BEYOND LOGICAL DECOMPOSITION: PRIORITIZING RETRIEVAL ACTIONS IN MULTI-HOP QA

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-hop retrieval-augmented generation systems typically plan retrieval by the logical dependencies in a question, but logically valid retrieval actions can differ greatly in how much they narrow the search. We study this mismatch as the SEARCH-PRIORITY GAP: when several valid actions are available, current planners do not reliably start from the most selective one. To isolate this behavior, we construct 2,541 question pairs from 2WikiMultiHopQA by adding true but less selective facts about the bridge entity from Wikidata, while keeping the original identifying relation, the anchor, and thus the answer unchanged. Across decomposition-based, iterative, and agentic retrieval systems, these added facts cause substantial accuracy drops, e.g., from 67.1% to 31.0% for PlanRAG. The drop depends on which facts are added and on where the anchor appears in the question, even though its position does not change the problem. Trajectory analysis shows that added facts pull planners away from the anchor, so the bridge is found late or not at all. A simple few-shot prompt that prioritizes the anchor recovers part of the drop, raising AceSearcher-14B accuracy from 50.8% to 66.7%. These results suggest that multi-hop retrieval planning should account not only for logical dependency, but also for the retrieval priority of candidate actions.

1 INTRODUCTION

LLM-based agentic retrieval-augmented generation (RAG) systems have achieved strong performance on complex queries, especially multi-hop questions that require composing evidence across multiple passages (Singh et al., 2025; Zhang et al., 2026b; Li et al., 2025c). These systems typically treat retrieval as a sequential decision process. They answer complex questions by generating intermediate questions or reasoning steps, retrieving evidence, and using the results to continue, either by following a plan generated upfront (Xu et al., 2025b) or by deciding each step adaptively (Press et al., 2023; Trivedi et al., 2023a; Yao et al., 2023a). In practice, these planners order retrieval steps by logical dependency, deciding what needs to be found first so that later steps can be answered. But a step that is logically valid is not necessarily an effective one to retrieve. A subquery may match many candidate entities, and the specific evidence needed can easily get lost among them. This problem is familiar in information retrieval. TF–IDF gives lower weight to terms that appear in many documents (Sparck Jones, 1972), and database query optimizers run the most selective operations first (Ali et al., 2022). LLM planners, however, usually choose among retrieval actions while ignoring how large these candidate spaces are.

Consider the HotpotQA (Yang et al., 2018a) example in Figure 1. Retrieval can begin from either constraint, and both choices are logically valid, yet they induce substantially different candidate spaces. Thus, logical validity alone does not determine which retrieval action should be executed first. This distinction motivates our investigation of the SEARCH-PRIORITY GAP: the gap between constructing logically valid plans and prioritizing effective retrieval actions. It remains unclear whether current planners reliably distinguish among valid subqueries according to their retrieval implications and choose actions that effectively resolve the current information need.

We study this question with controlled question pairs built from 2WikiMultiHopQA (Ho et al., 2020a). For each two-hop question, we add other true facts about the bridge entity from Wikidata. For example, "Where was the director of film Thomas Jefferson born?" becomes "Where was the filmmaker who received the Nichols-Chancellor's Medal and directed the film Thomas Jefferson

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

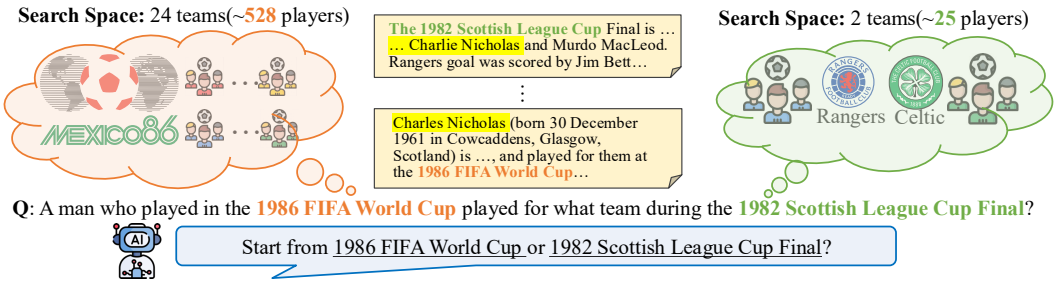

Figure 1: Two logically valid retrieval choices for the same intermediate entity induce substantially different candidate spaces.

born?" The original relation (the *anchor*) is kept, so the answer is unchanged, but the planner now has several valid starting points that differ in how many entities they match. This yields 2,541 original–rewritten pairs.

The SEARCH-PRIORITY GAP appears across all evaluated retrieval baselines. Although the rewritten questions keep the anchor, so the original reasoning path and answer are unchanged, accuracy drops substantially, e.g., from 67.1% to 31.0% for PlanRAG and from 74.0% to 50.8% for AceSearcher-14B. The anchor identifies the bridge uniquely, while every added fact matches more candidates. This shows that planners choose among valid facts without considering their retrieval priority. They are also sensitive to where the anchor appears in the question, even though its surface position does not change the problem. Bridge hit rates confirm the gap: planners find the bridge far less often in the first step, and many never find it at all.

Contributions. Our contributions are threefold: (1) we identify the SEARCH-PRIORITY GAP, where planners fail to prioritize the most selective among valid retrieval actions; (2) we construct 2,541 controlled question pairs that add valid but less selective facts while preserving the anchor and answer; and (3) we trace the resulting failures across multiple retrieval paradigms and show that a simple few-shot prompt prioritizing the anchor recovers part of the drop. These results suggest that retrieval planning should consider not only which actions are logically valid, but also which one to retrieve first.

2 CONTROLLED EVALUATION FRAMEWORK

2.1 QUESTION PAIRS CONSTRUCTION

We measure relation selectivity by the number of entities that satisfy the corresponding constraint. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a knowledge graph and $z \in \mathcal{V}$ the target entity. Hiding z in a factual triple defines a retrieval constraint: an outgoing triple (z, r, v) becomes $(?, r, v)$, while an incoming triple (u, r, z) becomes $(u, r, ?)$. For a constraint c, let $\mathcal{C}_{\mathcal{G}}(c)$ denote the entities that can fill the unknown position, and define its *directional candidate cardinality* as $\kappa(c) = |\mathcal{C}_{\mathcal{G}}(c)|$. As both the relation and known endpoint are fixed, candidate cardinality depends on retrieval direction. For example, "born in Rome" may match many people, whereas "the birthplace of Albert Einstein" has only one place. We use $\kappa(c)$ to characterize *selectivity*: smaller candidate spaces correspond to more selective constraints. This quantity measures relational compatibility in Wikidata, rather than lexical frequency or evidence availability in the text corpus.

We use this definition to construct controlled question pairs from 2WikiMultiHopQA. For each question q with annotated bridge entity z, let c_{anchor} denote the original bridge-identifying constraint. We retain questions for which the anchor uniquely identifies the bridge, $\kappa(c_{\text{anchor}}) = 1$, and collect additional one-hop Wikidata constraints $\mathcal{C}_{\text{add}}(q)$ such that

$$z \in \mathcal{C}_{\mathcal{G}}(c) \quad \text{and} \quad \kappa(c) > 1, \qquad \forall c \in \mathcal{C}_{\text{add}}(q).$$

Thus, every added constraint is valid for the same bridge but less selective than the anchor, with added constraints spanning different candidate cardinalities.

An LLM incorporates these constraints into a rewrite $\widetilde{q}$ while retaining the anchor and the requested attribute. The rewrite therefore preserves the gold bridge, answer, and original two-hop path while introducing additional valid retrieval alternatives. We vary the number and selectivity of the added constraints and the position of the anchor relation in the rewrite. For example, for "Where was the director of film *Thomas Jefferson* born?", the anchor is the film–director relation. We place this relation before, between, or after the added descriptions, corresponding to the anchor-first, anchor-middle, and anchor-last conditions; anchor-last is the default. After validation, we obtain 2,541 aligned original–rewrite pairs. Appendix B provides the full construction, cardinality ranges, and validation details.

2.2 Experimental Setup

We evaluate all baselines on the same original–rewrite pairs, keeping each baseline's retrieval configuration, decoding settings, and inference budget fixed across conditions. All retrieval systems use E5-large-v2 embeddings and a shared FAISS index over the 2WikiMultiHopQA corpus, with $k = 10$ passages per retrieval call except for Self-RAG's second round ($k = 5$). Appendix C provides full implementation details, prompts, and method-specific adaptations.

Baselines. We include Llama-3.1-8B-Instruct, Qwen2.5-7B-Instruct, Qwen2.5-14B-Instruct, and GPT-5.6 Terra without retrieval to assess parametric answering. Prompt-based baselines use Qwen2.5-14B-Instruct without retrieval-specific training and cover one-time decomposition (Vanilla RAG), interleaved reasoning and retrieval (IRCoT (Trivedi et al., 2023a)), sequential decomposition (DecomP-ODQA (Khot et al., 2022)), replanning (PlanRAG (Lee et al., 2024a)), and tree-structured reasoning (RT-RAG (Shi et al., 2026)). We further evaluate trained retrieval systems: Self-RAG (Asai et al., 2024), DQG (Han & Gardent, 2025), Search-R1 (Jin et al., 2025b) (3B and 7B, Base and Instruct), and AceSearcher (Xu et al., 2025b) (1.5B, 8B, 14B, and 32B). GPT-5.6 Terra with retrieval serves as an additional reference configuration.

Evaluation. We report Accuracy, normalized Exact Match (EM), and token-level F1. For paired comparisons, ΔAcc denotes the change in accuracy from the original to the rewritten question (Rewrite $-$ Original); negative values indicate performance degradation under the added valid constraints. To trace where this degradation arises, we additionally measure recovery of the gold bridge entity. Psg@1 and Psg@all record whether the bridge appears in passages returned by the first or any retrieval action, respectively, while Bri@1 and Bri@all record its occurrence in the first or any generated intermediate state. These metrics measure bridge occurrence only and do not establish that the retrieved evidence supports the required relation or that the bridge is used correctly downstream.

3 Investigating Retrieval Prioritization

3.1 Main Results

If planners prioritized retrieval actions by selectivity, the added constraints should have minimal effect on accuracy, since the anchor still identifies the bridge uniquely. Table 1 reveals a clear SEARCH-PRIORITY GAP. The rewritten questions preserve the gold bridge, answer, anchor, and original two-hop solution, but introduce additional valid relations through which the same bridge could be identified. Despite this, every evaluated retrieval system loses accuracy on the rewrites. The degradation is substantial across different planning paradigms: Vanilla RAG drops from 45.42% to 17.04%, PlanRAG from 67.14% to 31.01%, RT-RAG from 63.79% to 18.22%, and AceSearcher-14B from 73.95% to 50.77%. Thus, making additional valid retrieval actions available can substantially reduce performance even when the original successful retrieval path remains unchanged.

The no-retrieval results show that this degradation is not explained simply by the rewritten questions being harder to answer. Direct-answer performance remains low overall, but the added descriptions improve Qwen2.5-7B, Qwen2.5-14B, and GPT-5.6 Terra. Most notably, Qwen2.5-14B-Instruct improves from 6.81% to 10.55% without retrieval, while the same backbone in Vanilla RAG drops from 45.42% to 17.04%. The additional information can therefore be useful for answering, yet harmful once it must be translated into retrieval decisions. This contrast localizes the observed gap to the retrieval process rather than to a general loss of question understanding.

Table 1: End-task performance on paired original and rewritten questions. Positive changes are shown in green and performance drops in red. †Search-R1-Instruct is affected by context overflow and is reported for completeness.

	Accuracy (%)			EM (%)		F1 (%)	
Model	Orig.	Rewrite	Δ	Orig.	Rewrite	Orig.	Rewrite
No retrieval							
Llama-3.1-8B-Instruct	8.38	6.30	-2.08	5.19	3.54	8.15	5.78
Qwen2.5-7B-Instruct	5.35	9.21	+3.86	4.01	7.56	6.75	10.35
Qwen2.5-14B-Instruct	6.81	10.55	+3.74	4.96	8.97	7.81	12.48
GPT-5.6 Terra	10.07	17.16	+7.08	5.39	8.85	10.79	16.78
Prompted							
Vanilla RAG	45.42	17.04	-28.38	25.38	10.11	36.68	15.12
IRCoT	61.39	**39.91**	-21.48	31.92	**21.33**	49.37	**33.45**
DecomP-ODQA	62.22	33.81	-28.41	33.96	18.81	50.40	27.53
PlanRAG	**67.14**	31.01	-36.13	**36.40**	17.67	**53.67**	25.38
RT-RAG	63.79	18.22	-45.57	34.16	10.70	50.84	15.65
Trained							
Self-RAG-7B	16.17	12.99	-3.18	0.20	0.67	4.92	3.66
DQG	13.89	9.29	-4.60	0.55	0.55	2.57	1.53
Search-R1-3B-Base	31.84	17.04	-14.80	23.57	13.73	31.59	18.13
Search-R1-3B-Instruct†	30.74	15.98	-14.76	19.56	11.57	27.60	15.27
Search-R1-7B-Base	70.92	**57.89**	-13.03	**48.52**	**40.73**	**62.64**	**51.78**
Search-R1-7B-Instruct†	51.83	42.70	-9.13	38.88	33.18	50.30	42.13
AceSearcher-1.5B	68.95	43.05	-25.90	42.38	26.88	57.89	36.81
AceSearcher-8B	**74.14**	55.37	-18.77	46.48	35.46	62.09	47.54
AceSearcher-14B	73.95	50.77	-23.18	45.18	31.33	61.71	43.34
AceSearcher-32B	71.94	50.33	-21.61	44.27	31.76	60.79	43.42
Reference							
GPT-5.6 Terra	79.69	75.44	-4.25	38.61	36.91	62.43	59.62

The gap also persists under retrieval-specific training and larger model scale. Search-R1-7B-Base drops from 70.92% to 57.89%, and all four AceSearcher variants drop substantially, from 68.95% to 43.05% at 1.5B and from 71.94% to 50.33% at 32B; scaling AceSearcher does not remove the effect. GPT-5.6 Terra with retrieval is far more stable, dropping only from 79.69% to 75.44%. The drop is therefore not inevitable, but depends strongly on the retrieval policy. These results establish the SEARCH-PRIORITY GAP in aggregate; we next examine which properties of the added constraints amplify it and how they change retrieval and bridge identification.

3.2 WHAT DRIVES THE SEARCH-PRIORITY GAP?

We examine two properties introduced by the controlled construction: the number and selectivity of the added retrieval alternatives, and the surface position of the anchor relation.

3.2.1 COMPETITION AMONG RETRIEVAL ALTERNATIVES

The candidate cardinality $\kappa(c)$ defined in Section 2.1 characterizes the selectivity of an individual constraint. A rewritten question, however, contains multiple added constraints that can differ substantially in cardinality. We therefore need a question-level measure that evaluates the set of retrieval alternatives presented to the planner. Directly grouping questions by the selectivity categories used during construction is unsuitable for this analysis as the resulting groups are highly imbalanced. This motivates aggregating the contribution of each added constraint at the question level.

Retrieval Competition Score. We define the *Retrieval Competition Score* (RCS) as

$$\mathrm{RCS}(q) = \sum_{c \in \mathcal{C}_{\mathrm{add}}(q)} h(\bar{\kappa}(c)), \qquad h(k) = \left(\frac{\log(1001/k)}{\log 1001} \right)^2. \tag{1}$$

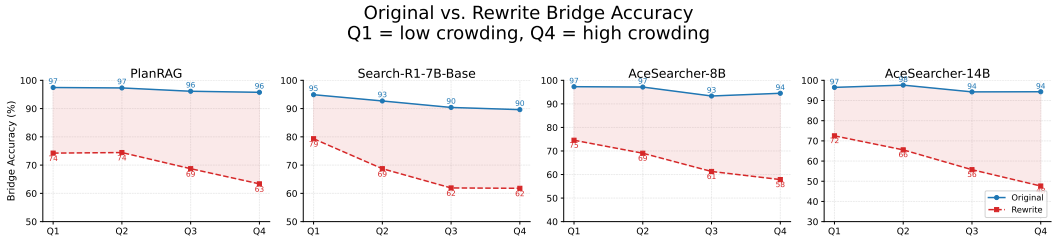

Figure 2: First-step bridge identification (Bri@1) across RCS quartiles.

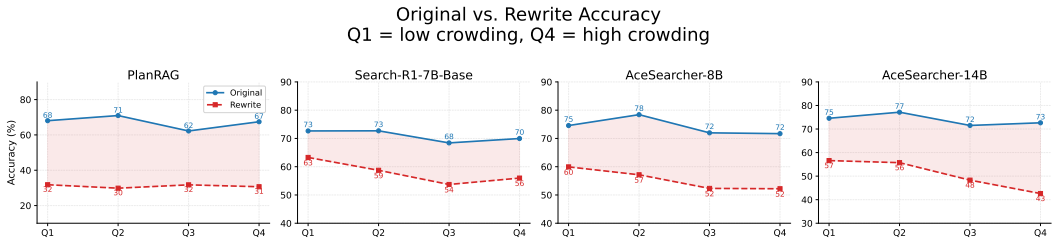

Figure 3: Answer accuracy across RCS quartiles.

Following the selectivity categories and counting convention in Appendix B, $\bar{\kappa}(c)$ retains cardinalities up to 1,000 and maps larger values to 1,001. We treat >1,000 as non-selective because such relations provide little discrimination; for example, a constraint such as "is an actor" may match thousands of entities. Since the construction separates selectivity by orders of magnitude, the logarithmic weighting assigns progressively greater weight to smaller candidate sets. Constraints mapped to 1,001 receive zero weight. Singleton added constraints do not occur because construction requires $\kappa(c) > 1$. Higher RCS therefore reflects more plausible competing alternatives and/or greater selectivity among them.

To test whether greater retrieval competition is associated with harder bridge identification, we rank the 2,541 matched pairs by RCS and divide them into four approximately equal-sized quartiles. This avoids the strong imbalance of the original selectivity buckets while preserving an ordered comparison from lower to higher competition. Each original question remains paired with its rewrite in the same quartile, allowing us to distinguish rewrite-specific effects from differences in baseline question difficulty. We report four representative systems spanning replanning, trained interleaved search, and one-time decomposition; full results and alternative summaries based on minimum and mean cardinality appear in Appendix D.

First-step bridge identification. Rewrite Bri@1 declines across RCS quartiles for all four displayed baselines (Figure 2). From Q1 to Q4, it decreases from 74.1% to 63.2% for PlanRAG, from 79.2% to 61.8% for Search-R1-7B-Base, and from 74.2% to 57.7% for AceSearcher-8B. AceSearcher-14B shows a larger decline, from 72.2% to 47.2%, while its original-question Bri@1 remains between 94.0% and 97.6%. The corresponding original results vary less across groups. Thus, the association with RCS is stronger on rewrites than on their paired originals, suggesting that baseline question difficulty alone does not explain the pattern.

Answer accuracy. The relationship with final accuracy is less consistent (Figure 3). PlanRAG remains near 30–32% rewrite accuracy despite declining Bri@1. Search-R1-7B-Base has lower accuracy in Q4 than in Q1, but its trend is not monotonic. Both displayed AceSearcher variants decline across quartiles. For AceSearcher-14B, rewrite accuracy decreases from 56.6% to 42.6%, while original accuracy changes from 74.6% to 72.6%.

For these baselines, higher RCS is more consistently associated with reduced first-step bridge identification than with lower final accuracy. RCS therefore characterizes a difficulty at an intermediate stage, while later retrieval and answering can produce different end-task outcomes. This analysis

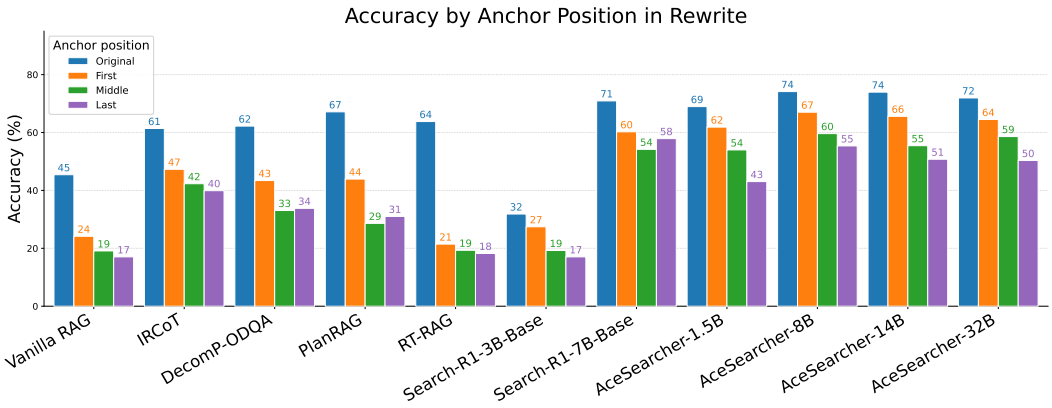

Figure 4: Answer accuracy on original questions and rewrite variants with the anchor relation placed first, middle, or last.

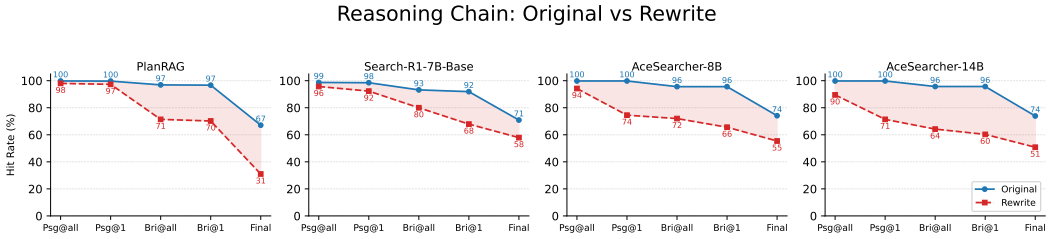

Figure 5: Passage coverage, bridge identification, and final answer accuracy on original and rewritten questions. Psg@1/Bri@1 refer to the first retrieval step, while Psg@all/Bri@all refer to any retrieval step. Final denotes end-task accuracy.

measures an association with constraint composition; it does not isolate the contributions of constraint number and selectivity.

3.2.2 Anchor Position

We next test whether performance depends on where the anchor relation appears in the rewritten question. The anchor-first, anchor-middle, and anchor-last variants preserve the same bridge-identifying relation and answer, but place the anchor before, between, or after the added descriptions. If planning were determined only by the logical dependencies of the question, this surface position should not systematically change performance. Because the variants are generated separately, they may also contain minor wording and constraint-realization differences.

Figure 4 shows that earlier anchor placement improves accuracy for most retrieval systems. IRCoT, DecomP-ODQA, PlanRAG, Search-R1-3B-Base, Search-R1-7B-Base, and all four AceSearcher variants perform best with the anchor first; for AceSearcher-14B, accuracy drops from 65.60% to 55.45% to 50.77% as the anchor moves from first to middle to last. RT-RAG remains weak across positions. Thus, even when the same valid retrieval path remains available, its surface position can substantially change final performance. This positional sensitivity indicates that current retrieval plans are influenced by presentation order, rather than being determined solely by logical dependency. Accuracy alone does not reveal how the plan changes, motivating our next analysis of anchor use, retrieval, and bridge identification.

3.3 Tracing the Failure Mechanism

The preceding analyses show when rewriting reduces first-step bridge identification. We now examine where the reasoning process fails: whether the first query uses the anchor, whether retrieval

Table 2: Bridge entity occurrence among questions answered correctly in the original form but incorrectly after rewriting. n counts eligible trajectories.

		First bridge occurrence (%)		
Baseline	n	First step	Later step	Never
PlanRAG	1,005	53.7	1.8	44.5
Search-R1-7B-Base	457	31.5	12.0	56.5
AceSearcher-8B	528	8.0	15.2	76.9
AceSearcher-14B	646	6.7	10.5	82.8

returns passages containing the bridge, and whether the model identifies the bridge before answering. We report the same four baselines as in the RCS analysis; Appendix F provides the full results.

Anchor use. If planners prioritized by selectivity, their first query would use the anchor alone, since it identifies the bridge uniquely. This rarely happens after rewriting. Anchor-only first queries account for only 7.6% of rewritten questions for PlanRAG, 1.2% for Search-R1-7B-Base, and 12.9% for AceSearcher-14B. Instead, PlanRAG and Search-R1-7B-Base mostly combine the anchor with added constraints (85.8% and 88.3%), while AceSearcher-14B often omits it altogether: its first-query anchor use falls from 93.9% on originals to 64.2% on rewrites. In contrast, GPT-5.6 Terra, the most robust system in Table 1, issues anchor-only first queries in 49.2% of rewritten questions. Planners that start from the anchor are thus the ones least affected by the added constraints. We next examine how the remaining first queries affect retrieval and bridge identification.

Retrieval and identification. Figure 5 separates bridge occurrence in retrieved passages from bridge identification in intermediate outputs. Psg@1 and Psg@all measure occurrence in first-step retrieval and in any retrieved passage, respectively. Bri@1 and Bri@all measure occurrence in the first and in any intermediate output; Final denotes answer accuracy. These rates are measured separately, rather than conditioned on success at the preceding stage.

For PlanRAG, Psg@1 remains nearly unchanged, from 99.7% to 97.3%, while Bri@1 falls from 96.7% to 70.2%. Search-R1-7B-Base shows a similar pattern: Psg@1 decreases from 98.5% to 92.3%, whereas Bri@1 falls from 91.9% to 67.9%. The decline in bridge identification therefore cannot be explained only by missing bridge mentions in retrieval. Both systems mostly issue mixed first queries, which still retrieve the passage containing the bridge but likely also retrieve other entities that satisfy the less selective added constraints, leaving the model to choose among several plausible candidates. For AceSearcher-8B and AceSearcher-14B, first-step retrieval also suffers: Psg@1 falls from 99.8% to 74.5% and 71.4%, respectively, consistent with their more frequent omission of the anchor. These results reveal two distinct difficulties: retrieving a passage containing the bridge and identifying the bridge in an intermediate answer.

Bridge recovery in failed rewrites. A bridge missed at the first step may be identified later, while early identification may still lead to a wrong answer. We therefore examine pairs answered correctly originally but incorrectly after rewriting. Table 2 groups these failed rewrites by when the bridge first appears in an intermediate output: at the first step, at a later step, or never. This distinguishes failures where the bridge remains missing from failures that occur despite its identification.

For AceSearcher-8B and AceSearcher-14B, the bridge never appears in 76.9% and 82.8% of failed rewrites, respectively. In most of their failures, later steps therefore do not recover the missing bridge. PlanRAG shows a different pattern: 53.7% of its failed rewrites already identify the bridge at the first step. Search-R1-7B-Base shows both patterns, with 56.5% never identifying the bridge and 31.5% identifying it at the first step. Failures with early bridge identification show that identifying the intermediate entity is not sufficient to answer correctly. The available measurements do not distinguish errors in later retrieval from errors in answer generation.

The key finding is that retrieval coverage, bridge identification, and correct answering are separate obstacles, and that the first two trace back to how the first query is formed. Planners that omit the anchor, such as AceSearcher, often never retrieve or identify the bridge. Planners that mix the anchor with added constraints, such as PlanRAG, usually retrieve it but identify it less reliably, and over half

Table 3: Anchor-first prompting on rewritten questions (%). Base and Ours denote the baseline and demonstration prompts; Δ is Ours $-$ Base.

	Accuracy (%)			Bri@1 (%)			Bri@all (%)		
Baseline	Base	Ours	Δ	Base	Ours	Δ	Base	Ours	Δ
Vanilla RAG	17.04	**64.38**	+47.34	8.03	**91.89**	+83.86	83.08	**92.52**	+9.44
AceSearcher-1.5B	43.05	**57.34**	+14.29	54.74	**78.71**	+23.97	55.37	**79.54**	+24.17
AceSearcher-8B	55.37	**58.24**	+2.87	65.41	**65.68**	+0.27	71.74	**74.03**	+2.29
AceSearcher-14B	50.77	**66.69**	+15.92	60.02	**88.94**	+28.92	63.91	**88.98**	+25.07
AceSearcher-32B	50.33	**67.10**	+16.77	59.94	**88.90**	+28.96	63.75	**88.94**	+25.19

of PlanRAG's failed rewrites still fail after identifying it at the first step. The SEARCH-PRIORITY GAP therefore arises mainly in how the first query is formed, with additional failures after the bridge is identified.

3.4 RETRIEVAL-AWARE MITIGATION

The trajectory analysis shows that many failed rewrites never identify the bridge or identify it only in a later step. We test anchor-first prompting using few-shot demonstrations whose first sub-question uses only the anchor relation to identify the bridge, leaving out the added constraints. Each demonstration identifies a filmmaker through a named film before asking for an attribute. This tests whether demonstrating a direct route to the bridge improves first-step identification and final accuracy.

We evaluate Vanilla RAG and the four AceSearcher variants, which share a one-time decomposition procedure. This allows the same prompt modification to be evaluated across an untuned backbone and trained planners while keeping each baseline's execution procedure fixed. We replace the zero-shot decomposition prompt with the four demonstrations, without additional training or gold bridge or anchor annotations for the evaluated questions. The question text and retrieval settings remain unchanged. Table 3 reports Bri@1, Bri@all, and accuracy; Appendix G provides the full prompt and additional results.

Bridge identification and answering. Anchor-first prompting improves accuracy for all five baselines. For Vanilla RAG, baseline Bri@all is 83.08%, but Bri@1 is only 8.03%. The demonstrations raise Bri@1 to 91.89%, close to Bri@all at 92.52%, while accuracy increases from 17.04% to 64.38%. Thus, the accuracy gain accompanies a clear shift toward identifying the bridge at the first step. AceSearcher-1.5B, 14B, and 32B improve both Bri@1 and Bri@all, showing that the intervention also reduces the share of questions where the bridge is never identified. For the 14B and 32B variants, Bri@1 rises from approximately 60% to 89%, and accuracy reaches 66.69% and 67.10%, respectively. AceSearcher-8B responds less: Bri@1 changes from 65.41% to 65.68%, while accuracy increases from 55.37% to 58.24%. The benefit therefore varies across models and is smaller when early bridge identification changes little.

Extent of recovery. The gains reduce, but do not uniformly remove, the performance loss on rewrites. AceSearcher-14B and 32B remain below their original-question baseline accuracies of 73.95% and 71.94%; these are reference scores obtained with the baseline prompt. For Vanilla RAG, we additionally evaluate anchor-first prompting on original questions. With the same prompt on both splits, accuracy reaches 62.57% on originals and 64.38% on rewrites, removing the aggregate rewrite deficit in this setting. Together, these results show that a simple few-shot planning intervention can improve bridge identification and answering without changing the questions or model parameters. Because the intervention also introduces demonstrations and specifies a decomposition structure, the results do not isolate anchor selection or bridge timing as the sole source of improvement.

4 RELATED WORK

Question decomposition. Multi-hop QA requires combining evidence across intermediate reasoning steps. HotpotQA, 2WikiMultiHopQA, and MuSiQue provide benchmarks for evaluating this

ability through supporting facts, explicit reasoning paths, or connected single-hop questions (Yang et al., 2018b; Ho et al., 2020b; Trivedi et al., 2022). Question decomposition makes these intermediate information needs explicit by expressing a complex question as simpler, dependent subquestions (Min et al., 2019b; Wolfson et al., 2020). Decomposed prompting and least-to-most prompting apply this approach to LLM reasoning (Khot et al., 2022; Zhou et al., 2023), while retrieval-oriented methods use tree structures, subquestion retrieval and reranking, or alternating decomposition and retrieval (Zhang et al., 2024; Ammann et al., 2025; He et al., 2026). These approaches establish subquestions as useful units of reasoning and execution. Our study examines how planners formulate a subquestion when several valid relations can be used to identify the same intermediate entity.

Retrieval planning. Once intermediate questions guide retrieval, systems must decide what to search for and how to proceed from the results. IRCoT and ReAct interleave reasoning with retrieval or external actions, while PlanRAG and Plan*RAG make planning explicit through replanning or structured subquery dependencies (Trivedi et al., 2023b; Yao et al., 2023b; Lee et al., 2024b; Verma et al., 2025). Adaptive-RAG selects retrieval strategies according to question complexity, and query-refinement methods improve the formulation of individual retrieval actions (Jeong et al., 2024; Chan et al., 2024). Search-R1 and AceSearcher further learn search and decomposition behavior through reinforcement learning (Jin et al., 2025a; Xu et al., 2025a). Retrieval utility is also addressed directly through bandit-based subquery selection and cost-guided construction of logical query trees (Petcu et al., 2025; Xu et al., 2026). We complement these methods with a controlled investigation: whether existing planners retain effective retrieval behavior when additional valid constraints introduce alternative starting points without removing the original reasoning path.

Retrieval difficulty and failures. Retrieval complexity measures the difficulty of obtaining sufficient evidence, while BRIGHT and RAR-b evaluate retrieval tasks that require reasoning (Gabburo et al., 2024; Su et al., 2025; Xiao et al., 2024). These studies show that reasoning depth alone does not determine retrieval difficulty. Failures also depend on how intermediate entities and retrieved evidence are used. ChainRAG addresses missing entities in decomposed subquestions through progressive retrieval and rewriting (Zhu et al., 2025), while trajectory analysis of long-horizon search agents distinguishes failures to retrieve relevant evidence from failures to use it (Liu et al., 2026). Our study examines how such failures arise when true descriptions are added to the same bridge entity. Wikidata provides a measure of each relation's selectivity, and paired trajectories reveal changes in query construction, passage coverage, bridge identification, and final answering. Appendix A provides additional discussion of these research directions and graph-based retrieval.

5 Conclusion

We identify a Search-Priority Gap in multi-hop retrieval: adding valid constraints reduces accuracy even when the original identifying relation and reasoning path remain available. Across 2,541 paired questions, performance varies with constraint composition and anchor position. Trajectory analysis reveals failures both in identifying the bridge and in answering after its identification. Few-shot demonstrations that use only the anchor in the first sub-question improve accuracy across all five tested decomposition-based systems. These findings motivate retrieval planning that considers the effectiveness of candidate actions alongside their logical validity.

Limitations. The controlled two-hop setting allows us to study retrieval choices while preserving the gold reasoning path. Extending this evaluation to longer paths, other domains, and different retrievers would clarify how broadly the observed patterns hold. KG cardinality characterizes relation selectivity; combining it with measures of textual evidence availability could better guide query selection. Likewise, evaluating relation support and downstream evidence use would extend the bridge-occurrence analysis. Our few-shot intervention provides an initial mitigation; matched demonstrations with different retrieval orders would help isolate the benefit of anchor prioritization and inform more general retrieval-aware planners.

AI Use Disclosure. Generative AI was used for writing assistance and to generate controlled question rewrites from predefined KG constraints. All generated rewrites were automatically validated as described in Appendix B. The authors take responsibility for the final content of this work.

REFERENCES

Waqas Ali, Muhammad Saleem, Bin Yao, Aidan Hogan, and Axel-Cyrille Ngonga Ngomo. A survey of rdf stores & sparql engines for querying knowledge graphs. *The VLDB Journal*, 31(3):1–26, 2022.

Paul J. L. Ammann, Jonas Golde, and Alan Akbik. Question Decomposition for Retrieval-Augmented Generation, July 2025.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 9112–9141, 2024.

Ingeol Baek, Jimin Lee, Joonho Yang, and Hwanhee Lee. Crafting the Path: Robust Query Rewriting for Information Retrieval, August 2024.

Chi-Min Chan, Chunpu Xu, Ruibin Yuan, Hongyin Luo, Wei Xue, Yike Guo, and Jie Fu. RQ-RAG: Learning to Refine Queries for Retrieval Augmented Generation, March 2024.

Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. The Power of Noise: Redefining Retrieval for RAG Systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 719–729, July 2024. doi: 10.1145/3626772.3657834.

Matteo Gabburo, Nicolaas Jedema, Siddhant Garg, Leonardo Ribeiro, and Alessandro Moschitti. Measuring Retrieval Complexity in Question Answering Systems. In *Findings of the Association for Computational Linguistics ACL 2024*, pp. 14636–14650, Bangkok, Thailand and virtual meeting, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.872.

Kai Guo, Harry Shomer, Shenglai Zeng, Haoyu Han, Yu Wang, and Jiliang Tang. Empowering GraphRAG with Knowledge Filtering and Integration, March 2025.

Kelvin Han and Claire Gardent. Generating complex question decompositions in the face of distribution shifts. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1189–1211, 2025.

Jiawei He, Zhanpeng Guan, Fuwei Zhang, Zhao Zhang, and Yongjun Xu. Bootstrapping in the Loop: Multi-hop Question Answering via Alternating Decomposition and Retrieval. In *Proceedings of the Nineteenth ACM International Conference on Web Search and Data Mining*, pp. 1145–1149, Boise ID USA, February 2026. ACM. ISBN 979-8-4007-2292-9. doi: 10.1145/3773966.3779382.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-Retriever: Retrieval-Augmented Generation for Textual Graph Understanding and Question Answering, May 2024.

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*, pp. 6609–6625, 2020a.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing A Multi-hop QA Dataset for Comprehensive Evaluation of Reasoning Steps, November 2020b.

Rolf Jagerman, Honglei Zhuang, Zhen Qin, Xuanhui Wang, and Michael Bendersky. Query Expansion by Prompting Large Language Models, May 2023.

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, March 2024.

Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, Yang Song, Chen Zhu, Hengshu Zhu, and Ji-Rong Wen. KG-Agent: An Efficient Autonomous Agent Framework for Complex Reasoning over Knowledge Graph, February 2024.

Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Zheng Li, Ruirui Li, Xianfeng Tang, Suhang Wang, Yu Meng, and Jiawei Han. Graph Chain-of-Thought: Augmenting Large Language Models by Reasoning on Graphs, October 2024.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning, August 2025a.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning. In *Second Conference on Language Modeling*, 2025b. URL https://openreview.net/forum?id=Rwhi91ideu.

Ashutosh Joshi, Sheikh Muhammad Sarwar, Samarth Varshney, Sreyashi Nag, Shrivats Agrawal, and Juhi Naik. REAPER: Reasoning based Retrieval Planning for Complex RAG Systems. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 4621–4628, Boise ID USA, October 2024. ACM. ISBN 979-8-4007-0436-9. doi: 10.1145/3627673.3680087.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*, 2022.

Myeonghwa Lee, Seonho An, and Min-Soo Kim. PlanRAG: A plan-then-retrieval augmented generation for generative large language models as decision makers. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 6537–6555. Association for Computational Linguistics, 2024a. doi: 10.18653/v1/2024.naacl-long.364. URL https://aclanthology.org/2024.naacl-long.364/.

Myeonghwa Lee, Seonho An, and Min-Soo Kim. PlanRAG: A Plan-then-Retrieval Augmented Generation for Generative Large Language Models as Decision Makers. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6537–6555, Mexico City, Mexico, 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.364.

Mufei Li, Siqi Miao, and Pan Li. Simple Is Effective: The Roles of Graphs and Large Language Models in Knowledge-Graph-Based Retrieval-Augmented Generation, February 2025a.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic Search-Enhanced Large Reasoning Models, January 2025b.

Yangning Li, Weizhi Zhang, Yuyao Yang, Wei-Chieh Huang, Yaozu Wu, Junyu Luo, Yuanchen Bei, Henry Peng Zou, Xiao Luo, Yusheng Zhao, et al. Towards agentic rag with deep reasoning: A survey of rag-reasoning systems in llms. *arXiv preprint arXiv:2507.09477*, 2, 2025c.

Haochen Liu, Song Wang, Yaochen Zhu, Yushun Dong, and Jundong Li. Knowledge Graph-Enhanced Large Language Models via Path Selection. In *Findings of the Association for Computational Linguistics ACL 2024*, pp. 6311–6321, Bangkok, Thailand and virtual meeting, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.376.

Qi Liu, Jiaxin Mao, Fengbin Zhu, and Tat-Seng Chua. Diagnosing Search Behavior and Failure Modes in Long-Horizon Search Agents, August 2026.

Haoran Luo, Haihong E, Guanting Chen, Qika Lin, Yikai Guo, Fangzhi Xu, Zemin Kuang, Meina Song, Xiaobao Wu, Yifan Zhu, and Luu Anh Tuan. Graph-R1: Towards Agentic GraphRAG Framework via End-to-end Reinforcement Learning, July 2025.

Sewon Min, Eric Wallace, Sameer Singh, Matt Gardner, Hannaneh Hajishirzi, and Luke Zettlemoyer. Compositional Questions Do Not Necessitate Multi-hop Reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4249–4257, Florence, Italy, 2019a. Association for Computational Linguistics. doi: 10.18653/v1/P19-1416.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. Multi-hop Reading Comprehension through Question Decomposition and Rescoring. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 6097–6109, Florence, Italy, 2019b. Association for Computational Linguistics. doi: 10.18653/v1/P19-1613.

Ethan Perez, Patrick Lewis, Wen-tau Yih, Kyunghyun Cho, and Douwe Kiela. Unsupervised Question Decomposition for Question Answering, October 2020.

Roxana Petcu, Kenton Murray, Daniel Khashabi, Evangelos Kanoulas, Maarten de Rijke, Dawn Lawrie, and Kevin Duh. Query Decomposition for RAG: Balancing Exploration-Exploitation, October 2025.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. Association for Computational Linguistics, 2023. URL https://arxiv.org/abs/2210.03350.

Rulin Shao, Rui Qiao, Varsha Kishore, Niklas Muennighoff, Xi Victoria Lin, Daniela Rus, Bryan Kian Hsiang Low, Sewon Min, Wen-tau Yih, Pang Wei Koh, and Luke Zettlemoyer. ReasonIR: Training Retrievers for Reasoning Tasks, April 2025.

Yuling Shi, Maolin Sun, Zijun Liu, Mo Yang, Yixiong Fang, Tianran Sun, and Xiaodong Gu. Reasoning in trees: Improving retrieval-augmented generation for multi-hop question answering. In *Companion Proceedings of the ACM Web Conference 2026*, WWW Companion '26, pp. 626–635, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400723087. doi: 10.1145/3774905.3794658. URL https://doi.org/10.1145/3774905.3794658.

Aditi Singh, Abul Ehtesham, Saket Kumar, Tala Talaei Khoei, and Athanasios V Vasilakos. Agentic retrieval-augmented generation: A survey on agentic rag. *arXiv preprint arXiv:2501.09136*, 2025.

Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han-yu Wang, Haisu Liu, Quan Shi, Zachary S. Siegel, Michael Tang, Ruoxi Sun, Jinsung Yoon, Sercan O. Arik, Danqi Chen, and Tao Yu. BRIGHT: A Realistic and Challenging Benchmark for Reasoning-Intensive Retrieval, March 2025.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multi-hop Questions via Single-hop Question Composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, May 2022. ISSN 2307-387X. doi: 10.1162/tacl_a_00475.

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)*, pp. 10014–10037, 2023a.

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)*, pp. 10014–10037, Toronto, Canada, 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.557.

Prakhar Verma, Sukruta Prakash Midigeshi, Gaurav Sinha, Arno Solin, Nagarajan Natarajan, and Amit Sharma. Plan*RAG: Efficient Test-Time Planning for Retrieval Augmented Generation, February 2025.

Liang Wang, Nan Yang, and Furu Wei. Query2doc: Query Expansion with Large Language Models, October 2023.

Liang Wang, Haonan Chen, Nan Yang, Xiaolong Huang, Zhicheng Dou, and Furu Wei. Chain-of-Retrieval Augmented Generation, October 2025.

Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. Break It Down: A Question Understanding Benchmark, January 2020.

Shirley Wu, Shiyu Zhao, Michihiro Yasunaga, Kexin Huang, Kaidi Cao, Qian Huang, Vassilis N. Ioannidis, Karthik Subbian, James Zou, and Jure Leskovec. STaRK: Benchmarking LLM Retrieval on Textual and Relational Knowledge Bases, October 2024.

Chenghao Xiao, G. Thomas Hudson, and Noura Al Moubayed. RAR-b: Reasoning as Retrieval Benchmark, May 2024.

Guangzhi Xiong, Qiao Jin, Xiao Wang, Yin Fang, Haolin Liu, Yifan Yang, Fangyuan Chen, Zhixing Song, Dengyu Wang, Minjia Zhang, Zhiyong Lu, and Aidong Zhang. RAG-Gym: Systematic Optimization of Language Agents for Retrieval-Augmented Generation, May 2025.

Ganlin Xu, Linghao Zhang, Zhitao Yin, Hongda Xi, Chen Yang, Jiaqing Liang, Weijia Lu, Sihang Jiang, Yanghua Xiao, and Deqing Yang. When RAG Meets Query Planning: Logical Query Trees for Resolving Exploratory Reasoning Problems, July 2026.

Ran Xu, Yuchen Zhuang, Zihan Dong, Jonathan Wang, Yue Yu, Joyce C. Ho, Linjun Zhang, Haoyu Wang, Wenqi Shi, and Carl Yang. AceSearcher: Bootstrapping Reasoning and Search for LLMs via Reinforced Self-Play, September 2025a.

Ran Xu, Yuchen Zhuang, Zihan Dong, Ruiyu Wang, Yue Yu, Joyce C. Ho, Linjun Zhang, Haoyu Wang, Wenqi Shi, and Carl Yang. AceSearcher: Bootstrapping Reasoning and Search for LLMs via Reinforced Self-Play. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=jSgCM0uZn3.

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*, pp. 2369–2380, 2018a.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering, September 2018b.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023a. URL https://arxiv.org/abs/2210.03629.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models, March 2023b.

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. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17): 19560–19568, March 2024. ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v38i17.29928.

Liangliang Zhang, Zhuorui Jiang, Hongliang Chi, Haoyang Chen, Mohammed Elkoumy, Fali Wang, Qiong Wu, Zhengyi Zhou, Shirui Pan, Suhang Wang, et al. Diagnosing and addressing pitfalls in kg-rag datasets: Toward more reliable benchmarking. *Advances in Neural Information Processing Systems*, 38, 2026a.

Wenlin Zhang, Xiangyang Li, Kuicai Dong, Yichao Wang, Pengyue Jia, Xiaopeng Li, Yingyi Zhang, Derong Xu, Zhaocheng Du, Huifeng Guo, et al. Process vs. outcome reward: Which is better for agentic rag reinforcement learning. *Advances in Neural Information Processing Systems*, 38: 58701–58729, 2026b.

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, April 2023.

Rongzhi Zhu, Xiangyu Liu, Zequn Sun, Yiwei Wang, and Wei Hu. Mitigating Lost-in-Retrieval Problems in Retrieval Augmented Multi-Hop Question Answering, May 2025.

Ziyuan Zhuang, Zhiyang Zhang, Sitao Cheng, Fangkai Yang, Jia Liu, Shujian Huang, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. EfficientRAG: Efficient Retriever for Multi-Hop Question Answering, September 2024.

A Related Work

A.1 Multi-Hop Evaluation and Decomposition

Multi-hop QA benchmarks provide different forms of supervision for studying evidence composition. HotpotQA annotates supporting facts across documents, 2WikiMultiHopQA provides explicit reasoning paths, and MuSiQue constructs questions from connected single-hop components to reduce shortcut solutions (Yang et al., 2018b; Ho et al., 2020b; Trivedi et al., 2022). However, compositional questions do not always require the intended multi-hop reasoning process, and benchmark construction can introduce additional evaluation artifacts (Min et al., 2019a; Zhang et al., 2026a). These findings motivate evaluating intermediate behavior in addition to final-answer accuracy. We use the annotated reasoning paths and bridge entities in 2WikiMultiHopQA to construct paired questions that preserve the same reasoning target while altering the retrieval alternatives available to the system.

Question decomposition makes intermediate information needs explicit. Early approaches decompose multi-hop questions into simpler single-hop subquestions, while BREAK represents complex questions as sequences of simpler operations (Min et al., 2019b; Perez et al., 2020; Wolfson et al., 2020). Decomposed prompting and least-to-most prompting extend this principle to LLM reasoning (Khot et al., 2022; Zhou et al., 2023). Retrieval-oriented methods further use tree-structured decomposition, subquestion retrieval and reranking, or alternating decomposition and retrieval (Zhang et al., 2024; Ammann et al., 2025; He et al., 2026). These approaches establish subquestions as units of both reasoning and retrieval. Our study instead asks how a planner chooses among multiple valid relations that can be used to identify the same intermediate entity.

A.2 Query Construction and Retrieval Planning

Retrieval performance depends not only on which information is needed, but also on how that information need is expressed as a query. Query expansion and refinement methods improve individual retrieval queries through prompting, pseudo-document generation, iterative refinement, or robust rewriting (Jagerman et al., 2023; Wang et al., 2023; Chan et al., 2024; Baek et al., 2024). At the trajectory level, IRCoT and ReAct interleave reasoning with retrieval or external actions (Trivedi et al., 2023b; Yao et al., 2023b). Other methods make retrieval planning more explicit: PlanRAG supports planning and replanning, REAPER plans retrieval across sources, and Plan*RAG represents subquery dependencies as a directed acyclic graph (Lee et al., 2024b; Joshi et al., 2024; Verma et al., 2025). Adaptive-RAG selects retrieval strategies according to question complexity, while EfficientRAG and CoRAG develop iterative retrieval procedures (Jeong et al., 2024; Zhuang et al., 2024; Wang et al., 2025).

Learning-based approaches further optimize search and reasoning behavior. Search-R1 learns search-enabled reasoning through reinforcement learning, Search-O1 integrates search into the reasoning process, and RAG-Gym studies retrieval-agent optimization (Jin et al., 2025a; Li et al., 2025b; Xiong et al., 2025). AceSearcher trains decomposer and solver behaviors through reinforced self-play (Xu et al., 2025a). More directly related to retrieval prioritization, bandit-based decomposition allocates search across alternative subqueries, while cost-guided query planning organizes atomic queries according to their execution cost (Petcu et al., 2025; Xu et al., 2026). Our study complements these approaches with a controlled evaluation: we keep the original reasoning path available, introduce additional valid retrieval alternatives, and measure how existing planners change their retrieval actions and downstream behavior.

A.3 Retrieval Difficulty and Evidence Use

Retrieval difficulty is not determined by reasoning depth alone. Retrieval-complexity measures characterize how difficult it is to obtain sufficient evidence for a question (Gabburo et al., 2024), while BRIGHT and RAR-b evaluate retrieval problems that themselves require reasoning, and ReasonIR trains retrievers for such settings (Su et al., 2025; Xiao et al., 2024; Shao et al., 2025). Retrieved context can also affect downstream answer quality even when relevant evidence is present (Cuconasu et al., 2024). In multi-hop settings, ChainRAG addresses missing entities in decomposed subquestions through progressive retrieval and rewriting (Zhu et al., 2025), while analyses of long-horizon search agents distinguish failures to retrieve relevant evidence from failures to use retrieved evidence correctly (Liu et al., 2026). Our trajectory analysis makes a related distinction by separately measuring whether retrieved passages contain the bridge entity and whether the model subsequently identifies that entity in its intermediate state. This allows us to localize failures without treating entity occurrence as sufficient evidence for the required relation.

Knowledge graphs provide a complementary setting in which relations and candidate entities are explicit. KG-Agent and Graph Chain-of-Thought combine reasoning with graph interaction, while G-Retriever retrieves graph structures for question answering (Jiang et al., 2024; Jin et al., 2024; He et al., 2024). Other work improves graph retrieval through path selection, knowledge filtering, subgraph construction, or learned retrieval policies (Liu et al., 2024; Guo et al., 2025; Li et al., 2025a; Luo et al., 2025). STaRK further evaluates retrieval over textual and relational knowledge (Wu et al., 2024). In our setting, Wikidata is used only to construct valid constraints and measure their directional candidate cardinality; the evaluated systems retrieve from a textual corpus. This separation lets us characterize the relational selectivity of a constraint independently from the behavior of the text retriever.

B Controlled Question Construction

This appendix describes how we construct additional retrieval constraints from Wikidata and incorporate them into matched question rewrites. We distinguish the KG facts available during construction, the constraints retained in the released annotations, and their natural-language realization in the rewritten questions.

B.1 Source Questions and Preserved Reasoning Path

We start from 2,756 bridge-type questions in the development split of 2WikiMultiHopQA. Each source record contains the original question, gold answer, two-hop evidence chain, and Wikidata QID of the intermediate bridge entity. We therefore use the bridge entity provided by 2WikiMultiHopQA rather than identifying it ourselves.

Each question follows a two-hop path from a named anchor entity through the bridge entity to the requested answer. For example, the question "Where was the director of film Highway Dragnet born?" has the path

$$\textit{Highway Dragnet} \xrightarrow{\text{director}} \textit{Nathan Juran} \xrightarrow{\text{place of birth}} \textit{Gura Humorului}.$$

Here, *Nathan Juran* is the bridge entity, the *director* relation from *Highway Dragnet* is the anchor, and *place of birth* is the requested relation. The rewrite adds other true descriptions of Nathan Juran while retaining the anchor and asking for the same attribute.

Let q denote an original question and $\widetilde{q}$ one of its rewrites. The construction preserves

$$z(\widetilde{q}) = z(q), \qquad y(\widetilde{q}) = y(q),$$

where z is the benchmark-provided gold bridge and y is the gold answer. The original anchor and requested relation are also retained, so the original two-hop reasoning path remains available in the rewritten question.

Rewrites are retained only when they pass the generation-time quality checks within the allowed retry budget. Some bridge entities do not provide enough usable KG constraints to produce a rewrite satisfying these requirements, and repeated generation attempts can therefore fail validation. We discard such cases, leaving 2,541 questions in the final aligned evaluation set.

Table 4: Selectivity categories used during construction.

Tag	Category	Candidate count
VS	Very selective	1–10
S	Selective	11–100
M	Medium	101–1,000
NS	Non-selective	$> 1{,}000$

B.2 KG Constraints and Candidate Cardinality

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote the Wikidata graph used for construction. A constraint leaves the bridge entity unknown in an otherwise instantiated factual triple. Its candidate set is

$$\mathcal{C}_{\mathcal{G}}(c) = \begin{cases} \{e \in \mathcal{V} : (e, r, v) \in \mathcal{E}\}, & c = (?, r, v), \\ \{e \in \mathcal{V} : (u, r, e) \in \mathcal{E}\}, & c = (u, r, ?). \end{cases} \tag{2}$$

The directional candidate cardinality is

$$\kappa(c) = |\mathcal{C}_{\mathcal{G}}(c)|.$$

Because each selected constraint is a true fact about the gold bridge, its candidate set contains that bridge.

For a bridge entity z, the set of valid one-hop constraints is

$$\mathcal{A}(z) = \{(?, r, v) : (z, r, v) \in \mathcal{E}\} \cup \{(u, r, ?) : (u, r, z) \in \mathcal{E}\}. \tag{3}$$

The selected added constraints satisfy

$$\mathcal{C}_{\text{add}}(q) \subseteq \mathcal{A}(z).$$

Consequently,

$$z \in \mathcal{C}_{\mathcal{G}}(c_{\text{anchor}}) \cap \bigcap_{c \in \mathcal{C}_{\text{add}}(q)} \mathcal{C}_{\mathcal{G}}(c). \tag{4}$$

The added constraints therefore remain compatible with the same gold bridge. They do not change the target intermediate entity; instead, they introduce additional valid relations through which a retrieval planner may attempt to identify it.

We collect one-hop incoming and outgoing facts for each bridge through the Wikidata API and SPARQL endpoint and cache the results. For each collected constraint, we count how many Wikidata entities satisfy the corresponding directional relation pattern. Candidate counts are grouped into four selectivity categories for construction, as shown in Table 4.

The released annotations represent broad counts as `>1000`. For numerical aggregation, we use

$$\bar{\kappa}(c) = \begin{cases} \kappa(c), & \kappa(c) \leq 1{,}000, \\ 1{,}001, & \kappa(c) > 1{,}000. \end{cases} \tag{5}$$

This preserves exact counts in the more selective ranges while assigning a common numerical representation to constraints whose candidate counts exceed the counting threshold. The Retrieval Competition Score defined in the main paper uses this capped representation.

B.3 Rewrite Generation and Anchor Position

The generator receives the original question, gold answer, bridge label, and an ordered list of selected KG constraints. The gold answer and bridge label are supplied as generation-time information and are not intended to appear in the rewritten question. Constraint metadata also includes the corresponding selectivity category, which is omitted from the generated text.

The generation prompt asks the model to express the supplied constraints as a single description of the bridge while preserving the original information request. The constraints are presented in a specified order, and the model is instructed not to reorder them. In the default anchor-last condition,

the added descriptions precede the clause expressing the original anchor relation. Category-specific framing is used for affiliation, domain, distinction, prominence, and generic constraint sets.

We use `gpt-4o-mini` as the primary generator, with `gpt-4o` as a fallback when validation fails. Generation uses temperature 0.3 and a maximum of 600 output tokens. When an otherwise valid rewrite is missing the anchor, an anchor-append editing step uses temperature 0.2 and a maximum of 300 output tokens. The pipeline allows up to three attempts per record.

We additionally generate anchor-first and anchor-middle variants for comparison with the default anchor-last rewrite. These conditions change the intended surface position of the original identifying relation. Because the variants are produced through separate LLM calls, their wording and realization of the added constraints can also differ. The position labels therefore describe how the anchor is presented in the question, not the order of retrieval actions subsequently chosen by a baseline.

B.4 Validation and Constraint Annotation

Each generated rewrite is checked automatically before it is retained. A rewrite is rejected if the bridge entity is named, the gold answer is explicitly disclosed, a required NS- or S-bucket constraint entity is missing, the anchor is absent or incorrectly positioned, or internal generation terms such as "bridge entity" or "hidden entity" appear in the output. For the anchor-last condition, the anchor entity is required to occur within the final 40% of the rewrite text.

The answer-disclosure check includes an exception when the gold answer also serves as a constraint entity. Failed generations are retried, and an anchor-append editing prompt can repair an otherwise valid rewrite in which the anchor is missing.

After generation, each question is joined with its original 2WikiMultiHopQA evidence chain and the KG triples collected for the bridge. The released constraint annotations retain only triples whose constraint entity appears in the final rewrite. Because multiple relations may share the same entity endpoint, the retained annotations need not correspond one-to-one to distinct surface clauses.

These checks operate primarily on textual realization and do not by themselves establish semantic equivalence between every generated variant. Constraint statistics and RCS are computed from the retained added-constraint annotations, excluding the original anchor.

After generation and quality filtering, the final release contains 2,541 aligned records for the anchor-first, anchor-middle, and anchor-last conditions.

B.5 Rationale for Constraint Expansion

The paired construction separates performance on the original benchmark question from retrieval behavior when additional valid constraints are introduced. Original-question performance provides a reference for each baseline's ability to solve the underlying task. The rewrite preserves the same gold bridge, answer, anchor, and two-hop reasoning path, while introducing additional descriptions of the bridge whose selectivity can be measured through Wikidata.

This distinction is particularly relevant to retrieval policies optimized using final-answer supervision. A correct final answer shows that a trajectory succeeded, but does not establish whether the policy reliably prioritizes effective retrieval actions when multiple valid alternatives are available. Our paired evaluation probes this behavior without changing the gold reasoning target. We do not assume that all evaluated systems were trained on 2WikiMultiHopQA, nor that preserving the reasoning target should leave model performance unchanged.

B.6 Dataset Statistics

The anchor-last dataset contains 12,374 retained added-constraint annotations across 2,541 questions, with a mean of 4.87 and a median of 5 constraints per question. Table 5 reports the number of retained constraints per question, while Table 6 summarizes their selectivity categories.

Table 5: Number of retained added-constraint annotations per question.

Constraints	Questions	Share (%)
2	133	5.2
3	380	15.0
4	578	22.7
5	723	28.5
6	421	16.6
7+	306	12.0

Table 6: Selectivity distribution of retained added constraints. The final column reports the number of questions containing at least one constraint from each category.

Tag	Constraints	Share (%)	Questions
VS	619	5.0	328
S	2,162	17.5	1,785
M	547	4.4	472
NS	9,046	73.1	2,523

B.7 Released Files and Schemas

The annotated anchor-last file retains the original record identifier, question, answer, bridge QID, and two-hop evidence chain, together with the rewritten question and retained KG constraint annotations. The anchor-position files contain the original identifier, original question, rewritten question, and gold answer. The shared identifier links each position variant to its corresponding source question.

C Baseline Implementation and Evaluation Details

This appendix describes the evaluated configurations in the same order as Table 1. We use released checkpoints and method-specific inference procedures, with the adaptations described below. Within each baseline, the retrieval configuration, decoding settings, and execution procedure are held fixed across the original and rewritten questions.

C.1 Shared Retrieval Environment

All retrieval baselines use `intfloat/e5-large-v2` embeddings and a shared FAISS index over the 2WikiMultiHopQA corpus. The index uses `faiss.IndexFlatIP`. Query and corpus representations are L2-normalized, so inner-product search is equivalent to cosine similarity. Each retrieval query is encoded with the `query:` prefix.

The corpus is stored as `corpus.tsv`, with one Wikipedia paragraph per row and separate identifier, text, and title fields. Retrieved passages contain the paragraph text; the title is not prepended to the passage provided to the model. Each retrieval action returns ten passages, except the second Self-RAG retrieval round, which returns five.

The evaluation contains 2,541 matched original–rewrite pairs from the 2WikiMultiHopQA development split. Within each baseline, the retriever, decoding configuration, execution procedure, and inference budget remain fixed across the two conditions. These choices can differ across baselines.

The shared environment requires several adaptations from the original implementations. Search-R1 uses the 2WikiMultiHopQA corpus and E5-large-v2 retriever instead of its original Wikipedia-2018 corpus and E5-base-v2 retriever. IRCoT and DecomP-ODQA use dense retrieval in place of their original BM25 configurations. The same shared index is used for retrieval queries generated by the GPT-5.6 Terra reference model.

Table 7: Fields in the annotated anchor-last release.

Field	Contents
`_id`	Original 2WikiMultiHopQA identifier.
`question`	Original question.
`rewrite_q`	Anchor-last rewritten question.
`answer`	Original gold answer.
`bridge_id`	Wikidata QID of the bridge entity.
`evidences`	Original reasoning-chain triples with entity and relation labels.
`evidences_id`	Original reasoning-chain triples with Wikidata entity and property IDs.
`added_evidences`	Retained added-constraint triples with candidate counts and selectivity tags.
`added_evidences_id`	Corresponding added-constraint triples with Wikidata entity and property IDs.

C.2 Backbones without Retrieval

Llama-3.1-8B-Instruct, Qwen2.5-7B-Instruct, and Qwen2.5-14B-Instruct answer each question directly without retrieved passages. These models correspond to selected backbones or components of the retrieval baselines. We additionally evaluate GPT-5.6 Terra with medium reasoning effort in the same no-retrieval setting.

These runs measure the specified models' parametric answering ability. They are not matched no-retrieval ablations of every retrieval-specific trained checkpoint.

C.3 Prompt-Based Retrieval

Vanilla RAG. Vanilla RAG uses Qwen2.5-14B-Instruct without retrieval-specific fine-tuning. It follows a decompose-then-execute procedure: the model first generates the complete set of dependent subquestions, then resolves references to previous answers and executes the subquestions sequentially. Each subquestion is answered using ten retrieved passages.

The final answer is generated from the original question and the subquestion–answer pairs. Our evaluated configuration uses `-add-passage 0`, so the retrieved passages are not provided again during final synthesis.

IRCoT. IRCoT uses Qwen2.5-14B-Instruct with zero-shot prompting and a maximum of eight reasoning–retrieval steps. At each step, the model observes the accumulated passages and previous reasoning, then either issues another retrieval query or produces a final answer. The prompt requires at least two search actions, and each search returns ten passages.

This differs from the original few-shot configuration and from implementations that begin by retrieving directly with the original question.

DecomP-ODQA. DecomP-ODQA uses Qwen2.5-14B-Instruct to generate dependent single-hop subquestions. Previous answers are substituted into later subquestions as needed, and each instantiated subquestion retrieves ten passages. Final synthesis uses the resulting subquestion–answer history.

PlanRAG. PlanRAG uses the text-QA adaptation provided in the AceSearcher baseline suite with Qwen2.5-14B-Instruct. The planner represents dependencies among subqueries as a reasoning DAG, while natural-language passage search replaces the database queries used in the original setting. The resulting subqueries are executed using the shared retrieval environment.

RT-RAG. RT-RAG uses Qwen2.5-14B-Instruct and constructs a reasoning tree before retrieval. The tree records dependencies among simple retrieval targets and is resolved from its leaves upward. When a child cannot be resolved, its parent can become a retrieval target. Each search returns ten passages.

Our configuration generates one reasoning tree per question at temperature zero rather than using the multi-sample tree and answer generation of the original configuration.

C.4 Trained Retrieval

Self-RAG. Self-RAG uses the released Llama-2-7B checkpoint in always-retrieve mode with two fixed retrieval rounds. The first round retrieves ten passages using the input question. The second retrieves five passages using a query conditioned on the first-round output. Candidate continuations are scored using the model's learned reflection tokens.

This configuration fixes the two retrieval rounds rather than using the original threshold-based decision over whether to retrieve.

DQG. DQG uses the released LoRA decomposer on Qwen2.5-7B-Instruct, trained on BreakHigh with the Qwen-specific `qwen_prmptv2` template. The identifier is reproduced exactly as it appears in the released implementation. The decomposer generates the complete sequence of subquestions with references to earlier answers.

Our executor resolves these references sequentially, retrieves ten passages for each subquestion, and uses a separate untuned Qwen2.5-14B-Instruct reader. The reader generates the final answer from the subquestion–answer pairs. This retrieve-and-read execution is our adaptation of the released decomposition model.

Search-R1. Search-R1 uses the released PPO checkpoints for Qwen2.5-3B and Qwen2.5-7B, including Base and Instruct variants. The model interleaves reasoning with search actions emitted inside `<search>` tags. Retrieved passages are inserted into the trajectory before generation continues. We retain the four-action budget and retrieve ten passages per search from the shared retrieval environment.

For the Instruct configurations marked with †, some examples produce excessively long reasoning trajectories and terminate with an empty final-answer field. These outputs are retained as produced by the evaluated configuration and score as incorrect on the final-answer metrics.

AceSearcher. AceSearcher uses the released Qwen2.5-Instruct-based 1.5B, 14B, and 32B checkpoints and the Llama-3.1-8B-Instruct-based 8B checkpoint. The same checkpoint serves as decomposer and solver under different prompts. It generates the complete decomposition before retrieval and then answers the subquestions sequentially. Earlier answers resolve references in later subquestions, and each step retrieves ten passages.

The decomposition is not revised during execution. The evaluated configuration uses temperature-zero decoding with `top_p`=0.99 and repetition penalty 1.05 rather than describing the procedure simply as greedy decoding. The final answer is generated from the original question and the subquestion–answer pairs without supplying the retrieved passages again.

C.5 Reference Model Evaluation Cost

GPT-5.6 Terra uses medium reasoning effort and interacts with the same retrieval index through our evaluation harness. The model may issue retrieval queries adaptively, subject to a hard cap of three searches. When the limit is reached, subsequent search attempts receive an instruction to answer using the evidence already available. The no-retrieval configuration uses the same reasoning-effort setting without exposing the search interface.

Table 8 reports the recorded API usage for the retrieval evaluation. Costs use the rates recorded in our runner: $2.00 per million uncached input tokens, $0.20 per million cached input tokens, and $12.00 per million output tokens. These totals cover the 2,541 original questions and 2,541 matched rewrites and exclude question generation and the no-retrieval evaluation.

Table 8: Recorded GPT-5.6 Terra API usage for the retrieval evaluation.

Condition	Input (M)	Cached (%)	Output (M)	Reasoning (%)	Cost ($)
Original	9.74	5.5	0.234	31	21.31
Rewrite	11.55	4.9	0.854	75	32.32
Combined	–	–	–	–	53.63

Table 9: RCS groups used for both answer accuracy and first-step bridge identification.

Group	RCS range	Pairs
Q1	< 1.10	629
Q2	1.10–1.50	630
Q3	1.50–1.88	646
Q4	> 1.88	636

C.6 Measurement and Trajectory Eligibility

Final-answer metrics. We report Accuracy, exact match (EM), and token F1 using the AceSearcher evaluation implementation. Before scoring, predictions and gold answers are lowercased, punctuation is removed, the articles *a*, *an*, and *the* are removed, and whitespace is normalized.

Accuracy is a normalized answer-containment measure: a prediction is counted as correct when the normalized gold answer occurs as a substring of the normalized prediction. EM requires the normalized prediction and gold answer to match exactly. F1 is the token-overlap F1 score between the normalized prediction and gold answer. These metrics use the single gold answer supplied with each example; the alias-expanded evaluator is not used for the results reported in Table 1.

Trajectory metrics. Psg@1 records whether the gold bridge occurs in the passages returned by the first retrieval action. Psg@all applies the same criterion across all recorded retrieval actions. Bri@1 records bridge occurrence in the first generated intermediate answer or corresponding observable state, while Bri@all considers all recorded intermediate states. For AceSearcher and Vanilla RAG, these states are the generated subquestion answers and are evaluated through substring matching.

Unavailable observations are excluded from the corresponding trajectory metric, so eligible counts can differ across baselines and analyses. A passage hit records occurrence of the bridge entity but does not establish that the passage provides sufficient evidence for the required relation. Similarly, bridge occurrence in an intermediate output does not establish that the model uses that entity correctly in subsequent retrieval or final answering. The reported Psg and Bri metrics are marginal rates rather than sequential conditional probabilities.

For the failed-rewrite analysis, we select pairs answered correctly on the original question and incorrectly after rewriting, subject to the availability of trajectory observations. Each failed rewrite is assigned according to when the bridge first appears in an intermediate output: at the first step, only at a later step, or never. The three percentages use the same eligible denominator within each baseline.

Self-RAG is excluded from the sequential trajectory analysis because its logged branch outputs do not form a single sequential retrieval trajectory.

The implementation adaptations described above are held fixed within every original–rewrite comparison. They can nevertheless affect absolute performance and sensitivity to rewriting. The results therefore characterize the evaluated configurations, not every possible implementation of each method.

D Additional Selectivity Analysis

RCS grouping. We compute the Retrieval Competition Score (RCS) from the retained added constraints only, excluding the original anchor relation. The 2,541 matched pairs are ranked using the

Table 10: Final Accuracy (%) across RCS quartiles. Each entry reports Original → Rewrite, with Δ denoting Rewrite − Original Accuracy.

	Q1 ($n = 629$)		Q2 ($n = 630$)		Q3 ($n = 646$)		Q4 ($n = 636$)	
Model	Orig.→Rewrite	Δ	Orig.→Rewrite	Δ	Orig.→Rewrite	Δ	Orig.→Rewrite	Δ
Prompted								
Vanilla-RAG	46.7→18.1	-28.6	45.2→16.3	-28.9	45.0→17.2	-27.9	44.7→16.5	-28.1
IRCoT	64.4→42.0	-22.4	61.0→38.7	-22.2	59.9→41.8	-18.1	60.4→37.1	-23.3
DecomP-ODQA	62.6→33.7	-28.9	64.8→32.9	-31.9	59.8→34.1	-25.7	61.8→34.6	-27.2
PlanRAG	68.0→31.8	-36.2	71.0→29.8	-41.1	62.2→31.7	-30.5	67.5→30.7	-36.8
RT-RAG	64.5→14.0	-50.6	67.6→16.8	-50.8	60.4→21.2	-39.2	62.7→20.8	-42.0
Trained								
Self-RAG-7B	16.1→11.3	-4.8	17.9→13.5	-4.4	17.6→14.4	-3.3	13.1→12.7	-0.3
DQG	9.9→6.8	-3.0	11.4→8.3	-3.2	18.1→12.2	-5.9	16.0→9.7	-6.3
Search-R1-3B-Base	33.1→17.6	-15.4	30.8→17.8	-13.0	32.5→17.2	-15.3	31.0→15.6	-15.4
Search-R1-3B-Instruct	31.2→17.3	-13.8	34.1→18.6	-15.6	28.5→14.6	-13.9	29.2→13.5	-15.7
Search-R1-7B-Base	72.7→63.3	-9.4	72.7→58.7	-14.0	68.4→53.7	-14.7	70.0→56.0	-14.0
Search-R1-7B-Instruct	35.0→27.0	-7.9	40.0→25.9	-14.1	36.8→27.7	-9.1	38.1→23.3	-14.8
AceSearcher-1.5B	71.9→49.1	-22.7	71.7→44.8	-27.0	66.1→41.2	-24.9	66.2→37.3	-28.9
AceSearcher-8B	74.6→59.9	-14.6	78.4→57.1	-21.3	72.0→52.3	-19.7	71.7→52.2	-19.5
AceSearcher-14B	74.6→56.6	-18.0	77.1→55.7	-21.4	71.5→48.3	-23.2	72.6→42.6	-30.0
AceSearcher-32B	72.5→56.8	-15.7	75.2→54.8	-20.5	69.0→46.4	-22.6	71.1→43.6	-27.5
Reference								
GPT-5.6 Terra	79.5→75.8	-3.7	80.6→76.3	-4.3	77.4→72.0	-5.4	81.3→77.7	-3.6

Q1: RCS < 1.10; Q2: 1.10–1.50; Q3: 1.50–1.88; Q4: RCS > 1.88.

Table 11: First-step bridge identification (Bri@1, %) across RCS quartiles. Each entry reports Original → Rewrite, with Δ denoting Rewrite − Original.

	Q1 ($n = 629$)		Q2 ($n = 630$)		Q3 ($n = 646$)		Q4 ($n = 636$)	
Model	Orig.→Rewrite	Δ	Orig.→Rewrite	Δ	Orig.→Rewrite	Δ	Orig.→Rewrite	Δ
Prompted								
Vanilla-RAG	77.7→11.8	-66.0	76.5→9.0	-67.5	71.8→5.6	-66.3	75.3→5.8	-69.5
IRCoT	78.1→41.5	-36.6	76.0→32.4	-43.7	76.0→34.7	-41.3	73.7→31.8	-42.0
DecomP-ODQA	96.2→83.1	-13.0	96.0→78.3	-17.8	95.5→74.8	-20.7	95.4→69.0	-26.4
PlanRAG	97.5→74.1	-23.4	97.3→74.0	-23.3	96.3→68.1	-28.2	95.8→63.2	-32.5
RT-RAG	97.1→62.8	-34.3	96.7→59.4	-37.3	96.1→54.2	-42.0	95.6→47.6	-48.0
Trained								
Self-RAG-7B	19.4→10.5	-8.9	21.0→10.5	-10.5	19.2→14.7	-4.5	19.7→18.4	-1.3
DQG	72.8→52.9	-19.9	74.0→49.0	-24.9	79.4→37.3	-42.1	77.0→23.4	-53.6
Search-R1-3B-Base	63.0→48.5	-14.5	65.9→46.5	-19.4	64.4→40.2	-24.1	61.2→46.2	-14.9
Search-R1-3B-Instruct	53.9→30.7	-23.2	54.8→27.0	-27.8	50.9→25.5	-25.4	53.5→25.5	-28.0
Search-R1-7B-Base	94.9→79.2	-15.7	92.7→68.7	-24.0	90.6→61.9	-28.6	89.6→61.8	-27.8
Search-R1-7B-Instruct	51.5→37.8	-13.7	49.4→34.9	-14.4	49.2→34.4	-14.9	51.3→31.0	-20.3
AceSearcher-1.5B	92.8→61.2	-31.6	94.1→58.6	-35.6	90.2→50.2	-40.1	90.3→48.1	-42.1
AceSearcher-8B	97.1→74.2	-22.9	97.1→68.9	-28.3	93.2→60.4	-32.8	94.2→57.7	-36.5
AceSearcher-14B	96.2→72.2	-24.0	97.6→65.4	-32.2	94.0→54.6	-39.3	94.0→47.2	-46.9
AceSearcher-32B	97.0→74.4	-22.6	97.6→66.3	-31.3	93.3→52.2	-41.2	93.9→46.5	-47.3
Reference								
GPT-5.6 Terra	98.1→92.4	-5.7	97.5→92.9	-4.6	96.3→88.9	-7.4	97.0→90.4	-6.6

Q1: RCS < 1.10; Q2: 1.10–1.50; Q3: 1.50–1.88; Q4: RCS > 1.88.

rewritten questions' RCS values and divided into four approximately equal-sized groups. Each original question is assigned to the same group as its rewrite, and group membership is fixed across all baselines. This preserves the original–rewrite pairing within each level of retrieval competition.

Interpretation of RCS. For consistency with the main paper, RCS is

$$\mathrm{RCS}(q) = \sum_{c \in \mathcal{C}_{\mathrm{add}}(q)} h(\bar{\kappa}(c)), \qquad h(k) = \left(\frac{\log(1001/k)}{\log 1001} \right)^2,$$

Table 12: Spearman correlation between RCS summaries and AceSearcher-14B rewrite Accuracy.

Metric	n	ρ	p
$\mathrm{RCS}_{\mathrm{sum}}$	2,541	-0.1150	6.15×10^{-9}
$\mathrm{RCS}_{\max}$	2,541	-0.0820	3.47×10^{-5}
Mean κ_q $[10, 1000)$	2,104	+0.0444	4.18×10^{-2}

with the cardinality convention defined in Appendix B.2. Constraints with candidate counts above 1,000 are represented as 1,001 and therefore contribute zero. Singleton constraints also receive zero weight because they already identify the bridge uniquely.

RCS is a question-level summary of the retrieval alternatives introduced by the rewrite. It is not an estimate of the candidate set produced by any executed retrieval query. Each non-singleton constraint contributes according to its candidate cardinality, and the contributions accumulate when multiple such alternatives occur in the same question. Higher RCS therefore indicates a larger aggregate contribution from plausible competing constraints, rather than simply a larger minimum or mean candidate count.

For comparison, we also consider two simpler cardinality summaries. For the added constraints of question q,

$$\kappa_{\min}(q) = \min_{c \in \mathcal{C}_{\mathrm{add}}(q)} \bar{\kappa}(c), \qquad \kappa_{\mathrm{mean}}(q) = \frac{1}{|\mathcal{C}_{\mathrm{add}}(q)|} \sum_{c \in \mathcal{C}_{\mathrm{add}}(q)} \bar{\kappa}(c).$$

For example, constraints with candidate counts 10, 40, and 10 have $\kappa_{\min} = 10$ and $\kappa_{\mathrm{mean}} = 20$. These summaries characterize the smallest and average candidate sets available in a question, but they do not capture how multiple constraints jointly contribute to the set of retrieval alternatives.

Results across additional baselines. The full-model results show both extensions of and exceptions to the patterns reported in the main text. AceSearcher-32B behaves similarly to AceSearcher-14B: rewrite Bri@1 decreases from 74.4% in Q1 to 46.5% in Q4, while rewrite accuracy decreases from 56.8% to 43.6%.

For DecomP-ODQA and RT-RAG, rewrite Bri@1 decreases from 83.1% to 69.0% and from 62.8% to 47.6%, respectively. Their final-answer accuracy does not exhibit the same ordering: DecomP-ODQA remains near 33–35%, whereas RT-RAG increases from 14.0% in Q1 to 20.8% in Q4. Thus, a stronger association between RCS and first-step bridge identification does not necessarily induce a corresponding monotonic relationship with final-answer accuracy.

GPT-5.6 Terra is comparatively stable, with rewrite Bri@1 between 88.9% and 92.9% across the four groups. More broadly, the strength of the association between RCS and bridge identification varies across systems. These full-model results therefore support using RCS as a descriptive summary of the retrieval alternatives introduced by the rewrite, rather than as a universal predictor of end-task performance.

Correlation analysis. We additionally examine question-level correlations with AceSearcher-14B rewrite accuracy. In Table 12, $\mathrm{RCS}_{\mathrm{sum}}$ is the RCS used in the main analysis, i.e., the sum of $h(\bar{\kappa}(c))$ over all retained added constraints:

$$\mathrm{RCS}_{\mathrm{sum}}(q) = \sum_{c \in \mathcal{C}_{\mathrm{add}}(q)} h(\bar{\kappa}(c)).$$

$\mathrm{RCS}_{\max}$ instead isolates the largest individual constraint contribution:

$$\mathrm{RCS}_{\max}(q) = \max_{c \in \mathcal{C}_{\mathrm{add}}(q)} h(\bar{\kappa}(c)).$$

The final comparison uses $\kappa_{\mathrm{mean}}(q)$, the arithmetic mean of the capped candidate cardinalities across the added constraints. The reported $[10, 1000)$ analysis is restricted to the 2,104 questions whose mean cardinality falls in that range.

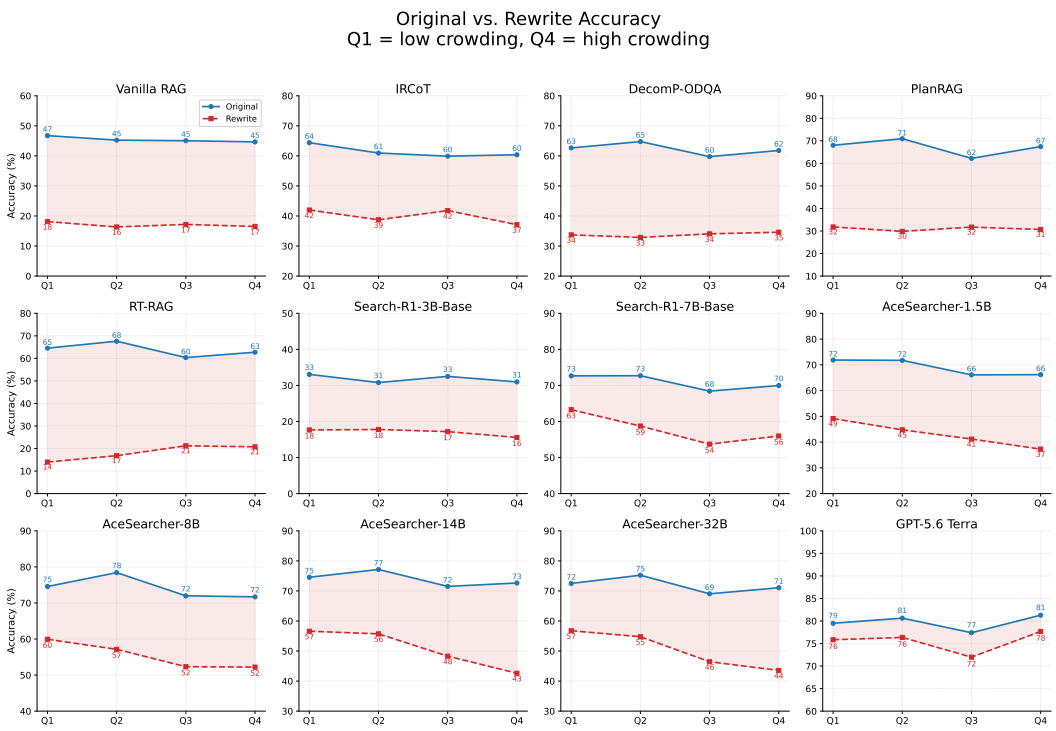

Figure 6: Answer accuracy across RCS quartiles. Original and rewritten questions are evaluated on the same examples within each quartile.

Both RCS summaries have statistically significant but weak negative Spearman correlations with rewrite accuracy ($\rho = -0.115$ for RCS_{sum} and $\rho = -0.082$ for RCS_{max}). The restricted mean-cardinality analysis has only a small positive correlation ($\rho = 0.044$). These weak question-level correlations are consistent with the quartile results: RCS more clearly organizes variation in first-step bridge identification than final-answer accuracy, which also depends on later retrieval and answer generation.

E Alternative Selectivity Measures

The main analysis uses the Retrieval Competition Score (RCS) to summarize the set of retrieval alternatives introduced by the rewrite. Here, we compare RCS with two simpler summaries based directly on candidate cardinality:

$$k_{\min}(q) = \min_{c \in \mathcal{C}_{\text{add}}(q)} \bar{\kappa}(c), \qquad k_{\text{mean}}(q) = \frac{1}{|\mathcal{C}_{\text{add}}(q)|} \sum_{c \in \mathcal{C}_{\text{add}}(q)} \bar{\kappa}(c),$$

where $\mathcal{C}_{\text{add}}(q)$ contains the retained added constraints and $\bar{\kappa}(c)$ follows the cardinality convention in Appendix B.2. Candidate counts above 1,000 are therefore represented as 1,001.

The two summaries retain different information about the constraint set. $k_{\min}$ records the smallest candidate set available among the added constraints, whereas k_{mean} describes their average candidate cardinality. Neither accounts directly for how many constraints with different cardinalities are simultaneously available to the planner. We examine both final-answer accuracy and first-step bridge identification under these groupings.

E.1 Minimum Constraint Cardinality

We first group questions by $k_{\min}$ using the same cardinality ranges as the construction: VS ($k_{\min} \leq 10$), S (11–100), M (101–1,000), and NS ($> 1{,}000$). The four groups contain 328, 1,457, 370, and

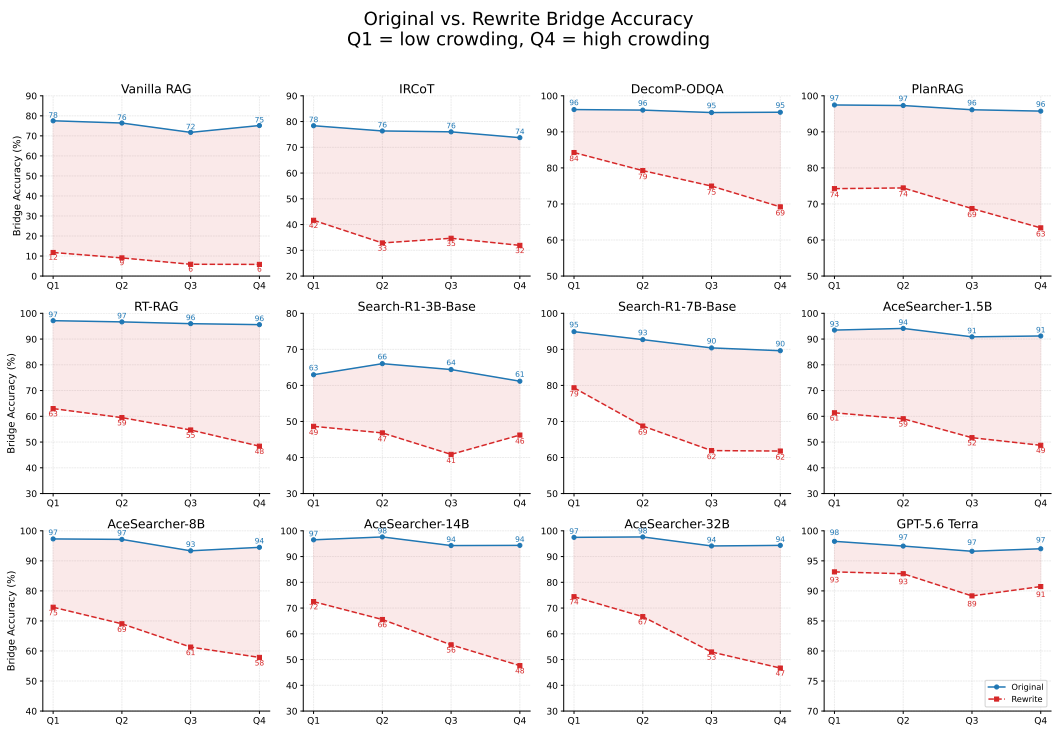

Figure 7: First-step bridge identification (Bri@1) across RCS quartiles. Original and rewritten questions are evaluated on the same examples within each quartile.

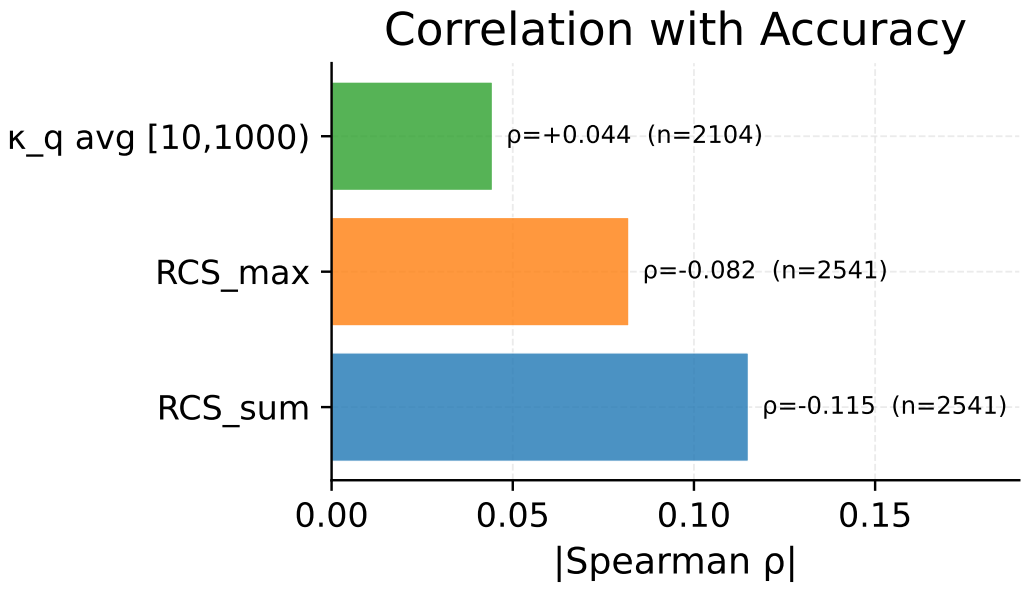

Figure 8: Spearman correlation between RCS summaries and rewrite accuracy for AceSearcher-14B. Negative values indicate that higher RCS is associated with lower end-task accuracy.

386 questions, respectively. Because this grouping is determined by the minimum cardinality, the VS group contains questions with at least one very selective added constraint, while the NS group contains questions for which every retained added constraint is non-selective.

Table 13: Bridge Hit Rate (%) across $k_{\min}$ groups. Each entry reports Original → Rewrite, with Δ denoting Rewrite − Original.

	VS ($\kappa \leq 10$)		S (11–100)		M (101–1000)		NS (> 1000)	
Model	Orig.→Rewrite	Δ	Orig.→Rewrite	Δ	Orig.→Rewrite	Δ	Orig.→Rewrite	Δ
Prompted								
Vanilla-RAG	74.4→7.9	-66.5	74.6→6.9	-67.6	78.2→8.6	-69.6	75.3→12.2	-63.1
IRCoT	71.0→37.5	-33.5	75.8→33.4	-42.3	78.1→35.1	-43.0	79.8→40.4	-39.4
DecomP-ODQA	94.8→80.8	-14.1	95.5→71.9	-23.6	97.0→85.9	-11.1	96.4→84.0	-12.3
PlanRAG	96.3→76.2	-20.1	96.0→65.8	-30.2	98.4→77.0	-21.4	97.7→74.9	-22.8
RT-RAG	96.6→55.5	-41.1	95.9→51.9	-43.9	97.3→64.1	-33.2	96.9→66.3	-30.6
Trained								
Self-RAG-7B	19.8→21.0	+1.2	20.7→13.9	-6.9	17.8→10.3	-7.6	19.9→9.8	-10.1
DQG	79.3→37.1	-42.1	76.7→37.8	-39.0	73.8→58.2	-15.6	71.5→59.9	-11.6
Search-R1-3B-Base	59.8→58.5	-1.2	64.0→41.3	-22.7	66.8→48.9	-17.8	62.4→47.7	-14.8
Search-R1-3B-Instruct	51.5→31.1	-20.4	52.8→25.0	-27.9	54.1→27.8	-26.2	55.7→31.9	-23.8
Search-R1-7B-Base	88.1→63.4	-24.7	91.4→65.1	-26.3	95.1→70.8	-24.3	94.0→79.5	-14.5
Search-R1-7B-Instruct	40.5→36.6	-4.0	50.6→32.9	-17.7	54.9→36.5	-18.4	53.6→38.3	-15.3
AceSearcher-1.5B	89.3→59.8	-29.6	92.2→51.5	-40.6	95.9→57.6	-38.4	92.5→62.7	-29.8
AceSearcher-8B	93.3→61.3	-32.0	95.1→61.4	-33.8	97.8→73.0	-24.9	96.9→78.5	-18.4
AceSearcher-14B	93.3→55.2	-38.1	95.5→54.1	-41.4	97.8→72.7	-25.1	96.4→76.2	-20.2
AceSearcher-32B	93.6→58.8	-34.8	95.7→53.6	-42.1	98.1→71.9	-26.2	96.4→74.4	-22.0
Reference								
GPT-5.6 Terra	97.3→92.1	-5.2	97.0→90.0	-7.1	98.1→95.1	-3.0	97.7→93.0	-4.7

$k_{\min}$ is the minimum candidate count across constraints. Group sizes are VS: $n = 328$, S: $n = 1457$, M: $n = 370$, and NS: $n = 386$. Bridge-hit coverage can differ slightly across systems because some runs do not expose a valid first observable retrieval state for every example.

Table 14: Bridge Hit Rate (%) across k_{mean} groups. Each entry reports Original → Rewrite, with Δ denoting Rewrite − Original.

	Low ($\bar{\kappa} \leq 600$)		Mid (601–800)		High (801–1000)		All-NS (> 1000)	
Model	Orig.→Rewrite	Δ	Orig.→Rewrite	Δ	Orig.→Rewrite	Δ	Orig.→Rewrite	Δ
Prompted								
Vanilla-RAG	72.8→8.7	-64.1	78.8→8.9	-70.0	73.0→5.6	-67.3	75.3→12.2	-63.1
IRCoT	70.9→32.8	-38.1	76.7→35.3	-41.4	76.1→34.1	-42.0	79.8→40.4	-39.4
DecomP-ODQA	94.4→77.3	-17.1	96.5→76.6	-19.9	95.4→74.1	-21.3	96.4→84.0	-12.3
PlanRAG	95.2→73.1	-22.1	97.0→70.3	-26.7	96.5→67.1	-29.4	97.7→74.9	-22.8
RT-RAG	95.8→50.4	-45.4	96.5→54.8	-41.7	96.2→56.0	-40.2	96.9→66.3	-30.6
Trained								
Self-RAG-7B	20.7→18.5	-2.2	20.4→13.2	-7.2	19.6→13.8	-5.9	19.9→9.8	-10.1
DQG	78.1→31.7	-46.4	76.0→44.0	-32.0	76.6→42.4	-34.2	71.5→59.9	-11.6
Search-R1-3B-Base	61.3→50.7	-10.6	64.1→48.4	-15.7	64.6→40.6	-24.0	62.4→47.7	-14.8
Search-R1-3B-Instruct	51.8→29.4	-22.4	52.8→25.6	-27.2	53.3→26.0	-27.3	55.7→31.9	-23.8
Search-R1-7B-Base	89.6→60.2	-29.4	92.0→65.7	-26.3	91.8→68.0	-23.8	94.0→79.5	-14.5
Search-R1-7B-Instruct	44.8→31.7	-13.2	49.5→35.2	-14.3	51.8→34.0	-17.9	53.6→38.3	-15.3
AceSearcher-1.5B	89.1→49.3	-39.8	93.2→56.4	-36.8	92.9→53.3	-39.6	92.5→62.7	-29.8
AceSearcher-8B	93.8→50.4	-43.4	96.2→62.5	-33.7	95.1→68.8	-26.3	96.9→78.5	-18.4
AceSearcher-14B	93.3→45.4	-47.9	96.4→58.1	-38.2	95.7→61.3	-34.4	96.4→76.2	-20.2
AceSearcher-32B	93.8→49.9	-44.0	96.2→62.7	-33.5	96.1→56.0	-40.1	96.4→74.4	-22.0
Reference								
GPT-5.6 Terra	96.6→91.3	-5.3	98.2→93.6	-4.6	96.7→89.1	-7.6	97.7→93.0	-4.7

k_{mean} is the mean candidate count across constraints; counts above 1000 are represented as 1001 before averaging. Group sizes are Low: $n = 357$, Mid: $n = 824$, High: $n = 974$, and All-NS: $n = 386$. Bridge-hit coverage can differ slightly across systems because some runs do not expose a valid first observable retrieval state for every example.

Figure 9 shows that $k_{\min}$ does not induce a consistent monotonic ordering of rewrite difficulty. In particular, the presence of a highly selective constraint does not necessarily make the rewritten question easier. For several planning-based systems, the largest performance drop occurs in the S or M group rather than at either cardinality extreme.

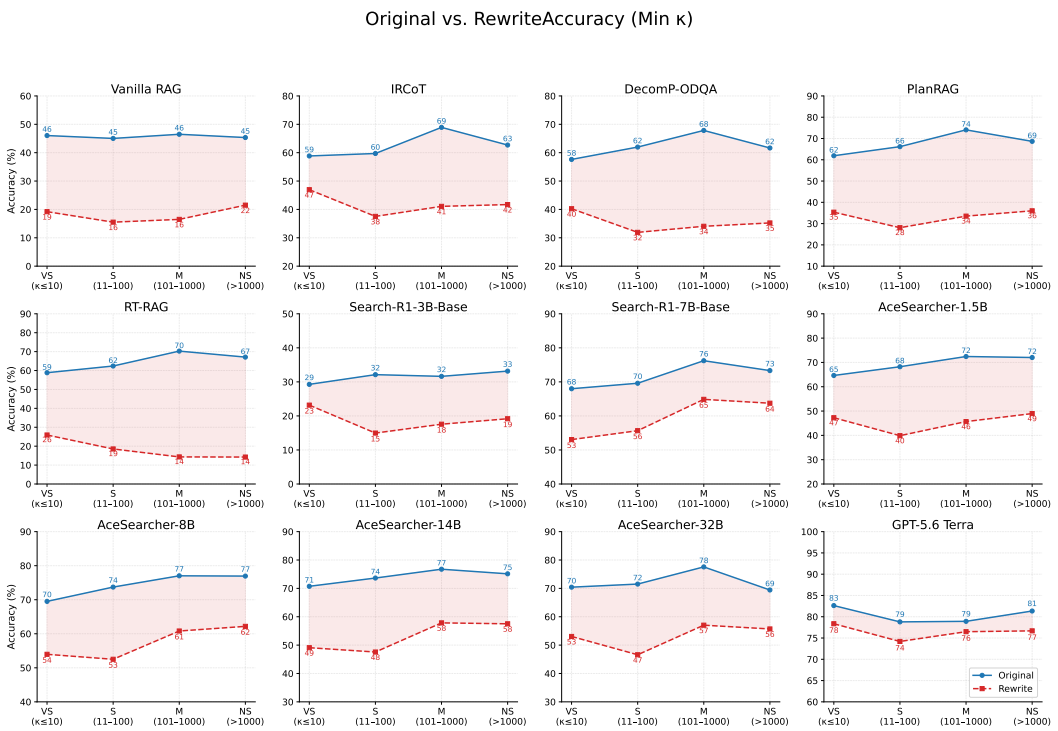

Figure 9: End-task accuracy by minimum added-constraint cardinality $k_{\min}$ for all evaluated systems. Original and rewritten questions are shown separately.

This result is important because $k_{\min}$ measures what is *available* to the planner, not what the planner actually selects or executes. A question may contain a highly discriminative constraint while the generated retrieval plan prioritizes another relation instead. The existence of one selective alternative is therefore insufficient to guarantee effective retrieval.

The first-step results in Figure 10 make this limitation more explicit. Rewriting substantially reduces Bri@1 for most multi-step retrieval systems, but the reduction is not monotonic in $k_{\min}$. For AceSearcher-14B, for example, Bri@1 decreases from 95.5% to 54.1% in the S group, compared with 96.4% to 76.2% in the NS group. Thus, questions containing a more selective available constraint can still exhibit substantially poorer first-step bridge identification.

These results show that the most selective constraint alone does not characterize the retrieval problem created by the rewrite. Whether that constraint helps depends on whether the planner recognizes and prioritizes it among the other valid retrieval alternatives.

E.2 Mean Constraint Cardinality

We next summarize the added constraints by their mean candidate cardinality. Questions are grouped into Low ($k_{\text{mean}} \leq 600$), Mid (601–800), High (801–1,000), and All-NS ($> 1{,}000$). These groups contain 357, 824, 974, and 386 questions, respectively.

Because cardinalities above 1,000 are represented as 1,001, the All-NS group consists precisely of questions for which all retained added constraints are non-selective. The other groups summarize increasingly larger average candidate sets while allowing individual constraints within a question to differ substantially in selectivity.

As with $k_{\min}$, mean cardinality does not produce a consistent monotonic ordering of end-task degradation (Figure 11). For AceSearcher-14B, the accuracy drops are 23.2, 23.4, 25.2, and 17.6 points from Low through All-NS, respectively. AceSearcher-32B similarly exhibits its largest drop outside the All-NS group. Larger average candidate sets therefore do not by themselves imply greater sensitivity to rewriting.

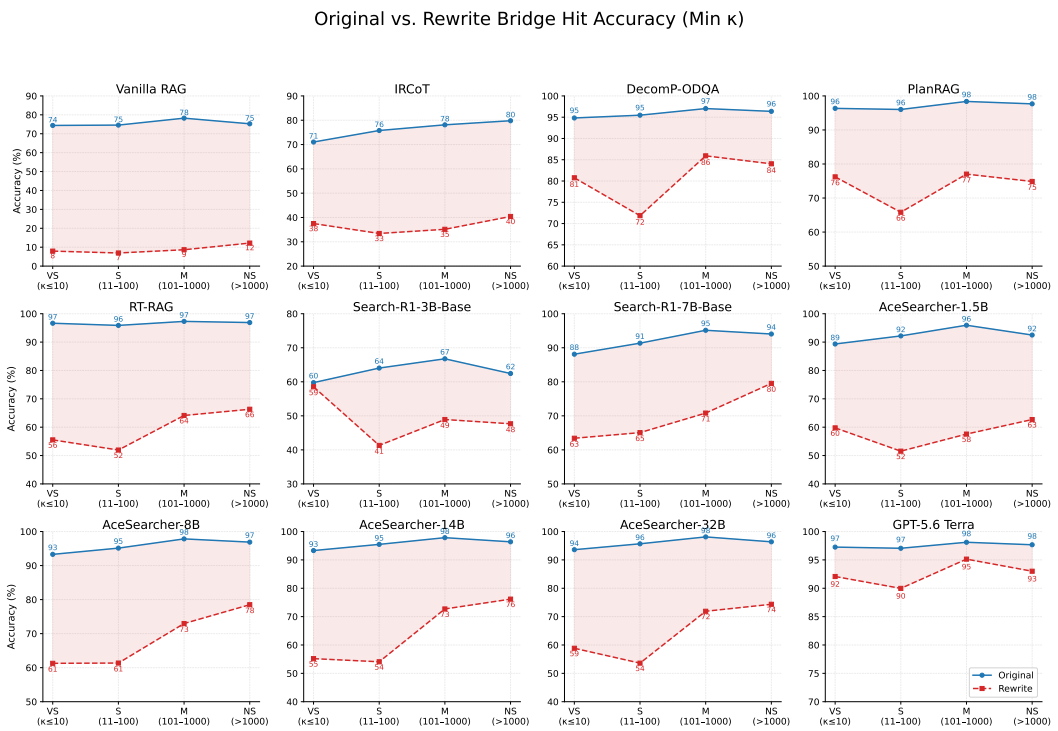

Figure 10: First-step bridge identification (Bri@1) by minimum added-constraint cardinality $k_{\min}$. Original and rewritten questions are shown separately.

This result also illustrates a limitation of averaging. Two questions can have similar k_{mean} while containing very different mixtures of selective and non-selective constraints. The average candidate count collapses this internal composition into a single value and does not represent how many alternative retrieval actions are presented to the planner.

The first-step results again expose this limitation more directly. Figure 12 shows substantial rewrite-induced reductions in Bri@1 across the Low, Mid, and High groups, while the All-NS group is often less affected. For AceSearcher-14B, rewrite Bri@1 increases from 45.4% in the Low group to 76.2% in the All-NS group even though the latter contains only constraints with larger candidate sets. Search-R1-7B-Base shows the same direction, increasing from 60.2% to 79.5%.

Taken together, the $k_{\min}$ and k_{mean} analyses show that retrieval difficulty is not determined by a single extremal or average cardinality statistic. $k_{\min}$ ignores the remaining retrieval alternatives once the most selective constraint is identified, while k_{mean} collapses the composition of the entire constraint set into one average.

RCS instead aggregates the contributions of the individual added constraints, allowing multiple plausible retrieval alternatives to accumulate at the question level. The alternative analyses therefore provide complementary evidence for the main result: the behavior of a retrieval planner depends on the composition of the available actions, not simply on the cardinality of the single most selective constraint or their arithmetic mean.

E.3 Anchor Position

The main analysis also varies where the original anchor relation appears in the rewritten question. The anchor-first, anchor-middle, and anchor-last variants preserve the same bridge-identifying relation, gold bridge, and answer, while placing the anchor before, between, or after the added descriptions. Anchor-last is the default rewrite condition used elsewhere in the paper.

This comparison tests whether access to the original dependency-resolving relation depends on its surface position. If retrieval planning were determined only by the logical dependencies of the

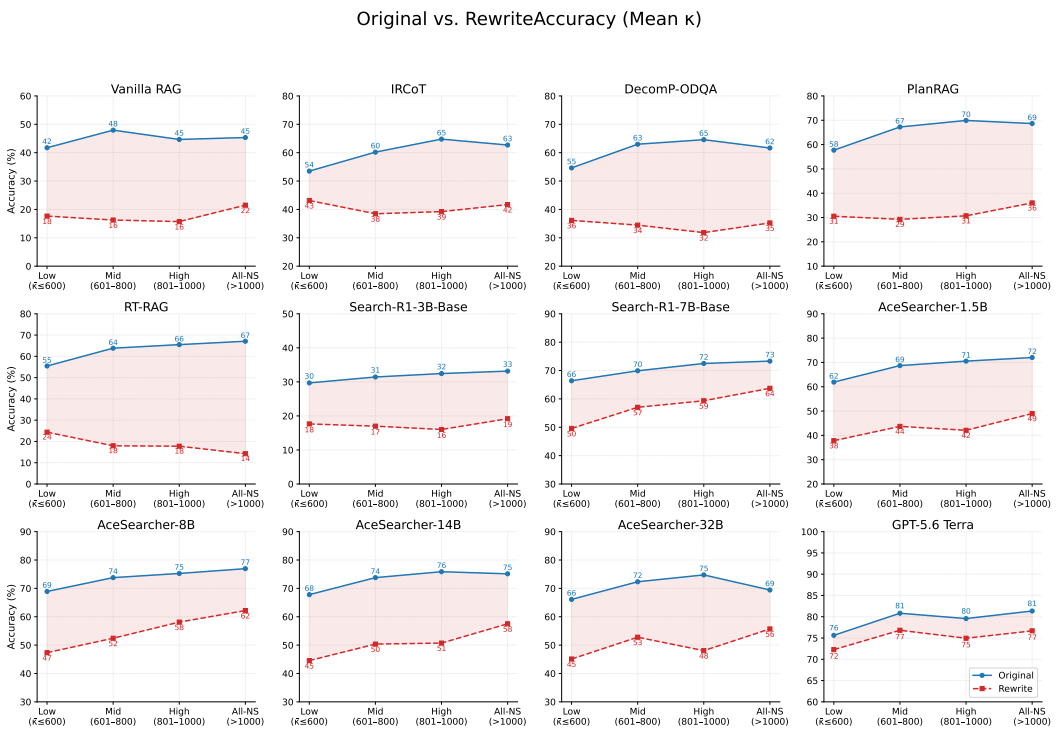

Figure 11: End-task accuracy by mean added-constraint cardinality k_{mean} for all evaluated systems. Candidate counts above 1,000 are represented as 1,001 before averaging. Original and rewritten questions are shown separately.

question, moving the same anchor relation should not systematically change performance. The comparison is not a strictly lexical position intervention, however: the three variants are generated through separate LLM calls and can therefore differ slightly in wording and constraint realization in addition to anchor position.

Table 15 reports the complete Accuracy, EM, and F1 results. For most retrieval systems, placing the anchor first substantially improves end-task performance relative to placing it in the middle or last. For example, AceSearcher-14B achieves 65.60% accuracy with the anchor first, compared with 55.45% in the middle and 50.77% with the anchor last. AceSearcher-1.5B, 8B, and 32B show the same ordering, as do Vanilla RAG, IRCoT, DecomP-ODQA, PlanRAG, and Search-R1-3B-Base.

The pattern is not universal. Search-R1-7B-Base performs better with the anchor last than in the middle, Self-RAG changes little across positions, and GPT-5.6 Terra is most accurate in the anchor-last condition. RT-RAG remains weak across all three positions. Thus, anchor position is not a deterministic predictor of performance across systems.

Nevertheless, the broad improvement from earlier anchor placement across many planning-based baselines shows that the same valid bridge-identifying relation can have different operational effects depending on how it is presented among the additional constraints. This positional sensitivity complements the RCS analysis: retrieval behavior depends not only on which valid constraints are available, but also on how the dependency-resolving action is exposed to the planner.

F Additional Failure Analysis

This appendix extends the mechanism analysis in Section 3.3 to the remaining baselines. We trace the effect of rewriting through four stages: whether the generated retrieval plan uses the anchor, whether the first retrieval contains the bridge, whether the model identifies the bridge from the retrieved evidence, and whether the trajectory ultimately answers correctly. Unless stated otherwise, all measurements refer to rewritten questions.

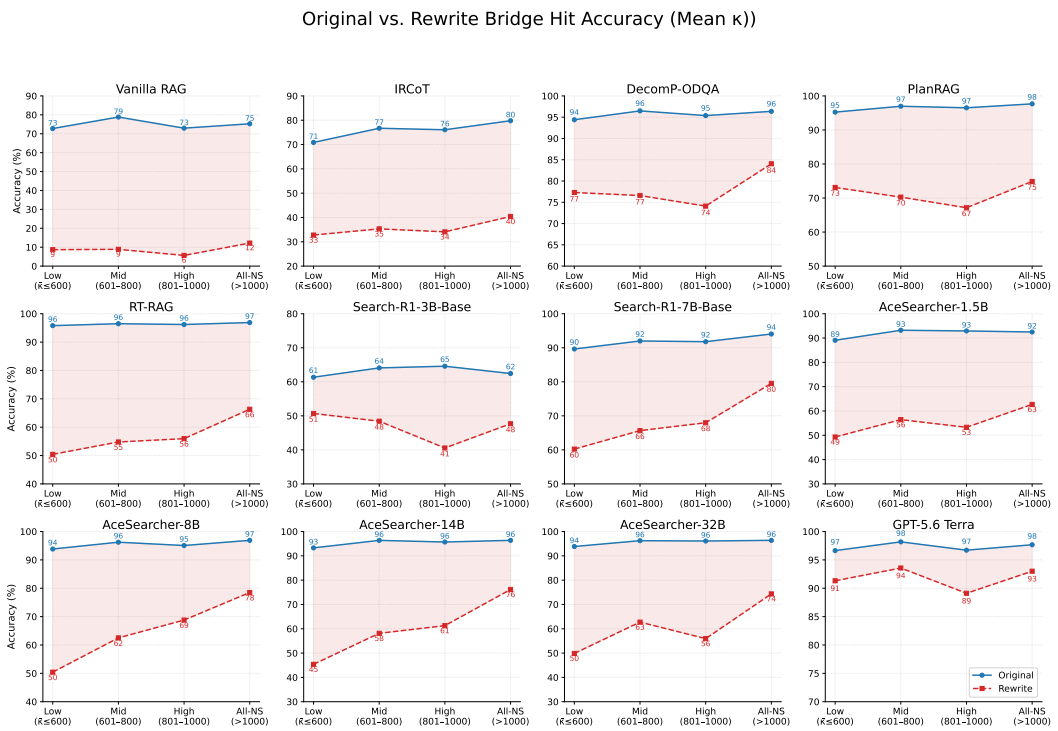

Figure 12: First-step bridge identification (Bri@1) by mean added-constraint cardinality k_{mean}. Original and rewritten questions are shown separately.

Figure 14 provides the expanded trajectory comparison corresponding to the representative systems shown in the main paper.

Anchor use. We first examine how rewriting changes the retrieval action chosen by the planner. Because every rewrite retains the original bridge-identifying anchor relation, we record when that relation first appears in the generated retrieval trajectory.

We classify a trajectory as *Clean* when the first query uses the anchor without any added constraint, *Mixed* when the first query combines the anchor with one or more added constraints, *Delayed* when the anchor first appears only in a later query, and *Absent* when it does not appear in any recorded query. Table 16 reports this distribution over all 2,541 rewritten questions.

The systems respond to the added constraints in different ways. Vanilla RAG delays the anchor in 78.2% of rewritten plans. DecomP-ODQA and PlanRAG instead place the anchor in a mixed first query in 75.9% and 85.8% of cases, respectively. AceSearcher exhibits another pattern: for AceSearcher-14B, first-query anchor use (Clean + Mixed) falls from 93.9% on the original questions to 64.2% after rewriting.

These differences show that rewriting does not induce a single planning failure mode. Some systems postpone the dependency-resolving relation, some combine it with competing constraints, and others omit it from the initial retrieval action. We therefore next separate failures in retrieving bridge-containing evidence from failures in identifying the bridge after that evidence has been retrieved.

Evidence retrieval and bridge identification. Figures 5 and 14 report passage coverage and bridge identification as separate marginal measurements. To examine their relationship within individual trajectories, we restrict this analysis to questions answered correctly in their original form and divide them according to whether the corresponding rewrite is answered correctly.

Table 17 reports two quantities. *Bridge retrieved* is the percentage of questions whose first-step retrieved passages contain the gold bridge. *Bridge identified given retrieved* conditions further on those first-step passage hits and measures the percentage for which the first intermediate output

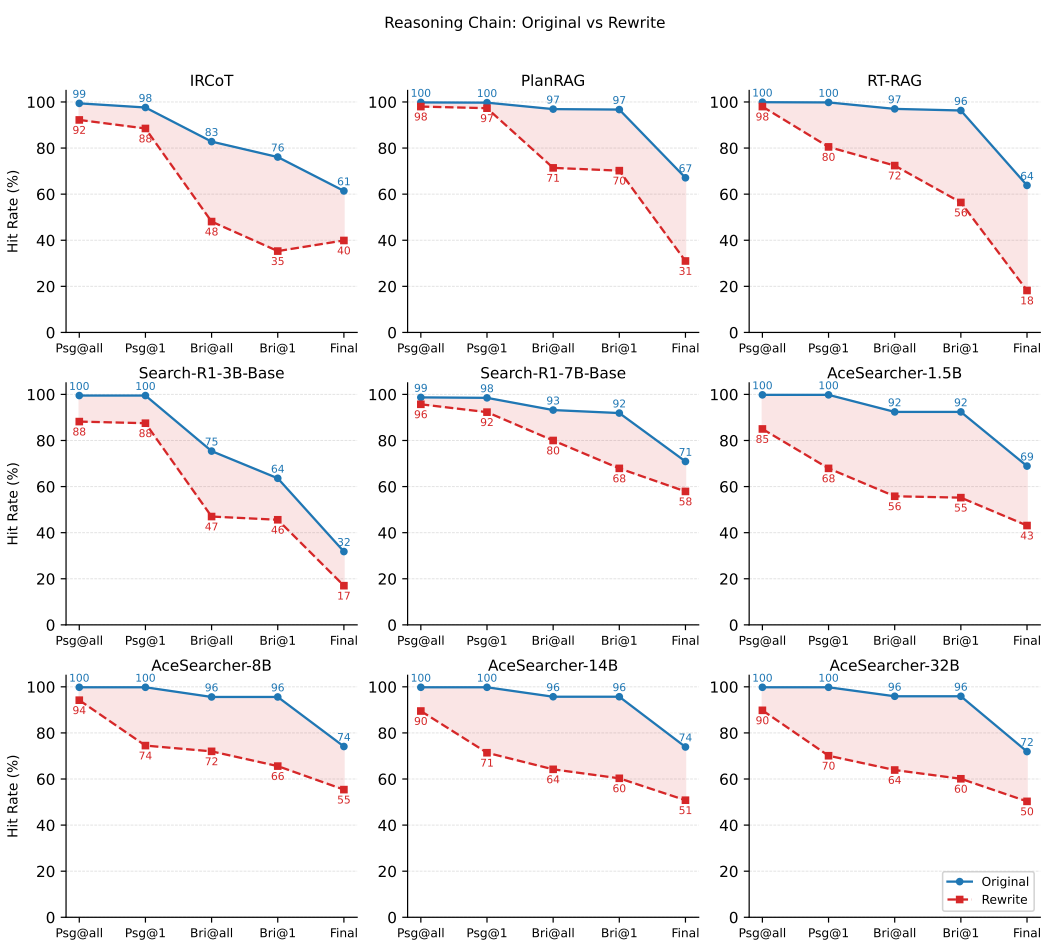

Figure 13: Passage coverage, bridge identification, and answer accuracy on original and rewritten questions. Psg measures bridge occurrence in retrieved passages, Bri measures occurrence in generated intermediate outputs, and Final denotes answer accuracy. The suffixes @1 and @all refer to the first execution step and any execution step, respectively. Values are marginal rates, not sequential conditional probabilities.

contains the bridge. The latter therefore corresponds to

$$P(\text{Bri@1} \mid \text{Psg@1}),$$

rather than to a comparison between the two marginal rates.

The distinction exposes different failure mechanisms across systems. For AceSearcher-14B, first-step bridge retrieval is substantially lower for failed rewrites than for successful rewrites, at 31.6% versus 93.8%. Moreover, even when a retrieved passage contains the bridge, the model identifies it in only 21.1% of failed rewrites, compared with 96.9% of successful rewrites.

PlanRAG exhibits a different pattern. Its first-step passage coverage remains high in both groups, at 96.7% for failed rewrites and 98.7% for successful rewrites, while conditional bridge identification differs sharply, at 55.5% and 96.2%. Thus, failure can arise either before the bridge reaches the model through retrieval or after bridge-containing evidence is already available.

Because these groups are defined by final-answer correctness, the differences are descriptive associations rather than causal effects. They nevertheless show why passage retrieval and bridge identification must be measured separately.

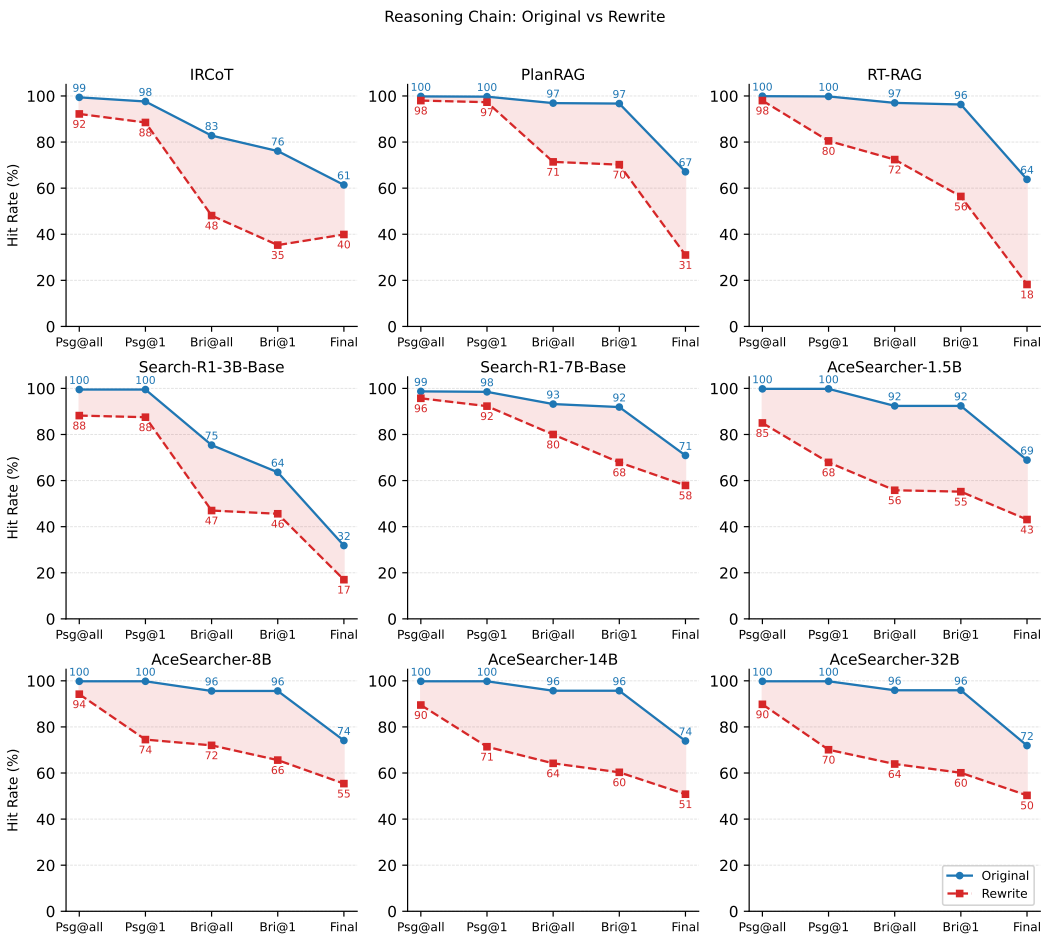

Figure 14: Passage coverage, bridge identification, and final-answer accuracy on original and rewritten questions. Psg measures bridge occurrence in retrieved passages, Bri measures bridge occurrence in generated intermediate outputs, and Final denotes answer accuracy. The suffixes @1 and @all refer to the first retrieval step and any retrieval step, respectively. These are marginal rates rather than sequential conditional probabilities.

Bridge identification among failed rewrites. A bridge missed at the first step may still be recovered later, while identifying it immediately does not guarantee a correct final answer. We therefore examine the subset of questions answered correctly in their original form but incorrectly after rewriting.

Table 18 extends Table 2 to all baselines with eligible trajectory observations. Each failed rewrite is assigned according to the first recorded occurrence of the bridge in an intermediate output: *First step* if it appears immediately, *Only later* if it first appears after the initial step, and *Never* if it does not appear anywhere in the recorded trajectory. The percentages describe the distribution of eligible failed rewrites and therefore sum over these three mutually exclusive categories within each baseline.

The expanded results reveal substantially different failure profiles. Persistent bridge misses account for 93.3% of failed rewrites for AceSearcher-1.5B, 86.3% for AceSearcher-32B, and 79.1% for IRCoT. For these systems, later retrieval rarely recovers the missing intermediate entity once the initial trajectory moves away from it.

Vanilla RAG shows the opposite pattern: 77.1% of its failed rewrites identify the bridge only after the first step. DecomP-ODQA and RT-RAG frequently fail even though the bridge is already identified at the first step, which occurs in 61.8% and 53.7% of their failed rewrites, respectively. Early bridge

identification is therefore an important intermediate milestone, but it is not sufficient for correct answering.

Final accuracy conditional on bridge identification. The previous analysis conditions on final failure and asks where the bridge first appeared. We now reverse the view: over all eligible rewritten questions, we group trajectories by bridge-identification timing and measure final accuracy within each group.

Table 19 reports both the number of eligible trajectories and their answer accuracy for *Never*, *Only later*, and *First step*. Unlike Table 18, these percentages measure correctness *within* each bridge-identification category rather than the distribution of failed examples.

For AceSearcher-14B, final accuracy is 75.0% when the bridge is identified at the first step, compared with 11.0% when it is identified only later and 14.3% when it is never observed in the intermediate outputs. The same qualitative separation is visible for several other AceSearcher and Search-R1 configurations.

The relationship is weaker for other architectures. First-step bridge identification is associated with only 42.7% accuracy for DecomP-ODQA, 42.0% for PlanRAG, and 22.9% for RT-RAG. These cases show that correctly resolving the dependency- critical bridge does not eliminate downstream failure: subsequent retrieval or final answer generation can still go wrong.

Together, the trajectory analyses separate three distinct failure points. A planner may fail to retrieve evidence containing the bridge, may retrieve such evidence without identifying the bridge, or may identify the bridge correctly but fail in the subsequent reasoning and answering stages. The relative contribution of these failure modes varies substantially across retrieval architectures.

Measurement limits. All passage and bridge measurements follow the observable-state definitions in Appendix C.6. Unavailable observations are excluded from the corresponding estimate rather than treated as misses, so denominators can differ across systems and measurements.

A passage containing the bridge does not establish that it supports the relation required by the current retrieval step. Likewise, bridge occurrence in an intermediate output does not establish that the entity is subsequently used correctly. Nonzero accuracy in the *Never* category therefore does not contradict the trajectory measurements: under the observable-state criterion, a system can answer correctly without explicitly producing the bridge in a recorded intermediate output.

Similarly, answer occurrence anywhere in the retrieved trajectory does not by itself establish successful downstream retrieval. The answer may appear before the bridge is identified or in a passage that does not support the requested relation. Distinguishing these downstream cases would require aligning the subsequent query, retrieved evidence, and generated intermediate answer with the second gold hop.

Self-RAG-7B is excluded from these sequential trajectory analyses because its logged branch outputs do not form a single ordered retrieval trajectory. As in the main evaluation, † marks Search-R1-Instruct runs affected by context overflow.

G ANCHOR-FIRST PROMPTING

This appendix provides additional details for the anchor-first prompting intervention in Section 3.4. We describe the evaluated configurations, define the bridge-identification measurements used for this intervention, report additional reference and answer metrics, and provide the complete prompt.

Setup. Vanilla RAG uses Qwen2.5-14B with the shared 2WikiMultiHopQA E5-large-v2 retrieval backend. AceSearcher-1.5B, AceSearcher-14B, and AceSearcher-32B use Qwen2.5 backbones, whereas AceSearcher-8B uses Llama-3.1-8B.

Base denotes the baseline zero-shot decomposition prompt, and Ours denotes the four-example anchor-first prompt. The intervention changes only the decomposition prompt; the evaluated question, model checkpoint, retriever, and execution procedure remain unchanged.

The AceSearcher-14B anchor-first rewrite run contains 2,540 questions because one record is missing. All other reported runs contain 2,541 questions. The AceSearcher-14B Base–Ours comparison is therefore not strictly paired.

Bridge-identification measurements. For this intervention, we inspect the generated subquestion answers to determine when the gold bridge is identified. Bri@1 is the percentage of questions for which the gold bridge entity occurs as a substring of the first subquestion answer, while Bri@all applies the same criterion to any generated subquestion answer.

A trajectory counted by Bri@all but not Bri@1 therefore identifies the bridge only after the first subquestion. These measurements concern the generated answers rather than the retrieval queries themselves: they do not establish which relation produced the retrieval, whether the bridge answer was supported by the retrieved passages, or whether later retrieval correctly used the identified bridge. They should therefore be distinguished from the passage coverage measurements in Section 3.3 and Appendix F.

Original-question reference. Table 20 provides original-question reference results. For the AceSearcher variants, these scores use the Base prompt only. They provide a reference for the models' original-question performance but are not matched Base–Ours comparisons across original and rewritten questions.

Vanilla RAG is additionally evaluated with the anchor-first prompt on the original questions, allowing the same prompt to be compared across both conditions. Its accuracy is 62.57% on original questions and 64.38% on rewrites. Despite the similar end-task accuracy, bridge identification remains slightly lower on rewrites: Bri@1 is 91.89%, compared with 95.79% on originals, and Bri@all is 92.52%, compared with 96.26%. Thus, removing the aggregate accuracy deficit does not imply identical intermediate behavior.

Additional answer metrics. Table 21 reports EM and F1 on rewritten questions. Both metrics improve under anchor-first prompting for all five evaluated systems, consistent with the accuracy improvements reported in the main text. As with accuracy and Bri@1, the smallest change occurs for AceSearcher-8B.

Remaining failures. The intervention changes bridge-identification behavior substantially for several models but does not eliminate all failures. For AceSearcher-14B, Bri@1 and Bri@all under the anchor-first prompt are 88.94% and 88.98%, respectively; for AceSearcher-32B they are 88.90% and 88.94%. The near equality indicates that almost all observed bridge identification occurs at the first step rather than being recovered later. It does not imply that every generated trajectory contains exactly two steps or that all subsequent retrieval is correct.

AceSearcher-8B responds differently. Its Bri@1 changes only from 65.41% to 65.68%, while Bri@all increases from 71.74% to 74.03%. Thus, a nontrivial fraction of bridge identifications still occur after the first subquestion, and the anchor-first demonstrations have little effect on first-step identification for this model.

These aggregate results localize where the intervention has an effect, but they do not by themselves identify the cause of the remaining errors. In particular, they do not establish that reader quality, downstream retrieval, or error propagation from earlier answers is the principal residual bottleneck. Distinguishing these explanations would require trajectory-level analysis of the subsequent retrieval and answer-generation steps.

Full prompt. The intervention replaces the baseline decomposition prompt with four demonstrations in which the first subquestion identifies the filmmaker from a named film and the second subquestion asks for the requested attribute. At inference time, the evaluated question replaces `{question}`. No gold bridge entity, gold answer, or explicit anchor annotation is provided for the evaluated question.

Prompt

```
Decompose each question into sub-questions. Mark each sub-question
    with ### Q<N>:. Use #N to refer to the answer of sub-question N.

Question: "Who is the mother of the filmmaker who received the
    Paszport Polityki award, works as a film director, was educated
    at the National Film School, and directed the film Polish-
    Russian War?"
### Q1: Who directed the film Polish-Russian War?
### Q2: Who is the mother of #1?

Question: "What is the nationality of the filmmaker who was
    educated at the University of Wisconsin-Madison, is a rower,
    participated in the 2000 Summer Olympics -men's coxless four,
    and directed the film World and Time Enough?"
### Q1: Who directed the film World and Time Enough?
### Q2: What is the nationality of #1?

Question: "Where was the filmmaker who is a member of the American
    Academy of Arts and Sciences, works as a film director, received
     The Nichols-Chancellor's Medal, and directed the film Thomas
    Jefferson born?"
### Q1: Who directed the film Thomas Jefferson?
### Q2: Where was #1 born?

Question: "Who is the spouse of the filmmaker who has works in the
    collection of the Groninger Museum, is an actor, held seat 32 of
     the Academie francaise, and directed the film Eden and After?"
### Q1: Who directed the film Eden and After?
### Q2: Who is the spouse of #1?

Question: "{question}"
```

The demonstrations specify both a preferred first retrieval relation and a particular decomposition structure. Moreover, all four demonstrations use the film–director relation. The experiment therefore shows that this prompt configuration can redirect retrieval toward the anchor and improve bridge identification, but it does not isolate a domain-general ability to infer retrieval priority from selectivity alone.

H PROMPTED-BASELINES

Table 22 summarizes decoding settings specified by the evaluated runners. A slash in the maximum-token column separates planning and subquestion-answering budgets when the two stages use different limits.

IRCoT. At each retrieval step, IRCoT uses the following system prompt:

Prompt

```
You are a multi-hop question answering assistant. Answer by
    reasoning step by step. When you need to retrieve a fact, output
     a single line: SEARCH: <your query>. When you are ready to give
     the final answer, output: ANSWER: <short answer>. You MUST
    issue at least 2 SEARCH calls before writing ANSWER.
```

The corresponding user message is:

Prompt

```
Context passages:

{accumulated_passages}

Question: {question}

Previous steps:

{prior_steps}

Next step:
```

When a separate final synthesis is required, the runner uses:

Prompt

```
You have the following passages:

{passages}

You are also given some reasoning steps:

{sub_answers}

Please answer the question '{question}' using the passages and
    reasoning steps as reference.

Think step by step, then put ONLY the final short answer inside <
    answer></answer> tags.
```

DecomP-ODQA. The decomposition prompt is:

Prompt

```
You are a system that decomposes complex questions into simpler
    single-hop questions that can be answered using Wikipedia.

Given a complex question (QC), generate a sequence of sub-questions
     (QS) that help answer it step-by-step.

Each sub-question should retrieve information needed for the next
    step.

Format:

Question: What is the profession of the person who directed the
    movie 'The Godfather'?

Decomposition:

1. Who directed the movie 'The Godfather'?

2. What is the profession of #1?

Question: Which city is the capital of the country where the Eiffel
     Tower is located?

Decomposition:

1. In which country is the Eiffel Tower located?

2. What is the capital of #1?

Question: Which author who was born in US and won the Booker Prize
    wrote 'Tenth of December'?

Decomposition:

1. Which author was born in the US and won the Booker Prize?

2. Did #1 write 'Tenth of December'?

Question: {question}

Decomposition:
```

PlanRAG. The PlanRAG planning system prompt is:

Prompt

```
You are a reasoning DAG generator expert. The goal is to make a
    reasoning DAG with minimum nodes. Given a query, if it is
    complex and requires a reasoning plan, split it into smaller,
    independent, and individual subqueries. The query and subqueries
     are used to construct a rooted DAG so make sure there are NO
    cycles and all nodes are connected, there is only one leaf node
    with a single root and one sink. DAG incorporates Markov
    property i.e. you only need the answer of the parent to answer
    the subquery. The main query should be the parent node of the
    initial set of subatomic queries such that the DAG starts with
    it. Return a Python list of tuples of parent query and the
    subatomic query which can be directly given to eval(). Strictly
    follow the below template for output. For the subquery
    generation, input a tag <AI.J> where the answer of the parent
    query should come to make the query complete. NOTE: Make the DAG
     connected and for simple queries return the original query only
     without any reasoning DAG.

Example: Query: Who is the current PM of India? DAG: "Q: Who is the
     current PM of India?" Query:What is the tallest mountain in the
     world and how tall is it? DAG: [ ("Q: What is the tallest
    mountain in the world and how tall is it?", "Q1.1: What is the
    tallest mountain in the world?"), ("Q1.1: What is the tallest
    mountain in the world?", "Q2.1: How tall is <A1 .1>?") ] Query:
    What percentage of the worlds population lives in urban areas?
    DAG: [ ("Q: What percentage of the worlds population lives in
    urban areas?", "Q1.1: What is the total world population?"), ("Q
    : What percentage of the worlds population lives in urban areas
    ?", "Q1.2: What is the total population living in urban areas
    worldwide?"), ("Q1.1: What is the total world population?", "Q2
    .1: Calculate the percentage living in urban areas worldwide
    when total population is <A1.1> and population living in urban
    areas is <A1.2>?"), ("Q1.2: What is the total population living
    in urban areas worldwide?", "Q2.1: Calculate the percentage
    living in urban areas worldwide when total population is <A1.1>
    and population living in urban areas is <A1.2>?") ]
```

The raw question is supplied as the user message.

RT-RAG. The reasoning-tree planning prompt is:

Prompt

```
You are an expert at breaking down complex, multi-hop questions
    into a "Reasoning Tree" of simple, independent sub-queries.

**Task:**
Analyze the user's complex query and decompose it into a series of
    logical steps (nodes).
- Each step must be a simple question that can be answered by a
    search engine or knowledge graph.
- Determine if a step depends on the answer of a previous step.

**Output Format:**
Return ONLY a valid JSON object with a list of "steps".
Each step should have:
- "id": A number (1, 2, 3...)
- "question": The sub-query text.
- "dependency": The ID of the step this question depends on (or
    null if independent).
- "entity": The main entity this step focuses on.

**Example:**
Query: "Who is the director of the movie that won the Best Picture
    Oscar in 2024?"

Output:
{
  "steps": [
    {
      "id": 1,
      "question": "What movie won the Best Picture Oscar in 2024?",
      "dependency": null,
      "entity": "Best Picture Oscar 2024"
    },
    {
      "id": 2,
      "question": "Who is the director of [Movie from Step 1]?",
      "dependency": 1,
      "entity": "Movie from Step 1"
    }
  ]
}
```

Shared DecomP-ODQA, PlanRAG, and RT-RAG reader prompts. After planning, these methods use the following subquestion reader:

Prompt

```
You have the following context passages:

{context}

Please answer the question '{question}' with a short span using the
     context as reference.

If no answer is found in the context, use your own knowledge. Your
    answer needs to be as short as possible.
```

When retrieved passages are included in final synthesis, the shared prompt is:

Prompt

```
You have the following passages:

{passages}

You are also given some subquestions and their answers:

{sub_answers}

Please answer the question '{question}' with a short span using the
     documents and subquestions as reference.

Make sure your response is grounded in documents and provides clear
     reasoning followed by a concise conclusion. If no relevant
    information is found, use your own knowledge.

Wrap your answer with <answer> and </answer> tags.
```

The corresponding no-passage final-synthesis prompt is:

Prompt

```
You are given some subquestions and their answers:

{sub_answers}

Please answer the question '{question}' with a short span using the
     subquestions as reference. Provides clear reasoning followed by
     a concise conclusion. If no relevant information is found, use
    your own knowledge.

Wrap your answer with <answer> and </answer> tags.
```

DQG. The decomposer uses the following system message:

Prompt

```
You are an intelligent assistant that can decompose a complex
    question into simpler sub-questions.
```

The user instruction is:

Prompt

```
I will provide you with one complex multi-hop question. Decompose
    it into a set of simpler sub-questions so that it will be easier
     to identify and retrieve the information for answering them
    instead of the complex question. It is important that the sub-
    questions (i) are not ambiguously worded, (ii) are logically
    connected, and (iii) that they cover all the necessary
    information and steps to answer the original complex question.
    It is very important that the decompositions must be as simple
    as possible, i.e. (i) there should only be as many sub-questions
     as absolutely necessary; and (ii) each sub-question and its
    answer should cover only a single atomic fact. Please provide
    the sub-questions in a single line, in the order that you would
    like them to be answered. They should be separated from each
    other by a ";".

When composing a sub-question, you must ensure that answers of
    earlier sub-questions are properly referred to whenever possible
    ; when you need to do this, use a variable (e.g. "#1") for
    referring to the previous answer. It is very important to get
    the numbering for the answer variable correct. A sub-question
    should never refer to the answer of a sub-question that comes
    after it.

Start your reply with your reasoning (in less than 50 words), and
    then immediately give the sequence of sub-questions, and stop
    immediately after; do not say anything else. Limit your
    reasoning to less than 50 words. It is very important that you
    prefix the start of each sub-question with a "[SQX]" marker that
     indicates the position number (X) of the sub-question, i.e. "[
    SQ1] ... [SQ2] ... [SQ3] ...".

{question}
```

The DQG reader uses the AceSearcher reader prompts given below.

Search-R1. Search-R1 prepends the following instruction to the user message:

Prompt

```
Answer the given question. You must conduct reasoning inside <think
    > and </think> first every time you get new information. After
    reasoning, if you find you lack some knowledge, you can call a
    search engine by <search> query </search> and it will return the
     top searched results between <information> and </information>.
    You can search as many times as your want. If you find no
    further external knowledge needed, you can directly provide the
    answer inside <answer> and </answer>, without detailed
    illustrations. For example, <answer> Beijing </answer>.
```

AceSearcher. The baseline decomposition prompt is:

Prompt

```
Please break down the question "{question}" into multiple specific
    sub-questions that address individual components of the original
     question.

Mark each sub-question with ### at the beginning. If you need to
    refer to answers from earlier sub-questions, use #1, #2, etc.,
    to indicate the corresponding answers.

Decomposed Question:
```

The per-subquestion reader prompt is:

Prompt

```
You have the following context passages:

{context_passages}

Please answer the question '{sub_question}' with a short span using
     the context as reference.

If no answer is found in the context, use your own knowledge. Your
    answer needs to be as short as possible.
```

The evaluated `--add-passage 0` final-answer prompt is:

Prompt

```
You are given some subquestions and their answers:

### Q1: {q1_text}, Answer for Q1: {a1}
### Q2: {q2_text}, Answer for Q2: {a2}
...

Please answer the question '{question}' with a short span using the
     subquestions as reference. Provides clear reasoning followed by
     a concise conclusion. If no relevant information is found, use
    your own knowledge.

Wrap your answer with <answer> and </answer> tags.
```

Retrieval and relevance reflection tokens are generated as part of the checkpoint's learned behavior rather than introduced through an additional instruction string.

Table 15: Full results for the anchor-position evaluation. Each positional rewrite is compared with the corresponding original question. The variants preserve the same anchor relation and answer but are generated separately and can differ slightly in wording and constraint realization. ΔAcc denotes Rewrite − Original Accuracy. † denotes Search-R1-Instruct runs affected by context overflow.

		Accuracy (%)			EM (%)		F1 (%)	
Model	Position	Orig.	Rewrite	Δ	Orig.	Rewrite	Orig.	Rewrite
Prompted								
Vanilla-RAG	First	45.42	24.16	-21.26	25.38	14.21	36.68	20.73
	Middle	45.42	19.09	-26.33	25.38	11.29	36.68	16.26
	Last	45.42	17.04	-28.38	25.38	10.11	36.68	15.12
IRCoT	First	61.39	47.30	-14.09	31.92	25.93	49.37	39.27
	Middle	61.39	42.31	-19.08	31.92	23.61	49.37	35.51
	Last	61.39	39.91	-21.48	31.92	21.33	49.37	33.45
DecomP-ODQA	First	62.22	43.41	-18.81	33.96	23.26	50.40	34.93
	Middle	62.22	33.06	-29.16	33.96	17.99	50.40	26.53
	Last	62.22	33.81	-28.41	33.96	18.81	50.40	27.53
PlanRAG	First	67.14	43.92	-23.22	36.40	24.05	53.67	36.16
	Middle	67.14	28.61	-38.53	36.40	15.66	53.67	23.21
	Last	67.14	31.01	-36.13	36.40	17.67	53.67	25.38
RT-RAG	First	63.79	21.45	-42.34	34.16	11.02	50.84	17.88
	Middle	63.79	19.32	-44.47	34.16	10.19	50.84	16.04
	Last	63.79	18.22	-45.57	34.16	10.70	50.84	15.65
Trained								
Self-RAG-7B	First	16.17	12.71	-3.46	0.20	0.67	4.92	3.91
	Middle	16.17	11.65	-4.52	0.20	0.71	4.92	3.47
	Last	16.17	12.99	-3.18	0.20	0.67	4.92	3.66
DQG	First	13.89	13.81	-0.08	0.55	0.35	2.57	1.36
	Middle	13.89	9.45	-4.44	0.55	0.39	2.57	1.36
	Last	13.89	9.29	-4.60	0.55	0.55	2.57	1.53
Search-R1-3B-Base	First	31.84	27.43	-4.41	23.57	21.61	31.59	27.95
	Middle	31.84	19.24	-12.60	23.57	14.80	31.59	20.03
	Last	31.84	17.04	-14.80	23.57	13.73	31.59	18.13
Search-R1-3B-Instruct†	First	30.74	26.37	-4.37	19.56	17.75	27.60	24.71
	Middle	30.74	20.86	-9.88	19.56	14.21	27.60	19.45
	Last	30.74	15.98	-14.76	19.56	11.57	27.60	15.27
Search-R1-7B-Base	First	70.92	60.25	-10.67	48.52	42.62	62.64	54.47
	Middle	70.92	54.19	-16.73	48.52	38.72	62.64	48.83
	Last	70.92	57.89	-13.03	48.52	40.73	62.64	51.78
Search-R1-7B-Instruct†	First	51.83	48.80	-3.03	38.88	38.45	50.30	48.64
	Middle	51.83	43.64	-8.19	38.88	34.55	50.30	43.55
	Last	51.83	42.70	-9.13	38.88	33.18	50.30	42.13
AceSearcher-1.5B	First	68.95	61.87	-7.08	42.38	38.84	57.89	52.69
	Middle	68.95	53.99	-14.96	42.38	33.96	57.89	45.77
	Last	68.95	43.05	-25.90	42.38	26.88	57.89	36.81
AceSearcher-8B	First	74.14	67.02	-7.12	46.48	41.95	62.09	56.40
	Middle	74.14	59.62	-14.52	46.48	38.06	62.09	50.89
	Last	74.14	55.37	-18.77	46.48	35.46	62.09	47.54
AceSearcher-14B	First	73.95	65.60	-8.35	45.18	39.71	61.71	55.03
	Middle	73.95	55.45	-18.50	45.18	34.08	61.71	46.44
	Last	73.95	50.77	-23.18	45.18	31.33	61.71	43.34
AceSearcher-32B	First	71.94	64.50	-7.44	44.27	40.46	60.79	54.91
	Middle	71.94	58.60	-13.34	44.27	36.25	60.79	49.94
	Last	71.94	50.33	-21.61	44.27	31.76	60.79	43.42
Reference								
GPT-5.6 Terra	First	79.69	70.88	-8.81	38.61	34.59	62.43	56.06
	Middle	79.69	73.16	-6.53	38.61	36.09	62.43	57.82
	Last	79.69	75.44	-4.25	38.61	36.91	62.43	59.62

Table 16: Anchor use in rewritten retrieval plans (%; $n = 2,541$ per model).

Model	Clean	Mixed	Delayed	Absent
Prompted				
Vanilla RAG	2.6	4.1	78.2	15.1
IRCoT	11.5	75.9	2.2	10.5
DecomP-ODQA	15.9	75.9	1.7	6.5
PlanRAG	7.6	85.8	0.3	6.3
RT-RAG	20.4	53.6	18.9	7.0
Trained				
DQG	12.3	39.8	18.5	29.4
Search-R1-3B-Base	7.9	74.7	0.5	16.9
Search-R1-3B-Instruct†	9.0	37.2	3.4	50.5
Search-R1-7B-Base	1.2	88.3	1.7	8.8
Search-R1-7B-Instruct†	8.6	29.5	2.4	59.5
AceSearcher-1.5B	1.7	58.6	15.3	24.3
AceSearcher-8B	8.9	58.1	19.2	13.8
AceSearcher-14B	12.9	51.3	17.0	18.8
AceSearcher-32B	9.8	52.7	17.7	19.9
Reference				
GPT-5.6 Terra	49.2	42.3	1.6	6.9

Table 17: First-hop retrieval and bridge identification among originally correct questions, grouped by rewrite correctness (%).

	Bridge retrieved		Bridge identified given retrieved	
Model	Rewrite fails	Rewrite succeeds	Rewrite fails	Rewrite succeeds
Prompted				
Vanilla RAG	8.7	27.2	65.8	77.9
IRCoT	81.5	97.3	17.3	58.4
DecomP-ODQA	95.7	99.3	64.5	96.7
PlanRAG	96.7	98.7	55.5	96.2
RT-RAG	76.9	96.7	69.4	74.3
Trained				
DQG	45.4	76.6	65.0	86.3
Search-R1-3B-Base	84.2	96.2	42.6	79.0
Search-R1-3B-Instruct†	39.8	91.5	39.8	85.5
Search-R1-7B-Base	80.1	98.1	39.3	85.8
Search-R1-7B-Instruct†	23.2	92.7	65.0	90.8
AceSearcher-1.5B	27.3	97.1	21.0	97.0
AceSearcher-8B	29.0	94.5	27.5	97.5
AceSearcher-14B	31.6	93.8	21.1	96.9
AceSearcher-32B	25.2	94.3	26.5	96.9
Reference				
GPT-5.6 Terra	82.5	98.7	44.7	98.3

Table 18: Bridge identification among originally correct questions answered incorrectly after rewriting. Percentages use the eligible failed-pair count n. † denotes runs affected by context overflow.

		Bridge identification (%)		
Baseline	n	Never	Only later	First step
Prompt-based				
Vanilla RAG	904	17.1	77.1	5.8
IRCoT	772	79.1	6.7	14.1
DecomP-ODQA	830	35.4	2.8	61.8
PlanRAG	1,005	44.5	1.8	53.7
RT-RAG	1,251	27.3	18.9	53.7
Trained				
DQG	203	43.3	23.6	33.0
Search-R1-3B-Base	571	63.2	0.9	35.9
Search-R1-3B-Instruct†	593	79.9	4.2	15.9
Search-R1-7B-Base	457	56.5	12.0	31.5
Search-R1-7B-Instruct†	530	83.4	1.5	15.1
AceSearcher-1.5B	713	93.3	1.0	5.8
AceSearcher-8B	528	76.9	15.2	8.0
AceSearcher-14B	646	82.8	10.5	6.7
AceSearcher-32B	615	86.3	7.0	6.7
Reference				
GPT-5.6 Terra	171	58.5	4.1	37.4

Table 19: First-step retrieval and bridge identification among originally correct questions, grouped by rewrite correctness (%).

	Never		Only later		First Step	
Model	n	Acc.	n	Acc.	n	Acc.
Prompted						
Vanilla RAG	428	3.5	1,906	17.7	206	39.3
IRCoT	1,318	18.7	327	60.6	896	63.6
DecomP-ODQA	514	4.7	57	15.8	1,898	42.7
PlanRAG	727	5.2	31	6.5	1,783	42.0
RT-RAG	695	16.5	404	5.2	1,421	22.9
Trained						
DQG	915	4.6	389	8.2	1,033	14.9
Search-R1-3B-Base	1,348	5.8	34	35.3	1,159	29.6
Search-R1-3B-Instruct†	1,714	4.1	135	25.9	692	43.5
Search-R1-7B-Base	509	11.0	307	58.3	1,725	71.7
Search-R1-7B-Instruct†	1,514	1.9	145	62.1	882	61.3
AceSearcher-1.5B	1,123	6.2	16	0.0	1,402	73.0
AceSearcher-8B	711	13.6	162	16.7	1,668	76.9
AceSearcher-14B	909	14.3	100	11.0	1,532	75.0
AceSearcher-32B	917	12.1	97	21.6	1,527	75.1
Reference						
GPT-5.6 Terra	188	19.1	29	55.2	2,324	80.2

Table 20: Original-question performance (%). Vanilla RAG is additionally evaluated with the anchor-first prompt.

Model	Prompt	Accuracy	Bri@1	Bri@all
Vanilla RAG	Base	45.42	75.32	81.31
	Ours	62.57	95.79	96.26
AceSearcher-1.5B	Base	68.95	91.89	91.93
AceSearcher-8B	Base	74.14	95.40	95.40
AceSearcher-14B	Base	73.95	95.47	95.47
AceSearcher-32B	Base	71.94	95.43	95.43

Table 21: EM and F1 on rewritten questions (%). Bold indicates the higher score within each model.

	EM (%)		F1 (%)	
Model	Base	Ours	Base	Ours
Vanilla RAG	10.11	**35.58**	15.12	**52.04**
AceSearcher-1.5B	26.88	**35.85**	36.81	**48.59**
AceSearcher-8B	35.46	**37.23**	47.54	**49.98**
AceSearcher-14B	31.33	**41.10**	43.34	**56.08**
AceSearcher-32B	31.76	**42.35**	43.42	**57.27**

Table 22: Decoding settings used in the evaluated open-model configurations.

Method	Temperature	Top-p	Repetition penalty	Maximum tokens
Vanilla RAG	0.0	0.99	1.05	2048 / 512
IRCoT	0.0	0.99	1.05	1024 / 512
DecomP-ODQA	0.0	0.99	1.05	1024 / 512
PlanRAG	0.0	0.99	1.05	1024 / 512
RT-RAG	0.0	0.99	1.05	1024 / 512
DQG	0.0	1.00	–	500 / 512
Search-R1	0.7	1.00	–	1024 per step
AceSearcher	0.0	0.99	1.05	512
Self-RAG	0.0	1.00	–	100