subq2:* What was his father's field of study? → Answer 2: Failed to link "father" to retrieved docs.
*Subq3:* At which institution did his father study? → Answer 3: Failed to link "father" to retrieved docs.
**Ours (SDRAG):** ✓
**Graph Construction:**

- $q_1$: Who is George Lane-Fox (Mp)'s father?
- $q_2$: What was his father's field of study?
- $q_3$: At which institution did his father study?

**Graph Pruning:** Node 1 (Redundant) → Pruned Structure: $0 \rightarrow 2$
**Graph Traversal:**

- *Node 0 (Subq):* Who is George Lane-Fox (Mp)'s father? → *Answer:* James Fox-Lane.
- *Node 2 (Rewritten):* At which institution did **James Fox-Lane** pursue his studies? → *Answer:* He pursued studies at Christ's College, Cambridge.

**Final Answer:** Christ's College, Cambridge

Figure 13: Case Study II.