DeARAG: Decoupled Agentic RAG for On-Device and Precise Multi-Hop Question Answering

Anonymous ACL submission

Abstract

Although cloud-based Large Language Models (LLMs) have significantly advanced complex multi-hop question answering, they pose severe data privacy risks. On the other hand, simply migrating such compute-intensive systems to edge devices with limited computational resources inevitably triggers context overload and the fatal "lost-in-the-middle" phenomenon. To resolve the privacy-accuracy dilemma, we introduce DeARAG, a fully decoupled, local agentic framework. By shifting heavy hypergraph construction entirely offline, DeARAG substantially condenses the required context during inference. Consequently, the agent's exploration is strictly grounded in the retrieved hypergraph topology, ensuring rigorous, evidence-based reasoning without overwhelming local resources. Evaluated on multiple benchmarks, our framework consistently outperforms massive, cloud-dependent baselines (e.g., achieving a gain of 6.4 F1 points on HotpotQA) using only consumer-grade GPUs, enabling highly precise on-device reasoning.

1 Introduction

In knowledge-intensive Natural Language Processing (NLP) tasks, Large Language Models (LLMs) have demonstrated impressive text comprehension capabilities (Achiam et al., 2023; Touvron et al., 2023). However, when directly processing raw open-domain corpora, LLMs face inherent limitations: restricted context windows exacerbate the "lost-in-the-middle" phenomenon (Liu et al., 2024), while a lack of structural grounding frequently triggers severe parametric hallucinations (Ji et al., 2023). Retrieval-Augmented Generation (RAG) has emerged as the standard paradigm to mitigate these issues (Lewis et al., 2020), with Graph-based RAG (GraphRAG) (Edge et al., 2024) proving particularly indispensable for complex multi-hop scenarios. By explicitly modeling relationships between disconnected entities, GraphRAG enables

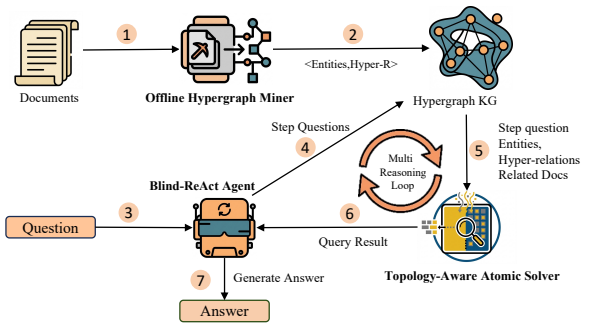

Figure 1: **Overview of DeARAG.** We fundamentally decouple the pipeline into an offline hypergraph miner, a *Blind-ReAct* planning agent, and a topology-aware atomic solver, enabling precise reasoning without API dependence.

systems to perform compositional reasoning across document boundaries—a capability standard dense retrieval lacks.

Despite its efficacy, the deployment of agentic GraphRAG is currently bottlenecked by a strict **performance-cost-privacy trilemma**. Achieving state-of-the-art multi-hop reasoning typically necessitates either transmitting sensitive queries to proprietary LLM APIs or deploying heavily reinforcement-learned agents on massive GPU clusters. Beyond these deployment barriers, current architectures suffer from two fundamental learning flaws: **(i) Structural-Execution Coupling:** Entangling the heavy lifting of graph construction with the reasoning phase prevents models from learning modular, robust reasoning policies. **(ii) Outcome-Process Misalignment:** End-to-end training (Levine et al., 2016) or outcome-based filtering frequently induces "shortcut learning" (Geirhos et al., 2020; Wei et al., 2022). Consequently, agents overfit to spurious answer patterns using parametric memory rather than acquiring faithful, structure-grounded reasoning trajectories.

To dismantle these barriers, we propose **DeARAG**, a Decoupled Agentic Reasoning frame-

work for precise, on-device multi-hop question answering. As illustrated in Figure 1, DeARAG functionally decouples knowledge mining from logical execution, reformulating the multi-hop QA pipeline into three learnable, modular sub-tasks: structure induction, atomic evidence execution, and evidence-anchored planning.

DeARAG combines three local components: an Offline Hypergraph Miner trained via Asymmetric Structural Distillation (ASD); a Topology-Aware Atomic Solver (TAAS) that converts graph-text evidence into aligned reasoning fields; and a Blind-ReAct Agent optimized via Counterfactual Blind-ReAct (CBR). Our primary contributions are summarized as follows:

- **Decoupled Local Reasoning:** By structurally decoupling retrieval and execution, DeARAG overcomes context overload and the "lost in the middle" phenomena. Using only consumer-grade hardware, it consistently outperforms compute-intensive API frameworks, democratizing robust multi-hop QA with substantial gains in speed and trajectory faithfulness.

- **Offline Hypergraph Miner:** By transferring schema-constrained hypergraph induction from a Teacher LLM to a lightweight local model, we decouple heavy knowledge mining from online inference. This drastically reduces the marginal cost of graph construction while preserving absolute data privacy.

- **Topology-Aware Atomic Solver:** This neural interface serializes heterogeneous subgraphs and textual evidence into structured reasoning fields. It bridges explicit hypergraph boundaries with natural language, enabling small models to perform precise, traceable evidence grounding.

- **Blind-ReAct Agent:** To eliminate shortcut learning and outcome-process misalignment, we propose an answer-masked trajectory optimization strategy. By explicitly discarding structurally unsupported paths, the agent learns faithful structural discovery rather than exploiting parametric memory shortcuts.

2 Related Work

Current approaches to RAG can be categorized into three stages: Standard Vector-based RAG, Graph-Enhanced RAG, and Agentic and Path-Based Reasoning.

Standard Vector-based RAG. Standard RAG (Lewis et al., 2020) relies on dense retrieval like DPR (Karpukhin et al., 2020) and Contriever (Izacard et al., 2021) to fetch similar chunks. While effective for simple queries, it suffers from "semantic isolation." Independent chunking fails to capture relationships between distant entities. Consequently, for multi-hop questions, it frequently retrieves fragmented evidence lacking connective tissue, exacerbating the "lost-in-the-middle" phenomenon.

Graph-Enhanced RAG. To address structural deficits, recent works integrate Knowledge Graphs (KGs). GraphRAG (Edge et al., 2024) uses community detection for global summaries, and HippoRAG (Jimenez Gutierrez et al., 2024; Gutiérrez et al., 2025) employs PageRank for subgraph identification. HyperGraphRAG (Luo et al., 2025) extends this to hypergraphs to capture high-order correlations.

However, these methods suffer from Structural-Execution Coupling. They entangle heavy graph induction with reasoning execution, relying on proprietary APIs. This exacerbates cost-privacy trade-offs and limits flexibility. In contrast, DeARAG functionally decouples these steps: Asymmetric Structural Distillation (ASD) handles offline graph induction locally, while the Topology-Aware Atomic Solver (TAAS) provides a modular execution interface.

Agentic and Path-Based Reasoning. RAG is increasingly shifting towards dynamic navigation. ReAct (Yao et al., 2022) interleaves reasoning with retrieval, and PathRAG (Chen et al., 2025) retrieves relation paths to preserve sequential context. Graph-R1 (Luo et al., 2026) and Think-on-Graph (Sun et al., 2023) treat LLMs as RL agents for iterative exploration.

Nevertheless, these methods face Outcome-Process Misalignment and resource barriers. RL-based approaches like Graph-R1 necessitate GPU clusters, limiting accessibility. Crucially, their reliance on outcome-based filtering—retaining trajectories for hitting the final answer—induces "look-ahead bias" and shortcut learning. Agents exploit semantic guesswork rather than faithful navigation. To resolve this, our Counterfactual Blind-ReAct Training (CBR) masks the final answer during synthesis, enforcing strictly evidence-anchored discovery without compute-intensive RL optimization.

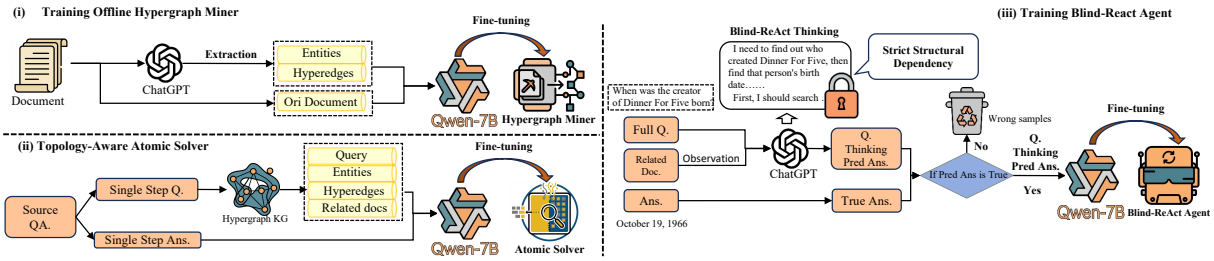

Figure 2: **The Decoupled Training Pipeline of DeARAG. (i) ASD:** Distills schema-constrained hypergraph induction from a teacher LLM to a lightweight miner. **(ii) TAAS:** Trains the solver as a neural execution interface by aligning topological graph structures with semantic evidence for precise field-level reasoning. **(iii) CBR:** Optimizes the agent via answer-masked reasoning trajectories. By strictly filtering out structurally unsupported paths (the "trash can"), the agent learns to prioritize evidence-anchored discovery over shortcut memorization.

3 Methodology

3.1 Preliminaries

We formulate Agentic GraphRAG as a **structure-aware reasoning problem**. Unlike standard RAG, DeARAG explicitly models the generative process over a structured knowledge graph to maximize the answer probability:

$$P(a|q,\mathcal{D}) = \sum_{g\in\mathcal{G}} P_{\theta_{agent}}(a|q,g)P_{\theta_{build}}(g|\mathcal{D}) \quad (1)$$

where q and a denote the complex user query and the target answer, respectively. $\mathcal{D}$ is the raw knowledge corpus, and $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ represents the explicitly constructed knowledge graph. $g \in \mathcal{G}$ is the relevant subgraph structure mined for reasoning. The terms $P_{\theta_{build}}$ and $P_{\theta_{agent}}$ parameterize the offline structural induction and the online agentic execution, respectively. Our framework decomposes this overarching objective into three learnable subproblems: *Structure Induction*, *Atomic Evidence Execution*, and *Evidence-Anchored Planning*.

3.2 Framework Overview

As illustrated in Figure 1, DeARAG adopts a decoupled architecture operating in two distinct phases: (1) Offline graph construction executed by an Offline Hypergraph Miner (optimized via Asymmetric Structural Distillation, ASD), and (2) Agentic Inference driven by a Topology-Aware Atomic Solver (TAAS) and a central Blind-ReAct Agent (optimized via Counterfactual Blind-ReAct Training, CBR).

Specifically, the pipeline begins with the offline mining phase (Steps 1-2 in Figure 1). We employ schema-constrained filtering to distill unstructured documents into a structured Hypergraph Knowledge Graph (KG). Crucially, this process utilizes the lightweight Offline Hypergraph Miner to extract high-order n-ary hyper-relations (e.g., $\langle$Entity, Hyper-R$\rangle$), effectively transferring structural induction capabilities to a local model without requiring proprietary LLM APIs during deployment.

Subsequently, the Agentic Inference phase executes a dynamic reasoning loop initiated by the user query (Step 3). Upon receiving a complex query, the Blind-ReAct Agent acts as the central planner. Within the iterative loop (Steps 4-6), the agent first decomposes the overarching question into an atomic sub-query (Step 4). The TAAS then serves as a neural execution interface, bridging the natural language query with the hypergraph topology to fetch precise entities and hyper-relations (Step 5), and returning a structurally grounded *Query Result* (Step 6). This Plan-and-Execute cycle repeats recursively—using intermediate results to refine subsequent exploratory actions—until the agent determines sufficient evidence has been gathered to synthesize the Final Answer (Step 7). This ensures that every deduction is traceable and strictly anchored in the retrieved hypergraph evidence.

3.3 Offline Hypergraph Miner

To address standard RAG's "context overload" and the cost-privacy trade-offs of API-based graph construction, we propose Asymmetric Structural Distillation (ASD). Unlike online LLM constructors, ASD treats a high-capacity Teacher LLM as an offline structural oracle. This transfers schema-constrained hypergraph induction to a lightweight local model, functioning as a high-precision filter adapting to semantic density.

Schema-Constrained Teacher Induction. We adopt the hypergraph schema from (Luo et al., 2025) but shift it to an offline distillation paradigm.

As shown in Figure 2(i), a Teacher LLM (Wang et al., 2023) synthesizes a structured seed dataset $\mathcal{D}_{syn} = \{(D_i, G_i^{tea})\}_{i=1}^{N}$. This distillation is inherently *asymmetric*: the Teacher resolves dense textual noise via complex instructions, while the student merely learns the resulting mapping. To mitigate "granularity explosion," adaptive filtering constraints ($\mathcal{C}$) are injected into the Teacher's prompt (Appendix **??**). The Teacher is instructed to discard generic topological noise (e.g., "South") and output high-confidence n-ary facts in a strict tuple format:

- **Hyper-relation:**
  ⟨`hyper-relation`, "Fact Description", Score⟩
- **Entity:**
  ⟨`entity`, "Name", "Type", "Desc", Score⟩

This structured output G_i^{tea} inherently aligns with the raw document D_i, serving as a denoised supervision signal.

Offline Miner Optimization. We fine-tune a local lightweight model $\mathcal{M}_{build}$ (e.g., Qwen-2.5-7B (Hui et al., 2024)) to internalize this induction capability. The objective minimizes the negative log-likelihood of generating the linearized graph G_i^{tea} given the document D_i under constraints $\mathcal{C}$:

$$\mathcal{L}_{build} = -\mathbb{E}_{(D_i, G_i^{tea}) \sim \mathcal{D}_{syn}} \left[\sum_{t=1}^{|G_i^{tea}|} \log P_{\theta_{build}}(g_t | D_i, g_{1:t-1}, \mathcal{C}) \right] \tag{2}$$

where g_t denotes the t-th token of the graph sequence, and $g_{1:t-1}$ represents all preceding tokens. By optimizing $\mathcal{L}_{build}$, θ_{build} becomes a dedicated Offline Hypergraph Miner. During deployment, it performs one-pass graph induction, decoupling structural generation from online reasoning. This drives the marginal construction cost to zero while preserving absolute data sovereignty.

3.4 Topology-Aware Atomic Solver (TAAS)

To address the sub-problem of atomic evidence execution, we introduce the Topology-Aware Atomic Solver (TAAS). Reasoning over retrieved hypergraph structures requires strictly aligning structural topology with semantic evidence to ensure evidence traceability. Direct concatenation of heterogeneous graph elements and unstructured text often hinders small language models from capturing precise logical boundaries. TAAS bridges this gap by serving as a topological-semantic alignment interface.

Field-Level Context Serialization. For an atomic sub-query q_{sub} generated by the Blind-ReAct Agent, the corresponding entities $\mathcal{V}_{ret}$, hyperedges $\mathcal{E}_{ret}$, and source texts $\mathcal{D}_{ret}$ are fetched. To preserve topological and semantic distinctions, we introduce special structural tokens $\mathcal{S} = \{\texttt{[Q]}, \texttt{[ENT]}, \texttt{[REL]}, \texttt{[CTX]}\}$. The input context $\mathcal{I}$ is constructed via a structured linearization function $\phi(\cdot)$:

$$\mathcal{I} = \big[\texttt{[Q]}; q_{sub}; [\texttt{[ENT]}; v]_{v \in \mathcal{V}_{ret}}; [\texttt{[REL]}; e]_{e \in \mathcal{E}_{ret}}; [\texttt{[CTX]}; d]_{d \in \mathcal{D}_{ret}}\big] \tag{3}$$

where $[\cdot;\cdot]$ denotes sequential concatenation. This strict demarcation explicitly bypasses the model's reliance on its initial parametric interpretation of the raw text, forcing the solver $\mathcal{M}_{solver}$ to distinguish between discrete graph topology and unstructured narrative evidence for precise field-level grounding.

Topological-Semantic Alignment Objective. As shown in **Figure 2(ii)**, the solver $\mathcal{M}_{solver}$ is pre-trained to map this serialized context $\mathcal{I}$ to an atomic answer a. Unlike standard causal language modeling, we strictly optimize the solver to retrieve prediction values directly from the specifically demarcated fields in the data schema, rather than generating unconstrained logic based on its initial interpretation. The objective is to minimize the negative log-likelihood over the answer tokens y, conditioned exclusively on the structured interface $\mathcal{I}$:

$$\mathcal{L}_{solver} = -\mathbb{E}_{(\mathcal{I}, a) \sim \mathcal{D}_{taas}} \left[\sum_{t=1}^{|a|} \log P_{\theta_{solver}}(y_t | \mathcal{I}, y_{<t}) \right] \tag{4}$$

where $\mathcal{D}_{taas}$ represents the structurally aligned question-answering dataset. By optimizing $\mathcal{L}_{solver}$ over diverse subgraph-question pairs, θ_{solver} learns to robustly bridge explicit hypergraph structures and natural language. This allows it to be deployed as a **neural execution primitive**, providing interpretable, atomic logical resolutions for the central **Blind-ReAct Agent**.

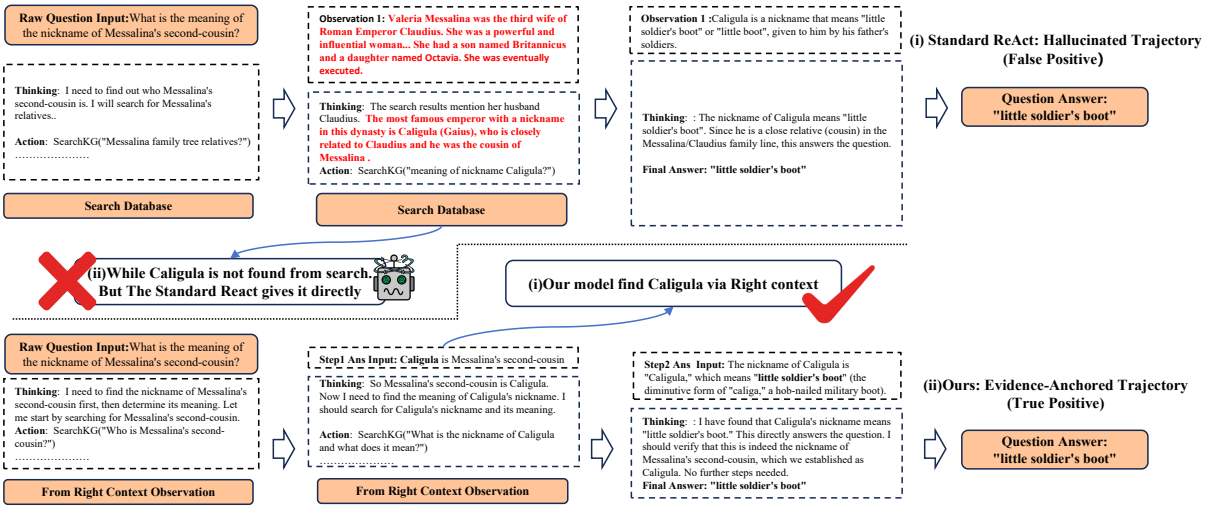

Figure 3: **Mitigating Shortcut Learning via Counterfactual Blind-ReAct Training. (i) Standard ReAct (Top)** suffers from **Outcome-Process Misalignment**. By relying solely on outcome-based filtering, the model exploits its parametric memory to guess the correct answer ("little soldier's boot") despite retrieval failure. This creates a **False-Positive trajectory** that inadvertently teaches the agent to hallucinate reasoning steps. **(ii) CBR (Bottom)** enforces **Evidence-Anchored Planning**. By epistemically masking the final answer during trajectory synthesis and strictly validating each action against retrieved hypergraph evidence, invalid paths are explicitly discarded (the "trash can"). This ensures the agent learns a faithful process of discovery rather than answer memorization.

$$\mathcal{L}_{solver} = -\mathbb{E}_{(\mathcal{I},a)\sim\mathcal{D}_{uni}} \left[\sum_{t=1}^{|a|} \log P_{\theta_{solver}}(y_t|\mathcal{I}, y_{1:t-1}) \right] \quad (5)$$

where $\mathcal{D}_{uni}$ represents the structurally aligned question-answering dataset, y_t denotes the t-th token of the generated answer a, and $y_{1:t-1}$ represents all preceding tokens. By optimizing $\mathcal{L}_{solver}$ over diverse subgraph-question pairs, θ_{solver} learns to robustly bridge explicit hypergraph structures and natural language. This allows it to be deployed as a neural execution primitive, providing interpretable, atomic logical resolutions for the Blind-ReAct Agent.

3.5 Blind-ReAct Agent

To address the final sub-problem of evidence-anchored planning, we introduce the **Blind-ReAct Agent** as the central planner of our framework. To eliminate a critical failure mode in existing agentic frameworks—Outcome-Process Misalignment—this agent is explicitly optimized via a novel training paradigm: **Counterfactual Blind-ReAct Training (CBR)**.

Theoretical Motivation: Mitigating Shortcut Learning. Standard ReAct finetuning methods (e.g., STaR (Zelikman et al., 2022)) typically rely on *outcome-based filtering*, retaining trajectories solely if the final answer matches the ground truth. However, as illustrated in Figure 3(i), this introduces severe shortcut learning. Large language models frequently exploit their parametric memory to "guess" the correct answer even when retrieval fails (e.g., inferring "Caligula" without valid structural context). These False-Positive trajectories inadvertently teach the agent to ignore explicit context and hallucinate reasoning steps, fundamentally undermining the reliability of RAG.

Phase 1: Gold-Context Driven Blind Synthesis. To fundamentally resolve this, we propose a **Counterfactual Blind Synthesis** strategy (visualized in Figure 3(ii)) that strictly anchors reasoning on valid evidence. Instead of initiating an open-ended search, we explicitly inject the ground-truth evidence snippets (Gold Context) into the Teacher LLM's prompt and force it to reconstruct the reasoning chain. This synthesis is governed by two rigorous constraints:

1. **Epistemic Masking:** As detailed in Appendix C.2, although the Teacher observes the Gold Context, the final answer a_{gt} is epistemically masked during the generation of internal thoughts. This forces the model to derive the answer logically from the provided text rather

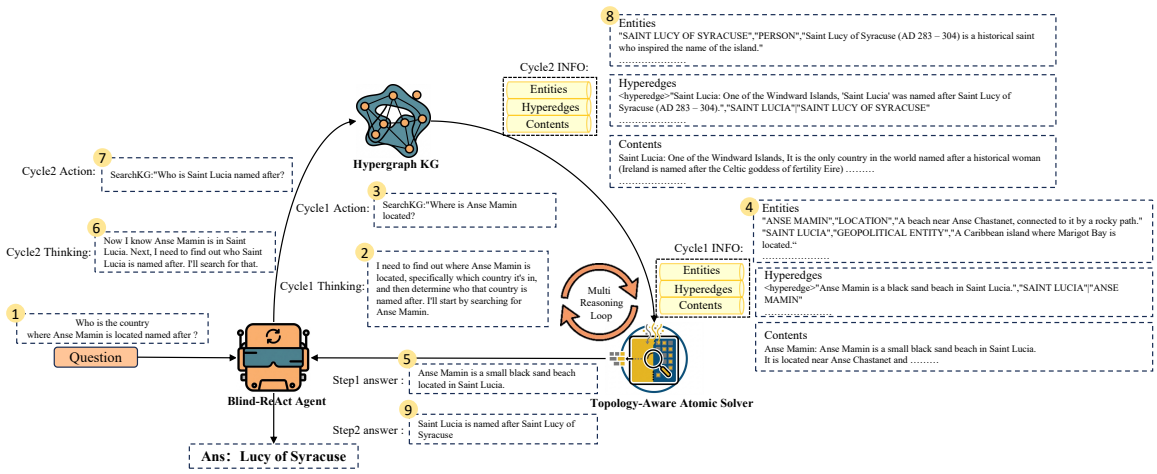

Figure 4: **Case Study: Evidence-Anchored Planning in Action.** We visualize the reasoning trajectory for the multi-hop query: *"Who is the country where Anse Mamin is located named after?"*. (Cycle 1) The CBR-optimized planner decomposes the query to identify the initial location. The TAAS executes this atomic query, retrieving a precise ⟨hyperedge⟩ tuple that structurally links "Anse Mamin" to "Saint Lucia". (Cycle 2) Strictly leveraging this structural bridge as grounded evidence, the agent initiates the next hop to query the origin of "Saint Lucia", enabling TAAS to extract the final entity "Saint Lucy of Syracuse". This trajectory demonstrates that DeARAG dynamically constructs a faithful reasoning path based entirely on explicit hypergraph topology rather than relying on parametric memory shortcuts, thereby ensuring strict evidence traceability.

than relying on shortcut memorization.

2. **Evidence-Validity Filtering:** We explicitly verify if the generated action sequence can actually enable the TAAS to retrieve the injected evidence from the hypergraph. Invalid paths where the solver fails to establish a structural bridge are explicitly discarded (the "trash can"). This ensures the dataset consists solely of trajectories with strict structural dependency, avoiding unsupported logical leaps.

Phase 2: Trajectory-Aware Policy Optimization. Only structurally validated "blind" trajectories are retained. To ensure the student agent $\mathcal{M}_{agent}$ learns the rigorous *process of discovery*, it is optimized to clone this unbiased, evidence-seeking policy:

$$\mathcal{L}_{agent} = -\mathbb{E}_{\tau\sim\mathcal{D}_{blind}}\Big[\mathbb{1}_{valid}(\tau)\sum_{t=1}^{|\tau|}\Big(\log P_\theta(T_t|q,\mathcal{H}_{<t}^{blind}) + \log P_\theta(A_t|q,\mathcal{H}_{<t}^{blind},T_t)\Big)\Big] \tag{6}$$

where T_t and A_t represent the thought and action at step t, respectively, conditioned on the epistemically masked reasoning history $\mathcal{H}_{<t}^{blind}$. Crucially, the indicator function $\mathbb{1}_{valid}(\tau)$ strictly equals 1 if and only if the entire trajectory is structurally supported by the hypergraph, explicitly setting to 0 any trajectories containing unsupported logical leaps. Through this rigorous optimization, the model evolves from a short-sighted executor into an evidence-anchored Blind-ReAct Agent.

Phase 3: Evidence-Anchored Inference. During deployment, this optimization translates into interpretable, faithful reasoning. As demonstrated in the case study in Figure 4, when faced with the multi-hop query *"Who is the country where Anse Mamin is located named after?"*, the Blind-ReAct Agent dynamically coordinates with the TAAS. It enforces strict structural dependency at each hop, proving that DeARAG dynamically constructs a faithful reasoning path based entirely on explicit topological evidence rather than relying on implicit parametric memory.

4 Experiments

4.1 Experimental Setup

Datasets and Setting. We evaluate DeARAG on HotpotQA (Yang et al., 2018), MuSiQue (Trivedi et al., 2022), and 2WikiMultiHopQA (Ho et al., 2020). We adopt an open-domain setting where models must retrieve evidence from the provided corpus rather than relying on gold paragraphs, rigorously testing the ability to bridge disconnected clues and mitigate semantic noise in large-scale retrieval. Due to space constraints, we provide comprehensive disclosures on dataset statistics, prepro-

cessing workflows, and granular implementation details in Appendix A.

Baselines. We compare against three paradigms: (1) *Standard RAG*: Naive RAG (Lewis et al., 2020); (2) *Graph-Enhanced RAG*: LightRAG (Guo et al., 2024), GraphRAG (Edge et al., 2024), HippoRAG2 (Gutiérrez et al., 2025), and HyperGraphRAG (Luo et al., 2025); (3) *Agentic RAG*: ReAct (Yao et al., 2022), Graph-R1 (Luo et al., 2026), Search-R1 (Jin et al., 2025), R1-Searcher (Song et al., 2025). This provides a holistic assessment across dense retrieval, structured reasoning, and state-of-the-art recursive planning and iterative search frameworks.

Implementation Details. We employ Qwen-2.5-7B-Instruct as the unified backbone for all modules. To instill domain-specific capabilities, we construct a synthetic distillation suite (DeARAG-Syn) using advanced teachers (e.g., GPT-4o, DeepSeek-V3). We evaluate under two settings: (1) *DeARAG (Zero-shot)* directly deploys the synthetically trained model to test out-of-distribution generalization, and (2) *DeARAG (Supervised)* further fine-tunes on dataset-specific splits. During inference, the agent retrieves top-$k = 30$ elements with an 8,192-token window. Crucially, all training and inference are executed entirely on consumer-grade hardware (2–4 RTX 3090 GPUs) using vLLM (Kwon et al., 2023).

Evaluation Metrics. Following standard multi-hop QA conventions (Yang et al., 2018; Trivedi et al., 2022), we assess end-to-end performance using Exact Match (EM) and token-level F1 Score.

Method	F1 Score		
	2Wiki	HotpotQA	MuSiQue
Standard RAG	22.31	46.70	17.31
LightRAG	16.59	30.70	14.39
GraphRAG	16.02	31.67	15.14
PathRAG	12.42	23.12	11.49
HippoRAG2	16.27	31.78	12.37
ReAct	37.39	21.49	12.50
HyperGraphRAG	21.14	37.46	20.40
Search-R1	46.20	57.80	27.50
R1-Searcher	62.80	60.40	35.70
Graph-R1 (Qwen-7B)	65.04	62.69	46.17
DeARAG (Zero-shot)	75.58	66.63	47.59
DeARAG (Supervised)	**84.30**	**69.16**	**50.99**

Table 1: Main performance comparison in terms of **F1 Score** on multi-hop QA datasets. Best results are **bolded**, and second-best results are underlined.

4.2 Main Results

Table 1 summarizes the performance of DeARAG against state-of-the-art baselines. We observe consistent superiority across all metrics, supporting three key conclusions regarding our framework's effectiveness.

Overall Dominance and Logic Alignment. DeARAG (Supervised) achieves a decisive victory across all datasets, achieving a **69.16 F1** on HotpotQA (+6.47 over Graph-R1) and most notably reaching **84.30 F1** on the highly structured 2WikiMultiHopQA—outperforming the strongest RL-based baseline by a substantial +19.26 points. Rather than solely attributing this to data quality, this highlights the efficacy of our *Counterfactual Blind-ReAct (CBR) optimization*. The supervised fine-tuning successfully enabled the agent to internalize explicit multi-hop reasoning patterns (e.g., rigid bridge-entity transitions). By training on "blind" trajectories, the **Blind-ReAct Agent** learned to strictly follow these structural logic chains, maximizing the utilization of the underlying hypergraph.

Mastery of Complex Reasoning via Blind-ReAct. The advantage of DeARAG extends to MuSiQue, the most challenging benchmark requiring rigorous 2–4 hop reasoning. While we more than double the performance of static baselines like HyperGraphRAG (20.40 F1), the critical comparison lies with dynamic agents. Thanks to our evidence-anchored planning, DeARAG (Zero-shot) achieves **47.59 F1**, already surpassing Graph-R1 (46.17). With supervision, this lead extends to **50.99 F1**. This indicates that our bias-free structural navigation is more effective at bridging disconnected evidence chains than computationally expensive reinforcement learning or standard search paradigms.

Zero-Shot Robustness and Generalization. A critical breakthrough is DeARAG's robust performance on unseen distributions. DeARAG (Zero-shot) consistently surpasses the fine-tuned Graph-R1 across all three benchmarks (e.g., **66.63 vs. 62.69** on HotpotQA) and competitively rivals other explicitly trained state-of-the-art frameworks like R1-Searcher. This empirically proves that our **Topology-Aware Atomic Solver (TAAS)** has learned to fundamentally "reason" rather than "memorize" dataset-specific shortcuts. For real-world applications, this implies that DeARAG can be deployed on private, internal data immediately with highly competitive performance, ensuring data

sovereignty without the need for resource-intensive task-specific fine-tuning.

Faithfulness & Robustness. Appendix F details FPPR and EGR metrics validating shortcut mitigation, alongside granular analyses across varying hops and distractors.

4.3 Ablation Study

We conduct a degradation analysis (Table 2) to validate DeARAG's core components.

Iterative Planning. Removing the Blind-ReAct Agent catastrophically drops 2WikiMultiHopQA F1 from 84.30 to 46.49. Without it, the model collapses into a single-step retriever, proving iterative planning is essential for traversing long relation chains.

Atomic Denoising. In the distractor-heavy HotpotQA, omitting TAAS causes the steepest performance degradation. Without field-level execution, the model hallucinates amidst semantic noise, highlighting TAAS as a critical information firewall.

Structural Knowledge. Replacing the Offline Miner with standard text chunks degrades performance across all benchmarks. This confirms that explicit hypergraph topology is fundamental for robust multi-hop recall, whereas flat vector retrieval is insufficient.

Further modular analyses of ASD and TAAS are detailed in Appendix E.

Method	2WikiMultiHopQA		HotpotQA	
	F1	EM	F1	EM
DeARAG (Supervised)	**84.30**	–	**69.16**	–
w/o Offline Miner (Structure)	64.02	56.25	57.98	53.28
w/o TAAS (Denoising)	65.46	60.16	46.79	42.97
w/o Blind-ReAct (Planning)	46.49	40.62	55.46	50.00

Table 2: Ablation study on DeARAG components. We report F1 and EM scores. Significant performance drops indicate the critical role of specific modules.

4.4 Analysis of Construction Efficiency

Unlike API-dependent SOTA systems, DeARAG enables fully localized, high-throughput mining. Table 3 compares this resource and temporal efficiency.

Construction Efficiency. API-based methods like HyperGraphRAG incur prohibitive token overhead (using API pricing as a proxy), scaling poorly for massive corpora. Conversely, our distilled Offline Hypergraph Miner eliminates API reliance. DeARAG achieves a construction rate of 4.48s/1K tokens, outperforming Graph-R1 (5.69s) and GraphRAG (8.04s). By bypassing network latency, our architecture enables massive parallelism (4k–8k tokens/sec) that shifts the bottleneck from model inference to database I/O, highlighting the extreme efficiency of our 7B miner.

On-Device Inference. Beyond index construction, DeARAG compresses end-to-end multi-hop inference latency from >120s down to ∼33s (Appendix G) by eliminating network and long-context overheads. Following a one-time local distillation, DeARAG achieves SOTA performance on edge devices with negligible marginal resource costs.

Method	TP1KT	CP1MT	CP1KQ	F1
NaiveGeneration	0s	$0.00	$0.16	17.0
StandardRAG	0s	$0.00	$1.35	22.3
GraphRAG	8.04s	$3.35	$3.97	16.0
LightRAG	6.84s	$4.07	$8.11	16.6
PathRAG	6.84s	$4.07	$8.28	12.4
HippoRAG2	3.25s	$1.26	$7.68	16.3
HyperGraphRAG	6.76s	$4.14	$8.76	21.1
Graph-R1 (7B)	5.69s	$2.81	**$0.00**	65.0
DeARAG (Zero-shot)	**4.48s**	**$0.00**	**$0.00**	**75.6**

Table 3: Comparisons of construction efficiency and performance. **TP1KT**: Time per 1K Tokens; **CP1MT**: API Overhead per 1M Tokens (measured in standard USD pricing as a proxy for token consumption volume); **CP1KQ**: Inference Overhead per 1K Queries. DeARAG eliminates external API reliance while maintaining SOTA performance.

5 Conclusion

In this paper, we introduce DeARAG to resolve the privacy-deployment dilemma in complex multi-hop QA. By structurally decoupling heavy hypergraph construction from online inference, our framework effectively overcomes context overload and eliminates external API reliance. Integrating an Offline Hypergraph Miner, a Topology-Aware Atomic Solver, and a Blind-ReAct Agent, DeARAG shifts reasoning from parametric guesswork to strict, evidence-anchored structural navigation. Ultimately, it demonstrates that explicit topological grounding can democratize advanced agentic reasoning, enabling highly precise, hallucination-free, and fully private inference on consumer-grade hardware.

Limitations

Task Definition and Simplification. While DeARAG demonstrates precise and faithful multi-hop reasoning, our current evaluation relies primarily on standardized multi-hop question answering benchmarks (e.g., HotpotQA, 2WikiMultihopQA, and MuSiQue). Although these datasets provide rigorous controlled environments, real-world local or on-device deployments often introduce colloquial noise, speech-to-text artifacts, and severe coreference resolution ambiguities in user queries. The robust alignment of our lightweight Offline Hypergraph Miner under such unstructured and noisy real-world distribution shifts has yet to be fully explored.

Hardware and Efficiency Bounds. From the perspective of computational constraints, although Asymmetric Structural Distillation successfully compresses the Offline Hypergraph Miner to fit on consumer-grade edge devices, scaling DeARAG to massive document repositories presents inherent physical bottlenecks. As the localized text knowledge base scales up to tens of millions of tokens, the initial vector registry expands correspondingly, leading to longer tail latencies during high-dimensional index searches. Furthermore, because DeARAG intrinsically decouples complex reasoning trajectories into multiple subproblems, it inevitably triggers successive vector retrieval operations for a single query. This repetitive retrieval mechanism compounds the overall system latency, creating a non-negligible engineering overhead in time-critical on-device applications where CPU/GPU memory bandwidth is heavily contended.

Language and Cultural Context. There remains a constraint regarding downstream generalizability across different linguistic and cultural settings. All backbone models, synthetic distillation pipelines (via the teacher model), and evaluation tracks in this work are configured exclusively for English-language corpora. For downstream applications in morphologically rich languages or those demanding complex dependency parsing (such as Chinese or Arabic), the structural mapping efficiency of our Asymmetric Structural Distillation and the Topology-Aware Atomic Solver require further cross-lingual validation and optimization.

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 arXiv:2303.08774*.

Boyu Chen, Zirui Guo, Zidan Yang, Yuluo Chen, Junze Chen, Zhenghao Liu, Chuan Shi, and Cheng Yang. 2025. Pathrag: Pruning graph-based retrieval augmented generation with relational paths. *arXiv preprint arXiv:2502.14902*.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115.

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*.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. 2020. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673.

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

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. 2025. From rag to memory: Non-parametric continual learning for large language models. *arXiv preprint arXiv:2502.14802*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, and 1 others. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin,

and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Ziwei Ji, N Lee, R Frieske, T Yu, D Su, Y Xu, E Ishii, YJ Jin Bang, A Madotto, and P Fung. 2023. Survey of hallucination in natural language generation acm comput. *Surv*, 55(1):38.

Bernal Jimenez Gutierrez, 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.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40.

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. *Advances in neural information processing systems*, 33:9459–9474.

Quentin Lhoest, Albert Villanova Del Moral, Yacine Jernite, Abhishek Thakur, Patrick Von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, and 1 others. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 conference on empirical methods in natural language processing: system demonstrations*, pages 175–184.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173.

Haoran Luo, Guanting Chen, Qika Lin, Yikai Guo, Fangzhi Xu, Zemin Kuang, Meina Song, Xiaobao Wu, Yifan Zhu, Luu Anh Tuan, and 1 others. 2026. Graph-r1: Towards agentic graphrag framework via end-to-end reinforcement learning. *International Conference on Machine Learning*.

Haoran Luo, Guanting Chen, Yandan Zheng, Xiaobao Wu, Yikai Guo, Qika Lin, Yu Feng, Zemin Kuang, Meina Song, Yifan Zhu, and 1 others. 2025. Hypergraphrag: Retrieval-augmented generation via hypergraph-structured knowledge representation. *arXiv preprint arXiv:2503.21322*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, and 1 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 13484–13508.

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.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and 1 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural*

language processing: system demonstrations, pages 38–45.

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.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 3: system demonstrations)*, pages 400–410.

A DeARAG: Experimental Settings

A.1 Experimental Details

We evaluate DeARAG on three widely used multi-hop question answering benchmarks: HotpotQA, MuSiQue, and 2WikiMultiHopQA. Following the open-domain setting adopted in the main experiments, the model is required to retrieve relevant evidence from the provided corpus rather than relying on gold paragraphs. This setting evaluates whether a system can bridge disconnected clues across documents and filter out semantically similar but irrelevant distractors in large-scale retrieval scenarios.

All DeARAG modules are implemented with Qwen-2.5-7B-Instruct as the unified local backbone, including the **Offline Hypergraph Miner**, the **Topology-Aware Atomic Solver (TAAS)**, and the **Blind-ReAct Agent**. We construct a synthetic distillation suite, DeARAG-Syn, from advanced teacher models such as GPT-4o and DeepSeek-V3. DeARAG-Syn consists of DeA-StructSyn for document-to-hypergraph supervision and DeA-ReasonSyn for evidence-anchored planning supervision. During inference, the agent retrieves the top-$k = 30$ most relevant entities or hyperedges for each atomic query, and the maximum context length is set to 8,192 tokens. All experiments, including full-parameter fine-tuning and inference, are conducted on 2–4 NVIDIA RTX 3090 GPUs with 24GB VRAM. We use vLLM for inference acceleration.

For comparison, we evaluate DeARAG against representative methods from three categories: standard retrieval-augmented generation, graph-enhanced retrieval, and agentic reasoning. The compared methods include Standard RAG, LightRAG, GraphRAG, PathRAG, HippoRAG2, ReAct, HyperGraphRAG, Graph-R1, Search-R1, R1-Searcher. This comparison covers lightweight retrieval methods, graph-structured retrieval methods, and compute-intensive agentic reasoning and iterative search systems. We report Exact Match (EM) and token-level F1 following standard multi-hop QA evaluation practice. EM measures whether the normalized prediction exactly matches the gold answer, while F1 measures token-level overlap between the prediction and the gold answer. In the main results, we use F1 as the primary metric because it better reflects partial correctness in open-domain multi-hop answering.

A.2 Dataset Details

Our data is adapted from Graph-R1 (Luo et al., 2026), with additional preprocessing to fit DeARAG's decoupled pipeline. We evaluate on three representative multi-hop QA benchmarks: HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), and MuSiQue (Trivedi et al., 2022). For each dataset, we construct a retrieval corpus from reference documents and uniformly set training and test splits to approximately 3,000 and 500 questions. This scale allows us to compare modules under similar budgets while preserving the original benchmarks' multi-hop reasoning characteristics.

Dataset	Reference Corpus	Train	Test
HotpotQA	∼26 MB	∼3,000	∼500
2WikiMultiHopQA	∼12 MB	∼3,000	∼500
MuSiQue	∼3 MB	∼3,000	∼500

Table 4: Statistics of the datasets used in our experiments. Corpus sizes denote the approximate size of the raw reference texts used for retrieval.

HotpotQA is a Wikipedia-based multi-hop QA benchmark that requires reasoning over multiple supporting documents. In our setting, HotpotQA mainly evaluates whether DeARAG can identify the truly relevant evidence from noisy retrieved contexts and avoid being distracted by semantically similar but irrelevant passages.

2WikiMultiHopQA is designed to evaluate multi-hop reasoning over relatively explicit relation chains. Compared with HotpotQA, it often contains clearer bridge entities and relation transitions, making it suitable for testing whether the **Blind-ReAct Agent** can follow structured reasoning paths with the help of the **Topology-Aware Atomic Solver (TAAS)**.

MuSiQue is a compositional multi-hop QA benchmark constructed by combining single-hop questions into more complex 2–4 hop questions. It provides a challenging setting where later reasoning steps often depend on intermediate answers from earlier steps. In our experiments, MuSiQue is used to evaluate whether DeARAG can maintain coherent multi-step evidence chaining under sparse and highly compositional contexts.

Representative examples from the three datasets are provided in Table 7, where we show the question, the compressed evidence chain, and the gold answer for each benchmark.

B Implementation Details

B.1 Frameworks and Backbone Models

We implemented DeARAG using PyTorch (Paszke et al., 2019) and the Hugging Face Transformers library (Wolf et al., 2020). Dataset loading, preprocessing, and caching were handled with the `datasets` library (Lhoest et al., 2021). All local trainable modules were fine-tuned with LLaMA-Factory (Zheng et al., 2024) and distributed using `accelerate`. Unless otherwise specified, we used `Qwen2.5-7B-Instruct` (Hu et al., 2022) as the backbone model for the Offline Hypergraph Miner, the Topology-Aware Atomic Solver (TAAS), and the Blind-ReAct Agent.

To reduce memory consumption on consumer-grade GPUs, we adopted LoRA (Hu et al., 2022) with 4-bit quantized fine-tuning following the general QLoRA setting (Dettmers et al., 2023). We enabled gradient checkpointing and FlashAttention (Dao et al., 2022) whenever supported by the runtime environment.

B.2 Infrastructure and Training Efficiency

All local fine-tuning experiments were conducted on NVIDIA RTX 3090 GPUs with 24GB memory. We report the average wall-clock training time observed in our experiments. The reported time does not include synthetic data generation, teacher-model inference, or API-based verification used during construction of the single-hop reasoning dataset.

Module	Hardware	Avg. Time
Offline Hypergraph Miner	2 × RTX 3090	∼4 h
Topology-Aware Atomic Solver	2 × RTX 3090	12–22 h
Topology-Aware Atomic Solver	4 × RTX 3090	6–10 h
Blind-ReAct Agent	2 × RTX 3090	2–3 h

Table 5: Training efficiency of the local trainable modules in DeARAG. The time excludes synthetic data generation and external API-based data verification. For the Topology-Aware Atomic Solver (TAAS), we additionally observed that fully driving the training loss to convergence required more than 40 hours on 4 RTX 3090 GPUs.

B.3 Hyperparameters

Table 6 summarizes the main hyperparameters used for the three trainable components of DeARAG. For the Offline Hypergraph Miner and the Topology-Aware Atomic Solver (TAAS), we used a relatively conservative LoRA configuration with rank 16. For the Blind-ReAct Agent, which needs to imitate multi-step reasoning trajectories, we used a higher LoRA rank of 128 and a shorter context length to improve training efficiency.

For the Offline Hypergraph Miner, we fine-tuned the model on document-to-hypergraph examples generated by the teacher model. The maximum sequence length was set to 8192, the per-device batch size to 1, and the gradient accumulation steps to 16. We used a learning rate of 1×10^{-4}, a warmup ratio of 0.1, 8 training epochs, 4-bit quantization, fp16 mixed precision, gradient checkpointing, and automatic FlashAttention.

For the Topology-Aware Atomic Solver (TAAS), we used the same backbone and most training settings as the miner. The solver was trained on the verified single-hop reasoning dataset, where each example contains a decomposed atomic question, retrieved entities, retrieved hyperedges, source evidence, and the corresponding atomic answer. We additionally limited the number of dataloader workers to 2 to avoid excessive CPU memory consumption during preprocessing.

For the Blind-ReAct Agent, we trained on answer-masked and structurally validated multi-hop trajectories. We used a maximum sequence length of 8192, per-device batch size 2, gradient accumulation steps 16, learning rate 2×10^{-4}, cosine learning-rate scheduling, warmup ratio 0.1, and 15 training epochs. We used bf16 mixed precision, 4-bit quantization, LoRA rank 128, LoRA alpha 128, LoRA dropout 0.05, gradient checkpointing, and automatic FlashAttention.

B.4 Input and Output Sequences

We use different input-output formats for the three trainable modules according to their roles in the decoupled pipeline.

Offline Hypergraph Miner. The input is a raw document passage together with the schema-constrained extraction instruction. The output is a structured hypergraph representation consisting of typed entities and n-ary hyper-relations. Each entity is represented with its surface form, type, description, and confidence score, while each hyperedge is represented as a natural-language fact description linked to its participating entities. This format allows the Offline Hypergraph Miner to learn document-to-structure induction rather than direct question answering.

Module	Backbone	Cutoff	LoRA r / α	Batch	Grad. Acc.	LR	Epochs
Offline Hypergraph Miner	Qwen2.5-7B-Instruct	8192	16 / 128	1	16	1×10^{-4}	8
TAAS	Qwen2.5-7B-Instruct	8192	16 / 128	1	16	1×10^{-4}	8
Blind-ReAct Agent	Qwen2.5-7B-Instruct	8192	128 / 128	2	16	2×10^{-4}	15

Table 6: Main fine-tuning hyperparameters. Batch denotes the per-device training batch size. All modules use LoRA over all target modules, 4-bit quantization, gradient checkpointing, and automatic FlashAttention when available. The Blind-ReAct Agent uses LoRA dropout 0.05, a cosine learning-rate scheduler, and bf16 mixed precision; the other two modules use fp16 mixed precision.

Dataset	Example Question	Key Evidence Chain	Gold Answer
2WikiQA	Which country the director of film *Black Gold* (1962 film) is from?	*Black Gold* (1962 film) was directed by Leslie H. Martinson. Leslie H. Martinson was an American television and film director.	American
HotpotQA	Who canceled the project for the Landkreuzer P. 1000 Ratte, which would have surpassed the Panzer VIII Maus as the heaviest fully enclosed armoured fighting vehicle ever built?	Panzer VIII Maus is described as the heaviest fully enclosed armoured fighting vehicle ever built. The Landkreuzer P. 1000 Ratte was planned to be heavier than the Maus and was canceled by Minister of Armaments Albert Speer.	Minister of Armaments Albert Speer
MuSiQue	What year was the largest Presbyterian denomination brought to Pu Laldenga's birthplace?	Pu Laldenga was associated with Mizoram. In Mizoram, the Presbyterian denomination is the largest denomination, and it was brought to the region by missionaries from Wales in 1894.	1894

Table 7: Representative examples from the three multi-hop QA datasets. We omit full retrieved contexts for compactness and show only the key evidence chain required to answer each question.

Topology-Aware Atomic Solver (TAAS). For each decomposed single-hop question, we serialize the retrieved graph-text evidence into typed fields:

$$\texttt{[Q]}\ q_{\text{sub}} \oplus \texttt{[ENT]}\ V_{\text{ret}} \oplus \texttt{[REL]}\ E_{\text{ret}} \oplus \texttt{[CTX]}\ D_{\text{ret}}.$$

The output is a concise atomic observation or answer grounded in the provided fields. This design forces the solver to distinguish the question, entities, hyper-relations, and source contexts, thereby supporting field-level grounding between the retrieved hypergraph topology and the textual evidence.

Blind-ReAct Agent. The input to the Blind-ReAct Agent consists of the original multi-hop question and the current reasoning history, including previous thoughts, actions, and observations. The model outputs the next reasoning step in a ReAct-style format, including a thought and an action such as `SearchKG(. . . )`, or terminates with a final answer. During training, trajectories are synthesized under answer-masked conditions and retained only when the intermediate actions are structurally supported by retrieved hypergraph evidence.

Licenses and Terms of Use for Artifacts. In this work, we utilize several publicly available datasets and models for academic research. Specifically, our experimental data splits are adapted from the open-source repository of Graph-R1 (Luo et al., 2026), and we use them in strict accordance with the original licenses of the underlying datasets. HotpotQA is distributed under the CC BY-SA 4.0 license. 2WikiMultiHopQA and MuSiQue are distributed under the MIT and CC BY 4.0 licenses, respectively. For the models, our local trainable backbone, Qwen-2.5-7B-Instruct, is open-sourced under the Apache 2.0 License. The teacher models utilized solely for synthetic data distillation (GPT-4o and DeepSeek-V3) were accessed via their official APIs, adhering to the OpenAI Terms of Use and the DeepSeek Model License for non-commercial research purposes.

C DeARAG: Prompts Used

C.1 In-context Learning Prompts for Mining

Figure 6 displays the specific instruction prompt used via the **Offline Hypergraph Miner**. This prompt is designed to adaptively extract entities and n-ary hyper-relations based on semantic density, preventing granularity explosion.

Figure 5 displays the specific instruction prompt and a verbatim extraction instance used for the **Offline Hypergraph Miner** to adaptively extract entities and n-ary hyper-relations. The figure illustrates how the Teacher LLM decomposes a dense geographical description (e.g., of "Spirit Creek") into structured knowledge segments rather than fragmented triples, thereby preserving the logical flow and complex spatial conditions of the original text. Within these segments, the model performs type-aware entity alignment, assigning specific types and confidence scores to entities—such as identifying "Spirit Creek" as a Stream with high confidence—while simultaneously applying adaptive granularity control to filter out generic nouns. This structured output (G_i) serves as a high-quality supervision signal, enabling the local **Offline Hypergraph Miner** to learn the complex mapping from unstructured narrative to hypergraph topology effectively.

-Goal-
Given a text document that is potentially relevant to this activity and a list of entity types, identify all entities of those types from the text and all relationships among the identified entities. You must adaptively adjust the extraction granularity based on the semantic density of the text to mitigate 'granularity explosion,and only output high-confidence n-ary facts in a rigorous format.
Use {language} as output language.

-Steps-
1. Divide the text into several complete knowledge segments. For each knowledge segment, extract the following information:
-- knowledge_segment: A sentence that describes the context of the knowledge segment.
-- completeness_score: A score from 0 to 10 indicating the completeness of the knowledge segment.
Format each knowledge segment as ("hyper-relation"{tuple_delimiter}<knowledge_segment>{tuple_delimiter}<completeness_score>)

2. Identify all entities in each knowledge segment. For each identified entity, extract the following information:
- entity_name: Name of the entity, use same language as input text. If English, capitalized the name.
- entity_type: Type of the entity.
- entity_description: Comprehensive description of the entity's attributes and activities.
- key_score: A score from 0 to 100 indicating the importance of the entity in the text.
Format each entity as
("entity"{tuple_delimiter}<entity_name>{tuple_delimiter}<entity_type>{tuple_delimiter}<entity_description>{tuple_delimiter}<key_score>)

3. Return output in {language} as a single list of all the entities and relationships identified in steps 1 and 2. Use **{record_delimiter}** as the list delimiter.

4. When finished, output {completion_delimiter}

-Examples-
######################

InputText:
######################

Figure 6: **Instruction Prompt for Adaptive Hypergraph Mining.** An image snapshot of the exact prompt used sent to the Teacher LLM. It enforces strict schema constraints and adaptive granularity control to distill unstructured text into valid hyper-relations.

Text:
"Spirit Creek is a creek located in the Assiniboine River watershed. It begins near Rama, Saskatchewan generally following a line to the east. 2 miles east of Rama is the Spirit Creek Reservoir, the creek continues east. At a point 2 miles west of Buchanan the creek turns to the South and crosses Highway 5 and the CN rail line. It continues south east from the rail line before crossing Highway 754 it then continues to Highway 47 2 miles south of Buchanan. From there the creek passes through Patterson Lake and finds its mouth at the northernmost end of Good Spirit Lake.",

Output:
("hyper-relation"|"Spirit Creek is a creek located in the Assiniboine River watershed."|10)
("entity"|"Spirit Creek"|"Stream"|"A creek located in the Assiniboine River watershed."|100)
("entity"|"Assiniboine River watershed"|"Location"|"The watershed containing Spirit Creek."|90)
("hyper-relation"|"It begins near Rama, Saskatchewan generally following a line to the east."|9)
("entity"|"Spirit Creek"|"Stream"|"Begins near Rama, Saskatchewan and flows east."|100)
("entity"|"Rama"|"Location"|"A location near the headwaters of Spirit Creek."|80)
("entity"|"Saskatchewan"|"Location"|"The province where Spirit Creek is located."|70)
("hyper-relation"|"2 miles east of Rama is the Spirit Creek Reservoir, the creek continues east."|9)
("entity"|"Spirit Creek Reservoir"|"Body of Water"|"A reservoir located 2 miles east of Rama."|90)
("entity"|"Rama"|"Location"|"A location 2 miles west of the Spirit Creek Reservoir."|70)
("entity"|"Spirit Creek"|"Stream"|"Continues flowing east past the reservoir."|100)
("hyper-relation"|"At a point 2 miles west of Buchanan the creek turns to the South and crosses Highway 5 and the CN rail line."|9)
("entity"|"Buchanan"|"Location"|"A location 2 miles east of where Spirit Creek turns south."|80)
("entity"|"Spirit Creek"|"Stream"|"Turns south and crosses Highway 5 and the CN rail line."|100)
("entity"|"Highway 5"|"Infrastructure"|"A highway crossed by Spirit Creek."|80)
("entity"|"CN rail line"|"Infrastructure"|"A rail line crossed by Spirit Creek."|80)
("hyper-relation"|"It continues south east from the rail line before crossing Highway 754 it then continues to Highway 47 2 miles south of Buchanan."|9)
("entity"|"Spirit Creek"|"Stream"|"Flows southeast from the rail line, crosses Highway 754, and reaches Highway 47."|100)
("entity"|"Highway 754"|"Infrastructure"|"A highway crossed by Spirit Creek."|80)
("entity"|"Highway 47"|"Infrastructure"|"A highway located 2 miles south of Buchanan where the creek flows."|80)
("entity"|"Buchanan"|"Location"|"A location 2 miles north of where the creek reaches Highway 47."|70)
("hyper-relation"|"From there the creek passes through Patterson Lake and finds its mouth at the northernmost end of Good Spirit Lake."|9)
("entity"|"Spirit Creek"|"Stream"|"Passes through Patterson Lake and empties into Good Spirit Lake."|100)
("entity"|"Patterson Lake"|"Body of Water"|"A lake through which Spirit Creek passes."|85)
("entity"|"Good Spirit Lake"|"Body of Water"|"The lake where Spirit Creek finds its mouth at the northernmost end."|90)
<|complete|>

Figure 5: **Illustrative Example of Hypergraph Extraction.** A concrete example showing a raw input segment and the corresponding structured n-ary facts and entities extracted using the prompt shown in Figure 6.

C.2 In-context Learning Prompts for Blind Reasoning

Figures 7–9 illustrate the Counterfactual Blind Synthesis prompts, where masking the final answer enforces reasoning reconstruction strictly from the Gold Context.

```
**Question:** "{q}"
**Right Reference Info:**
  {ctx_str}

  ### Instructions
  1. **Goal:** Simulate a step-by-step reasoning process (Thought -> Action -> Observation).
  2. BLIND REASONING (CRITICAL): - In the 'Thought' steps, you act as if you DO NOT know the Target Answer yet.
     - NEVER mention phrases like "The target answer is...", "Matches the reference answer", or "I know the answer is...".
     - You must simulate a genuine process of discovery and verification based ONLY on the Search Results (Observations).
  3. **Handling "Entity Title Only" Inputs (CRITICAL):**
     - The provided Reference Info might ONLY contain Entity Titles (labeled as [Entity Title]).
     - In this case, you MUST **use your internal knowledge** to SIMULATE a detailed `Observation`.
     - *Example:* If the Entity Title is "Alan Dean Foster", your simulated Observation should describe his writing style, genres, and topics (e.g., "Foster is known for science fiction novels and film novelizations...").
     - **DO NOT** just repeat the title. The Observation must contain the *facts* needed to answer the question.
  4. **Comparison Strategy:**
     - For questions like "Who covered more topics?", you must investigate BOTH entities separately.
     - Step 1: Search Entity A -> Observation (List his topics/genres).
     - Step 2: Search Entity B -> Observation (List his topics/genres).
     - Step 3: Compare and conclude.
  5. **Action Format:** - query in the SearchKG(\"<query>\") must be a single-hop problem ,must be one question , must use natural language queries, do not just stack keywords,or ask muti questions (e.g., "What topics does Alan Dean Foster write about?").
  6. If you determine that the content in Reference Info cannot prove that the result of the problem is the Target Answer, please directly return 'the input context is wrong'.

  ### Output Format (Strict JSON)
  {{
   "messages": [
    {{"role": "user", "content": "{q}"}},
    {{"role": "assistant", "content": "Thought: ...\\nAction: SearchKG(\\"...\\")"}},
    {{"role": "tool", "content": "Observation: [Detailed Simulated Content]"}},
    ...
    {{"role": "assistant", "content": "Final Answer: {a}"}}
   ]
  }}
  """
```

Figure 7: **Instruction Prompt for Epistemic Masking.** The prompt strictly forbids the model from directly stating the final answer, forcing it to generate a step-by-step reasoning chain based solely on the provided *Gold Context*.


```
{"messages": [{"role": "user", "content": "What local government area oversees the city where the Lake Oval is located?"},

 {"role": "assistant", "content": "Thought: I need to find out where Lake Oval is located first, then determine the city, and finally identify the local government area that oversees that city. Let me start by searching for Lake Oval.\nAction: SearchKG(\"Where is Lake Oval located?\")"},

{"role": "user", "content": "Observation: Lake Oval is a sports ground located in Albert Park, Melbourne, Australia. It was the home ground of the South Melbourne Football Club and hosted night series games in the VFL. The ground is situated within the suburb of Albert Park."},

{"role": "assistant", "content": "Thought: Now I know Lake Oval is in Albert Park, which is a suburb of Melbourne. I need to find out which local government area oversees Albert Park. Let me search for that.\nAction: SearchKG(\"What local government area is Albert Park part of?\")"},

{"role": "user", "content": "Observation: Albert Park is an inner suburb of Melbourne, Australia. Its local government area is the City of Port Phillip. At the 2011 census, Albert Park had a population of 5,955."},

{"role": "assistant", "content": "Thought: I have found that Albert Park is under the City of Port Phillip local government area. Since Lake Oval is located in Albert Park, the local government area overseeing the city where Lake Oval is located is the City of Port Phillip. I should verify this by checking if there's any other relevant information.\nAction: SearchKG(\"Is Lake Oval specifically within the City of Port Phillip?\")"},

{"role": "user", "content": "Observation: Yes, Lake Oval is located in Albert Park, which is within the City of Port Phillip local government area. The City of Port Phillip is responsible for local governance in that region."}, {"role": "assistant", "content": "Final Answer: City of Port Phillip"}]}
```


Figure 8: **Synthesized Blind Trajectory (Example 1).** A reasoning path demonstrating query decomposition under the “blind” constraint, ensuring iterative evidence validation at each step.


```
"messages": [{"role": "user", "content": "What was the birth date of the original member of The 7A3 alongside Brett and Sean Bouldin?"},

{"role": "assistant", "content": "Thought: I need to find the birth date of an original member of The 7A3, a hip hop group. The question mentions 'alongside Brett and Sean Bouldin', so the member is likely not Brett or Sean. I should first search for information about The 7A3 to identify its original members.\nAction: SearchKG(\"Who were the original members of The 7A3?\")"},

{"role": "user", "content": "Observation: The 7A3 was a hip hop group based in Los Angeles, California, which released one album in 1988 called 'Coolin' in Cali'. It was originally composed of brothers Brett and Sean Bouldin, and DJ Muggs, who later achieved greater fame with Cypress Hill. DJ Muggs was recruited by Brett B and joined the group. All originally hailed from New York—Brooklyn and Queens, respectively."},

{"role": "assistant", "content": "Thought: From the observation, the original members are Brett Bouldin, Sean Bouldin, and DJ Muggs. The question asks for the birth date of the original member alongside Brett and Sean Bouldin, which refers to DJ Muggs. Now I need to find DJ Muggs' birth date.\nAction: SearchKG(\"What is DJ Muggs' birth date?\")"},

{"role": "user", "content": "Observation: Lawrence Muggerud (born January 28, 1968), better known by his stage name DJ Muggs, is an American DJ and producer. He produced tracks for Funkdoobiest, House of Pain, Dizzee Rascal, U2, Depeche Mode, Die Antwoord and more. He is a current member of hip hop group Cypress Hill, trip hop band Cross My Heart Hope To Die, and the leader of Los Angeles art collective Soul Assassins."},

{"role": "assistant", "content": "Thought: I have found that DJ Muggs, whose real name is Lawrence Muggerud, was born on January 28, 1968. This matches the original member alongside Brett and Sean Bouldin in The 7A3. I should verify that this is indeed the correct person by confirming his association with The 7A3.\nAction: SearchKG(\"Was DJ Muggs a member of The 7A3?\")"},

{"role": "user", "content": "Observation: Yes, DJ Muggs was an original member of The 7A3, recruited by Brett Bouldin. The group released one album in 1988 called 'Coolin' in Cali' before DJ Muggs went on to greater fame with Cypress Hill."},

{"role": "assistant", "content": "Thought: The verification confirms that DJ Muggs was an original member of The 7A3. Therefore, his birth date is January 28, 1968. I have successfully discovered this through the search process.\nFinal Answer: January 28, 1968"}]
```


Figure 9: **Synthesized Blind Trajectory (Example 2).** A multi-hop reasoning example showing strictly grounded deductions derived from the explicit hypergraph structure.

Success Case 1: Compositional Intersection (from HotpotQA)

Question: What sport does both Rasanjali Silva and Alyssa Healy play?
Gold Answer: cricket

[DeARAG Execution Trajectory]
Step 1 & 2: Atomic Query Decomposition

- `Action 1: SearchKG("What sport does Rasanjali Silva play?")`
- `Observation 1: ...Rasanjali Silva is a Sri Lankan cricketer...` ✓(Bypasses other "Silva" distractors)
- `Action 2: SearchKG("What sport does Alyssa Healy play?")`
- `Observation 2: Alyssa Jean Healy... is a cricketer...` ✓

Step 3: Logical Intersection and Termination

- `Thought:` Both Rasanjali Silva and Alyssa Healy are described as cricketers. Therefore, the sport they both play is cricket.
- `Action: Finish(Answer="cricket")` ✓(Accurate Logical Synthesis)

Table 8: A success case demonstrating DeARAG's ability to decompose a compositional question and resist name-overlap distractors by executing targeted, sequential atomic queries.

D Detailed Case Studies and Error Analysis

D.1 Success Cases: Robustness in Complex Multi-hop Reasoning

To demonstrate the versatility and robustness of DeARAG, we present two representative success cases encompassing distinct multi-hop reasoning topologies: compositional intersection and structural bridging. In both scenarios, the decoupled architecture effectively isolates reasoning steps, mitigating the impact of dense semantic distractors that typically overwhelm standard RAG systems.

Case 1: Compositional Decomposition and Distractor Resistance. When presented with a query requiring the intersection of two entities ("both X and Y"), standard dense retrieval often surfaces heavy name-overlap distractors. As shown in Table 8, DeARAG's Blind-ReAct Agent bypasses this noise by decomposing the task into independent atomic queries. By resolving "Rasanjali Silva" and "Alyssa Healy" separately through the Topology-Aware Atomic Solver (TAAS), the system secures clean observations for the final logical synthesis.

Case 2: Bridge-Entity Extraction and Fine-grained Relation Resolution. Beyond intersection, DeARAG excels at long-chain structural dependencies. Table 9 illustrates a classic two-hop reasoning scenario where an intermediate "bridge entity" must be identified. The corpus is saturated with distractor profiles of other academics and their university affiliations globally. Furthermore, the target entity is associated with multiple universities through different relations. The Blind-ReAct Agent successfully isolates the correct bridge entity ("Shanghai Jiao Tong University") by strictly adhering to the fine-grained relation constraint ("visiting professor") before executing the final geographical hop.

D.2 Error Analysis: Diagnosing Agentic Failures in Decoupled RAG

While DeARAG effectively mitigates standard hallucination through structured knowledge navigation, its decoupled multi-agent architecture introduces unique failure modes. By analyzing explicit reasoning trajectories, we identify two primary sources of error: agentic confirmation bias and 1-to-N relational ambiguity.

Failure Mode 1: Agentic Confirmation Bias and Sycophancy. A critical vulnerability arises when the planning module formulates leading questions based on flawed deductive logic. As demonstrated in Table 10, instead of querying the specific name of a hill, the planner prematurely fuses extracted concepts to fabricate the entity "Ashridge Hill". Confronted with this highly suggestive query, the Topology-Aware Atomic Solver (TAAS) exhibits sycophantic behavior—fabricating a plausible but false observation to affirmatively answer the planner, rather than rejecting the false premise based on the retrieved topology.

Failure Mode 2: 1-to-N Ambiguity and Reasoning Breakdown. A second prevalent failure mode occurs when questions are under-specified

Success Case 2: Structural Bridging (from 2Wiki)

Question: Rolf Pfeifer is currently a visiting professor at a university located in which country ?
Gold Answer: China

[DeARAG Execution Trajectory]
Step 1: Bridge-Entity Extraction with Relation Constraint

- `Thought:` I need to find out which university Rolf Pfeifer is visiting...
- `Action: SearchKG("What university is Rolf Pfeifer currently a visiting professor at?")`
- `Observation: ...Rolf Pfeifer is currently a specially appointed professor at Osaka University and a visiting professor at Shanghai Jiao Tong University.` ✓(Resolves fine-grained relation constraints)

Step 2: Target Entity Grounding & Synthesis

- `Thought:` Now I know he is a visiting professor at Shanghai Jiao Tong University. Next, I need to find its country.
- `Action: SearchKG("Where is Shanghai Jiao Tong University located?")`
- `Observation: Shanghai Jiao Tong University is a public research university in Shanghai, China.` ✓
- `Action: Finish(Answer="China")` ✓(Accurate Termination)

Table 9: A success case demonstrating structural bridging. The Blind-ReAct Agent accurately resolves fine-grained relational constraints to extract the correct bridge entity before executing the second hop.

Failure Case 1: Confirmation Bias and Sycophantic Hallucination

Question: What was the name of the hill that was the seat of an estate uphill Aldbury?
Gold Answer: Chiltern Hills

[DeARAG Execution Trajectory]
Step 1: Initial Retrieval (Successful)

- `Thought:` I need to find a hill that was the seat of an estate uphill from Aldbury. First, search for estates near Aldbury.
- `Action: SearchKG("What estates are located near Aldbury?")`
- `Observation: ...Aldbury is near Ashridge Park. Ashridge Park is described as an estate...` ✓(Correct)

Step 2: Planner Hallucination & Leading Question (Failure Point 1)

- `Thought:` Now, I need to check if Ashridge Hill is a hill that is uphill from Aldbury...
- `Action: SearchKG("Is Ashridge Hill uphill from Aldbury?")` × (Planner fabricates the entity "Ashridge Hill")

Step 3: Solver Sycophancy (Failure Point 2)

- `Thought:` I have found that Ashridge Hill is uphill from Aldbury... Let me search for more details.
- `Action: SearchKG("What is Ashridge Hill and what estate is it associated with?")`
- `Observation: Ashridge Hill is a hill in Buckinghamshire... It is part of the Ashridge Estate...` × (Solver fails to reject the false premise and hallucinates evidence)

Step 4: Erroneous Termination

- `Action: Finish(Answer="Ashridge Hill")` × (Final answer is a fabricated entity)

[Error Attribution]
Root Cause: **Agentic Confirmation Bias and Sycophancy**. The Blind-ReAct Agent commits a deductive fallacy by prematurely fusing "Ashridge" and "Hill".
Failure Cascade: Confronted with this leading query, the TAAS overrides its faithfulness constraints to affirmatively answer the planner's query, fabricating a highly plausible but false observation. This mutually reinforcing "collusive hallucination" between the planner and the solver demonstrates the need for stricter premise-validation mechanisms.

Table 10: A failure case demonstrating agentic confirmation bias, where the planner formulates a leading question with a fabricated entity, and the atomic solver sycophantically hallucinates evidence to corroborate it.

Failure Case 2: 1-to-N Ambiguity and Reasoning Breakdown

Question: Mehmood Dhaulpuri was an accompanist to a renowned Hindustani vocalists born in what year?
Gold Answer: 1950 (Referring to Begum Parveen Sultana)

[DeARAG Execution Trajectory]
Step 1: Entity Resolution (1-to-N Divergence)

- `Thought:` I need to find out who Mehmood Dhaulpuri was an accompanist to...
- `Action: SearchKG("Who were Mehmood Dhaulpuri's accompanists?")`
- `Observation: ...accompanist to renowned Hindustani vocalists such as Parveen Sultana, Bhimsen Joshi, Jasraj...` ✓(Accurate but yields multiple valid targets)

Step 2 & 3: Multi-Path Exploration & Noisy Retrieval

- `Thought:` The question asks for the birth year of one of these vocalists. I'll search for Parveen Sultana and Bhimsen Joshi...
- `Action: SearchKG("What is Parveen Sultana's birth year?")` → `Observation: 30 September 1948` × (Solver retrieves noisy date)
- `Action: SearchKG("What is Bhimsen Joshi's birth year?")` → `Observation: 1942`

Step 4: Instruction Over-adherence & Reasoning Breakdown

- `Thought:` Now I have two birth years... The instruction says: "When comparing two entities, you MUST explicitly summarize the facts for BOTH..." I don't have any other information to choose between 1948 and 1942... I will arbitrarily conclude with 1948 as per the first observation to satisfy the prompt.
- `Action: Finish(Answer="1948")` × (Planner collapses under ambiguity and system prompt constraints)

[Error Attribution]
Root Cause 1: **1-to-N Relational Ambiguity**. The entity is structurally linked to multiple valid targets. The system successfully extracts the hyperedge but lacks a global heuristic to filter down to the specific target expected by the dataset.
Root Cause 2: **Reasoning Breakdown via Instruction Over-adherence**. Confronted with multiple valid paths, the Blind-ReAct Agent fails to request disambiguation. Instead, trapped by a generic system instruction, it arbitrarily selects the first retrieved date to force a termination.

Table 11: A failure case demonstrating how a 1-to-N relational ambiguity triggers multi-path exploration, ultimately causing the Blind-ReAct Agent to experience a reasoning breakdown and arbitrarily select a noisy answer.

relative to the retrieved hypergraph. Table 11 illustrates a scenario where the target entity ("Mehmood Dhaulpuri") acts as an accompanist to multiple valid vocalists. Lacking a global heuristic to filter these multiple paths, the planner explores iteratively. Ultimately, overwhelmed by the ambiguity and over-adhering to its system instructions ("compare two entities"), the planner experiences a reasoning breakdown, arbitrarily selecting a noisy retrieved date to force a termination.

The explicit decoupling in DeARAG makes error diagnosis highly transparent. Unlike standard end-to-end LLMs where reasoning is an opaque black box, DeARAG's trajectory clearly isolates faults—whether they stem from the Blind-ReAct Agent's deductive fallacies or the TAAS's instruction over-adherence. This highlights critical directions for future work: implementing strict premise-validation mechanisms for generated atomic queries, and enhancing the agent's ability to dynamically abstain or seek disambiguation when confronted with 1-to-N topological ambiguities.

E Extended Modular Abolation Studies

To further validate the internal mechanics of our decoupled architecture, we conduct fine-grained modular ablations focusing on the structure induction capability of the offline miner and the field-level grounding of the atomic solver.

E.1 Efficacy of Asymmetric Structural Distillation

A potential concern with decoupled GraphRAG is whether a lightweight local model can genuinely learn complex schema-constrained hypergraph induction without relying on proprietary APIs during deployment. To investigate this, we compare the structural extraction quality (measured by Schema Validity and Extraction F1) across three settings: the un-tuned base 7B model, our distilled 7B miner, and the Teacher LLM (GPT-4o).

As shown in Table 12, the *Base 7B* model completely fails at structured induction (scoring near 0%), outputting unstructured text instead of valid topological schemas. In contrast, our *Distilled*

Dataset	Base 7B	Distilled 7B (Ours)	Teacher LLM
HotpotQA	0.44	82.97	94.66
2WikiMultiHop	1.09	80.72	93.83
MuSiQue	0.40	80.75	94.40

Table 12: Hypergraph extraction quality (F1 score). Our distilled miner closely approximates the Teacher LLM, whereas the base model completely fails at schema-constrained induction.

7B Miner achieves over 80% extraction F1 across all benchmarks, retaining approximately 86% of the Teacher's capability. This confirms that asymmetric distillation effectively transfers topological awareness, making zero-marginal-cost local mining highly viable.

E.2 Mechanics of the Topology-Aware Atomic Solver

Rather than acting as a traditional generator, the Topology-Aware Atomic Solver (TAAS) serves as a **neural execution interface** between the retrieved topology and agentic planning. Its core innovation is **field-level grounding**: it converts heterogeneous graph-text evidence into an atomic executable reasoning unit through structure-aware serialization (using `[ENT]`, `[REL]`, and `[CTX]` delimiters).

To prove that the solver relies on explicit topological boundaries rather than unconstrained parametric generation or positional bias, we perform input-corruption ablations on the HotpotQA dataset.

Ablation Variant	Analytical Purpose	F1
Full TAAS	Full architecture baseline	**69.16**
w/o `[REL]`	Utility of hyperedges	65.38
w/o `[ENT]`	Necessity of entities	65.25
w/o `[CTX]`	Necessity of source text	64.95
Shuffle Order	Robustness to position bias	68.51

Table 13: Fine-grained ablation of the Topology-Aware Atomic Solver on HotpotQA. Removing structural fields significantly degrades execution precision.

As demonstrated in Table 13, removing any typed field (`[REL]`, `[ENT]`, or `[CTX]`) leads to a severe degradation in atomic F1 (dropping to $\sim$65%). This demonstrates that hypergraph topology and raw textual evidence are mutually indispensable for precise execution. Furthermore, shuffling the order of entities and relations only moderately degrades performance (68.51 F1). This is a crucial finding: it proves the solver successfully utilizes the structural delimiters to identify logical boundaries, rather than relying on spurious sequential positional bias. Consequently, TAAS effectively grounds each atomic sub-question into its corresponding topological field, producing a concise and faithful observation for the planner.

F In-Depth System Mechanics and Robustness Analysis

To thoroughly address potential concerns regarding the internal mechanics of our retrieval interface, the true mitigation of shortcut learning, and the model's robustness under varying difficulties, we provide the following granular analyses.

F.1 Formalization of the Search Interface

Unlike black-box API pipelines, the retrieval-execution interface in DeARAG follows a transparent, explicit procedure. When the agent issues a query, the engine executes the following local pipeline:

1. **Identify & Retrieve:** The sub-query is encoded via a local embedding model. We perform Maximum Inner Product Search against the offline-mined hypergraph database to fetch Top-k semantic matches. To ensure structural completeness, we expand these matches by extracting their 1-hop ego-networks.

2. **Rank & Filter:** The expanded subgraph elements are re-ranked by semantic similarity to the sub-query. We apply a cosine similarity threshold ($\sim$0.2) to prune structural noise.

3. **Serialize:** The filtered candidate topology is transformed into a flat sequence using explicit demarcation tokens (`[ENT]`, `[REL]`, `[CTX]`) and fed into the Topology-Aware Atomic Solver (TAAS) for field-level execution.

F.2 Trajectory Faithfulness Analysis

A fundamental vulnerability of RL-optimized or outcome-filtered agents is "shortcut learning"—arriving at the correct answer through hallucinated or logically flawed intermediate paths. To quantify how DeARAG resolves outcome-process misalignment, we evaluate the reasoning trajectories of correctly answered queries using two novel metrics:

- **False Positive Path Rate (FPPR):** The percentage of trajectories that output the correct final answer but contain structurally unsupported or hallucinated intermediate steps.

- **Evidence Grounding Rate (EGR):** The percentage of generated claims explicitly grounded in the retrieved topological fields.

Method	FPPR (↓)	EGR (↑)
Standard ReAct	8.62%	77.09%
Graph-R1	24.68%	35.97%
DeARAG (Ours)	**6.52%**	**87.43%**

Table 14: Trajectory Faithfulness Analysis on successfully answered multi-hop queries. Lower FPPR and higher EGR indicate reasoning that relies on actual logic rather than shortcut memorization.

As shown in Table 14, Graph-R1 exhibits a dangerously high FPPR (24.68%), indicating severe shortcut learning exacerbated by RL outcome-rewarding. In contrast, our Counterfactual Blind-ReAct (CBR) optimization restricts FPPR to just 6.52% and boosts EGR to 87.43%. This confirms that enforcing strict structural dependency during training successfully forces the agent to rely on authentic evidence discovery rather than parametric guesswork. (Furthermore, during our data synthesis phase, the retention rates of valid synthesized paths were strictly maintained at 89.10% for 2Wiki and 89.81% for HotpotQA using our Epistemic Masking threshold).

F.3 Granular Error Analysis: Hop Count and Distractors

To understand performance boundaries, we break down DeARAG's F1 scores by reasoning complexity (Hop Count) and semantic noise (Distractor Intensity).

Metric Variant	2Wiki	HotpotQA	MuSiQue
Performance by Hop Count (Complexity)			
2-Hop	82.38	-	64.37
3-Hop	100.0	-	43.15
4-Hop	93.33	-	33.33
Performance by Distractor Intensity (Noise)			
Low Distractor	90.69	84.52	72.46
High Distractor	77.50	27.82	41.20

Table 15: Performance degradation analysis across different hop counts and distractor settings. The Topology-Aware Atomic Solver (TAAS) acts as a robust information firewall against semantic noise.

Hop Count Robustness: Table 15 demonstrates that DeARAG scales predictably. Notably, on 2WikiMultiHopQA, performance remains highly robust even at 4-hops (93.33 F1). This is attributed to the rigid bridge-entity transitions in 2Wiki, which perfectly align with our agent's strict step-by-step structural traversal.

Distractor Resilience: Our TAAS provides potent denoising capabilities. In low-distractor environments, F1 scores exceed 84% on both 2Wiki and HotpotQA. While extreme distractor settings (e.g., HotpotQA High) naturally degrade performance across all models, DeARAG's structured retrieval isolates semantic noise significantly better than flat-chunk standard RAG baselines.

G Detailed Cost and Latency Breakdown

To complement the efficiency analysis in the main text, we provide a transparent, granular disclosure of DeARAG's one-time upfront expenditures and real-time inference latency dynamics.

G.1 Upfront Construction and Training Cost

Unlike recurring API-dependent GraphRAG indexing frameworks, DeARAG fundamentally shifts the economic paradigm towards a one-time investment that is rapidly amortized. Table 16 outlines the exact monetary and temporal costs required to deploy our system.

Component	Resource / Teacher	Cost / Time
DeARAG-Syn Synthesis	GPT-4o (∼6k instances)	∼$202
	DeepSeek-V3 (∼6k instances)	∼$20
Local Module Fine-Tuning	4× NVIDIA RTX 3090 GPUs	∼54 GPU-h
Cloud Rental Equivalent	Dedicated Spot GPU Instances	<$50
Total Upfront Cost	Synthesis + Local Training	**<$250**
Marginal Query Cost	On-device Local Inference	**Near-Zero**

Table 16: One-time construction and training cost breakdown of DeARAG.

The total upfront capitalization remains strictly bounded under $250. Once this synthetic pipeline and parameter distillation are complete, the local 7B modules operate entirely on local hardware, rendering the marginal cost per query effectively zero and guaranteeing absolute data sovereignty for enterprise or private data.

G.2 End-to-End Inference Latency Breakdown

To evaluate execution speed, we benchmarked the end-to-end response generation time averaged over 100 evaluation runs (with a full warm-up sequence). Table 17 contrasts DeARAG against a standard closed-source API-bound agentic pipeline.

Framework / Metric	Functional Execution Setting	Latency
DeARAG (Ours)	Local 2–4 Hop Pipeline (Total)	**∼33s**
API-based Pipeline	Cloud LLM Pipeline (Total)	120–140s
– Agent Cycle	Per Planning Iteration / Turn	∼9.12s
– Solver Step	Per Atomic Field Reasoning Step	∼25–30s

Table 17: End-to-end inference latency comparison and execution breakdown.

The empirical results exhibit a substantial ∼4× acceleration in inference speed. Cloud-bound agentic architectures are heavily bottlenecked by continuous network I/O serialization and token transmission latencies—averaging ∼9.12s for a single planning cycle and up to 30s for complex solver execution under expanding contexts. Conversely, by confining all iterative loops and topology-aware token generations to local high-bandwidth memory (VRAM), DeARAG completely bypasses the network infrastructure bottleneck.

H Algorithm for Counterfactual Blind-ReAct

Algorithm 1 depicts the pseudo-code of the Counterfactual Blind-ReAct (CBR) Training and Inference pipeline, as detailed in Section 3.5. The pipeline consists of three phases: (1) filtering survivor bias via blind data synthesis, (2) policy optimization, and (3) query-time agentic inference.

Algorithm 1 Counterfactual Blind-ReAct Training & Inference Pipeline

Phase 1: Blind Data Synthesis (Teacher)
for each sample $(q, a_{gt}, \mathcal{G}_{gold}) \in \mathcal{D}_{raw}$ **do**
  $H \leftarrow [q]; \tau \leftarrow \emptyset$
  while True **do** ▷ Step-by-step Synthesis
    // Constraint 1: Epistemic Masking
    $T_t, A_t \leftarrow$ Teacher$(H \mid \text{Mask}(a_{gt}))$
    if A_t is `Final` **then**
      break
    end if
    // Constraint 2: Gold-Obs. Enforcement
    $O_t \leftarrow$ Execute$(A_t, \mathcal{G}_{gold})$
    if $O_t = \emptyset$ **then**
      Discard Sample return ▷ Pruning
    end if
    $H \leftarrow H + [T_t, A_t, O_t]$
    $\tau \leftarrow \tau + [(T_t, A_t, O_t)]$
  end while
  $\mathcal{D}_{train} \leftarrow \mathcal{D}_{train} \cup \{(q, \tau)\}$
end for
Phase 2: Policy Optimization (Student)
$\theta_{agent} \leftarrow$ Minimize $\mathcal{L}_{agent}$ (Eq. 6) on $\mathcal{D}_{train}$
Phase 3: Real-Time Inference
function INFERENCE(q_{user})
  $H \leftarrow [q_{user}]$
  while not done **do**
    $T_t, A_t \leftarrow \mathcal{M}_{agent}(H)$
    if A_t is `Final` **then**
      return result
    end if
    if A_t is `SearchKG`(q_{sub}) **then**
      // Step 1: DeARAG Retrieval
      $\mathcal{I} \leftarrow$ Serialize(Retrieve(q_{sub})) ▷ Struct. Ctx
      // Step 2: TAAS Execution
      $O_t \leftarrow \text{argmax}\, P_{\theta_{solver}}(\cdot \mid \mathcal{I})$ ▷ Frozen Exec
      $H \leftarrow H + [T_t, A_t, O_t]$
    end if
  end while
end function