Hi-Q: Hierarchical Evidence-guided Query Refinement for Multi-Hop Question Answering

Jueun Kim[1] **Sungho Park**[2] **Wook-Shin Han**[2*]
[1]Department of Computer Science and Engineering, POSTECH
[2]Graduate School of Artificial Intelligence, POSTECH
{jekim,shpark,wshan}@dblab.postech.ac.kr

Abstract

A central bottleneck in multi-hop Question Answering (QA) is that the granularity at which a question is expressed often differs from the granularity at which corpus evidence is retrievable. Existing methods address this mismatch by imposing fixed graph structures over the corpus, by iteratively reformulating the query, or by executing a generated program over it, but these strategies do not explicitly decide when a query unit is already supported by evidence and when it should be refined. We formulate this bottleneck as retrievable granularity discovery and introduce `Hi-Q`, an evidence-conditioned framework for hierarchical query refinement. At each query node, a resolution operator tests whether retrieved evidence supports the current query unit; resolved nodes terminate, while unresolved nodes are expanded by a dependency-preserving binary operator and checked by a semantic coverage verifier. `Hi-Q` therefore grows a query tree whose topology is determined by corpus support signals rather than by a fixed decomposition template or a pre-built graph. We evaluate `Hi-Q` on three multi-hop QA benchmarks over their complete corpora, where dependent evidence must be found among up to millions of open-domain passages. There it reaches 52.3 EM and 64.0 F1 on average, ahead of the iterative retrieval baseline IRCoT by 15.2 EM / 18.2 F1 and of the graph-based RAG baseline PropRAG by 11.6 EM / 12.0 F1 on MuSiQue, without corpus-wide graph construction. On the smaller controlled corpora used to compare graph-based methods, `Hi-Q` also attains the best accuracy, 57.9 EM and 69.3 F1 on average, ahead of PropRAG by 5.6 EM / 3.9 F1 and IRCoT by 13.7 EM / 15.8 F1. The project page is available at `https://hi-q-project.github.io/`.

1 Introduction

Multi-hop Question Answering (QA) requires resolving multiple interdependent reasoning steps embedded within a single natural language query. Consider the query in Figure 1: "When was the start of the battle of the birthplace of the performer of III?" Answering this question requires first identifying the performer of "III," then determining that person's birthplace, and finally finding the start date of the battle associated with that location. Thus, a single query implicitly compresses a chain of dependent informational needs into one surface form.

A central difficulty in multi-hop QA is that the unit at which a question can be *logically expressed* is often different from the unit at which evidence can be *reliably retrieved*. The facts needed to answer a multi-hop question are typically distributed across documents at a fine-grained level, while the input query presents them as a single coarse-grained sentence. As a result, even if a question admits a plausible logical decomposition, it remains unclear *a priori* which intermediate formulation is best aligned with the corpus for retrieval. Queries that are too coarse entangle multiple reasoning aspects

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

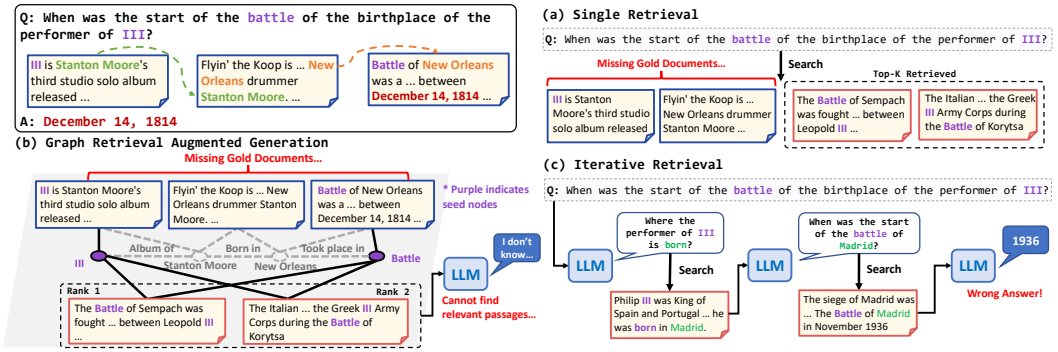


Figure 1: **Granularity failures in multi-hop retrieval.** (a) Single retrieval, (b) Graph RAG, and (c) iterative retrieval on the same query and corpus.

and cause retrieval interference, while queries that are too fine may lose contextual constraints and lead to over-decomposition. Therefore, the key challenge is not merely how to decompose a question logically, but how to discover the *retrievable granularity* at which each reasoning step becomes operationally answerable.

Figure 1 illustrates three concrete failure modes arising from this granularity mismatch. First, single-shot retrieval fails when a coarse query entangles several reasoning constraints: top-ranked passages may match surface terms such as "III" or "battle" without covering the full evidence chain. Second, Graph RAG methods add corpus-side structure, but their pre-computed graphs impose a fixed, query-agnostic granularity and may reduce reasoning to surface-level seed alignment. Third, iterative retrieval methods adapt the query over time, but their reformulations are not explicitly checked for evidence support, so an early wrong intermediate query can propagate through later retrieval steps. Together, these failures reveal the need for a control mechanism that decides which query unit is currently retrievable under the given corpus.

We introduce `Hi-Q`, a framework for coarse-to-fine, evidence-guided query refinement that answers these three failures with two coupled mechanisms. Failure-aware granularity control tests whether the current query unit is already supported by retrieved evidence before refining it, so a query that single retrieval or a fixed graph already answers is never decomposed. Dependency-preserving hierarchical decomposition then expands only the unresolved nodes, resolving prerequisite sub-queries before dependent ones so that an unchecked intermediate query cannot propagate, while a semantic coverage verifier confirms that each binary split preserves the intent of its parent. `Hi-Q` therefore does not assume the retrieval-aligned query unit in advance, but discovers it through retrieval-and-answering feedback.

Our contributions follow this challenge-to-component mapping:

- **Retrievable granularity discovery.** We formulate multi-hop RAG as the problem of identifying the query unit at which a reasoning step becomes both retrievable and answerable under a given corpus.
- **Failure-aware granularity control.** We propose an evidence-conditioned control policy that expands a query node only when a resolution operator detects insufficient evidence support, refining coarse queries while avoiding unnecessary decomposition of already answerable queries. We show that this choice is a cost-sensitive threshold on unresolved support.
- **Dependency-preserving hierarchical refinement.** We introduce a binary expansion operator that resolves prerequisite sub-queries first and propagates their results to downstream retrieval, reducing under-specified retrieval and error propagation.
- **Evaluation under full-corpus retrieval.** Across three multi-hop benchmarks, `Hi-Q` outperforms graph-based, iterative, and code-executing agent baselines without pre-built knowledge graphs or task-specific fine-tuning, and a cost-matched configuration is both cheaper and more accurate than iterative retrieval at the same number of LLM calls. We validate the unresolved-support signal through trigger diagnostics, ablations, and reader and embedding substitutions.

2 Related Work

Retrieval-Augmented Generation (RAG). Standard RAG holds both ends fixed: the query is issued as written, and the corpus is indexed as flat passages. It integrates external knowledge into LLMs [1], with dense retrievers such as DPR [2] and recent embedding models [3, 4] improving similarity-based evidence acquisition. This is effective when the question and the retrievable evidence already sit at a similar granularity. In multi-hop QA they often do not: a single coarse query matches passages on isolated surface terms while missing the complete reasoning chain [5–7]. This query–evidence granularity mismatch is the gap the remaining lines of work, and `Hi-Q`, attempt to close.

Graph RAG. Graph-based RAG moves the granularity decision to the corpus side, but makes it before any query arrives. GraphRAG [8] and RAPTOR [9] build hierarchical summaries, while HippoRAG [5, 6] and PropRAG [7] retrieve through graph- or proposition-level structures. These help when the pre-computed units happen to match a question's evidence needs, but the structure is query-agnostic by construction and carries a corpus-wide pre-computation cost. `Hi-Q` instead builds a dependency-ordered query tree online and expands only unresolved nodes, without corpus-wide graph construction.

Iterative Retrieval. Iterative retrieval issues a sequence of queries instead of one. IRCoT [10], ReAct [11], and Self-Ask [12] generate each follow-up query online from intermediate findings, whereas Least-to-Most prompting [13] and Decomposed Prompting [14] plan the sequence up front by decomposing the question, and TRQA [15] and Q-DREAM [16] learn such decompositions. None of them tests whether the query it issues is resolvable from what the corpus returns: a follow-up or a split can be linguistically valid yet unnecessary, over-fragmented, or misaligned with the atomic facts the corpus exposes, and once an early step selects the wrong bridge entity or drops a constraint, later queries amplify the error. Self-reflective and corrective RAG [17, 18] do assess whether retrieved documents support the generation, but they use that judgment to filter or re-retrieve documents for a flat answer, not to decide at what granularity the next query should be issued. `Hi-Q` instead starts from the original question and splits it step by step, testing each unit against retrieved evidence and refining only those the corpus fails to resolve, so the granularity of its queries is fitted to the corpus rather than fixed in advance.

Retrieval Strategy Selection. Retrieval strategy selection decides how retrieval is orchestrated for a question rather than how each query is expressed. Coding agents [19], Recursive Language Models [20], and PyRAG [21] read the question and write a program that decides when and how to call retrieval and answering tools, while Adaptive-RAG [22] trains a classifier of question complexity that routes each question to one of a few fixed strategies: no, single-step, or multi-step retrieval. Both choose the strategy from the question rather than from what retrieval returns. The route is fixed before any evidence is seen (Appendix D), and although an agent can recover from an execution failure at run time (Appendix M), the recovery re-runs or patches the program rather than identifying the granularity at which a retrieval call failed and refining the query to it. What separates `Hi-Q` is therefore the state on which control is conditioned, not the presence of learning.

3 Method

3.1 Problem Formulation

We formulate multi-hop QA as evidence-conditioned adaptive search over query granularity. `Hi-Q` first tests whether the current query unit is supported by retrieved evidence, and expands it into smaller dependency-aware sub-queries only when the query remains unresolved. This requires an explicit search state, because retrieval feedback determines whether a query should remain at its current coarse granularity or be refined.

Notation. Let C denote the retrieval corpus, Q the original question, and q a query node in the search tree. Let $R_k(q, C)$ be the retriever that returns the top-k passages for query q. We write $\mathcal{H}$ for the accumulated interaction history and d for the current recursion depth.

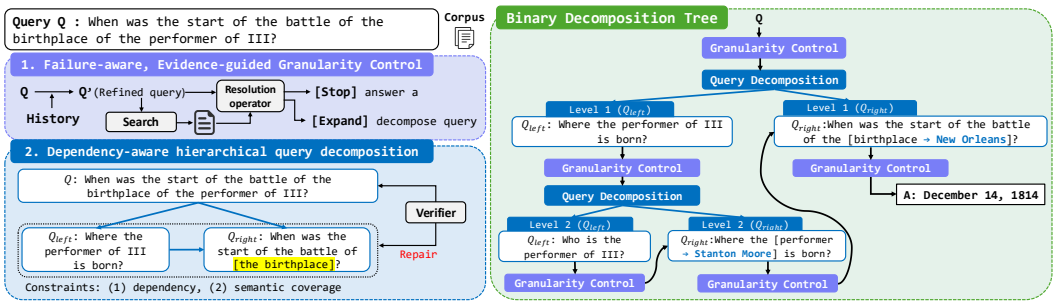


Figure 2: **Overview of Hi-Q.** Hi-Q treats multi-hop QA as evidence-conditioned search over query granularity. At each node, the resolution operator first tests whether retrieved evidence supports the current query unit. Resolved nodes terminate; unresolved nodes trigger dependency-ordered binary decomposition, where the prerequisite branch is resolved before the dependent branch. A semantic coverage verifier repairs invalid splits before recursion. The resulting binary decomposition tree is determined by corpus support signals rather than by a fixed graph or a predetermined decomposition template.

Search state. Hi-Q maintains a state $x = (q, \mathcal{H}, d)$, where q is the current query node, $\mathcal{H}$ is the accumulated interaction history, and d is the current recursion depth. The search objective is to find a leaf set whose queries are individually resolvable under corpus evidence and whose dependency composition preserves Q.

Resolution operator. Let $\mathcal{G}(q, \mathcal{H}, C) \rightarrow (s, a, D)$ be a resolution operator, where $s \in \{\text{RESOLVED}, \text{UNRESOLVED}\}$ is the resolution status, a is an answer when $s = \text{RESOLVED}$, and $D = R_k(q', C)$ is the evidence retrieved for the refined query $q' = \text{Refine}(q, \mathcal{H}, Q)$, which rewrites q from the history while keeping it anchored to Q (Section 3.2). Once $\mathcal{G}$ has run, the node carries that evidence as well, giving the post-resolution state $\tilde{x} = (x, D)$. For analysis only, we additionally assign an optional diagnostic label τ to unresolved cases; τ is not consumed by the control policy.

Policy. The policy acts on that state together with the resolution status, since whether to stop is decided by what the retrieved evidence supported:

$$\pi(\tilde{x}, s) = \begin{cases} \text{STOP} & \text{if } s = \text{RESOLVED} \\ \text{FAIL} & \text{if } d = d_{\max} \text{ or } q \text{ is non-decomposable} \\ \text{EXPAND} & \text{otherwise} \end{cases}$$

Expansion. When EXPAND is selected, a binary expansion operator $\mathcal{B}(q, \mathcal{H}, Q)$ proposes a pair $(q_{\text{left}}, q_{\text{right}})$ subject to two constraints: (i) a *dependency constraint* $q_{\text{left}} \prec q_{\text{right}}$, meaning the prerequisite branch must be resolved before the dependent branch; (ii) a *semantic coverage constraint* $\mathcal{V}(Q, q, q_{\text{left}}, q_{\text{right}})$, ensuring that resolving q_{left} followed by q_{right} recovers the intent of q without omission or drift. The answer to q_{left} updates $\mathcal{H}$ before q_{right} is resolved.

3.2 Failure-aware, evidence-guided granularity control

Hi-Q decides whether to decompose a query from the evidence it retrieves rather than from how complex the query looks. At each node, the resolution operator $\mathcal{G}$ first tries to resolve the query at its current granularity: it refines the query from the history $\mathcal{H}$ while keeping it anchored to Q, retrieves the top-k passages D, and has a retrieval-grounded reader answer from them. Refinement precedes retrieval because a dependent sub-query often contains a reference whose meaning is fixed only by an earlier step; once the performer of "III" is resolved to Stanton Moore, the query about "the birthplace of the performer of III" is issued as a query about the birthplace of Stanton Moore (Figure 2). This replaces abstract references with resolved entities, reducing retrieval interference while preserving the informational role of the current node in the original multi-hop question. The reader's output is the control signal. If the reader finds an answer, the retrieved evidence supports the query unit and the node is RESOLVED and terminates; if it does not, the evidence does not jointly support the unit, and the node is UNRESOLVED and becomes eligible for expansion.

Threshold rule. The STOP/EXPAND decision rests on an estimate of whether the current node is unresolved; we derive the decision rule from the two ways the decision can fail. A query that is too coarse entangles several constraints, so the retriever returns passages that match isolated terms and miss the full chain; decomposition then exposes smaller evidence needs that can be retrieved and verified separately. A query that is already answerable should not be split merely because it looks complex, since an unnecessary split can drop constraints and introduce spurious intermediate goals. Writing the two errors as costs turns the decision into a threshold. For an expansion-admissible node in state $\tilde{x}$, let $Z \in \{\mathrm{R}, \mathrm{U}\}$ denote whether the node is in fact resolvable (R) or unresolved (U) from its retrieved evidence, of which the operator's status s is an estimate. Let $\Delta_{\mathrm{R}}(\tilde{x}) > 0$ be the cost-to-go penalty of expanding a resolvable node and $\Delta_{\mathrm{U}}(\tilde{x}) > 0$ that of stopping at an unresolved one, each measured over the resulting subtree and therefore including descendant retrieval and LLM calls, drift risk, synthesis, and terminal answer loss. Minimizing conditional expected cost yields a threshold rule, derived in Appendix B:

$$\pi^*(\tilde{x}) = \textsc{expand} \iff \Pr[Z = \mathrm{U} \mid \tilde{x}] \;\geq\; \theta(\tilde{x}) := \frac{\Delta_{\mathrm{R}}(\tilde{x})}{\Delta_{\mathrm{R}}(\tilde{x}) + \Delta_{\mathrm{U}}(\tilde{x})}.$$

Because the objective includes downstream cost-to-go, this decision is node-wise but not myopic; we do not claim global optimality of the resulting query tree, and FAIL is handled separately as a budget or feasibility action.

`Hi-Q` instantiates the estimate without training, as a test that reads it off the reader's attempt to answer from the retrieved passages. The reader is instructed to answer from the retrieved passages, extracting the answer span at the root, so UNRESOLVED means that the retrieved passages do not support the query, not that the model declined to answer. This test is deliberately the simplest instantiation of the rule: it ignores depth and cost weights, which is defensible because stopping at an unresolved node forfeits the answer while an unnecessary expansion costs far less, an asymmetry the experiments measure (decomposing every node costs 3.4 EM on the ablation sample of Section 4.4, whereas never decomposing, i.e. single retrieval, trails `Hi-Q` by 9.7 EM in the controlled-corpus setting of Table 1), so $\Delta_{\mathrm{U}} \gg \Delta_{\mathrm{R}}$ places θ well below one half and favors expanding whenever support is incomplete. Keeping the estimator this simple also attributes the gains in Section 4 to the control structure rather than to the estimator. Any instantiation, however, must observe the evidence state: when the same query calls for opposite actions under different retrieved evidence, a policy that sees only the query incurs a strictly positive regret that no amount of query-only training data removes, as we prove in Appendix B; this is what separates `Hi-Q` from methods that choose a retrieval strategy from the question alone, whose learned routing collapses to a near-constant policy in Appendix D. The training-free test is a hard classifier $\hat{Z}$ and is not claimed to compute the posterior: provided it observes the evidence state, a calibrated classifier, an entailment model, or a trained cost-sensitive router can replace it without changing `Hi-Q`'s control semantics (Appendix C), and Section 4 evaluates how precise it is in practice.

3.3 Dependency-aware hierarchical query decomposition

Dependency-aware decomposition turns an unresolved query into an ordered split whose sub-goals can be tested against evidence. `Hi-Q` splits binary rather than N-ary (multi-way) to avoid over-fragmenting the question or skipping essential intermediate bridge facts: a single multi-way split jumps to the finest granularity at once, whereas binary expansion moves from coarse to fine one step at a time, so at each step the resolution test can check whether the current granularity is already retrievable before splitting further. Splitting in two restricts sequential depth rather than expressiveness: a plan with m leaf information needs is organized as at most $m - 1$ binary reductions, with the expansion operator and the verifier unchanged, and how many reductions occur is decided by the controller, since a node resolvable from its retrieved passages terminates there. Appendix L tests this on a controlled set whose leaf needs rise from two to five: binary expansion holds its accuracy across that range, branching with unrestricted arity does not improve on it, and combining the branches in one flat N-ary step instead of sequential binary reductions is substantially worse.

Expansion is subject to the two constraints of Section 3.1. The dependency constraint makes the left branch a prerequisite for the right: given an unresolved query q, $\mathcal{B}(q, \mathcal{H}, Q)$ proposes exactly two sub-queries, q_{left}, which resolves the bridge fact, and q_{right}, which uses that bridge to approach the parent query's answer. The split follows entity references and relational dependencies (compositional, temporal, or causal) rather than the surface syntax of the question, so the decomposition order

matches the order in which retrieval must proceed. That order has to be executed and not merely proposed, because a dependent sub-query is often under-specified until its prerequisite is resolved: `Hi-Q` resolves q_{left} first, writes its answer, if any, together with its retrieved evidence and intermediate context into the history $\mathcal{H}$, and then resolves q_{right} under the updated history. For the running example, the system must identify the performer of "III" before it can retrieve evidence about the relevant birthplace and battle. The dependency is enforced at the evidence-state level rather than as a brittle requirement that the left branch must always produce a final answer: even if q_{left} remains unresolved, its retrieved evidence and partial context are retained in $\mathcal{H}$ to ground the dependent branch.

The semantic coverage constraint is enforced by the verifier $\mathcal{V}$ and prevents local decomposition errors from becoming downstream reasoning errors. Before the two sub-queries are solved, $\mathcal{V}$ checks whether resolving q_{left} followed by q_{right} would recover the intent of q without adding, omitting, or reordering essential constraints. If the split is inconsistent, $\mathcal{V}$ repairs the two sub-queries before they enter the recursive solver; we allow at most one repair attempt, and if the revised split is still inconsistent, the node is marked non-decomposable and decomposition stops for that branch. This guardrail is especially important at deeper recursion depths, where a small semantic drift in one split can compound across later branches.

Recursive expansion is bounded so that refinement remains a controlled search rather than an open-ended reasoning loop. If a sub-query is resolved, recursion stops at that node. If a sub-query is non-decomposable or the maximum depth is reached, the policy takes FAIL: the branch terminates without an answer, and the solver proceeds with the evidence gathered so far. After both branches return, a synthesis step aggregates intermediate answers, retrieved evidence, unresolved sub-goals, and history into a final response consistent with the original query Q. When multiple evidence-supported candidates are retained from earlier resolution steps, synthesis disambiguates them using consistency with other sub-query results rather than committing to the highest-ranked passage alone. Thus, `Hi-Q` constructs a finite dependency-ordered binary tree whose leaves correspond to the evidence-aligned query units used for final reasoning. The full recursive procedure is presented in Algorithm 1 (Appendix A), and the prompt templates are listed in Appendix O.

4 Experiments

4.1 Experimental Setup

Datasets and evaluation protocol. We evaluate `Hi-Q` on HotpotQA [23], MuSiQue [24], and 2WikiMultiHopQA [25]. Following prior graph-based RAG methods [5–7], we sample 1,000 validation questions per dataset and evaluate each question under two retrieval regimes. The first, the *full-corpus* setting, is the benchmark's complete corpus, in which retrieval scales to 5,233,235 passages for HotpotQA, 139,416 for MuSiQue, and 430,225 for 2WikiMultiHopQA. The second, the *controlled-corpus* setting, is the corpus used by these graph-based methods, pooled from the supporting and distractor passages of the sampled questions, with 9,811 passages for HotpotQA, 11,656 for MuSiQue, and 6,119 for 2WikiMultiHopQA. EM and token-level F1 measure final QA accuracy; Recall@k and its pooling protocol for multi-query methods are defined in Appendix E. Appendix I repeats the evaluation under strict document-level grounding. Unless stated otherwise, differences between methods are tested with a paired question-level bootstrap over the per-question predictions behind each table, using $10{,}000$ resamples and 95% percentile intervals.

Baselines and implementation. We compare against four families of methods, the first three of which cover the three failure modes in Figure 1: single retrieval tests whether one coarse query suffices, graph-based methods test corpus-side structuring before query-specific reasoning, and iterative methods test predefined or model-driven follow-up query generation. Single retrieval issues the original question once to a dense retriever, represented by NV-Embed-v2 [4]. Graph-based RAG structures the corpus before any question arrives, through hierarchical summaries, entity graphs, or proposition paths, represented by RAPTOR, GraphRAG, HippoRAG, HippoRAG 2, and PropRAG [9, 8, 5–7]. Iterative methods issue a sequence of queries, generated online from intermediate reasoning or planned up front by decomposing the question, represented by IRCoT, Self-Ask, and Least-to-Most [10, 12, 13]. A comparison with the fine-tuned decomposer TRQA under its own protocol is in Appendix J. Agentic methods cast multi-hop QA as code generation and execution over the corpus, represented by a coding agent [19] and a Recursive Language Model [20]. In the full-corpus setting, PropRAG is omitted for HotpotQA-full, because running its LLM-based

graph construction over 5.2M passages would cost more than $2,500 in API calls alone, beyond the budget available to us. All methods use GPT-4o-mini as the reader unless otherwise stated, and `Hi-Q` uses NV-Embed-v2, $k = 5$, maximum recursion depth $d_{\max} = 4$, and temperature 0. Appendices F and G measure the robustness of these two choices by substituting other readers (Llama-3.3-70B, Qwen3-30B-A3B) and other embedding models (text-embedding-3-large, Qwen3-Embedding-8B).

4.2 Main Results

`Hi-Q` outperforms iterative, graph-based, and agentic baselines. Table 1 evaluates the 1,000-question subsets in the full-corpus setting. `Hi-Q` reaches 52.3 EM and 64.0 F1 on average, outperforming IRCoT by 15.2 EM / 18.2 F1, with per-benchmark margins of 5.4 EM / 6.3 F1 on HotpotQA-full, 13.0 EM / 15.7 F1 on MuSiQue-full, and 27.0 EM / 32.5 F1 on 2Wiki-full. It also surpasses PropRAG by 11.6 EM / 12.0 F1 on MuSiQue-full and by 9.9 EM / 12.1 F1 on 2Wiki-full, the two corpora over which PropRAG's graph could be constructed within our compute budget, and does so without constructing a graph over the corpus. `Hi-Q` also improves over the coding agent and the Recursive Language Model by 30.8 EM / 26.9 F1 and by 24.4 EM / 26.7 F1 on average. An agent's program controls when retrieval is called, but the query passed to each call is still written by the model with no check that it is expressed at a granularity the corpus can match, so a coarse or under-specified query returns the same distractors inside a program as in single-shot retrieval. `Hi-Q` makes that check the control signal, expanding a node only when its retrieved evidence fails to resolve it and rewriting each dependent sub-query with the resolved bridge entity before issuing it.

The gains also hold in the controlled-corpus setting of prior work. To compare against a wider range of graph-based methods, we also evaluate in the controlled-corpus setting used in the HippoRAG and PropRAG evaluations [5–7] (Table 1, Controlled corpus). `Hi-Q` achieves 57.9 EM and 69.3 F1 on average, above every graph-based method, including HippoRAG 2 by 7.3 EM / 5.9 F1 and PropRAG by 5.6 EM / 3.9 F1, and above IRCoT by 13.7 EM / 15.8 F1. The largest gains again occur on MuSiQue, where shortcut reasoning is limited and evidence dependencies must be resolved explicitly: 7.3 EM / 5.0 F1 over PropRAG and 13.1 EM / 14.8 F1 over IRCoT. The gold passages are present in both settings, but here they compete with far fewer candidates, which compresses the differences that the full corpora expose. Every improvement over IRCoT and PropRAG is significant, in all six controlled-corpus comparisons and all five full-corpus ones, with 19 of the 22 EM and F1 tests at $p < 10^{-4}$. The smallest margin, 3.2 F1 over PropRAG on 2Wiki in the controlled-corpus setting, reaches $p = 0.0033$, and the F1 gain over PropRAG on MuSiQue-full has a 95% confidence interval of $[+9.44, +14.57]$.

Higher pooled recall does not by itself produce better answers. Self-Ask attains the highest average Recall@5 in the controlled-corpus setting, but it issues more sub-queries and retrieves roughly 33% more documents per question than `Hi-Q`, and because k is applied per retrieval call while recall is computed over the union of all returned passages, more calls buy a larger effective budget at the same nominal k. A larger candidate pool therefore raises recall mechanically while handing the reader more distractors, and the pooled score says nothing about execution alignment: whether each required passage was visible at the reasoning step that needed it, and whether the information it carried reached final synthesis. Self-Ask accordingly reaches higher average recall with much lower answer accuracy, 35.7 EM against `Hi-Q`'s 57.9. `Hi-Q`'s advantage is therefore not raw retrieval volume; it targets query units whose retrieved evidence is also resolvable by the reader.

4.3 The Unresolved-Support Signal

The signal is actionable: refinement recovers evidence that root retrieval misses. Decomposition makes retrievable what the coarse root query could not retrieve. A trigger fires when the passages retrieved for the root query fail to resolve the question, so triggered cases are those in which the question at its original granularity did not retrieve its evidence. On the triggered MuSiQue cases, we test whether the sub-queries issued after decomposition retrieve the gold passages that the root query missed, comparing root@5 with **Leaf@5**, the top-5 of the pooled leaf retrievals ranked by retriever similarity and therefore budget-matched to root@5, and with **Leaf@*all***, the full pool. Figure 3 reports all-gold cover, the fraction of triggered questions for which every gold passage is retrieved. On the Clean subset, the root query covers all gold passages in only 7.9% of triggered questions; the sub-queries raise this to 42.7% at the same five-passage budget and to 57.7% over the full pool, and

Table 1: **Main results.** QA and retrieval performance in the full-corpus setting, over each benchmark's complete corpus (Full corpus), and in the controlled-corpus setting of prior work, over pooled supporting and distractor passages (Controlled corpus). *Results are reproduced. †PropRAG is reported only on MuSiQue and 2Wiki under full-corpus retrieval.

Method	HotpotQA				MuSiQue				2Wiki				Avg			
	EM	F1	R@2	R@5	EM	F1	R@2	R@5	EM	F1	R@2	R@5	EM	F1	R@2	R@5
Full corpus																
PropRAG*,†	N/A				25.8	38.6	44.4	57.9	52.1	59.3	56.3	74.7	N/A			
Self-Ask*	30.0	37.5	63.2	74.5	22.2	28.1	49.9	65.2	41.7	46.9	**62.0**	**89.9**	31.3	37.5	58.4	76.5
Least-to-Most*	45.4	59.1	58.8	71.0	24.3	34.0	44.4	56.3	30.7	34.0	57.8	68.2	33.5	42.4	53.7	65.2
IRCoT*	52.0	63.6	68.9	80.2	24.4	34.9	46.9	63.0	35.0	38.9	57.5	76.9	37.1	45.8	57.8	73.4
Coding agent*	31.1	47.1	64.9	76.6	16.8	29.6	43.7	58.4	16.5	34.7	59.9	81.1	21.5	37.1	56.2	72.0
RLM*	30.1	41.1	21.5	25.8	19.6	29.1	21.7	29.7	34.0	41.6	25.9	35.4	27.9	37.3	23.0	30.3
`Hi-Q`	**57.4**	**69.9**	**70.5**	**82.0**	**37.4**	**50.6**	**53.4**	**70.8**	**62.0**	**71.4**	61.0	86.3	**52.3**	**64.0**	**61.6**	**79.7**
Controlled corpus																
NV-Embed-v2	57.3	71.0	84.1	94.5	32.8	46.0	52.7	69.7	54.4	60.8	67.1	76.5	48.2	59.3	68.0	80.2
RAPTOR	50.6	64.7	78.6	90.2	27.7	39.2	49.1	61.0	39.7	48.4	58.4	66.0	39.3	50.8	62.0	72.4
GraphRAG	51.4	67.6	–	–	27.0	42.0	–	–	45.7	61.0	–	–	41.4	56.9	–	–
HippoRAG	46.3	60.0	60.1	78.5	24.0	35.9	41.8	52.4	59.4	67.3	68.4	87.0	43.2	54.4	56.8	72.6
HippoRAG 2	56.3	71.1	80.5	95.7	35.0	49.3	53.5	74.2	60.5	69.7	74.6	90.2	50.6	63.4	69.5	86.7
PropRAG*	59.1	73.8	85.1	**96.9**	37.7	52.2	55.2	75.8	60.2	70.2	74.8	91.3	52.3	65.4	71.7	88.0
Self-Ask*	45.4	62.3	82.5	92.6	29.2	43.8	**60.0**	**77.4**	32.6	58.7	**84.4**	**98.8**	35.7	54.9	**75.6**	**89.6**
Least-to-Most*	55.9	71.3	78.5	92.4	29.1	39.5	51.8	66.2	37.0	41.5	68.0	75.2	40.7	50.8	66.1	77.9
IRCoT*	59.9	72.0	85.2	95.2	31.9	42.4	55.9	74.0	40.9	46.0	76.8	90.8	44.2	53.5	72.6	86.7
`Hi-Q`	**64.3**	**77.4**	**85.8**	96.2	**45.0**	**57.2**	56.9	75.7	**64.5**	**73.4**	78.5	93.4	**57.9**	**69.3**	73.7	88.4

38.7% of these initially failed questions are ultimately answered correctly. Because most of the gain is already present at the matched Leaf@5 budget, it comes from the sub-queries ranking the gold passages highly rather than from retrieving more passages: the evidence was in the corpus all along, and refining the query to the granularity at which the corpus stores it is what makes it retrievable.

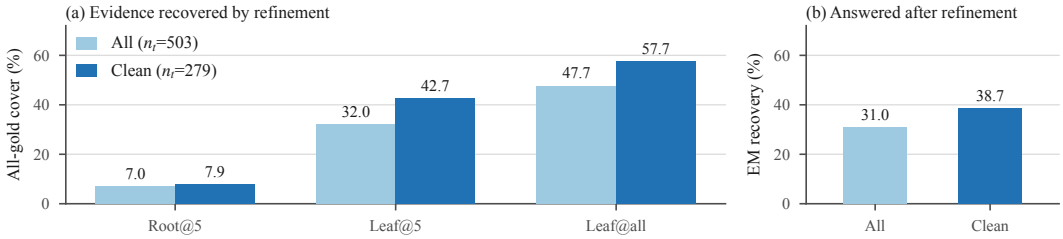


Figure 3: Trigger recovery on MuSiQue. (a) All-gold cover of Root@5, Leaf@5, and Leaf@*all* on all triggered questions (n_t=503) and on the Clean subset (n_t=279). (b) EM recovery, the fraction of triggered questions ultimately answered correctly.

The signal is precise enough to route decomposition without a learned classifier. `Hi-Q` uses a deterministic trigger: a node is expanded only when the resolution operator returns UNRESOLVED. What matters is not why each unresolved-support signal arose, but how often one is raised while the retrieved evidence does support the current query, since that is the only case in which withholding an answer is the wrong action. To test this, we manually analyze 100 randomly sampled UNRESOLVED triggers on MuSiQue and group them by cause in Figure 4(a). Annotation errors (14%) reflect defective benchmark evidence and malformed sub-queries (10%) are upstream formulation failures; in both the evidence genuinely fails to support the query, so abstaining is the correct action and neither counts as a trigger error. Of the remaining 76 triggers, 66 (87%) are granularity mismatches, the case the trigger is designed to catch, and only 10 are false abstentions, in which the operator returned UNRESOLVED although the retrieved passages did support the query. These 10 are the only false triggers: the signal fires on a genuinely unresolved query in 90% of cases, precise enough to route decomposition without a learned classifier.

The signal is conservative under both answerable and adversarially unanswerable inputs. The resolution operator can err in two directions. It may return UNRESOLVED although the retrieved evidence supports the current query, in which case a question that was already answerable at its current granularity is decomposed; or it may answer although the evidence does not support the query, in which case the node is never expanded and an unsupported answer propagates. We measure both directions on MuSiQue: the False Rejection Rate (FRR) on 100 normal queries, and the False Acceptance Rate (FAR) on 50 adversarially unanswerable queries constructed by replacing answerable conditions following the SQuAD 2.0 methodology [26]; for diagnosis only, each unresolved output carries a failure-type label that the policy does not use. Figure 4(b) shows FRR $= 11\%$ on normal queries, with all 11 null outputs labeled as missing or incomplete evidence rather than blind abstention. On adversarially unanswerable queries, `Hi-Q` identifies 49 of 50 as unanswerable, yielding FAR $= 2\%$. The trigger therefore rarely answers without evidence, and its errors fall on the withholding side, whose cost is an unnecessary expansion bounded by the always-decompose ablation in Table 2, rather than on the answering side, which would let an unsupported answer stand.

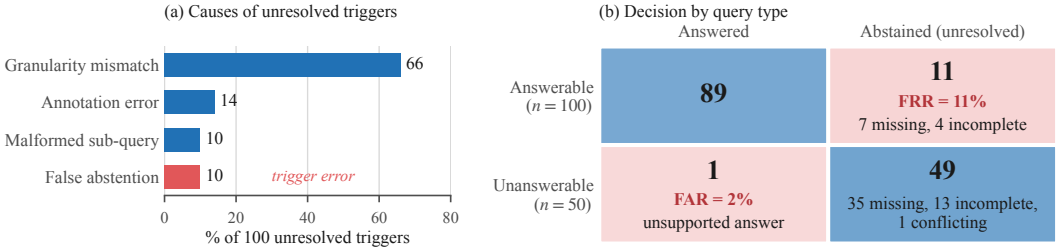


Figure 4: Trigger analysis on MuSiQue. (a) Causes of 100 sampled UNRESOLVED triggers. (b) Decisions on 100 answerable and 50 adversarially unanswerable queries; the off-diagonal cells are the two error types, false rejection (FRR) and false acceptance (FAR), and the UNRESOLVED cells list the failure-type labels.

4.4 Component Ablations and Reasoning Depth

Each part of the control loop earns its cost. Table 2 removes the components of `Hi-Q` one at a time. Replacing hierarchical refinement with a static one-shot decomposition lowers the average from 57.9 EM / 69.3 F1 to 51.5 EM / 63.7 F1, so retrieval feedback is needed to adjust granularity. Removing dependency awareness causes a larger drop, to 47.1 EM / 56.0 F1, because a dependent sub-query issued before its prerequisite is resolved is under-specified. Finally, replacing the failure-aware trigger with an always-decompose policy reduces performance from 60.7 EM / 70.8 F1 to 57.3 EM / 68.5 F1 on the sample of 100 queries per dataset. This shows that unconditional decomposition fragments already answerable queries, while `Hi-Q`'s trigger expands only when evidence support is insufficient. Because always-decompose is the limiting case of over-triggering, firing on every query, this 3.4-point drop also bounds what any trigger error can cost. Round-trip verification has a small overall effect but matters at depth: the gap from removing it widens with the number of hops (Table 3), reaching 1.0 F1 at 4 hops on MuSiQue, where a decomposition error can compound across successive splits. The verifier repairs 14.8–18.8% of triggered decompositions, with per-dataset trigger and repair frequencies in Appendix H. Consistently, generating several root decompositions and voting over them raises accuracy only modestly at near-linear token cost, indicating that early decomposition errors rarely survive to the final answer (Appendix K). Each component thus removes a different failure: refinement lets granularity follow what retrieval returns, dependency ordering keeps dependent sub-queries specified, the trigger spares already answerable queries at a fraction of the cost of decomposing every node, and the verifier keeps decomposition errors from compounding at depth.

`Hi-Q` degrades more gracefully than baselines as reasoning depth increases. Figure 5 stratifies MuSiQue performance by hop count. Every method loses accuracy as the number of hops grows, but `Hi-Q` keeps the highest F1 at each depth, reaching 55.2 and 39.2 F1 at 3 and 4 hops against 43.4 and 26.9 for IRCoT and 43.1 and 35.4 for PropRAG. Deeper questions are where an early wrong bridge entity or a dropped constraint propagates furthest, so this is where resolving prerequisite sub-queries before dependent ones is expected to help, as Section 3.3 argues.

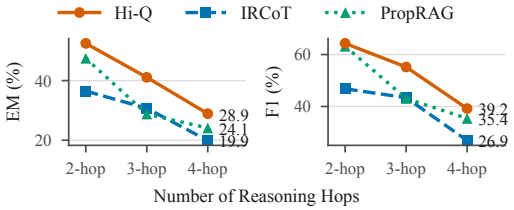
Table 2: Control-flow ablation of `Hi-Q`. Bold indicates the best value in each column. †100 randomly sampled queries per dataset (300 in total), due to the prohibitive cost of recursive always-decompose.

Setting	HotpotQA		MuSiQue		2Wiki		Avg	
	EM	F1	EM	F1	EM	F1	EM	F1
Decomposition variant (full subset):								
`Hi-Q`	**64.3**	**77.4**	**45.0**	**57.2**	**64.5**	**73.4**	**57.9**	**69.3**
w/o Hierarchical Decomp.	58.0	73.6	39.0	51.2	57.6	66.3	51.5	63.7
w/o Dependency-aware Decomp.	62.5	74.9	34.2	43.2	44.5	50.0	47.1	56.0
Triggering policy (N=100 per dataset†):								
`Hi-Q` (Failure-aware)	**62.0**	**76.6**	**49.0**	**57.2**	**71.0**	**78.7**	**60.7**	**70.8**
Always-decompose	58.0	75.3	45.0	53.4	69.0	76.9	57.3	68.5

Table 3: Verification ablation by hop count (F1 on MuSiQue).

Hop	`Hi-Q`	w/o RT
Overall	57.2	56.5
2-hop	64.3	63.7
3-hop	55.2	54.4
4-hop	39.2	38.2

Figure 5: EM and F1 on MuSiQue across hop counts.



4.5 Computational Cost

Table 4 reports the per-question cost. `Hi-Q` takes 13.5 seconds against PropRAG's 23.1, which covers graph traversal but not corpus-wide graph construction. Against IRCoT, `Hi-Q` uses 70.5% fewer tokens in total but about twice the latency and 7.5× the output tokens; this overhead comes from decomposition, verification, and descendant resolution, which run only for unresolved nodes, so it is paid by the queries that fail to resolve rather than by every query. A cost-matched configuration shortens the rationales and caps the recursion depth at 1, leaving the policy and every operator unchanged. At the same number of LLM calls as IRCoT (2.93 versus 2.92), it uses 9.4× fewer input tokens, runs 25% faster, and costs 8.5× less per question at GPT-4o-mini list prices (Appendix E), while scoring 10.4 EM / 11.6 F1 higher on average ($p \leq 0.0036$ on every benchmark). The default configuration adds a further 3.3 EM / 4.2 F1 at roughly one third of IRCoT's API token cost.

Table 4: Computational cost per question, macro-averaged over the three benchmarks. Latency is end-to-end wall-clock time, retrieval and reading included; LLM calls and tokens count reader usage. Corpus embedding and PropRAG's offline graph construction are excluded.

Method	Latency (s)	LLM Calls	Input toks	Output toks
PropRAG	23.1	1	1,411	88
IRCoT	6.4	2.92	43,701	74
Self-Ask	8.0	6.36	4,591	98
Least-to-Most	11.0	5.38	6,874	424
`Hi-Q` **(cost-matched)**	4.8	2.93	4,641	131
`Hi-Q`	13.5	5.7	12,343	558

5 Conclusion

The main finding of this work is that multi-hop RAG benefits from treating query granularity as a control variable rather than as a fixed design choice. `Hi-Q` operationalizes this idea by testing each query node against retrieved evidence, expanding only unresolved nodes, and preserving prerequisite-to-dependent ordering during refinement. Across the full-corpus and controlled-corpus settings, this evidence-conditioned control improves answer accuracy over graph-based, iterative, and code-executing agent baselines while avoiding corpus-wide knowledge graph construction. More broadly, multi-hop retrieval should be viewed not only as a problem of finding more passages, but as a problem of discovering the query granularity at which each reasoning step becomes retrievable and answerable.

Acknowledgments and Disclosure of Funding

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2024-00509258, Global AI Frontier Lab, 40%; No. IITP-2026-RS-2026-25616370, AI Star Fellowship Support Program, 20%) and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2025-00517736, 40%). This work utilized two GPUs provided by the National AI Project of the Ministry of Science and ICT (MSIT), Korea.

References

[1] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

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

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

[4] Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. NV-Embed: Improved techniques for training LLMs as generalist embedding models. In *The Thirteenth International Conference on Learning Representations*, 2025.

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

[6] Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From RAG to memory: Non-parametric continual learning for large language models. In *Forty-second International Conference on Machine Learning*, 2025.

[7] Jingjin Wang and Jiawei Han. PropRAG: Guiding retrieval with beam search over proposition paths. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 6212–6227, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.317. URL `https://aclanthology.org/2025.emnlp-main.317/`.

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

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

[10] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 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, 2023.

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

[12] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models, 2023. URL https://arxiv.org/abs/2210.03350.

[13] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. Least-to-most prompting enables complex reasoning in large language models, 2023. URL https://arxiv.org/abs/2205.10625.

[14] Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Ashish Sabharwal, and Peter Clark. Decomposed prompting: A modular approach for solving complex tasks. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

[15] Kun Zhang, Jiali Zeng, Fandong Meng, Yuanzhuo Wang, Shiqi Sun, Long Bai, Huawei Shen, and Jie Zhou. Tree-of-reasoning question decomposition for complex question answering with large language models. In *Proceedings of the AAAI Conference on artificial intelligence*, volume 38, pages 19560–19568, 2024.

[16] Linhao Ye, Lang Yu, Zhikai Lei, Qin Chen, Jie Zhou, and Liang He. Optimizing question semantic space for dynamic retrieval-augmented multi-hop question answering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17814–17824, Vienna, Austria, 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.acl-long.871/.

[17] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection, 2023. URL https://arxiv.org/abs/2310.11511.

[18] Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*, 2024.

[19] Weili Cao, Xunjian Yin, Bhuwan Dhingra, and Shuyan Zhou. Coding agents are effective long-context processors, 2026. URL https://arxiv.org/abs/2603.20432.

[20] Alex L. Zhang, Tim Kraska, and Omar Khattab. Recursive language models, 2026. URL https://arxiv.org/abs/2512.24601.

[21] Jiashuo Sun, Jimeng Shi, Yixuan Xie, Saizhuo Wang, Jash Rajesh Parekh, Pengcheng Jiang, Zhiyi Shi, Jiajun Fan, Qinglong Zheng, Peiran Li, Shaowen Wang, Ge Liu, and Jiawei Han. Retrieval is cheap, show me the code: Executable multi-hop reasoning for retrieval-augmented generation, 2026. URL https://arxiv.org/abs/2605.12975.

[22] Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity, 2024. URL https://arxiv.org/abs/2403.14403.

[23] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 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, 2018.

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

[25] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 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, 2020.

[26] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, 2018.

[27] Aaron Grattafiori et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[28] An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[29] Chia-Yuan Chang, Zhimeng Jiang, Vineeth Rakesh, Menghai Pan, Chin-Chia Michael Yeh, Guanchu Wang, Mingzhi Hu, Zhichao Xu, Yan Zheng, Mahashweta Das, and Na Zou. MAIN-RAG: Multi-agent filtering retrieval-augmented generation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2607–2622, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.131. URL https://aclanthology.org/2025.acl-long.131/.

[30] Ivan Stelmakh, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang. ASQA: Factoid questions meet long-form answers, 2022. URL https://arxiv.org/abs/2204.06092.

[31] Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. ELI5: Long form question answering, 2019. URL https://arxiv.org/abs/1907.09190.

[32] Akari Asai, Jungo Kasai, Jonathan H. Clark, Kenton Lee, Eunsol Choi, and Hannaneh Hajishirzi. XOR QA: Cross-lingual open-retrieval question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2021. URL https://arxiv.org/abs/2010.11856.

[33] Shayne Longpre, Yi Lu, and Joachim Daiber. MKQA: A linguistically diverse benchmark for multilingual open domain question answering, 2020. URL https://arxiv.org/abs/2007.15207.

[34] Anastasia Krithara, Anastasios Nentidis, Konstantinos Bougiatiotis, and Georgios Paliouras. BioASQ-QA: A manually curated corpus for biomedical question answering. *Scientific Data*, 10(170), 2023. doi: 10.1038/s41597-023-02068-4.

[35] Pranab Islam, Anand Kannappan, Douwe Kiela, Rebecca Qian, Nino Scherrer, and Bertie Vidgen. FinanceBench: A new benchmark for financial question answering, 2023. URL https://arxiv.org/abs/2311.11944.

[36] Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Wang. HybridQA: A dataset of multi-hop question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020. URL https://arxiv.org/abs/2004.07347.

[37] Wenhu Chen, Ming-Wei Chang, Eva Schlinger, William Wang, and William W. Cohen. Open question answering over tables and text. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://arxiv.org/abs/2010.10439.

[38] Alon Talmor, Ori Yoran, Amnon Catav, Dan Lahav, Yizhong Wang, Akari Asai, Gabriel Ilharco, Hannaneh Hajishirzi, and Jonathan Berant. MultiModalQA: Complex question answering over text, tables and images. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://arxiv.org/abs/2104.06039.

[39] Yingshan Chang, Mridu Narang, Hisami Suzuki, Guihong Cao, Jianfeng Gao, and Yonatan Bisk. WebQA: Multihop and multimodal QA, 2021. URL https://arxiv.org/abs/2109.00590.

A Algorithm

Algorithm 1 instantiates the evidence-conditioned search policy formalized in Section 3.1. Each line annotated with a π-action corresponds to one of the policy's three decisions (STOP, FAIL, EXPAND). The three operators are realized as prompted-LLM modules whose prompts are listed in Appendix O: the resolution operator $\mathcal{G}$ encapsulates query refinement, retrieval, and reference-aware answering, returning the resolution status $s \in \{\text{RESOLVED}, \text{UNRESOLVED}\}$ together with the answer a and retrieved evidence D; the binary expansion operator $\mathcal{B}$ proposes a dependency-ordered pair of sub-queries; the semantic coverage verifier $\mathcal{V}$ checks and repairs the split before recursion. The interaction history $\mathcal{H}$ is updated in place, so the right branch is resolved under an $\mathcal{H}$ already containing the answer to the left branch.

Algorithm 1 Evidence-conditioned hierarchical query refinement (`Hi-Q`).

Require: Original query Q, corpus C, retrieval size k, max recursion depth $d_{\max}$
Ensure: Final answer A

$\mathcal{H} \leftarrow [\,]$ {interaction history}
$A \leftarrow \textsc{Solve}(Q, \mathcal{H}, 0)$
return A
function $\textsc{Solve}(q, \mathcal{H}, d)$
  $(s, a, D) \leftarrow \mathcal{G}(q, \mathcal{H}, C)$ {resolution: refine $\rightarrow$ retrieve $\rightarrow$ answer}
  $\textsc{Append}(\mathcal{H}, (q, D, a))$
  if $s =$ RESOLVED **then**
    return a {$\pi(\tilde{x}, s) =$ STOP}
  end if
  if $d = d_{\max}$ **then**
    return $\perp$ {$\pi(\tilde{x}, s) =$ FAIL: depth budget exhausted}
  end if
  $(q_{\text{left}}, q_{\text{right}}) \leftarrow \mathcal{B}(q, \mathcal{H}, Q)$ {binary expansion}
  if $(q_{\text{left}}, q_{\text{right}}) = \perp$ **then**
    return $\perp$ {$\pi(\tilde{x}, s) =$ FAIL: non-decomposable}
  end if
  $(q_{\text{left}}, q_{\text{right}}) \leftarrow \mathcal{V}(Q, q, q_{\text{left}}, q_{\text{right}})$ {semantic coverage check / repair}
  if $(q_{\text{left}}, q_{\text{right}}) = \perp$ **then**
    return $\perp$ {$\pi(\tilde{x}, s) =$ FAIL: split not repairable}
  end if
  $a_{\text{left}} \leftarrow \textsc{Solve}(q_{\text{left}}, \mathcal{H}, d+1)$ {prerequisite branch first; $\mathcal{H}$ updated in place}
  $\textsc{Update}(\mathcal{H}, (q_{\text{left}}, a_{\text{left}}))$ {record the left answer before the dependent branch}
  $a_{\text{right}} \leftarrow \textsc{Solve}(q_{\text{right}}, \mathcal{H}, d+1)$ {dependent branch grounded by updated $\mathcal{H}$}
  $a \leftarrow \textsc{Synthesize}(q, a_{\text{left}}, a_{\text{right}}, \mathcal{H}, Q)$
  return a
end function

The procedure terminates, and the depth limit alone bounds its cost. Execution forms a binary tree rooted at depth 0 whose maximum depth is $d_{\max}$, so at most $\sum_{j=0}^{d_{\max}} 2^j = 2^{d_{\max}+1} - 1$ nodes are visited and the resolution operator runs at most once per visited node. Expansion, verification, and synthesis occur only at internal nodes, of which there are at most $\sum_{j=0}^{d_{\max}-1} 2^j = 2^{d_{\max}} - 1$. Every recursive call increases the depth by one and no node at depth $d_{\max}$ expands, so every execution path is finite and the recursion stack holds at most $d_{\max} + 1$ frames. These are worst-case bounds over a fully expanded tree: at $d_{\max} = 4$ they permit 31 visited nodes, each costing at least one LLM call, against a measured mean of 5.7 calls per question (Section 4.5), because most nodes resolve without expanding.

The execution order of the two branches is an invariant of the procedure rather than a property of any particular split. At every node, SOLVE appends the current query, its retrieved evidence, and its answer status to $\mathcal{H}$ as soon as the resolution operator returns, before any branching decision is taken; the right child is then invoked only after the left call has returned on that same history. When the dependent branch begins, $\mathcal{H}$ therefore already holds the current-node record together with every record produced by the prerequisite subtree, including its retrieved evidence and partial context,

whether or not that subtree returned an answer. This makes the dependency constraint of Section 3.3 an execution guarantee; it does not assert that the intermediate answers are correct.

B Derivation of the STOP/EXPAND Threshold Rule

This section derives the threshold rule stated in Section 3 and decomposes the excess risk of an imperfect estimator. A node is *expansion-admissible* when both actions are genuinely available to it, that is, when

$$x = (q, \mathcal{H}, d) \in \mathcal{X}_{\text{feas}} := \{\, (q, \mathcal{H}, d) \;:\; d < d_{\max} \text{ and } q \text{ is decomposable} \,\};$$

outside $\mathcal{X}_{\text{feas}}$, EXPAND is inadmissible: a node that resolves still stops, and an unresolved one is forced to FAIL; every statement below is conditioned on admissibility. Throughout, $\tilde{x} = (x, D)$ is the post-resolution state of an expansion-admissible node, with $D = R_k(q', C)$ the retrieved evidence, and $Z \in \{\mathrm{R}, \mathrm{U}\}$ indicates whether q is resolvable (R) or unresolved (U) from D together with $\mathcal{H}$. Write $p(\tilde{x}) = \Pr[Z = \mathrm{U} \mid \tilde{x}]$.

Cost model. Two actions are available at such a node. Taking EXPAND at a node that was in fact resolvable incurs $\Delta_{\mathrm{R}}(\tilde{x}) > 0$, and taking STOP at a node that was in fact unresolved incurs $\Delta_{\mathrm{U}}(\tilde{x}) > 0$. Both penalties are cost-to-go quantities measured over the subtree that the action induces, and therefore include descendant retrieval and LLM calls, drift risk, synthesis, and terminal answer loss. The action that matches the realized Z is taken as the reference and assigned zero excess cost, so the conditional expected costs are

$$\mathbb{E}[\,\mathrm{cost}(\textsc{expand}) \mid \tilde{x}\,] = \big(1 - p(\tilde{x})\big)\,\Delta_{\mathrm{R}}(\tilde{x}),$$
$$\mathbb{E}[\,\mathrm{cost}(\textsc{stop}) \mid \tilde{x}\,] = p(\tilde{x})\,\Delta_{\mathrm{U}}(\tilde{x}).$$

We write $\mathcal{R}(\pi)$ for the expected cost of a policy π over the post-resolution states $\tilde{x} = (x, D)$ with $x \in \mathcal{X}_{\text{feas}}$.

Threshold rule. EXPAND is optimal exactly when its conditional expected cost is no larger (state arguments omitted):

$$\big(1-p\big)\Delta_{\mathrm{R}} \;\le\; p\,\Delta_{\mathrm{U}} \iff \Delta_{\mathrm{R}} \;\le\; p\,\big(\Delta_{\mathrm{R}} + \Delta_{\mathrm{U}}\big) \iff p \;\ge\; \theta(\tilde{x}) := \frac{\Delta_{\mathrm{R}}(\tilde{x})}{\Delta_{\mathrm{R}}(\tilde{x}) + \Delta_{\mathrm{U}}(\tilde{x})},$$

which is the rule given in Section 3. The threshold is state-dependent because both penalties are. Since the penalties are cost-to-go quantities, the resulting decision is node-wise but not myopic; it does not follow, and we do not claim, that greedily applying the rule at every node yields a globally optimal query tree.

Dominance over unconditional routing. Before any estimator is fixed, the cost model already separates evidence-conditioned control from policies that commit to one action in advance. Let π_{orc} be the oracle that observes the realized Z itself, stopping when $Z = \mathrm{R}$ and expanding when $Z = \mathrm{U}$; it is a stronger reference than the π^* of the next paragraph, which sees only the posterior $p(\tilde{x})$. Always-stop agrees with π_{orc} except on $\{Z = \mathrm{U}\}$, where it gives up $\Delta_{\mathrm{U}}(\tilde{x})$, and always-expand agrees except on $\{Z = \mathrm{R}\}$, where it gives up $\Delta_{\mathrm{R}}(\tilde{x})$, so

$$\mathcal{R}(\pi_{\mathrm{stop}}) - \mathcal{R}(\pi_{\mathrm{orc}}) = \Pr[Z = \mathrm{U}]\,\mathbb{E}\big[\Delta_{\mathrm{U}}(\tilde{x}) \mid Z = \mathrm{U}\big],$$
$$\mathcal{R}(\pi_{\mathrm{exp}}) - \mathcal{R}(\pi_{\mathrm{orc}}) = \Pr[Z = \mathrm{R}]\,\mathbb{E}\big[\Delta_{\mathrm{R}}(\tilde{x}) \mid Z = \mathrm{R}\big].$$

Each difference is strictly positive whenever its state occurs with positive probability. This is the value of observing the evidence state, realized by the oracle; an operational policy realizes it only to the extent that the excess risk of its estimator, derived next, stays below these margins. Appendix D measures the converse: a learned router that never observes the evidence collapses onto a near-constant policy and scores below that constant policy itself.

Excess risk of an estimator. Let $\hat{Z}$ be any estimator inducing a policy $\pi_{\hat{Z}}$, and let π^* be the rule above. The two actions differ in conditional expected cost by

$$\big(1-p\big)\Delta_{\mathrm{R}} - p\,\Delta_{\mathrm{U}} = \big(\Delta_{\mathrm{R}} + \Delta_{\mathrm{U}}\big)\big(\theta - p\big),$$

so disagreeing with π^* at $\tilde{x}$ costs $(\Delta_{\mathrm{R}} + \Delta_{\mathrm{U}})\,|p - \theta|$ and agreeing costs nothing. Taking expectations,

$$\mathcal{R}(\pi_{\hat{Z}}) - \mathcal{R}(\pi^*) = \mathbb{E}\Big[\big(\Delta_{\mathrm{R}}(\tilde{x}) + \Delta_{\mathrm{U}}(\tilde{x})\big)\,\big|p(\tilde{x}) - \theta(\tilde{x})\big| \cdot \mathbf{1}\{\pi_{\hat{Z}}(\tilde{x}) \neq \pi^*(\tilde{x})\}\Big].$$

The indicator splits into the two error directions: *premature stopping*, where $p > \theta$ but the estimator stops, and *unnecessary expansion*, where $p < \theta$ but the estimator expands. Each contributes in proportion to how far p lies from the threshold, so errors on states where the evidence is genuinely ambiguous are cheap and errors on clear-cut states are expensive. This is the sense in which the resolution test is an estimator of a defined decision rather than a heuristic: it is a hard classifier $\hat{Z}$ whose excess risk is governed by the expression above, and any calibrated or trained replacement is evaluated on the same scale.

Why the estimator must observe the evidence state. The running example of Figure 2 instantiates the difficulty. For a fixed corpus and a deterministic retriever, a fixed query yields a single evidence set, so the argument is made over the distribution of problem instances (Q, C) on which a policy is evaluated: the same question can appear against different corpora, or different versions of a corpus, that rank its evidence differently. Take q^* to be "When was the start of the battle of the birthplace of the performer of III?", and hold the retriever and the gold answer fixed. In an instance whose root top-k contains the album, performer, birthplace, and battle passages, the node is resolvable and STOP is optimal. In an instance whose top-k omits the passage naming the performer, which is still in the corpus and merely not ranked into the budget, EXPAND is optimal, because the dependent sub-queries retrieve against a narrower target. The query is identical in the two instances and the difference lies entirely in which passages the retrieval budget surfaced, so no function of q^* alone separates them. Consider two post-resolution states $\tilde{x}_1 = (x, D_1)$ and $\tilde{x}_2 = (x, D_2)$ that share the search state x but differ in the retrieved evidence, $D_1 \neq D_2$, and suppose $\pi^*(\tilde{x}_1) = \text{STOP}$ while $\pi^*(\tilde{x}_2) = \text{EXPAND}$, each occurring with positive probability under the instance distribution. Any policy measurable with respect to q alone assigns the same action to both, and therefore disagrees with π^* on at least one of them. Its excess risk is bounded below by

$$\min_{i \in \{1,2\}} \Pr[\tilde{x}_i] \cdot \big(\Delta_{\mathrm{R}}(\tilde{x}_i) + \Delta_{\mathrm{U}}(\tilde{x}_i)\big)\big|p(\tilde{x}_i) - \theta(\tilde{x}_i)\big| \;>\; 0,$$

a quantity that no amount of query-only training data reduces, since the two states are indistinguishable to such a policy. Learning is therefore not what separates `Hi-Q` from query-level routing; conditioning on the realized evidence state is. Appendix C reports what happens when a learned, calibrated estimator that does observe $\tilde{x}$ replaces the resolution test.

C A Learned and Calibrated STOP/EXPAND Policy

`Hi-Q`'s routing decision is implemented as a training-free test: a node is expanded when the resolution operator returns UNRESOLVED. This section reports a controlled experiment showing that the same decision admits a learned, calibrated estimator, and that replacing the training-free test with one changes neither accuracy nor cost enough to justify the added machinery. The experiment therefore establishes that the STOP/EXPAND interface is instantiable by a trained controller, not that a trained controller is required.

All runs use the MuSiQue controlled-corpus setting. For each eligible saved state we force both STOP and EXPAND through to a final answer and label the state with the lower-loss action, collecting $3{,}060$ labelled decision states from 817 training questions. The LLM, the retriever, and the NV-Embed-v2 encoder are frozen, and only a logistic action head is trained on the resulting state representations. Model selection, Platt calibration, and testing use disjoint sets of 200, 400, and $1{,}000$ questions; the $1{,}417$ questions used for training, selection, and calibration are drawn from the MuSiQue development pool outside the evaluation subset.

On held-out decision states, Platt calibration reduces expected calibration error from 0.111 to 0.082 and decision regret by 0.024 (95% CI $[0.005, 0.047]$). The action head is therefore not merely a classifier of convenience: its probabilities carry usable calibration, which is what the threshold form of the routing decision requires.

We then replace only the root STOP/EXPAND rule, leaving every other component at the configuration used in the paper, and rerun all $1{,}000$ test questions. Relative to the training-free policy, two

thresholds on the calibrated probability, one chosen for accuracy alone and one that also weights cost, lose 0.30 EM (95% CI $[-2.10, +1.50]$) and 1.40 EM (95% CI $[-3.20, +0.40]$) respectively; neither difference is statistically significant. LLM calls per question rise from 7.63 to 10.41 for the accuracy-only threshold and to 8.64 for the cost-weighted one. The learned estimator thus neither improves accuracy nor lowers cost, so we retain the training-free configuration. This is a root-level proof of instantiation rather than a learned replacement for every recursive decision, and we do not claim that the trained head computes the posterior of the underlying decision.

D Query-Level Routing

Adaptive-RAG [22] predicts question complexity before retrieval and selects a global no-, single-, or multi-step strategy. We train its classifier on the $1{,}417$ MuSiQue development questions outside our evaluation subset, so the router operates in-distribution, and evaluate it on MuSiQue-full with a shared Qwen3.6-27B reader and retrieval stack; the numbers are therefore not comparable with Table 1. It falls far short of `Hi-Q` (Table 5). The routing behavior is more informative than the gap: 959 of the $1{,}000$ questions are routed to multi-step retrieval, 41 to single-step, and none to no-retrieval, and a control policy that always selects multi-step scores above Adaptive-RAG itself. The learned routing decision therefore contributes nothing over a constant policy. Whether a query is resolvable at its current granularity depends on the evidence actually returned, not on the apparent complexity of the original question. A controller that conditions on that evidence remains compatible with `Hi-Q`, as Appendix C shows.

Table 5: Query-level routing on MuSiQue-full, under a shared Qwen3.6-27B reader and retrieval stack. *Always multi-step* is a constant-policy control for Adaptive-RAG. These values are not comparable with Table 1, which uses GPT-4o-mini. Bold indicates the best value in each column.

Method	EM	F1
Adaptive-RAG [22]	26.4	33.9
Always multi-step (control)	27.3	34.6
`Hi-Q`	**47.9**	**59.3**

E Evaluation Details

Table 1 marks reproduced baselines with an asterisk, but the asterisk alone does not make the decision rule auditable. We quote a published number only when the source reports the same dataset subset, corpus regime, reader setting, retriever, and metric protocol as our evaluation, and otherwise reproduce the method ourselves. The criterion is evaluation compatibility, not the age or expected strength of a baseline.

Every controlled-corpus row of Table 1, quoted or reproduced, uses the same 1,000-question subsets per dataset, GPT-4o-mini as the reader, NV-Embed-v2 as the retriever with $k = 5$, and the pooled Recall@k protocol defined below. Retrieval for every run is performed on one NVIDIA Tesla P40 GPU, except HotpotQA-full, which uses four NVIDIA RTX A6000 GPUs; reading is served by the OpenAI API. Retrieval scores passages by L2-normalized dot product, and the depth limit $d_{\max} = 4$ matches the maximum reasoning depth analyzed in MuSiQue and bounds recursive cost.

Recall@k for $k \in \{2, 5\}$ measures whether the gold supporting documents appear in the retrieved top-k passages. For multi-query methods, we pool the passages retrieved across all node- or step-level queries, deduplicate them by retaining the highest embedding score, and globally re-rank the pool before computing Recall@k; this keeps the evaluation budget k fixed across methods while making retrieval volume explicit through the call and token statistics in Table 4. Because all runs use temperature-zero decoding, the bootstrap intervals reported in Section 4 quantify uncertainty over the question population rather than run-to-run variability.

Cost accounting for Table 4: API cost is priced at GPT-4o-mini list prices as of August 2026, \$0.15 per 1M input tokens and \$0.60 per 1M output tokens, which is the basis of the $8.5\times$ figure in Section 4.5. The cost-matched configuration caps recursion at depth 1, so the root may still be expanded once but its children are not. Its EM gain over IRCoT holds on every benchmark, from $+3.5$ to $+18.1$, and

its output generation nonetheless remains 77% higher than IRCoT's (131 versus 74 tokens), so that operating point is cheaper without being uniformly lighter.

The quoted entries are taken from the GPT-4o-mini rows of [6], which uses the same subsets and retriever, with the QA numbers and the passage recall coming from two different appendix tables of that paper. GraphRAG has no recall entry there, since it does not directly produce passage retrieval results, which is why its recall cells are empty in Table 1. All full-corpus results in Table 1 are reproduced, as no prior work reports these methods on the complete corpora under our protocol. The coding agent and the Recursive Language Model are run under the same standardization, which confines the comparison to their control logic; because the original studies use stronger models and different native corpus interfaces, these are controlled adaptations rather than reproductions of their published configurations.

Table 6: Source of every baseline number in Table 1. All rows share the reader, retriever, subsets, and metric protocol described above.

Method	Corpus regime	EM / F1	Recall@2 / @5
NV-Embed-v2 [4]	controlled	Quoted	Quoted
RAPTOR [9]	controlled	Quoted	Quoted
GraphRAG [8]	controlled	Quoted	Not reported
HippoRAG [5]	controlled	Quoted	Quoted
HippoRAG 2 [6]	controlled	Quoted	Quoted
PropRAG [7]	controlled, MuSiQue-full, 2Wiki-full	Reproduced	Reproduced
Self-Ask [12]	controlled, full	Reproduced	Reproduced
Least-to-Most [13]	controlled, full	Reproduced	Reproduced
IRCoT [10]	controlled, full	Reproduced	Reproduced
Coding agent [19]	full	Reproduced	Reproduced
RLM [20]	full	Reproduced	Reproduced
`Hi-Q`	controlled, full	This work	This work

F Robustness across Reader Backbones

`Hi-Q`'s gains are not specific to a single reader backbone. To test reader robustness, we re-run the evaluation pipeline in the controlled-corpus setting with two additional LLM readers spanning different architectures: Llama-3.3-70B [27] and Qwen3-30B-A3B [28]. Tables 7 and 8 show that `Hi-Q`'s gains are not specific to a single reader backbone. Across two additional readers, Llama-3.3-70B and Qwen3-30B-A3B, `Hi-Q` retains the best average QA performance, achieving 60.2 EM / 71.2 F1 and 58.3 EM / 68.9 F1, respectively. It outperforms the strongest reproduced graph-based baseline, PropRAG, by $+3.6$ EM with Llama-3.3-70B and $+11.0$ EM with Qwen3-30B-A3B, while also outperforming iterative/decomposition baselines. This supports that evidence-conditioned refinement is robust across dense and MoE reader architectures.

Table 7: QA performance with Llama-3.3-70B [27] in the controlled-corpus setting. Bold indicates the best value in each column. *Results are reproduced.

Method	HotpotQA		MuSiQue		2Wiki		Avg	
	EM	F1	EM	F1	EM	F1	EM	F1
NV-Embed-v2	62.8	75.3	34.7	45.7	57.5	61.5	51.7	60.8
RAPTOR	56.8	69.5	20.7	28.9	47.3	52.1	41.6	50.2
GraphRAG	55.2	68.6	27.3	38.5	51.4	58.6	44.6	55.2
HippoRAG	52.6	63.5	26.2	35.1	65.0	71.8	47.9	56.8
HippoRAG 2	62.7	75.5	37.2	48.6	65.0	71.0	55.0	65.0
PropRAG*	62.6	76.0	41.5	53.6	65.6	74.3	56.6	68.0
Self-Ask*	58.8	71.7	33.9	47.8	64.9	74.9	52.5	64.8
Least-to-Most*	57.6	72.9	29.4	43.5	44.2	48.9	43.7	55.1
IRCoT*	59.7	72.0	31.2	40.7	68.6	78.3	53.2	63.7
`Hi-Q`	**65.4**	**78.0**	**45.0**	**56.6**	**70.3**	**78.9**	**60.2**	**71.2**

Table 8: QA performance with Qwen3-30B-A3B [28] in the controlled-corpus setting. Bold indicates the best value in each column. *Results are reproduced.

Method	HotpotQA		MuSiQue		2Wiki		Avg	
	EM	F1	EM	F1	EM	F1	EM	F1
PropRAG*	55.0	69.2	32.2	44.7	54.6	63.8	47.3	59.2
Self-Ask*	52.8	67.3	37.6	47.9	66.0	**75.2**	52.1	63.5
Least-to-Most*	53.8	67.9	33.1	43.1	37.4	42.5	41.4	51.2
IRCoT*	62.9	75.3	34.4	44.8	52.6	58.5	50.0	59.5
Hi-Q	**65.1**	**77.3**	**43.4**	**54.6**	**66.3**	74.8	**58.3**	**68.9**

G Robustness to Embedding Models

Hi-Q's retrieval gains are not tied to a single embedding backbone. Table 9 evaluates Hi-Q with text-embedding-3-large, Qwen3-Embedding-8B [28], and NV-Embed-v2. Across the three retrievers, Hi-Q improves average Recall@5 by 9.4, 10.3, and 8.2 points, respectively, over dense retrieval alone. NV-Embed-v2 gives the strongest overall recall, which motivates its use as the default encoder. The consistent gains across backbones indicate that Hi-Q's improvement comes from evidence-guided query refinement rather than from a particular embedding model.

Table 9: Average Recall@5 across embedding models. Bold indicates the best value in each column.

Retriever	Dense	Hi-Q
text-embedding-3-large	74.5	83.9
Qwen3-Embedding-8B	68.7	79.0
NV-Embed-v2	**80.2**	**88.4**

H Trigger Rate and Round-trip Repair Statistics

Trigger and repair statistics show that Hi-Q decomposes mainly on benchmarks where query–evidence granularity mismatch is more frequent, and that verification is actively used rather than acting as a passive check. Table 10 reports how often decomposition is triggered and how often round-trip verification repairs a proposed decomposition across datasets. The trigger rate is much lower on HotpotQA (10.7%) than on MuSiQue (45.2%) and 2Wiki (48.6%), which is consistent with HotpotQA's greater shortcut availability and relatively simpler reasoning structure. Round-trip repair occurs in 14.8–18.8% of triggered cases, showing that the verifier corrects imperfect decompositions in a non-trivial fraction of recursive calls.

Table 10: Decomposition trigger rate and round-trip repair frequency across datasets.

Dataset	Trigger (%)	Repair (%)
HotpotQA	10.7	18.8
MuSiQue	45.2	14.8
2Wiki	48.6	17.0

I Strict Evidence-grounded Evaluation

Hi-Q's gains are not an artifact of parametric memorization by the reader. Although Hi-Q encourages evidence-grounded reasoning, the default evaluation does not require every intermediate answer to cite annotated gold documents. To test whether this permissiveness inflates performance, we evaluate a stricter variant that requires document-level grounding for each answer on a filtered subset of 634 MuSiQue queries, where gold annotations are reliable enough for strict citation enforcement [17, 29]. As shown in Table 11, the strict variant achieves nearly identical performance to Hi-Q. Moreover, all baselines in Table 1 use the same GPT-4o-mini reader and likewise enforce no document-level citation or entailment constraint during answering, so any contribution from parametric knowledge should

affect all methods similarly. The differential gains of Hi-Q over these baselines therefore reflect evidence-conditioned refinement and decomposition control rather than parametric memorization.

Table 11: Evaluation of a stricter evidence-grounded variant on the MuSiQue refined subset. Bold indicates the best value in each column.

Setting	EM	F1
Hi-Q	47.5	**61.9**
Strict evidence-grounded variant	**47.6**	61.7

Closed-book control. As an additional control for parametric knowledge, we evaluate GPT-4o-mini without retrieval, providing only the question as input on the same datasets. As shown in Table 12, the parametric-only model performs substantially below Hi-Q across all datasets, with 13.1 EM / 21.7 F1 on MuSiQue, 23.1 EM / 28.8 F1 on 2Wiki, and 29.4 EM / 39.7 F1 on HotpotQA. Although the model can answer some questions from memorized knowledge, these scores are far below the retrieval-augmented results, especially on MuSiQue where genuine multi-hop reasoning is required. Together with the strict grounding variant and the shared-reader comparison, this supports that Hi-Q's gains are driven primarily by evidence acquisition and decomposition control rather than parametric memorization.

Table 12: Closed-book GPT-4o-mini performance on the 1,000-question subsets.

Dataset	EM	F1
MuSiQue	13.1	21.7
2Wiki	23.1	28.8
HotpotQA	29.4	39.7

Backbone sensitivity. The controls above hold the reader fixed at GPT-4o-mini, and that choice is deliberate. Repeating the closed-book control on MuSiQue with GPT-5 yields 25.0 EM / 37.6 F1, well above GPT-4o-mini's 13.1 / 21.7, and when GPT-5 refines sub-queries it frequently supplies bridge entities that were never retrieved. A stronger backbone therefore answers a larger share of questions from parametric knowledge, which makes answer accuracy a weaker diagnostic of retrieval–query alignment. Holding the reader at GPT-4o-mini keeps model capability fixed and attributes differences between methods to their control and decomposition logic instead. We report this as a sensitivity analysis of the backbone choice, not as a performance ceiling for Hi-Q.

J Comparison with Dedicated Decomposition Baselines

This comparison tests whether Hi-Q remains competitive with dedicated decomposition methods under their reported evaluation setting. We focus on TRQA [15] because it is the most recent state-of-the-art query-decomposition baseline among the methods discussed in Section 2. Since both Q-DREAM [16] and TRQA are fine-tuned methods, we select TRQA as the stronger fine-tuned comparison. We also include Self-Ask [12] and Least-to-Most [13] as prominent prompt-based decomposition baselines. Decomposed Prompting [14] is excluded because its number of LLM calls is computationally prohibitive in our evaluation setting. Because TRQA is closed-source, we match its reported setting—GPT-3.5-Turbo reader, ColBERTv2 retriever, the full MuSiQue development set of 2,417 questions over the complete corpus, and the Answer Recall metric—and cite its reported score for a standardized comparison; Hi-Q is therefore run on all 2,417 questions here rather than on the 1,000-question subset used elsewhere. Under this setting, Hi-Q reaches 29.3 Answer Recall, improving over TRQA's 26.8 and over the prompt-based baselines Least-to-Most and Self-Ask, which obtain 13.4 and 15.6, respectively (Table 13).

K Voting over Multiple Decompositions

Hi-Q commits to a single decomposition at each expansion, which raises the question of whether an early root-level split that is locally plausible but poorly aligned with the corpus propagates to

Table 13: Comparison against dedicated decomposition baselines on MuSiQue-full (N=2,417) under TRQA's reported setting (GPT-3.5-Turbo reader, ColBERTv2 retriever). The metric is *Answer Recall* as defined and reported by TRQA; the baseline scores are those reported by TRQA, and `Hi-Q` is measured in this work on the same questions with the same reader and retriever. Bold indicates the best value.

Method	Answer Recall
Least-to-Most [13] (ICLR 2023)	13.4
Self-Ask [12] (EMNLP 2023)	15.6
TRQA [15] (AAAI 2024)	26.8
`Hi-Q`	**29.3**

the final answer. This section tests the natural remedy: generating several root decompositions and letting them vote. On MuSiQue-full (1,000 questions, GPT-4o-mini, the same retrieval stack as the rest of the paper), we generate b alternative root-level decompositions with distinct pivots, run each candidate independently through the full pipeline, and select the final answer by majority vote with a judge tie-break. Here $b = 1$ is the configuration used throughout the paper and reproduces its MuSiQue-full result in Table 1.

Accuracy increases monotonically but modestly, while token cost grows close to linearly in b (Table 14). Wall-clock latency need not grow proportionally, since the b candidates are independent and can be executed in parallel. That the gains are small is itself informative: it indicates that early decomposition errors rarely survive to the final answer. A sub-query that cannot be answered from the corpus returns unresolved rather than committing a wrong intermediate value, so a poor split tends to be absorbed by further refinement or by the verifier instead of being propagated. Multiple root decompositions are therefore a valid extension of `Hi-Q` rather than a correction to it, and $b = 1$ remains the better cost-effectiveness point, which is why we retain it.

Table 14: Root-level decomposition candidates on MuSiQue-full. Cost is the per-question API token cost relative to $b = 1$, the configuration used throughout the paper.

b	EM	F1	Cost
1	37.4	50.6	1.0×
2	38.3	51.4	1.8×
3	39.0	52.2	2.6×

L Binary Decomposition under High Arity

Binary expansion is an execution primitive, not an assumption that every question has a single linear reasoning chain. A plan with m leaf information needs is organized as at most $m - 1$ binary reductions, with the expansion operator, the verifier, and the execution framework unchanged; how many reductions actually occur is decided by the controller, since a node resolvable from its retrieved passages terminates there. This section tests whether that primitive degrades as the number of leaf needs grows.

No question among our 1,000 MuSiQue questions has a gold reasoning node with immediate fan-in 3 or higher, so we construct 400 controlled questions from 100 five-fact MuSiQue bundles with nested variants for $m = 2, 3, 4, 5$. Each branch carries one gold passage and three mined distractors, holding gold density at 25% so that retrieval difficulty stays constant as arity grows. The set was frozen before evaluation, and two annotators verified 100 sampled questions and their answers. Forced expansion prevents the controller from terminating without decomposing, so the expansion operator itself is under test rather than the routing decision; the oracle-plan control supplies gold leaf queries while retrieval and reading still run.

Branch generation is not the bottleneck. Table 15 shows that binary `Hi-Q` stays between 95.0 and 99.0 EM as arity rises from 2 to 5, losing 0.80 EM per additional branch, and its generated decompositions are statistically indistinguishable from the binary oracle-plan control (-1.50 EM, 95% CI $[-3.50, +0.50]$). Variable-arity branching gives no significant mean-EM improvement over

binary decomposition (binary $-$ variable $= +0.25$, 95% CI $[-2.00, +2.50]$), although it does reduce latency. IRCoT degrades fastest, at 5.00 EM per additional branch.

Arity is costly in aggregation rather than in branching. A separate comparison varies only the final combination step: both arms receive the gold plan and run identical retrieval and per-branch reading, but one combines by sequential binary reduction and the other by a single flat N-ary aggregation. The flat variant loses 27.50 EM (71.0 versus 98.5 overall; 95% CI $[24.00, 31.00]$), so the measured trade-off of binary refinement is additional sequential cost, not an observed high-arity accuracy failure, and we retain dependency-preserving binary decomposition.

Ordinary comparison questions are covered by the same primitive. The class often cited as requiring parallel aggregation, retrieving an attribute for one entity, retrieving the corresponding attribute for another, and comparing them, is represented by two branches followed by synthesis, with each resolved value written to $\mathcal{H}$ before the dependent branch runs. On the 235 comparison questions in 2Wiki, `Hi-Q` reaches 85.5 EM, which together with the depth-stratified results in Section 4 indicates that ordinary two-source comparison is within the evaluated scope.

Table 15: Forced-expansion results on the 400-question controlled high-arity set, where m is the number of leaf information needs. EM slope is the mean EM change per additional branch. In the binary conditions every EXPAND emits exactly two dependency-ordered children; the variable-arity condition lets it emit up to m children.

Method	EM@2	EM@3	EM@4	EM@5	EM slope
IRCoT	100.0	99.0	94.0	85.0	-5.00
Oracle-plan, binary	100.0	99.0	98.0	**97.0**	-1.00
`Hi-Q`, variable-arity (forced)	100.0	98.0	**99.0**	90.0	-2.90
`Hi-Q`, binary (forced)	97.0	**99.0**	97.0	95.0	$\mathbf{-0.80}$

M Comparison under a Code-Capable Backbone

PyRAG [21] specializes the corpus-as-environment paradigm for multi-hop RAG, representing reasoning as an executable program over retrieval and answering tools, which makes intermediate state explicit and supports execution-grounded repair. It requires a code-capable backbone, so we evaluate it and `Hi-Q` with a shared Qwen3.6-27B reader and retrieval stack on MuSiQue-full, a setting not comparable with Table 1: `Hi-Q` reaches 47.9 EM / 59.3 F1 against PyRAG's 41.2 / 52.9 (Table 16). Giving the agentic baseline a stronger code model therefore does not change the ordering.

Table 16: Comparison with PyRAG on MuSiQue-full, under a shared Qwen3.6-27B reader and retrieval stack. These values are not comparable with Table 1, which uses GPT-4o-mini. Bold indicates the best value in each column.

Method	EM	F1
PyRAG [21]	41.2	52.9
`Hi-Q`	**47.9**	**59.3**

N Limitations

Our experiments establish performance on English, Wikipedia-derived, passage-based short-answer multi-hop QA, and the results should not be read as evidence of transfer beyond that setting. The dependency structures the evaluation covers are acyclic and resolvable branch by branch; questions whose prerequisites are mutually dependent, or whose branches must be optimized jointly rather than settled in sequence, are outside what the current state representation models. The untested dimensions have established benchmarks of their own: long-form QA (ASQA [30], ELI5 [31]), multilingual QA (XOR-TyDi QA [32], MKQA [33]), domain-specific QA (BioASQ [34], FinanceBench [35]), structured table–text QA (HybridQA [36], OTT-QA [37]), and multimodal QA (MultiModalQA [38], WebQA [39]). None is a drop-in extension: the long-form, multilingual, and domain-specific settings change the answer format, language-access assumptions, or corpus domain without controlling for the passage dependency structure studied here, and the structured and multimodal ones additionally require heterogeneous evidence representations and retrieval operators.

`Hi-Q` depends on the reliability of LLM modules for resolution, decomposition, verification, and synthesis. Reader errors can affect the refinement policy: false abstention may cause over-refinement, while unsupported answer generation may stop refinement too early. Although our calibration and robustness analyses suggest these errors are bounded in the evaluated setting, the current binary resolved/unresolved signal does not capture finer states such as partial evidence, conflicting evidence, or multiple competing bridge entities. Appendix C replaces that test with a calibrated classifier without changing what the signal can express; representing partial or conflicting evidence would instead require a richer state, such as entailment-based checks over the retrieved passages.

`Hi-Q` expands a node only when the reader cannot answer from the retrieved evidence, so on queries the corpus cannot support it abstains rather than fabricates (49 of 50 adversarially unanswerable queries, Figure 4(b)). The same dependence on evidence is the main risk: the verifier checks that a split preserves the question, not that the retrieved passages are true, so a biased or inaccurate corpus yields answers that are well supported and wrong in the same way. Deployments should therefore use curated corpora and expose the retrieved passages alongside each answer.

O Prompt Templates

This appendix provides a consolidated overview of the prompt templates used to instantiate the operators in Algorithm 1. Rather than embedding the full prompt specifications in the main text, we summarize their roles here—grouped by the operator they realize—and present the concrete templates in the corresponding figures.

Resolution operator $\mathcal{G}$. The resolution operator is realized by three prompts that jointly perform query refinement, retrieval-grounded answering, and resolution-status determination:

- **Subquery Refinement**, which rewrites a query using the accumulated interaction history $\mathcal{H}$ so that implicit references and abstract expressions are grounded in previously resolved facts before retrieval (Figure 10).
- **Granularity Assessment (Root Version)**, which evaluates whether the original query Q can be answered directly under the retrieved evidence and returns either an answer or an unresolved-support signal (Figure 8).
- **Granularity Assessment (Recursive Version)**, which applies the same evidence-conditioned resolution check to each internal sub-query node during recursion (Figure 9).

Binary expansion operator $\mathcal{B}$.

- **Binary Query Decomposition**, which proposes a dependency-ordered pair of sub-queries $(q_{\text{left}}, q_{\text{right}})$ connected by a bridge entity, or returns $\perp$ when the query is non-decomposable (Figure 6).

Semantic coverage verifier $\mathcal{V}$.

- **Round-Trip Consistency Verification**, which checks whether the proposed split $(q_{\text{left}}, q_{\text{right}})$ jointly preserves the intent of q without omission, addition, or reordering, and repairs the split when this constraint is violated (Figure 7); a null `needs` field means the split passed and the original pair is kept, while a non-null field replaces it.

Final synthesis.

- **Final Answer**, which aggregates the resolved sub-answers a_{left}, a_{right} and the accumulated evidence into a final response consistent with Q (Figure 11).

Binary Query Decomposition

You are a Multi-hop Question Decomposition Agent. Your task is to analyze a given question and determine whether it exhibits a dependency structure that can be decomposed into **EXACTLY TWO** information needs connected by a conceptual or factual **BRIDGE**. A **BRIDGE** is a pivot fact that must be resolved first in order to answer the final question.

Typical BRIDGEs include:

- Key entities (persons, organizations, locations, dates)
- Important noun phrases (titles, concepts, objects)
- Logical or temporal relationships (cause, dependency, sequence)
- Explicit constraints stated in the question

Decomposition Instructions:

1. **If the question requires an intermediate BRIDGE:** Decompose it into **EXACTLY TWO** needs:
   - (N1) The pivot need that establishes the bridge
   - (N2) The dependent need that uses the resolved bridge to reach the final answer
2. **If there is no intermediate BRIDGE** and the question therefore cannot be decomposed into two dependent needs: Do **NOT** decompose.

Rules:

- **NEVER** produce more than two needs.
- **NEVER** produce fewer than two needs when decomposing.
- **N2 MUST** logically depend on N1.

For each need, specify:

- `id`: `N1` or `N2` only
- `text`: a declarative description of the information required
- `depends_on`: `[]` for N1, `["N1"]` for N2
- `subquery`: a concise natural-language question that retrieves the required information

Output Format:

- Always output a **JSON object**.
- Use **EXACTLY TWO** top-level keys: `"thought"` and `"needs"`.
- If the question cannot be decomposed into two dependent needs, return:

```
{"thought": "...", "needs": null}
```

Question: `{question}`

Return the results in a FLAT JSON format.
DO NOT include any explanations or notes in the output. **ONLY** return JSON.

Figure 6: Binary Query Decomposition Prompt.

Round-Trip Consistency Verification

You are a Decomposition Verification & Repair Agent. Your task is to verify and, if necessary, repair a proposed **TWO-NEED** decomposition so that it faithfully represents the intent and constraints of the original multi-hop question.

You will be given:

- An original multi-hop question
- A decomposition containing two needs: N1 and N2

Your tasks:

1. **VERIFY:** Check whether N1 and N2, when recomposed, are equivalent to the original question and capture *all* required constraints.
2. **REPAIR (only if invalid):** If the decomposition is invalid, produce a corrected decomposition that satisfies the full intent and constraints of the original question.

Decomposition Rules: Same as those defined in the *Binary Query Decomposition* prompt.

For each need, specify:

- `id`: `N1` or `N2` only
- `text`: a declarative description of the information required
- `depends_on`: `[]` for N1, `["N1"]` for N2
- `subquery`: a concise natural-language question that retrieves the required information

Output Format:

- Always output a **JSON object** with **EXACTLY TWO** top-level keys: `"thought"` and `"needs"`.
- If the decomposition is **valid** and already satisfies all constraints:
  - Set `"needs"` to `null`.
- If the decomposition is **invalid**:
  - Set `"needs"` to the corrected two-need decomposition.

Question: `{question}`

Decomposition: `{decomposition}`

Return the results in a FLAT JSON format.
DO NOT include any explanations or notes in the output. **ONLY** return JSON.

Figure 7: Round-Trip Consistency Verification Prompt.

Granularity Assessment (Root version)

You are an advanced reading comprehension assistant.
Your task is to analyze text passages and corresponding questions meticulously.

Always respond as a JSON object with the following structure:

```
{ "thought":  "<methodically break down the reasoning process,
illustrating how you arrive at conclusions>",
"answer":  "<a concise and definitive answer string>" }
```

- If there is an answer, extract the answer span from the text passages.
- If there is no answer, respond with `{ "thought": "<methodically break down the reasoning process, illustrating how you arrive at conclusions>", "answer": null }`

Question: `{question}`

Documents: `{documents}`

Return the results in a FLAT JSON format.
DO NOT include any explanations or notes in the output. **ONLY** return JSON.

Figure 8: Granularity Assessment Prompt (Root Version).

Granularity Assessment (Recursive version)

You are a QA assistant.

Context:

- The ORIGINAL QUESTION is being solved through multiple intermediate steps.
- Some intermediate facts have ALREADY been resolved in previous steps.
- These resolved facts are provided in the Current Document Context.
- The current SUBQUESTION is derived from that context and represents ONLY ONE intermediate step.
- There may be further steps after this one.
- The ORIGINAL QUESTION and the Current Document Context are provided ONLY to help you interpret the SUBQUESTION.

Your task:

- Answer ONLY the given SUBQUESTION.
- Do NOT attempt to answer the ORIGINAL QUESTION.
- You may use the ORIGINAL QUESTION to understand how the result will be used.
- If the SUBQUESTION is NOT the final step toward answering the ORIGINAL QUESTION, return the answer that will be most useful for subsequent reasoning steps.
- If the SUBQUESTION IS the final step, return the answer directly.

Always respond as a JSON object with the following structure:

```
{ "thought":  "<methodically break down the reasoning process,
illustrating how you arrive at conclusions>",
"answer":  "<a concise and definitive answer string>" }
```

- If there are MULTIPLE valid answers to the SUBQUESTION, return ALL of them.
- If there is no answer, respond with `{ "thought":  "<methodically break down the reasoning process, illustrating how you arrive at conclusions>", "answer":  null }`

Original Question: `{original question}`

Current Document Context: `{retrieved documents so far}`

Subquestion: `{sub question}`

Documents: `{documents}`

Return the results in a FLAT JSON format.
DO NOT include any explanations or notes in the output. **ONLY** return JSON.

Figure 9: Granularity Assessment Prompt (Recursive Version).

Subquery Refinement

You are a subquery planner for multi-hop QA.

You are given:

- the original question Q
- the current target need
- the retrieved documents accumulated so far

Your job:

- Construct a focused, concise text query q_{next} that will help satisfy the outstanding needs.
- The query should incorporate any necessary entities from known facts in the retrieved docs.

Output format:

- Return ONLY a JSON object: `{"query": "..."}`
- No explanations.

Original Question: `{original question}`

Target need: `{subquery}`

Retrieved documents: `{documents}`

Return the results in a FLAT JSON format.
DO NOT include any explanations or notes in the output. **ONLY** return JSON.

Figure 10: Subquery Refinement Prompt.

Final Answer

You are an advanced reading comprehension assistant.

You are given:

- Question
- A history of decomposed information needs and the answers found for each need
- The retrieved documents accumulated during multi-hop retrieval

Your task: Answer the Question using the provided history and retrieved documents.

Always respond as a JSON object with the following structure:

```
{ "thought": "<methodically break down the reasoning process,
illustrating how you arrive at conclusions>",
"answer": "<final answer>" }
```

Original Question: `{original question}`

History (needs and answers): `{history}`

Retrieved documents: `{documents}`

Return the results in a FLAT JSON format.
DO NOT include any explanations or notes in the output. **ONLY** return JSON.

Figure 11: Final Answer Prompt.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims are stated in the abstract and Section 1, and are supported by the experimental results in Section 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Appendix N discusses the main limitations, including dependence on the underlying LLM and the restriction of our evaluation to English, Wikipedia-derived, passage-based short-answer multi-hop QA.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not include formal theoretical results or proofs; our contributions are empirical.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide all key hyperparameters and configurations in Section 4, the full algorithm in Appendix A, and all prompt templates in Appendix O. All datasets and base models used are publicly available.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code, prompts, and inference scripts are publicly available at `https://github.com/juevn/Hi-Q` with instructions for reproducing the main experiments. All datasets used are publicly available.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 specifies all evaluation details, including dataset splits, retriever (NV-Embed-v2), reader (GPT-4o-mini), retrieval size $k = 5$, maximum recursion depth $d_{\max} = 4$, and the full-corpus sizes used in our open-domain evaluation.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Differences between `Hi-Q` and the strongest baselines are tested with a paired question-level bootstrap over per-question predictions (10,000 resamples, 95% percentile intervals); the protocol is stated in Section 4 and the outcome is reported with the main results. Because decoding is deterministic at temperature zero, the intervals quantify uncertainty over the question population rather than run-to-run variability, so standard deviations across repeated runs are not applicable.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Per-question latency, LLM call count, and token usage are reported in Table 4. Appendix E specifies the hardware: retrieval is performed on a single NVIDIA Tesla P40 GPU (four NVIDIA RTX A6000 GPUs for HotpotQA-full), and reading uses GPT-4o-mini via the OpenAI API.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research conforms with the NeurIPS Code of Ethics. We use only publicly available benchmark datasets and do not involve human subjects or sensitive data.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: `Hi-Q` can improve the reliability of multi-hop question answering by grounding answers in retrieved evidence, which mitigates LLM hallucination. As with any retrieval-augmented system, however, the quality of generated answers depends on the underlying corpus, and biased or inaccurate sources could propagate to outputs (Appendix N).

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release any new high-risk data or pre-trained models. Our contribution is an inference-time framework that operates over publicly available datasets and models.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All datasets (MuSiQue, HotpotQA, 2WikiMultiHopQA), retriever (NV-Embed-v2), reader (GPT-4o-mini), and baseline methods are properly cited in Section 4 and the references. We use them in accordance with their respective licenses and terms of use.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We release the `Hi-Q` implementation, prompts, and inference scripts at `https://github.com/juevn/Hi-Q` under the MIT license, with documentation describing usage and reproduction steps. No new datasets or pre-trained models are introduced.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This work does not involve crowdsourcing experiments or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This work does not involve human subjects research and therefore does not require IRB approval.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs are an integral component of `Hi-Q`: GPT-4o-mini (default reader), Llama-3.3-70B and Qwen3-30B-A3B (Appendix F), and Qwen3.6-27B (Appendices D and M) instantiate the resolution, refinement, decomposition, verification, and synthesis modules. The full algorithm and prompt templates are provided in Appendix A and Appendix O.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.