EAGER: Entity-Anchored Evidence Expansion and Refinement for Multi-Hop Retrieval-Augmented Generation

Anonymous ACL submission

Abstract

Multi-hop retrieval-augmented generation (RAG) requires recovering connected evidence paths, since later-hop support often becomes identifiable only after an intermediate entity or fact has been found. Existing decomposition-based methods make intermediate retrieval steps explicit but can over-commit to generated subquestion chains, while graph-based methods rely on offline structures whose topology or granularity may not match a query-specific evidence path. We propose EAGER, an entity-anchored framework for query-time evidence construction. EAGER separates candidate reachability from evidence selection: anchors from the question and evolving evidence state activate local candidate pools, and generated step-level signals guide hierarchical paragraph- and sentence-level refinement into compact support. Under a controlled HippoRAG-style shared-corpus protocol on three multi-hop QA benchmarks, EAGER improves semantic answer accuracy over iterative and structured retrieval controls evaluated under the same protocol, with the clearest gains on datasets that require explicit evidence-path recovery. Ablations further show that generated retrieval signals and hierarchical refinement are both important for turning expanded candidate pools into answer-supporting evidence.

1 Introduction

Retrieval-augmented generation (RAG) grounds language model outputs in external evidence and has become a standard paradigm for knowledge-intensive question answering (Lewis et al., 2020). In multi-hop settings, however, the key challenge is not only retrieving relevant passages, but recovering a connected evidence path: later-hop support often becomes identifiable only after earlier entities or facts have been found, a pattern central to benchmarks such as HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), and MuSiQue (Trivedi et al., 2022). This creates a retrieval bottleneck for standard RAG pipelines: a single-shot retriever may recover first-hop evidence or other topically related passages while missing the next hop that is not explicit in the original question. Prior work on multi-hop dense retrieval and interleaved retrieval-reasoning likewise suggests that complex questions require step-by-step evidence acquisition rather than a single static retrieval context (Xiong et al., 2021; Trivedi et al., 2023). Figure 1 illustrates this bottleneck and motivates viewing multi-hop RAG as an evidence path construction problem.

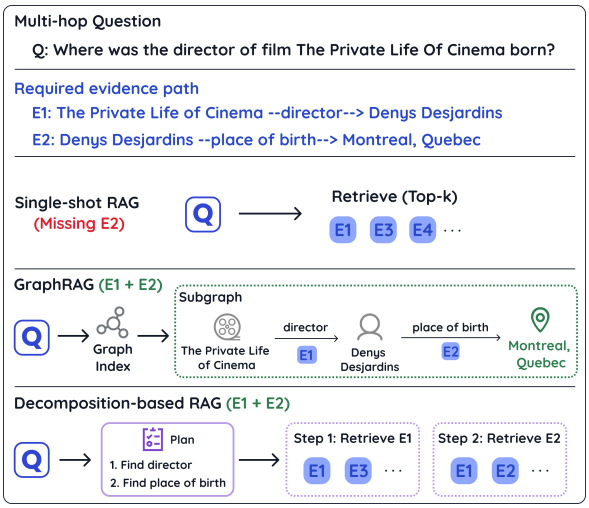

Figure 1: Evidence-path bottleneck in multi-hop RAG. Answering the example question requires both the film–director evidence (E1) and the director–birthplace evidence (E2). Single-shot retrieval misses E2, while graph-based and decomposition-based retrieval can recover both hops through structured evidence or planned retrieval steps.

Existing methods for multi-hop RAG usually add structure to recover missing evidence paths. Decomposition-based methods make intermediate reasoning explicit: Self-Ask (Press et al., 2023)

uses follow-up questions, and IRCoT (Trivedi et al., 2023) interleaves reasoning with retrieval. GraphRAG methods instead organize the corpus before retrieval: GraphRAG (Edge et al., 2024) builds entity graphs and summaries, while HippoRAG (Jiménez Gutiérrez et al., 2024) retrieves through knowledge-graph-based memory.

These methods address the missing-hop problem in Figure 1, but they also introduce new brittleness. Decomposition-based RAG may over-commit to an initial reasoning chain before each subquestion is verified as retrievable and answerable, so an early error can derail later hops. GraphRAG methods, in contrast, may over-commit to a pre-built graph whose entity extraction, topology, or granularity does not fully match the query-specific evidence path, especially when the answer depends on fine-grained sentence-level evidence rather than coarse graph neighborhoods or summaries.

To address these limitations, we propose EAGER, an entity-anchored framework that constructs evidence paths at query time rather than relying solely on a generated decomposition chain or an offline graph. The key design is to decouple where the candidate pool is expanded from how evidence is selected. EAGER extracts retrieval anchors from the question and retained evidence, maps them to entities in the index, and uses the matched entities to activate a local candidate sentence pool; generated signals then guide selection and refinement within that bounded pool rather than serving as the only plan for discovering the next evidence item. This design prevents each supplementation step from repeatedly searching or reranking the full corpus. Although the number of active anchors and the resulting candidate pool size vary by question, the query-time search cost scales with activated entity neighborhoods rather than with the entire shared corpus. EAGER therefore uses anchor-to-entity mapping to delimit a local candidate sentence pool, then refines the resulting candidates into compact sentence-level support. In this way, it preserves the structure needed for multi-hop reasoning while avoiding premature commitment to a brittle chain or a static graph. Experiments on HotpotQA, 2WikiMultiHopQA, and MuSiQue under the HippoRAG-style 1,000-question shared-corpus protocol show that EAGER improves semantic answer accuracy in this controlled multi-hop QA setting, especially when questions require explicit evidence-path recovery (Jiménez Gutiérrez et al., 2024; Zhuang et al., 2026).

Overall, our contributions are as follows:

1. We identify multi-hop RAG as an evidence path construction problem and propose EAGER, a query-time evidence-state construction framework that separates candidate reachability from evidence selection, so generated step signals guide refinement without being the sole determinant of the retrieval trajectory.

2. We introduce a hierarchical evidence refinement mechanism that performs evidence selection across paragraph and sentence levels, turning the expanded candidate pool into a compact and answer-oriented evidence state.

3. We characterize entity-anchored expansion as a retrieval-space control mechanism. By mapping anchors from the question and retained evidence to entity-associated sentences, EAGER defines a bounded local candidate range for each step, avoiding repeated retrieval or reranking over the full corpus at every hop.

2 Related Work

EAGER is closest to two lines of work that add structure to RAG for multi-hop reasoning: methods that control retrieval with generated reasoning or planning signals, and methods that organize evidence in graph-like or hierarchical structures. We focus on how each line constructs evidence paths, rather than surveying RAG systems broadly.

2.1 Active and Agentic Retrieval

Decomposition and agentic retrieval methods make intermediate search states explicit. Self-Ask (Press et al., 2023), ReAct (Yao et al., 2023), and IRCoT (Trivedi et al., 2023) use follow-up questions, actions, or chain-of-thought states to guide later retrieval, while Iter-RetGen (Shao et al., 2023), FLARE (Jiang et al., 2023), Self-RAG (Asai et al., 2024), Adaptive-RAG (Jeong et al., 2024), and LLM-Independent Adaptive RAG (Marina et al., 2025) further show that retrieval can be triggered, routed, or critiqued according to generated states or question complexity.

Other recent systems improve the quality of these control signals through efficient retrievers, filtering, process supervision, or planning. EfficientRAG (Zhuang et al., 2024), BlendFilter (Wang et al., 2024), and Q-PRM (Ye et al., 2025) emphasize that step-level retrieval decisions require filtering and supervision, while RoleRAG (Zhu et al.,

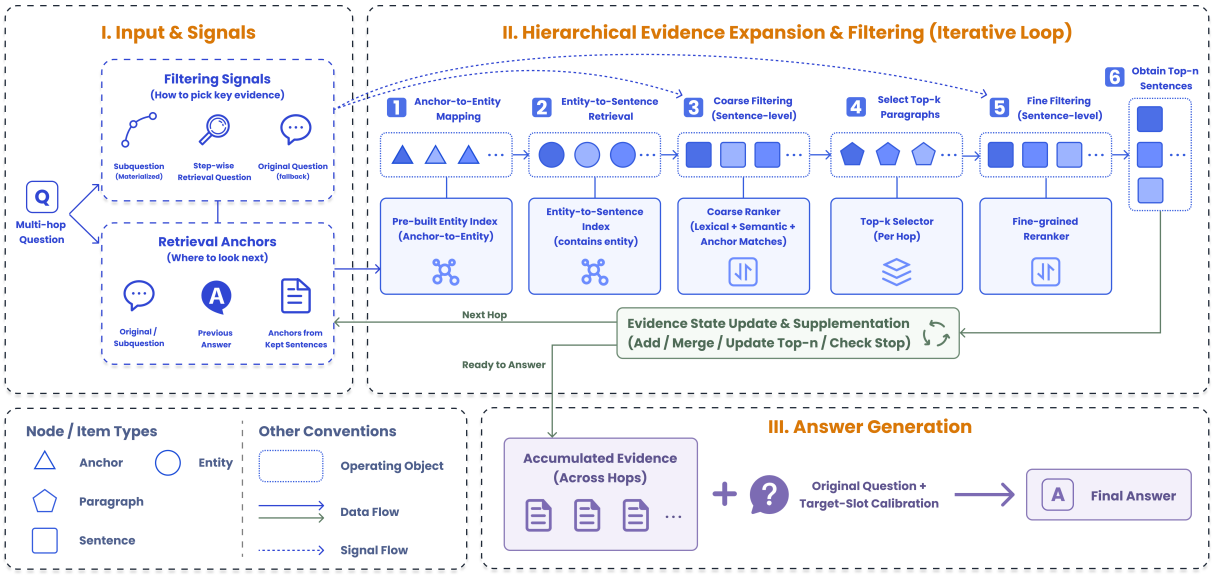

Figure 2: Overview of EAGER. Retrieval anchors determine where candidate evidence is expanded, filtering signals guide hierarchical refinement within the expanded pool, and the retained evidence state is iteratively updated before final answer generation.

2025), UniRAG (Kim et al., 2025), AirRAG (Feng et al., 2025), and LAG (Xiao et al., 2025) organize retrieval through roles, decomposition, strategic planning, or logical subquestion dependencies. EAGER is complementary to this line: it still uses question-like signals for ranking and refinement, but does not let generated questions or plans be the sole source of candidate expansion. Instead, candidate pools are expanded by mapping retrieval anchors from the question and retained evidence to entities in the index, so generated step-level signals operate within a bounded candidate pool rather than defining the entire reachable set.

2.2 Structured Evidence and GraphRAG

Structured RAG methods instead change the retrieval substrate. RAPTOR (Sarthi et al., 2024) builds a recursive summary tree; GraphRAG (Edge et al., 2024) constructs entity graphs and community summaries; HippoRAG (Jiménez Gutiérrez et al., 2024) uses graph-like long-term memory and personalized PageRank; and Think-on-Graph (Sun et al., 2024) and Think-on-Graph 2.0 (Ma et al., 2025) demonstrate the value of traversable entity and relation structures for complex QA.

Recent variants make graph-style retrieval lighter, more path-oriented, or more query-oriented: LightRAG (Guo et al., 2025), PropRAG (Wang and Han, 2025), HydraRAG (Tan et al., 2025), GFM-RAG (Luo et al., 2025), HippoRAG2 (Jiménez Gutiérrez et al., 2025), PathRAG (Chen et al., 2025), LinearRAG (Zhuang et al., 2026), and Youtu-GraphRAG (Dong et al., 2026) all revisit how entities, paths, or memories should support retrieval, while GraphRAG-Bench (Xiang et al., 2026) studies when such graph structures help. These works show that entity connections are useful for recovering multi-hop evidence, but pre-built structures can still suffer from extraction noise, static topology, and granularity mismatch with the query. Filtering-oriented work such as InfoGain-RAG (Wang et al., 2025) also suggests that retrieval must be followed by evidence selection. EAGER follows this entity-connection intuition, but uses entity links to define a query-time candidate range rather than as a fixed traversal result: the expanded pool is refined at paragraph and sentence levels, and the retained evidence state then supplies anchors for later hops.

3 Method

We propose EAGER, an entity-anchored framework that constructs multi-hop evidence paths at query time rather than committing to a fixed decomposition chain or a pre-built graph. As illustrated in Figure 2, EAGER maintains an evolving evidence state organized around four roles: entity expansion defines the reachable candidate range, step-level signals provide directional control within that range, hierarchical refinement performs budgeted evidence selection, and state update exposes anchors for later hops. Thus, generated signals

guide selection rather than define the only retrieval plan, and entity links determine what evidence is reachable rather than what evidence is finally accepted. Each iteration extracts anchors from the question or retained evidence, maps them to entities, expands the candidate pool from the matched entity neighborhoods, and refines the expanded pool with a query-like signal. The goal is not to expose a complete reasoning trace, but to delimit the query-specific evidence neighborhood that will be passed to hierarchical refinement.

3.1 Problem Formulation

Let $\mathcal{D} = \{p_i\}_{i=1}^{N}$ denote a paragraph corpus and $\mathcal{U}$ the sentence-level evidence units derived from it, where each sentence $c \in \mathcal{U}$ keeps its parent paragraph $\pi(c)$. Given a question q, EAGER constructs a compact retained evidence state $\widehat{\mathcal{E}}_T \subseteq \mathcal{U}$ under a budget B, then generates an answer from that constructed state. We view the required support as an implicit evidence path: later evidence often becomes identifiable only after an intermediate entity or fact has been recovered, which is why a single top-k retrieval pass can recover topical first-hop evidence yet miss the path as a whole.

We therefore formulate multi-hop RAG as evidence-state construction under a retrieval budget B. Ideally, the selected state covers the hidden supporting path while excluding distractors, and the final generator answers from the constructed approximation $\widehat{\mathcal{E}}$:

$$\mathcal{E}^* = \arg\max_{\substack{\mathcal{E} \subseteq \mathcal{U} \\ |\mathcal{E}| \le B}} \mathrm{Support}(\mathcal{E}, q),$$
$$\hat{y} = \mathrm{Generate}(q, \tau(q), \widehat{\mathcal{E}}).$$

Here $\mathrm{Support}(\mathcal{E}, q)$ is an idealized utility that is not observable during inference, and $\tau(q)$ denotes a lightweight answer focus inferred from the wording of q, such as the expected answer slot or type. In implementation, this is expressed as an answer-generation instruction: the generator is asked to return the shortest evidence-supported span for the target slot, reducing cases where a bridge entity is returned instead of the final answer. EAGER approximates the ideal objective by constructing $\widehat{\mathcal{E}}$ at query time rather than assuming oracle evidence, a fixed decomposition chain, or a static graph traversal.

3.2 Entity-Anchored Evidence Expansion

Retrieval anchors provide query-time expansion pivots without requiring a relation-complete knowledge graph. They are surface-form entries extracted at inference time and then mapped to a corpus-level entity index. The corpus representation links each sentence to normalized entity aliases, yielding an index from surface forms to entity-associated sentences:

$$\mathcal{I}_e = \{(v, \mathrm{alias}(v), \Gamma(v)) \mid v \in \mathcal{V}\},$$

where $\mathrm{alias}(v)$ contains canonical names and surface variants from entity clusters, title aliases, and string-normalized mentions, and $\Gamma(v)$ returns linked sentences. At iteration t, the anchor set is derived from the original question, an optional retrieval text r_t, and the retained evidence state $\widehat{\mathcal{E}}_{t-1}$. LLM-based extraction captures semantically salient mentions, while lightweight NER/string matching supplements high-recall lexical anchors. Mapping these anchors through $\mathcal{I}_e$ induces the candidate expansion:

$$\begin{aligned} \mathcal{A}_t &= \mathrm{Anchor}(q, r_t, \widehat{\mathcal{E}}_{t-1}), \\ \mathcal{V}_t &= \mathrm{Map}_{\mathcal{I}_e}(\mathcal{A}_t), \\ \mathcal{C}_t &= \bigcup_{v \in \mathcal{V}_t} \Gamma(v). \end{aligned}$$

Here $\mathcal{A}_t$ is a normalized, deduplicated set of surface-form anchors. The optional r_t is present when the active mode provides a materialized sub-question or step-wise retrieval question; otherwise $\mathcal{A}_t$ is derived from q and retained evidence. The index therefore controls reachability rather than final selection: anchors define the entity neighborhood made available for refinement, while the query-like signal in Section 3.3 determines which candidates from $\mathcal{C}_t$ are retained. Entity-anchored expansion should therefore be understood as a candidate-space construction step. It maps surface-form anchors to entity-associated sentence neighborhoods and uses these neighborhoods to define the bounded local candidate pool passed to downstream refinement, so each hop operates over an activated local corpus region instead of repeatedly searching or reranking the full shared corpus. This module does not by itself determine answer accuracy; downstream gains depend on whether signal-guided paragraph- and sentence-level refinement selects useful evidence within the activated range.

3.3 Query-Signal-Guided Hierarchical Refinement

After entity-anchored expansion, $\mathcal{C}_t$ contains sentences reachable from the current anchors. This set defines the candidate range for the current iteration; query-conditioned selection then determines which sentences enter the retained state. EAGER instantiates this filtering signal in three ways: a materialized subquestion in subquestion-chain mode, an online retrieval question generated from the original question and current evidence state in step-wise retrieval-question mode, or no additional step-level question in no-signal mode. The main system selects between the two generated-signal modes with a lightweight rule-based question classifier before inference (deterministic string-pattern rules over the question text, with the routing distribution and classifier rules summarized in Appendix C).

Let s_t denote the step-level text used for scoring at iteration t. When the same text also participates in anchor extraction, s_t corresponds to the retrieval text r_t in Section 3.2; when no step-level signal is available, the corresponding signal weight is set to zero. Thus $\mathcal{A}_t$ defines expansion into $\mathcal{C}_t$, while q and, when available, s_t, define ranking within that expanded pool. EAGER first computes a coarse sentence score:

$$\begin{aligned}\mathrm{Score}_{\mathrm{coarse}}(c) = \lambda_q \,&\mathrm{Rel}(c, q) \\ &+ \lambda_s \,\mathrm{Rel}(c, s_t) \\ &+ \lambda_a \,\mathrm{AnchorHit}(c, \mathcal{A}_t).\end{aligned}$$

The relevance term $\mathrm{Rel}(c, x)$ is a lexical–semantic hybrid score between candidate sentence c and query text x, and $\mathrm{AnchorHit}(c, \mathcal{A}_t)$ adds an anchor-matching feature when c, or its paragraph title, matches the current anchors. The nonnegative weights $\lambda_q, \lambda_s, \lambda_a$ are fixed per signal mode and balance original-question relevance, optional signal relevance, and anchor evidence; in no-signal mode, $\lambda_s = 0$.

EAGER then pools coarse scores at the paragraph level. Let $\pi(c)$ be the paragraph containing sentence c, and let $\mathcal{P}_t$ denote the top-k paragraphs selected by max pooling:

$$\begin{aligned}\mathrm{Score}_{\mathrm{para}}(p) &= \max_{c \in \mathcal{C}_t : \pi(c) = p} \mathrm{Score}_{\mathrm{coarse}}(c), \\ \mathcal{P}_t &= \mathrm{TopK}_p \; \mathrm{Score}_{\mathrm{para}}(p).\end{aligned}$$

This step preserves local context around promising sentences while avoiding fine reranking over the full expanded pool. Within $\mathcal{P}_t$, EAGER applies a fine-grained sentence reranker:

$$\begin{aligned}\mathrm{Score}_{\mathrm{fine}}(c) = \mu_q \,&\mathrm{Rerank}(q, c) \\ &+ \mu_s \,\mathrm{Rerank}(s_t, c) \\ &+ \mu_a \,\mathrm{AnchorHit}(c, \mathcal{A}_t).\end{aligned}$$

Here $\mathrm{Rerank}(x, c)$ denotes the local cross-encoder reranker score for query text x and candidate sentence c, and μ_q, μ_s, μ_a are fixed in the same way as the coarse weights. The no-signal setting again uses $\mu_s = 0$. EAGER finally retains the top-n sentences:

$$\mathcal{S}_t = \mathrm{TopN}_{c \in \mathcal{C}_t : \pi(c) \in \mathcal{P}_t} \, \mathrm{Score}_{\mathrm{fine}}(c).$$

The resulting $\mathcal{S}_t$ is the refined sentence-level evidence passed to the update stage.

3.4 Evidence-State Update and Adaptive Supplementation

Hierarchical refinement yields sentence-level evidence for the current iteration, while the evidence state accumulates across iterations. Let $\widehat{\mathcal{E}}_{t-1}$ denote the retained state before iteration t. The update uses ordered concatenation: previously retained sentences are listed before newly refined sentences, and stable deduplication keeps the first occurrence of each sentence id. The capped state is therefore:

$$\widehat{\mathcal{E}}_t = \mathrm{Cap}_B \left(\mathrm{StableDedup}([\widehat{\mathcal{E}}_{t-1}; \mathcal{S}_t]) \right).$$

Each retained sentence keeps its parent paragraph identifier, so the state preserves both fine-grained evidence units and their surrounding paragraph context. Termination is determined by mode-specific control signals and the iteration budget. Subquestion-chain mode consumes the materialized subquestions in order and stops when the chain is exhausted or $T_{\max}$ is reached; unresolved placeholders or malformed subquestions are treated as missing step-level signals, so the corresponding round uses no-signal expansion from the original question and retained evidence. Step-wise retrieval-question mode prompts the controller to either emit the next retrieval question or stop; if it emits no valid retrieval question, or if $T_{\max}$ is reached, EAGER answers from the current state. No-signal mode sets $s_t = \emptyset$ and runs for a fixed supplementation budget. Whenever another round is launched,

Method	HotpotQA		2WikiMultiHopQA		MuSiQue	
	ConAcc	GPTAcc	ConAcc	GPTAcc	ConAcc	GPTAcc
GPT-4o-mini zero-shot	35.9	32.9	36.9	24.2	13.3	13.9
Vanilla and iterative RAG methods						
Retrieval Top-5 (dense only)	54.3	55.4	42.9	38.4	25.0	28.5
Retrieval Top-5 (Qwen rerank)	53.5	55.2	42.7	36.6	27.5	30.5
IRCoT Top-5 (Prior)	58.4	59.6	53.0	36.8	22.6	26.1
IRCoT-style Top-5	63.1	68.9	67.2	62.7	34.9	38.3
Structured multi-hop RAG systems						
RAPTOR (Prior)	58.1	58.3	60.6	43.9	32.2	29.7
LightRAG (Prior)	61.5	60.5	55.2	39.0	27.7	28.6
HippoRAG (Prior)	57.0	59.3	66.1	59.9	29.3	24.1
GFM-RAG (Prior)	62.7	65.6	66.8	59.6	29.9	34.6
HippoRAG2 (Prior)	62.9	64.3	62.7	58.8	34.5	35.6
LinearRAG	**65.4**	65.4	70.2	64.1	32.3	36.2
EAGER-auto (signal router)	63.2	**70.3**	**74.4**	**70.9**	**41.3**	**45.8**

Table 1: Main results under the HippoRAG-style shared-corpus protocol. Unmarked rows are evaluated in this work on the fixed 1,000-question sets with the shared answer-generation and evaluation stack; rows marked "Prior" are protocol-matched results from prior work and serve as external references rather than same-stack evaluations. ConAcc is string-containment accuracy and GPTAcc is semantic answer accuracy from the shared HippoRAG-style judge.

expansion uses anchors exposed by q and $\widehat{\mathcal{E}}_t$. Failed or malformed signal steps may add no new evidence, but the stable merge keeps previously retained evidence in the state.

4 Experiments

We evaluate whether EAGER improves multi-hop answer generation and whether its evidence-construction stages recover more complete support. The experiments compare EAGER with recent multi-hop RAG baselines, then analyze fixed signal modes and component removals to identify where the gains come from.

4.1 Experimental Setup

Experiments are conducted on HotpotQA, 2WikiMultiHopQA, and MuSiQue under the HippoRAG-style shared-corpus protocol (Jiménez Gutiérrez et al., 2024; Zhuang et al., 2026). For each dataset, methods evaluated in this work answer the same fixed 1,000 questions and retrieve from a dataset-level corpus formed by pooling the passages associated with those questions, rather than from the smaller per-question context pools used in the original benchmarks (Yang et al., 2018; Ho et al., 2020; Trivedi et al., 2022). This setting tests evidence acquisition in a larger shared search space, and should be read as a controlled shared-corpus evaluation rather than an official full-dataset leaderboard. Within this benchmark suite, HotpotQA contains many bridge-style questions, whereas 2WikiMultiHopQA and MuSiQue more often require explicit composition across several support pieces. We therefore examine not only aggregate gains but also where gains concentrate across datasets.

We report ConAcc, GPTAcc, and Full-gold support coverage. ConAcc captures string containment between the prediction and the reference answer, while GPTAcc uses the HippoRAG judge prompt (Jiménez Gutiérrez et al., 2024) to assess semantic equivalence. For rows evaluated in this work, GPTAcc is computed with the same judge prompt, and the judge input contains only the prediction and reference answer, without method names, system identifiers, or retrieval traces. Full-gold support coverage is computed with dataset-specific matching rules and is used as an evidence-recovery diagnostic rather than a standalone answer metric.

The comparison combines rows evaluated in this work under a unified experimental stack with

protocol-matched rows from prior work. The former group includes GPT-4o-mini zero-shot, two Retrieval Top-5 controls, an IRCoT-style Top-5 control, LinearRAG, and EAGER variants; these rows share the fixed question sets, answer-generation and evaluation stack, and comparable retrieval/evidence-budget controls where applicable. Rows marked "Prior" are taken from prior work under related shared-corpus protocols and are included as external references, placing the results in the context of recent shared-corpus multi-hop RAG systems rather than serving as same-stack evaluations. Accordingly, claims that require direct control are drawn from the unified-stack rows, while the prior rows are used for contextual comparison. Additional details on the evaluation protocol, support matching, manual audit, comparison-row configurations, retrieval budgets, routing policy, and implementation settings are provided in Appendix A and Appendix C.

4.2 Main Results

Among the compared rows in Table 1, EAGER-auto achieves the highest GPTAcc on all three benchmarks, reaching 70.3 on HotpotQA, 70.9 on 2WikiMultiHopQA, and 45.8 on MuSiQue. The size of the gain, however, depends on the benchmark. On HotpotQA, EAGER-auto improves over IRCoT-style Top-5 by 1.4 GPTAcc points, suggesting that many bridge-style questions are already well served by iterative retrieval with a compact top-k context. The margins are larger on 2WikiMultiHopQA and MuSiQue, where EAGER-auto improves over IRCoT-style Top-5 by 8.2 and 7.5 points, respectively. The comparison with LinearRAG follows the same trend: EAGER-auto improves GPTAcc by 4.9, 6.8, and 9.6 points on the three datasets. This pattern is consistent with EAGER's evidence-state design: entity anchors expose a local candidate range, and signal-guided refinement selects evidence within that range as the reasoning focus changes. The larger gains on 2WikiMultiHopQA and MuSiQue suggest that this design is most useful when answering requires composing several dependent evidence pieces.

The metric pattern further clarifies where the gains appear. On 2WikiMultiHopQA and MuSiQue, EAGER-auto obtains the highest ConAcc as well as the highest GPTAcc, so the improvements are also visible under stricter string-containment accuracy. By contrast, on HotpotQA, LinearRAG has the highest ConAcc (65.4), while EAGER-auto has the highest GPTAcc (70.3). Since ConAcc depends on strict containment of the reference string, this gap suggests that EAGER often constructs evidence sufficient for a semantically correct answer, but the generated surface form can still differ from the reference answer. We therefore read Table 1 as evidence for improved semantic answerability in this shared-corpus setting, while leaving some answer-realization errors to be analyzed through the ablation and coverage results in Table 2.

4.3 Ablation Study

Table 2 shows that the most effective signal form depends on the question distribution. Step-wise retrieval questions give the best HotpotQA GPTAcc (70.9), while subquestion chains are strongest on 2WikiMultiHopQA and MuSiQue (71.1 and 47.5 GPTAcc). EAGER-auto remains close to these best fixed modes, trailing by 0.6, 0.2, and 1.7 GPTAcc points, respectively, which suggests that the pre-inference router provides a useful single operating point without oracle selection over signal modes. The coverage columns show a related but not identical pattern. On 2WikiMultiHopQA and MuSiQue, the subquestion-chain mode has both the highest Full-gold coverage and the highest answer accuracy, so the answer gains coincide with more complete evidence recovery. HotpotQA is more nuanced: subquestion-chain has the highest Full-gold coverage (83.7), but step-wise retrieval questions answer better. This suggests that complete annotated support alone is insufficient; answer quality also depends on whether the final evidence state remains focused on answer-relevant information.

The component removals clarify how EAGER turns candidate reachability into useful evidence. Removing step-level signals reduces GPTAcc by 7.1, 18.8, and 14.8 points relative to EAGER-auto, and Full-gold coverage drops by 16.7, 35.5, and 31.5 points. This pattern suggests that the generated signal contributes more than an auxiliary prompt: it supplies the query direction used to refine the activated candidate range. Removing hierarchical filtering gives smaller but consistent drops in both GPTAcc and coverage, suggesting that candidate expansion alone leaves too much irrelevant or weakly related material in the state. These results are consistent with the intended division of labor: entity expansion keeps a bounded candidate region reachable, generated signals point refinement toward the current reasoning need, and

Variant	HotpotQA			2Wiki			MuSiQue		
	Full-gold	ConAcc	GPTAcc	Full-gold	ConAcc	GPTAcc	Full-gold	ConAcc	GPTAcc
EAGER-auto (full)	82.0	63.2	70.3	81.4	74.4	70.9	54.2	41.3	45.8
EAGER w/ step-wise retrieval-question signal	79.6	**63.3**	**70.9**	80.9	72.3	68.8	29.9	30.4	35.7
EAGER w/ subquestion-chain signal	**83.7**	57.4	63.3	**87.3**	**74.6**	**71.1**	**60.4**	**43.2**	**47.5**
EAGER w/o step-level signal	65.3	57.3	63.2	45.9	53.9	52.1	22.7	26.3	31.0
EAGER-auto w/o hierarchical filtering	66.7	60.8	66.5	68.2	67.2	63.9	40.8	34.7	39.6

Table 2: EAGER signal-mode and refinement ablations on the fixed shared-corpus sets. All rows use the same answer generator, judge, corpus resources, and evidence budget except for the ablated component. Full-gold measures whether the final retained evidence contains all annotated support: sentence-level support for HotpotQA and 2WikiMultiHopQA, and supporting-paragraph coverage for MuSiQue.

Metric	HotpotQA	2Wiki	MuSiQue
Full corpus (sent.)	40,992	22,268	42,907
Avg. space / question	760.8	376.7	795.2
Compression ratio	1.86%	1.69%	1.85%
Full retained	92.6%	89.9%	80.9%

Table 3: Question-level retrieval-space scale for EAGER-auto. Avg. space / question is the de-duplicated candidate-sentence union before refinement; the final row reports pre-refinement reachability of all annotated support.

hierarchical filtering turns the activated range into evidence that is both complete and useful for answering.

4.4 Retrieval-Space Analysis

Table 3 reports the question-level retrieval space used by EAGER-auto. For each question, we take the de-duplicated union of candidate sentence ids activated across its routed retrieval steps before refinement. This measures the corpus region exposed by entity expansion, rather than the final evidence budget. Although the shared corpora contain 40,992, 22,268, and 42,907 sentences, EAGER activates only 760.8, 376.7, and 795.2 sentences per question on average, corresponding to 1.86%, 1.69%, and 1.85% of the full corpora. These numbers show that the implemented expansion stage exposes a much smaller query-time candidate range than the full shared corpus, rather than requiring every hop to score all corpus sentences.

The activated range still retains most annotated support before refinement: Full retained is 92.6% on HotpotQA, 89.9% on 2WikiMultiHopQA, and 80.9% on MuSiQue. This pre-refinement measure differs from Table 2's Full-gold, which is computed on the final retained evidence state after refinement and budget truncation; here it only asks whether all gold support remains reachable in the candidate range. Example-level analysis suggests that a common failure pattern is first-hop drift, where an incorrect initial neighborhood prevents later evidence-derived anchors from reaching the missing support. Table 3 should therefore be read as a reachability diagnostic: it evaluates the local search space exposed by entity expansion, while answer-level gains depend on signal-guided refinement selecting useful evidence within that space.

5 Conclusion

We introduced EAGER, an entity-anchored framework that constructs multi-hop evidence paths at query time by separating candidate reachability from evidence selection. Under the controlled HippoRAG-style shared-corpus protocol, a single routed EAGER configuration achieves 70.3, 70.9, and 45.8 GPTAcc on HotpotQA, 2WikiMultiHopQA, and MuSiQue, with clearer gains on datasets requiring explicit evidence-path recovery. Ablations and coverage analyses show that generated signals and hierarchical refinement are important for selecting useful support, while retrieval-space analysis shows that the implemented expansion stage exposes a compact candidate range with high pre-refinement support reachability. Overall, EAGER provides a query-time evidence construction alternative to relying solely on fixed decomposition chains or pre-built graph structures.

Limitations

The main practical limitations of EAGER are inference cost and reliance on a local reranker. Each query requires multiple LLM calls to generate step-level signals and answer-focused text, which is still manageable in our setting but remains more expensive than lightweight single-pass RAG pipelines. EAGER also depends on a local reranker whose quality and deployment requirements affect the final system design, so the method is less suitable for environments that cannot afford an additional local ranking stage or repeated model calls. Beyond these deployment costs, the method remains tied to the quality of the generated signals and the retained evidence budget, which means that noisy intermediate outputs can still affect later hops even when the update rule is stable. Finally, our experiments validate EAGER under a controlled HippoRAG-style shared-corpus protocol; extending the same evidence-construction design to broader open-domain corpora will require additional corpus indexing, alias normalization, and surface-matching validation.

References

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

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*.

Junnan Dong, Siyu An, Yifei Yu, Qian-Wen Zhang, Linhao Luo, Xiao Huang, Di Yin, Yunsheng Wu, and Xing Sun. 2026. Youtu-GraphRAG: Vertically unified agents for graph retrieval-augmented complex reasoning. In *The Fourteenth International Conference on Learning Representations*.

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 GraphRAG approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Wenfeng Feng, Chuzhan Hao, Yuewei Zhang, Guochao Jiang, and Jingyi Song. 2025. AirRAG: Autonomous strategic planning and reasoning steer retrieval augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 18934–18953, Suzhou, China. Association for Computational Linguistics.

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

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength natural language processing in Python.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. 2024. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7036–7050, Mexico City, Mexico. Association for Computational Linguistics.

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

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 59532–59569. Curran Associates, Inc.

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*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Gun Il Kim, Jong Wook Kim, and Beakcheol Jang. 2025. UniRAG: A unified RAG framework for knowledge-intensive queries with decomposition, break-down reasoning, and iterative rewriting. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 18795–18810, Suzhou, China. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Linhao Luo, Zicheng Zhao, Gholamreza Haffari, Dinh Phung, Chen Gong, and Shirui Pan. 2025. GFM-RAG: Graph foundation model for retrieval augmented generation. *arXiv preprint arXiv:2502.01113*.

Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, Cehao Yang, Jiaxin Mao, and Jian Guo. 2025. Think-on-graph 2.0: Deep and faithful large language model reasoning with knowledge-guided retrieval augmented generation. In *The Thirteenth International Conference on Learning Representations*.

Maria Marina, Nikolay Ivanov, Sergey Pletenev, Mikhail Salnikov, Daria Galimzianova, Nikita Krayko, Vasily Konovalov, Alexander Panchenko, and Viktor Moskvoretskii. 2025. LLM-independent adaptive RAG: Let the question speak for itself. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 8697–8709, Suzhou, China. Association for Computational Linguistics.

OpenAI. 2024. GPT-4o mini model.

Shon Otmazgin, Arie Cattan, and Yoav Goldberg. 2022. F-coref: Fast, accurate and easy to use coreference resolution. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 48–56, Taipei, Taiwan. Association for Computational Linguistics.

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

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. 2024. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *The Twelfth International Conference on Learning Representations*.

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

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations*.

Xingyu Tan, Xiaoyang Wang, Qing Liu, Xiwei Xu, Xin Yuan, Liming Zhu, and Wenjie Zhang. 2025. HydraRAG: Structured cross-source enhanced large language model reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 14431–14459, Suzhou, China. Association for Computational Linguistics.

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.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Haoyu Wang, Ruirui Li, Haoming Jiang, Jinjin Tian, Zhengyang Wang, Chen Luo, Xianfeng Tang, Monica Xiao Cheng, Tuo Zhao, and Jing Gao. 2024. BlendFilter: Advancing retrieval-augmented large language models via query generation blending and knowledge filtering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1009–1025, Miami, Florida, USA. Association for Computational Linguistics.

Jingjin Wang and Jiawei Han. 2025. PropRAG: Guiding retrieval with beam search over proposition paths. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 6212–6227, Suzhou, China. Association for Computational Linguistics.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems*, volume 33, pages 5776–5788.

Zihan Wang, Zihan Liang, Zhou Shao, Yufei Ma, Huangyu Dai, Ben Chen, Lingtao Mao, Chenyi Lei, Yuqing Ding, and Han Li. 2025. InfoGain-RAG:

Boosting retrieval-augmented generation through document information gain-based reranking and filtering. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 7190–7204, Suzhou, China. Association for Computational Linguistics.

Zhishang Xiang, Chuanjie Wu, Qinggang Zhang, Shengyuan Chen, Zijin Hong, Xiao Huang, and Jinsong Su. 2026. When to use graphs in RAG: A comprehensive analysis for graph retrieval-augmented generation. In *The Fourteenth International Conference on Learning Representations*.

Yilin Xiao, Chuang Zhou, Yujing Zhang, Qinggang Zhang, Su Dong, Shengyuan Chen, Chang Yang, and Xiao Huang. 2025. LAG: Logic-augmented generation from a cartesian perspective. *arXiv preprint arXiv:2508.05509*.

Wenhan Xiong, Xiang Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Scott Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oguz. 2021. Answering complex open-domain questions with multi-hop dense retrieval. In *International Conference on Learning Representations*.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Xiaopeng Ye, Chen Xu, Chaoliang Zhang, Zhaocheng Du, Jun Xu, Gang Wang, and Zhenhua Dong. 2025. Q-PRM: Adaptive query rewriting for retrieval-augmented generation via step-level process supervision. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 15113–15128, Suzhou, China. Association for Computational Linguistics.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 Embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Yutao Zhu, Jiajie Jin, Hongjin Qian, Zheng Liu, Zhicheng Dou, and Ji-Rong Wen. 2025. Single LLM, multiple roles: A unified retrieval-augmented generation framework using role-specific token optimization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 4837–4856, Suzhou, China. Association for Computational Linguistics.

Luyao Zhuang, Shengyuan Chen, Yilin Xiao, Huachi Zhou, Yujing Zhang, Hao Chen, Qinggang Zhang, and Xiao Huang. 2026. LinearRAG: Linear graph retrieval augmented generation on large-scale corpora. In *The Fourteenth International Conference on Learning Representations*.

Ziyuan Zhuang, Zhiyang Zhang, Sitao Cheng, Fangkai Yang, Jia Liu, Shujian Huang, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. 2024. EfficientRAG: Efficient retriever for multi-hop question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3392–3411, Miami, Florida, USA. Association for Computational Linguistics.

This appendix provides evaluation definitions, implementation details, corpus construction details, prompt-template documentation, diagnostic examples, and cost accounting for EAGER.

A Evaluation Protocol and Metrics

All EAGER variants share the same answer-generation and evaluation stack unless a component is explicitly ablated. Shared evaluation settings are as follows:

- **LLM stack:** `gpt-4o-mini` (OpenAI, 2024) for answer generation, signal generation, and GPT-based answer judging.
- **Answering:** target-slot answer prompt.
- **Evaluation:** HippoRAG-style GPTAcc judge prompt; ConAcc; gold evidence coverage.
- **LLM decoding:** temperature 0.0, maximum completion budget 1,600 tokens, 120-second timeout.
- **Retry policy:** failed LLM calls are retried up to two times.

For rows evaluated in this work, GPTAcc is scored with the same HippoRAG-style judge prompt. The judge input contains only the generated answer and the corresponding gold answer, together with a fixed evaluator instruction; it does not include the system name, signal mode, retrieval method, or any other method identifier. We therefore treat GPTAcc as a blind semantic correctness measure under a shared prompt for the unified-stack rows, and report it together with ConAcc rather than as a replacement for deterministic matching. ConAcc provides a conservative string-containment check, while GPTAcc is used to recognize semantically equivalent answers whose surface form differs because of aliases, date formatting, canonicalization, or answer granularity.

Dataset	Sample GPTAcc	Manual accuracy	Disagreements	GPT-judge accuracy
HotpotQA	74 / 100 (74.0%)	82 / 100 (82.0%)	8	92 / 100 (92.0%)
2WikiMultiHopQA	71 / 100 (71.0%)	76 / 100 (76.0%)	5	95 / 100 (95.0%)
MuSiQue	48 / 100 (48.0%)	50 / 100 (50.0%)	4	96 / 100 (96.0%)
Total	193 / 300 (64.3%)	208 / 300 (69.3%)	17	283 / 300 (94.3%)

Table 4: Manual audit of GPTAcc decisions on sampled EAGER-auto outputs. The audit uses 100 seed-0 sampled EAGER-auto predictions per dataset. It characterizes the observed GPTAcc judge behavior on the reported EAGER outputs and is not a full manual audit of all baseline rows.

Because GPTAcc is an automatic semantic-evaluation metric, its absolute values may still be affected by judge calibration and prompt sensitivity. We therefore use GPTAcc as one component of a triangulated evaluation rather than as the sole basis for interpretation. In the main results, GPTAcc is reported alongside ConAcc; in the diagnostic analyses, answer accuracy is further compared with evidence coverage and representative error cases. This combination separates surface-form mismatches, semantic answer correctness, evidence-recovery failures, and answer-generation failures more clearly than any single metric alone.

To characterize the GPT-based answer judge on the reported EAGER outputs, we additionally conduct a manual audit on the main EAGER-auto outputs. For each dataset, we sample 100 examples from the routed 1,000-question run using fixed seed 0 and manually check whether the predicted answer should be counted as semantically correct with respect to the reference answer and question intent. Table 4 reports four quantities: the original sample GPTAcc, the manually verified answer accuracy, the number of cases where the manual decision differs from GPTAcc, and the resulting GPT-judge accuracy, defined as agreement between the automatic judge and the manual check. The judge agrees with manual verification on 92% of HotpotQA examples, 95% of 2WikiMultiHopQA examples, and 96% of MuSiQue examples. Across the 300 audited EAGER-auto predictions, 283 automatic judgments agree with the manual check. The remaining disagreements are mostly conservative GPTAcc false negatives, where short but semantically acceptable variants–such as aliases, answer-unit reductions, location shorthand, numeric qualifiers, capitalization or punctuation differences, and answer-granularity variants–are rejected by the judge. Only one audited case is a permissive false positive. This audit supports the reliability of the observed GPTAcc behavior on the reported EAGER outputs, with the disagreements skewing toward slight under-counting of acceptable EAGER answers rather than inflation. It is not a full manual audit of every baseline row; we therefore interpret GPTAcc together with ConAcc and evidence-coverage diagnostics rather than as a standalone ground-truth label.

The ablation analysis uses Full-gold to diagnose whether the final retained evidence state recovers the support required by a question under the shared-corpus setting. The metric is computed only after all retrieval and supplementation steps have finished and the final evidence budget has been applied, so it reflects the evidence actually available to the answer generator rather than a larger intermediate pool. Although we also record Any-gold as a coarse reference, the main analysis emphasizes Full-gold because it more directly indicates whether the annotated evidence path has been recovered.

For HotpotQA and 2WikiMultiHopQA, supervision is sentence-level, since each question provides supporting facts as title–sentence-index pairs. After the shared corpus is constructed, each annotated fact is mapped to a shared-corpus sentence id by resolving its title and sentence position. A question is counted as Full-gold only when all mapped gold sentence ids appear in the final retained evidence state; retrieving the correct paragraph while failing to retain the supporting sentence, or retaining only a strict subset of the annotated support, is therefore not counted as full recovery. MuSiQue is evaluated at the supporting-paragraph level under our fixed shared-corpus protocol: the retained evidence remains sentence-level, but a question is counted as Full-gold when the final evidence contains at least one retained sentence from every gold supporting paragraph. These rules differ in support granularity, but they play the same diagnostic role across datasets. In all cases, the mappings are used only after inference to compute diagnostic coverage.

The main comparison combines rows evaluated in this work with protocol-matched results reported

in prior work. Under the unified stack, we evaluate GPT-4o-mini zero-shot, two Retrieval Top-5 variants, IRCoT-style Top-5, LinearRAG, and all EAGER configurations on the fixed 1,000-question sets, using the same ConAcc script and HippoRAG-style GPTAcc judge prompt. These unified-stack rows define the controlled comparisons used for the main claims, since they share the question sets, answer generator, judge prompt, corpus resources, and comparable retrieval/evidence-budget controls wherever the method structure permits. GPT-4o-mini zero-shot provides a direct-answering reference without retrieval. Retrieval Top-5 (dense only) uses the same `sentence-transformers/all-MiniLM-L6-v2` dense encoder (Reimers and Gurevych, 2019; Wang et al., 2020) to retrieve the top-5 passages from the dataset-level shared corpus, while Retrieval Top-5 (Qwen rerank) first retrieves a top-20 candidate set with the same dense retriever and then applies `Qwen/Qwen3-Reranker-0.6B` (Zhang et al., 2025) before passing the final top-5 passages to the shared answer-generation and evaluation stack. The reported IRCoT Top-5 (Prior) row is included as a protocol-matched external reference for the original IRCoT setting, rather than as a unified-stack evaluation.

The IRCoT-style Top-5 row is implemented as a same-stack iterative control rather than an official-code reproduction. Its controller prompt follows the IRCoT principle of interleaving reasoning and retrieval: at each step, the LLM receives the original question, previous reasoning states, and retrieved evidence, then emits a concise reasoning state, a standalone search query, and a stop flag. The first step is forced to retrieve; later steps may stop when the controller judges that the retrieved evidence is sufficient. The step budget is fixed to two retrieval steps. At each step, the search query retrieves 20 dense sentence candidates from the same shared corpus with the MiniLM sentence embeddings, reranks them with the same Qwen reranker, and adds the top-5 step evidence sentences to the accumulated pool. After the iterative trace, the final answer context is selected by reranking the accumulated pool with a query formed from the original question and the recent reasoning trace, then keeping the final top-5 evidence sentences for the shared answer-generation and evaluation stack. This makes the row a controlled same-stack IRCoT-style implementation for iterative reasoning-guided retrieval, while keeping the retrieval backend, answer generator, and judge aligned with the EAGER runs. The prior IRCoT row and the IRCoT-style control therefore play different roles: the former anchors the comparison to the reported method, while the latter controls for iterative reasoning-guided retrieval under the unified stack.

The LinearRAG row uses the LinearRAG implementation with the same MiniLM semantic encoder as EAGER, while keeping LinearRAG's dataset-specific retrieval settings. The remaining baseline rows are included from prior papers when they report results under a related HippoRAG-style shared-corpus protocol and compatible answer-evaluation format. We do not reimplement every prior system under EAGER's exact generator, reranker, prompting, and budget configuration, since doing so would require substantial system-specific engineering and could alter the original methods. This design keeps the question sets, corpus protocol, answer generator, and answer-evaluation format aligned where possible, while avoiding a misleading claim that all rows are unified-stack evaluations. The rows evaluated in this work therefore define the controlled comparisons, whereas prior-work rows provide protocol-matched external references for systems whose full processed resources or implementations are not part of the unified evaluation stack.

B Corpus and Entity Index Construction

This section documents how the shared corpora and entity indexes used by EAGER are constructed, and separates retrieval-time resources from evaluation-only annotations. The purpose is to make clear that entity-anchored expansion is built from the input corpus itself rather than from gold answers or oracle support labels.

For each benchmark, we first fix the same 1,000-question evaluation set used by the shared-corpus protocol. The dataset-level corpus is then constructed by pooling the paragraphs associated with those questions in the official benchmark resources and de-duplicating paragraphs by their title and text. HotpotQA and 2WikiMultiHopQA preserve paragraph sentence lists from the official dev resources; MuSiQue preserves paragraph text from the official answerable dev split and segments it into sentences using deterministic preprocessing. During inference, EAGER and the unified-stack baselines retrieve from this dataset-level shared corpus. They do not receive a question-specific context pool, gold supporting facts, gold answers,

answer aliases, or dataset-level evaluation results as retrieval input.

The entity index is built after shared-corpus construction. Each paragraph is processed with `spaCy` (`en_core_web_sm`) (Honnibal et al., 2020) to extract named entities, noun chunks, proper-noun spans, and pronoun mentions. When enabled, `fastcoref` (Otmazgin et al., 2022) is used only on paragraph text to add local coreference links. Mentions within the same paragraph are grouped into local entity clusters by exact normalized matches, head/alias overlap, and coreference links. For each cluster, the index stores a canonical surface form, observed aliases, mention ids, sentence ids, and the paragraph id. Sentence windows with radius 1 are also stored so that a matched entity can activate a small local evidence neighborhood rather than only the exact mention sentence.

Surface keys are normalized with Unicode normalization, lowercasing, whitespace normalization, punctuation trimming, and parenthetical-title variants. The retrieval index stores a title surface index, an entity surface index, entity-to-sentence/window maps, paragraph-to-window maps, and optional FAISS vector indexes (Johnson et al., 2019) over titles, entity texts, and sentence windows. Entity vector text is built only from the cluster's canonical name, observed aliases, paragraph title, and representative corpus sentences. It does not include gold answers, supporting-fact labels, or answer aliases. At query time, anchors extracted from the question, generated step signal, or retained evidence are normalized and matched against these surface indexes. Exact surface matches are supplemented by fuzzy/semantic matching with fixed thresholds and caps, and matched entities activate candidate sentence windows for the downstream paragraph- and sentence-level refinement stages.

Gold annotations are kept outside the paragraph, entity, and window records used by retrieval. Supporting-fact mappings and answer aliases are read only by evaluation and diagnostic code after inference. They are not serialized into surface indexes or vector-index texts, and they are not used for anchor generation, candidate expansion, reranking, or routing. The retrieval-time index and the evaluation-time support mapping are therefore separate: EAGER may retrieve a gold-supporting sentence only if it is reached through the same corpus-derived surface, entity, dense, and reranking mechanisms available to all non-oracle runs.

The released artifacts will include the fixed question identifiers, corpus manifests, index summaries, configuration files, and processed index artifacts needed to rebuild or audit this pipeline. Local filesystem paths and cache locations are not part of the method; they will be replaced by repository-relative paths and documented build commands in the release.

C Implementation Details

The retrieval stack is shared across datasets:

- **Dense retrieval:** `sentence-transformers/all-MiniLM-L6-v2` (Reimers and Gurevych, 2019; Wang et al., 2020).
- **Local reranking:** `Qwen/Qwen3-Reranker-0.6B` (Zhang et al., 2025).
- **Anchor extraction:** LLM anchors when needed, together with local anchors exposed by the question text and retained evidence.
- **Shared corpus and index resources:** HotpotQA, 2WikiMultiHopQA, and MuSiQue each use a fixed dataset-level shared corpus and a corresponding corpus-derived entity/title surface index; all variants within a dataset share the same resources.

At query time, entity matching normalizes anchors against the corpus-derived entity/title surface indexes. Fuzzy surface matching uses a similarity threshold of 88, keeps at most 8 similar surface matches, and limits each entity expansion to at most 12 surface keys and 32 linked clusters. Sentence embeddings are computed in batches of 256. In each retrieval round, entity-expanded candidates are first coarse-ranked by lexical–semantic relevance and anchor overlap; the top-20 candidates are then reranked by the local reranker with batch size 8, and the top-10 sentence evidence rows are retained for state update and answer generation. All signal modes share these retrieval budgets and are capped at 6 supplementation steps.

The scoring functions in Section 3.3 are implemented with fixed, non-tuned weights. The lexical–semantic relevance term first combines token-overlap and MiniLM cosine similarity as $0.6\,\mathrm{Lex}(c, x) + 0.4\,\mathrm{Sem}(c, x)$, where embeddings are L2-normalized and negative cosine values are clipped to zero. This relevance score is then combined with entity/title anchor features: a title-containment feature contributes 0.15 to the self-

Dataset	Step-wise retrieval question	Subquestion-chain
HotpotQA	681 / 1000 (68.1%)	319 / 1000 (31.9%)
2WikiMultiHopQA	487 / 1000 (48.7%)	513 / 1000 (51.3%)
MuSiQue	248 / 1000 (24.8%)	752 / 1000 (75.2%)

Table 5: Routing distribution of the fixed pre-inference signal router. The router is deterministic and uses only string-pattern cues in the question text. It does not inspect answer correctness, gold evidence, or dataset-level evaluation results.

match term, sentence/title overlap are mixed as $0.7/0.3$, and the final coarse score uses weights 0.55 for self-match, 0.30 for evidence-overlap, and 0.15 for bridge/entity-anchor bonus. When a step-wise retrieval question is available, coarse scores from the original question and the generated retrieval question are combined with weights $0.3/0.7$; no-signal sets the generated-signal weight to 0.0. Fine ranking uses the Qwen reranker score plus a 0.1 title-containment bonus before selecting the retained top-10 sentences. These settings are fixed across the reported runs, and additional engineering details will be available in the released code and configuration files after the review period.

The three signal modes use the same overall evidence construction pipeline. They differ mainly in how step-level retrieval text is supplied:

- **Step-wise retrieval-question:** generates one retrieval question at each supplementation step and answers from the accumulated evidence state.
- **Subquestion-chain:** first generates an ordered decomposition, executes each subquestion in order, injects intermediate answers into later subquestions, and falls back to expansion from the original question and retained evidence when a step-level signal cannot be reliably parsed.
- **No-signal:** does not generate step-level questions and expands from the original question together with anchors exposed by the current evidence state.

In all settings, candidate evidence is first obtained through entity-anchored expansion, then passed through paragraph-level and sentence-level reranking, and finally retained in a fixed-size evidence state for later expansion or answer generation. The number of executed rounds is determined by question type, subquestion-chain length, or the fixed supplementation budget, subject to the shared maximum above.

The main EAGER-auto system uses a fixed deterministic rule-based signal router before inference to choose between the two generated-signal modes. The router does not inspect answer correctness or dataset-level results; it only uses string-pattern cues in the question text. Table 5 reports the resulting routing distribution, which is consistent with the different question structures in the three benchmarks. HotpotQA is routed mostly to step-wise retrieval questions, reflecting many cases where the retrieval focus changes after an initial bridge entity or partial evidence item is found. 2WikiMultiHopQA is nearly balanced, which matches its mixture of comparison, bridge-comparison, compositional, and inference questions. MuSiQue is routed mostly to subquestion-chain mode, consistent with its more explicit compositional decomposition structure. The router therefore provides a fixed operating point rather than an oracle choice over signal modes.

The rule classifier assigns retrieval-question mode when the question contains comparison or parallel-evidence cues such as `both`, `same`, `different`, `which ... or`, comparative adjectives, or other patterns suggesting that evidence pieces can be collected and compared without a strict ordered chain. It assigns subquestion-chain mode when ordered slot-filling or dependency cues dominate, including nested `of` constructions, family or creator relations such as `father of`, `director of`, `author of`, relative clauses with `who`/`that`/`which`/`whose`, and repeated wh-dependencies. When neither side is strongly marked, the conservative fallback is retrieval-question mode.

The ablation rows use the same fixed 1,000-question evaluation sets and evaluation scripts as the main routed system unless otherwise noted. Signal-mode variants and component removals share the same answer generator, judge, evidence budget, and corpus resources, changing only the signal source or the ablated evidence-construction component. These fixed settings define the reported EAGER variants and relevant local-run configurations.

D Prompt Templates

Figure 3 summarizes the core prompts used by EAGER. The figure includes the prompts that define the main LLM-facing operations in the system: anchor extraction, step-wise retrieval-question gener-

ation, subquestion-chain decomposition, target-slot answer generation, and GPT-based answer judging. These templates are shown at the level used to specify each function, while minor dataset wrappers and historical prompt variants are omitted for clarity.

The same target-slot answer prompt is used when generating final answers across signal modes. In subquestion-chain mode, the answer prompt is also used to answer intermediate materialized subquestions before the final answer is produced. The no-signal mode does not introduce an additional step-level prompt; it reuses the shared anchor and answer prompts while expanding from the original question and evidence-derived anchors. For GPT-based judging, the system prompt and user prompt are shown together because the actual evaluation call uses a short evaluator system instruction together with a user message that compares the generated and gold answers.

E Error Analysis

The representative failures below are grouped by error mechanism to foreground EAGER's main residual failure modes in answer formation. Different evidence-coverage outcomes do not necessarily correspond to different error sources, and cases with similar coverage status may still diverge at the final answer stage. Each category is illustrated with a representative example, while evidence status and metric behavior are used only as auxiliary diagnostic labels.

E.1 Target-Slot Misbinding

Error pattern:
The system reaches a largely correct evidence neighborhood but binds the final answer to the wrong target slot, for example by returning an intermediate entity or a nearby attribute rather than the one requested by the question.

Why it matters:
This category is informative because it shows that a run may fail even after the relevant support has been recovered; the remaining error lies in mapping the retained evidence to the correct answer variable.

Case question:
"What is the date of birth of Mohammed Bin Nawwaf Bin Abdulaziz's father?"

Gold evidence status:
`Mohammed bin Nawwaf bin Abdulaziz` – found. The retained evidence identifies Mohammed as the relevant child entity. `Nawwaf bin Abdulaziz Al Saud` – found. The retained evidence contains the sentence "Nawwaf bin Abdulaziz (16 August 1932 – 29 September 2015)."

Prediction:
"22 May 1953"

Influential retained evidence:
The final evidence state also retains "Mohammed bin Nawwaf bin Abdulaziz ... born 22 May 1953," which is topically relevant but corresponds to the child rather than the queried father.

Interpretation:
The evidence path reaches the correct local neighborhood, but the answer generator binds the final slot to the wrong person.

Implication for EAGER:
This case suggests that when closely related entities remain simultaneously active in the final evidence state, the answer generator may over-rely on the most locally salient biographical field and under-enforce the target-entity constraint implied by the full relation chain.

E.2 Relation Composition Failures

Error pattern:
Local pieces of evidence are individually relevant, but the multi-hop composition is incorrect, such as confusing father with grandfather or mixing up adjacent roles and attributes.

Why it matters:
These failures are central to multi-hop QA because they reveal errors in relational composition rather than simple retrieval misses.

Case question:
"Who is Karl Von Habsburg's paternal grandmother?"

Gold evidence status:
`Karl von Habsburg` – found. The retained evidence states that Karl is the son of Otto von Habsburg and Princess Regina of Saxe-Meiningen. `Otto von Habsburg` – found. The retained evidence also states that Otto was the son of Charles I and his wife, `Zita of Bourbon-Parma`.

Prediction:
"Princess Regina of Saxe-Meiningen"

Influential retained evidence:
The sentence about Karl's parents is more locally salient in the final evidence state than the sentence establishing Otto's mother.

Anchor Extraction Prompt

```
You are the anchor extractor.

Input:
A JSON object containing the current question.

Task:
Extract short retrieval anchors from the current question. Prefer entity
names, titles, events, organizations, places, and distinctive noun
phrases. Keep anchors concrete and searchable. Do not answer the
question.

Return JSON only:
{
  "anchors": ["...",]
}
```

Target-Slot Answer Prompt

```
You are a multi-hop reading-comprehension assistant. Answer the
exact target slot asked by the question using only the provided
evidence.

Input:
Evidence passages followed by the question.

Task:
Briefly identify the question's target slot, connect the evidence facts
needed to resolve the bridge or comparison, and choose the shortest
supported final answer for that slot. Return one answer span, not an
explanatory sentence. Do not answer with a bridge entity when the
question asks for an attribute of that entity. If multiple nearby entities
appear in the evidence, choose the one that fills the target slot.
Preserve the full named span, date, number, or category modifier
when it is needed for specificity.

Output format:
Thought: brief evidence-based reasoning
Answer: final answer only
```

GPT Judge Prompt

```
System:
You are an expert evaluator.

User:
Please evaluate if the generated answer is correct by comparing it with
the gold answer.
Generated answer: {predicted_answer}
Gold answer: {gold_answer}

The generated answer should be considered correct if it:
1. Contains the key information from the gold answer
2. Is factually accurate and consistent with the gold answer
3. Does not contain any contradicting information

Respond with ONLY 'correct' or 'incorrect'.
Response:
```

Step-Wise Retrieval-Question Prompt

```
You generate one retrieval question for a multi-hop QA pipeline.

Input:
A JSON object containing the original question, question type or hop
information, current layer index, selected titles, previous retrieval
questions, and current evidence.

Task:
Write one concise standalone retrieval question for the next missing
evidence hop. Use the current evidence to identify what bridge entity,
relation, event, place, time, attribute, or supporting fact is still missing.
The retrieval question should help retrieve one additional supporting
page, paragraph, or sentence. Do not answer the original question, and
do not repeat a previous retrieval question.

Return JSON only:
{
  "reasoning": "brief reason",
  "retrieval_question": "one standalone retrieval question"
}
```

Subquestion-Chain Decomposition Prompt

```
You are a question decomposition assistant.

Input:
A JSON object containing the question id and original question.

Goal:
Break one multi-hop question into the smallest ordered chain of atomic
sub-questions needed to answer it. First resolve intermediate entities
or relations, then use them to ask for the final target attribute. Each
sub-question should ask for exactly one missing fact. Later sub-
questions may depend on earlier answers using placeholders such as
#1 and #2. Do not invent answers or cite evidence.

Return JSON only:
{
  "target_slot": "short phrase for the exact answer slot asked by the
question",
  "subquestions": [
    {
      "id": 1,
      "question": "...",
      "depends_on": [],
      "fills_slot": "..."
    }
  ]
}
```

Figure 3: Core prompt templates used by EAGER. The figure reports the main prompts used for anchor extraction, step-wise retrieval-question generation, subquestion-chain decomposition, target-slot answer generation, and GPT-based answer judging.

Interpretation:
The system remains within the correct family chain but composes the roles incorrectly, returning Otto's wife relation from the wrong generation rather than the paternal-grandmother relation required by the question.

Implication for EAGER:
This example indicates that once multiple family relations are simultaneously available, the answer generator may collapse nearby generational roles unless the final reasoning step more explicitly enforces relation direction and hop depth.

E.3 Answer-Form Mismatches

Error pattern:
The predicted answer is semantically close to the reference, but differs in canonical form, granularity, or formatting.

Why it matters:
These cases help explain why ConAcc and GPTAcc can diverge even when the underlying reasoning path is successful.

Case question:
"What is the date of death of the director of film The Organization (Film)?"

Gold evidence status:
`The Organization (film)` – found. The retained evidence identifies `Don Medford` as the director. `Don Medford` – found. The retained evidence contains the date "December 12, 2012" and a duplicate date expression in the later biographical sentence.

Prediction:
"12 December 2012"

Metric behavior:
The answer is judged correct by GPTAcc but does not satisfy strict string-based matching.

Interpretation:
The reasoning path is successful and the relevant support is retained, but the predicted answer differs from the reference only in canonical surface form.

Implication for EAGER:
This case indicates that some residual discrepancies arise after successful retrieval and reasoning, and are better understood as answer-form normalization differences than as evidence-construction failures.

E.4 Conservative or Failed Answer Extraction

Error pattern:
The retained evidence is sufficient or nearly sufficient, but the answer generator fails to convert it into an explicit final answer, often producing an empty or overly cautious response.

Why it matters:
These cases isolate answer-stage weakness more clearly than pure retrieval failures, because the run is already close to the correct evidence neighborhood.

Case question:
"What is the date of death of Thomas Barnard's father?"

Gold evidence status:
`Thomas Barnard` – found. The retained evidence identifies Thomas Barnard as the son of William Barnard. `William Barnard (bishop)` – partially retained. The final evidence keeps the correct father paragraph, but it does not retain the sentence "William Barnard (1697 – 10 January 1768) ..." that explicitly states the answer.

Prediction:
empty

Influential retained evidence:
The final evidence includes biographical sentences about Thomas Barnard and non-answer sentences from the father paragraph, such as burial information, but omits the most answer-bearing death-date sentence.

Interpretation:
The system reaches the correct parent entity but fails to preserve the most useful answer-bearing evidence, after which the generator defaults to an empty response.

Implication for EAGER:
This case suggests that once the retained evidence around the correct entity becomes too diffuse, the answer generator may back off instead of committing to an answer, even though the retrieval trajectory has already reached the appropriate local neighborhood.

E.5 Long-Hop Drift

Error pattern:
As the reasoning chain grows longer, the evidence state becomes progressively less stable and later-hop recovery drifts toward locally relevant but incomplete evidence.

Why it matters:
These cases illustrate a genuine boundary of the current method on long-chain questions, especially when later topical relevance overwhelms earlier bridge evidence.

Dataset	Signal mode	Avg. extra LLM calls	Avg. runtime / question
HotpotQA	Step-wise retrieval-question	1.00	10.59s
HotpotQA	Subquestion-chain	3.24	17.71s
HotpotQA	No-signal	0.00	7.01s
2WikiMultiHopQA	Step-wise retrieval-question	0.99	6.22s
2WikiMultiHopQA	Subquestion-chain	2.32	12.05s
2WikiMultiHopQA	No-signal	0.00	4.06s
MuSiQue	Step-wise retrieval-question	3.30	8.98s
MuSiQue	Subquestion-chain	4.73	15.76s
MuSiQue	No-signal	0.00	5.02s

Table 6: Query-time cost of the three EAGER signal modes under the shared-corpus setting. Extra LLM calls exclude the shared initial anchor extraction, final answer generation, and evaluation-only judge calls; runtime is measured on matched two-hop samples with the same local retrieval and reranking stack.

Case question:
"Where does the body of water by the city where the Southeast Library designer died empty into the Gulf of Mexico?"

Gold evidence status:
`Southeast Library` – not retained in the final evidence state. `Riverside Plaza` – not retained in the final evidence state. `Minneapolis` – not retained in the final evidence state. `Mississippi River` – retained.

Prediction:
"Mississippi River"

Influential retained evidence:
The final evidence is dominated by generic paragraphs about the `Gulf of Mexico`, plus one retained paragraph about the `Mississippi River`, while the earlier bridge paragraphs needed to reconstruct the full chain are no longer preserved.

Interpretation:
The run drifts toward the later topical neighborhood of the question but loses the earlier bridge structure required to complete the final hop from the river to its delta.

Implication for EAGER:
This example suggests that on longer chains, later topical relevance can remain strong enough to sustain a plausible local trajectory even after the earlier bridge evidence has thinned out, leaving the final answer one hop short of completion.

F Cost Analysis

This appendix reports the inference cost of the three EAGER signal modes under the shared-corpus setting. EAGER also requires offline corpus preprocessing, entity indexing, and construction of the resources used for entity-anchored expansion. These offline steps are performed once for a fixed corpus and are amortized across queries; Table 6 therefore focuses on query-time cost.

We report two quantities. The first is the average number of extra LLM calls, computed on the fixed 1,000-question evaluation set used for the main results. To isolate the cost introduced by the signal mechanism itself, the count excludes calls shared by all comparable pipelines, namely the initial anchor extraction call and the final answer generation call. Evaluation-only GPT judge calls are also excluded. The count includes generated retrieval questions or subquestions, decomposition calls, intermediate step-answer calls, anchor-extraction calls triggered during subquestion-chain execution, and fallback calls whenever they are triggered by the corresponding pipeline variant. Local entity extraction, dense retrieval, and reranking are not counted as LLM calls.

The second quantity is average runtime per question. Because the three benchmarks contain different reasoning-type distributions, latency is measured on matched two-hop samples rather than from the total wall-clock time of the full 1,000-question jobs. Runtime is recorded from the beginning of question processing to final answer generation and does not include evaluation-only judge calls, script initialization, or post-hoc metric computation. It does include local entity extraction, dense retrieval, reranking, signal generation, and answer generation. HotpotQA uses bridge questions, 2WikiMultiHopQA uses compositional questions, and MuSiQue uses questions from the `2hop` subset. All timing measurements are conducted on a single NVIDIA GeForce RTX 5060 Laptop GPU with 8151 MiB of memory. Since several components involve external LLM calls, the absolute latency can vary with API response time and transient system load; the table is intended to compare

the relative cost of the three signal modes under the same measurement setting.

For 2WikiMultiHopQA, the step-wise run uses the type-routed protocol from the main experiments. Comparison questions do not trigger a supplementation step, bridge-comparison questions trigger two generated signals, and compositional or inference questions trigger one generated signal. The reported 0.99 average therefore reflects the full 1,000-question type mixture, while the runtime column isolates latency on matched two-hop compositional questions.

Table 6 shows the expected cost ordering. No-signal provides the lowest-cost inference path because it does not add step-level LLM calls beyond the shared answer pipeline. Step-wise retrieval-question introduces a small number of additional signal calls and correspondingly increases latency. Subquestion-chain is the most expensive mode, since it adds explicit decomposition and per-step processing; this overhead is most visible on 2WikiMultiHopQA and MuSiQue, where the chain-style signal also gives larger gains in the main results. Overall, the additional cost remains bounded: the signal modes increase the number of LLM calls by only a small constant amount per question, while the runtime stays within a practical range for offline evaluation and controlled analysis.