A Survey of Agentic RAG: From Retrieval Decisions to Evidence-Grounded Answers

Abstract

Retrieval-augmented generation is moving from one-shot lookup toward systems that decide how to gather and use evidence. In a multihop question, an early search can reveal what to look for next, while additional passages can distract the model and add cost. Agentic RAG gives the model control over when and where to search, how to formulate follow-up queries, how to use retrieved evidence, and when to stop. This survey organizes the field around those decisions and connects them to evidence assessment, memory, synthesis, policy learning, and evaluation. We distinguish finding relevant material from assembling sufficient support and using that support correctly. Across question answering, web search, and long-form research, the literature shows that retrieval gains depend on evidence access, model and tool capabilities, and resource budgets. Search depth and retrieval-call savings therefore do not by themselves establish better answers or greater efficiency. We use these distinctions to compare reported results and identify open problems in adaptive acquisition, evidence use, and grounded generation.

1 Introduction

Large language models can answer a wide range of questions, but their knowledge is fixed when training ends, and they can state false claims with confidence. *Retrieval-augmented generation* (RAG) addresses both problems by retrieving relevant documents and generating an answer from them [1]. In its standard form, a RAG system searches once with the user's question and answers from the top results. This simple pipeline now underlies search assistants, enterprise question answering, and many other applications.

A single fixed retrieval step works well when the question itself says what to look for, but many questions do not. To answer "Where did the director of *Oppenheimer* go to university?", a system may need to identify Christopher Nolan as the director before searching for his education. Systems that interleave retrieval with reasoning, letting each finding shape the next search, handle such multihop questions far better than single-step retrieval [2]. Open-ended tasks, such as writing a research report, push further still, requiring many rounds of searching that no single step can provide.

These limits have pushed RAG systems from fixed pipelines toward systems that make their own retrieval decisions. Some decide when to retrieve, based on the question or on their own uncertainty [3, 12]. Others critique the passages they retrieve and correct course when the evidence is weak [6, 7], or alternate reasoning steps with search actions [13]. More recently, systems have begun to learn search behavior directly through reinforcement learning [9]. We refer to this family as *agentic RAG*: systems in which the model controls retrieval, deciding whether, when, and what to retrieve, judging the evidence it gathers, and determining when that evidence is enough to answer. Agentic RAG thus sits between fixed RAG and general tool-using agents [13, 14]: it keeps RAG's goal of answering from retrieved evidence, but gives the model an agent's freedom to decide how to gather that evidence. *Deep research agents*, which search at length before writing a report [15, 16], take this furthest.

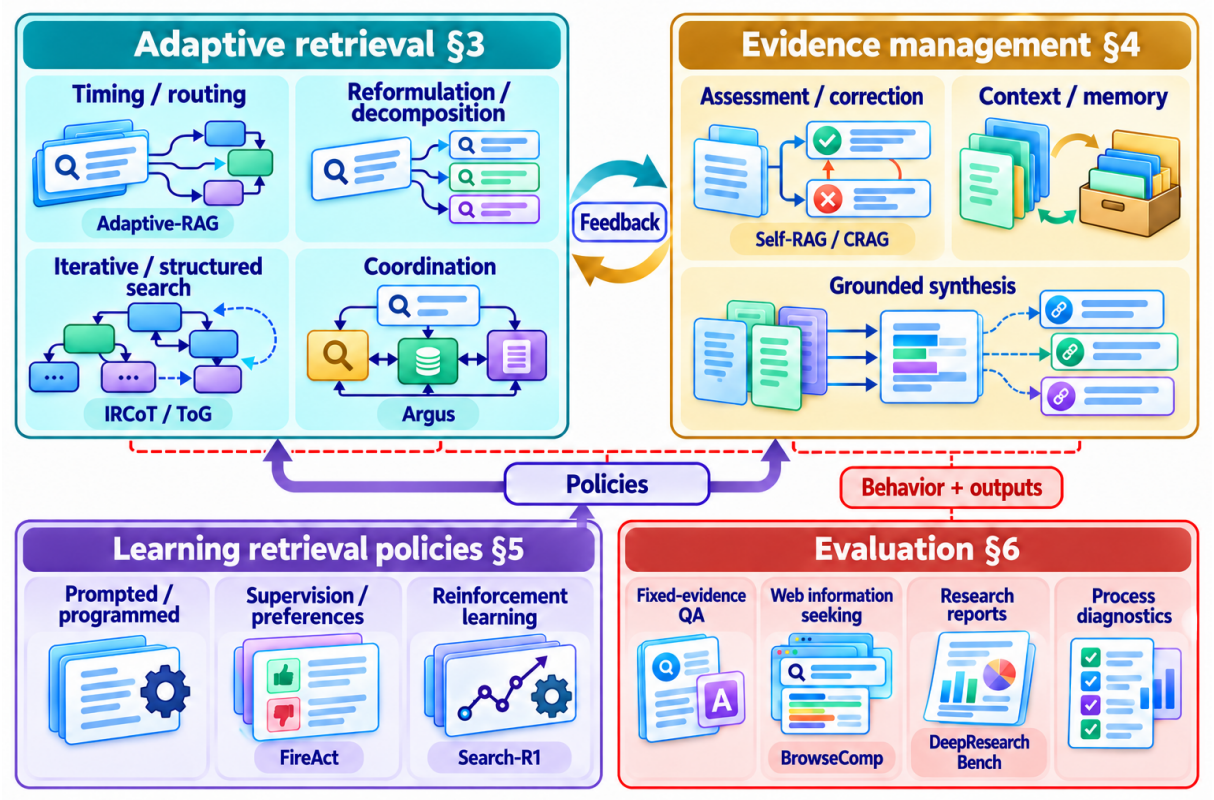


Figure 1: Overview of the survey. Adaptive retrieval (§3) and evidence management (§4) inform each other: what a system retrieves determines what it can claim, and gaps in its evidence prompt further searches. Learning (§5) produces the policies behind these decisions, and evaluation (§6) measures both the system's behavior and its answers. Labels name representative systems and benchmarks [2–11].

Making these decisions well is the central difficulty, for four reasons. (i) *Every decision has a cost.* Each search spends time and tokens, and even deciding whether to search does: a router that skips retrieval can still use more tokens overall than always retrieving, because the decision itself requires a preliminary generation [17]. (ii) *More retrieval is not always better.* Retrieved passages can distract a model from what it already knows [18]. (iii) *Stopping is hard to get right.* To judge when a search is complete, a system needs to know which facts the question requires [19, 20]. (iv) *Feedback is sparse.* Reinforcement learning methods often reward only the final answer [9], leaving the system to infer which searches helped. These pressures also complicate evaluation: because accuracy depends on how much search and synthesis a system is allowed [21], gains reported under different budgets are hard to compare.

Several recent surveys cover related ground: architectures for agentic RAG [22], the interplay of retrieval and reasoning [23, 24], deep research agents [15], and RAG evaluation [25]. These surveys categorize entire systems, making it difficult to compare the individual decisions each system gives the model. We organize the field around those decisions (Figure 1). Our contributions are:

- A decision-centered taxonomy of agentic RAG, covering retrieval timing, query formulation, iterative search, and coordination across searches and tools (§3).
- A review of how systems assess, retain, and synthesize retrieved evidence, from short-answer questions to long research reports (§4).
- A comparison of policy learning (§5) and evaluation (§6) that relates reported gains to model capabilities, evidence access, and computation budgets, and identifies open problems (§7).

2 Foundations

Standard and agentic RAG solve the same problem, answering a question with evidence drawn from an external collection (§2.1), but they solve it differently. A standard system retrieves once and then generates, so it fails whenever the passages an answer needs are not the ones that best match the question. An agentic system instead makes a sequence of decisions about what to retrieve, what to keep, and when to answer, letting what it finds shape what it searches for next (§2.2). Such systems differ mainly in which of those decisions they give the model, how the decisions are made, and where the system acts (§2.3).

2.1 Retrieval-augmented generation

Retrieval-augmented generation separates what a system knows from what its language model memorized during training. Knowledge lives in an external collection of text that can be updated or replaced, and the model's job is to read the relevant parts and answer from them. The standard pipeline has two components: a *retriever* that finds relevant text, and a *generator* that writes the answer.

Formally, let q be a question and $\mathcal{D}$ a corpus of passages. A retriever assigns each passage a relevance score $s(q, d)$ and returns the k highest-scoring passages,

$$C = \underset{d \in \mathcal{D}}{\text{top-}k} \; s(q, d), \tag{1}$$

and a generator, typically a large language model, produces an answer conditioned on both, $y \sim p(y \mid q, C)$ [1]. A good answer is correct and *grounded*: the claims it makes are supported by the passages in C. The same formulation covers open-ended tasks, in which q is a research request and y is a long report whose claims each need support.

This pipeline retrieves once, with the question as its only query. Suppose the top results describe the film and identify its director, but say nothing about his education. Those passages answer "Who directed *Oppenheimer*?" yet leave "Where did the director of *Oppenheimer* go to university?" unsupported:

Question q	Retrieved passages C	Answer y
Who directed *Oppenheimer*?	Passages about the film, naming Christopher Nolan as director	Christopher Nolan (supported)
Where did the director of *Oppenheimer* go to university?	Passages about the film, naming Christopher Nolan as director	Unsupported guess, or no answer

A biography might contain the university without mentioning the film. It may rank below passages that match the film's title, even though the fact it states is needed for the answer. A passage's relevance score against q can therefore differ from its usefulness for answering the question. An intermediate finding can reveal a useful follow-up query that the original question does not state. The pipeline is also inflexible in the other direction, retrieving even when the model already knows the answer and returning the same k passages whether the question needs one fact or dozens.

2.2 Agentic RAG

Agentic RAG addresses the same problem by letting the model gather evidence step by step, in the manner of language agents that interleave reasoning with actions and their observations [13]. We describe this process as a loop over states, actions, and observations. At step t, the system is in a *state* $s_t = (q, h_t, E_t)$, where h_t is the *history* of actions and observations so far and E_t is the

evidence the system currently retains, such as selected passages or extracted facts. A *policy* π, sometimes called the controller, chooses the next action from an action set $\mathcal{A}$:

$$a_t \sim \pi(\cdot \mid s_t), \qquad a_t \in \mathcal{A}. \tag{2}$$

Typical actions include searching with a new query, reading a source, assessing the retained evidence, and answering. Each action returns an *observation* o_t from the environment, such as a list of search results or the text of a page; the history records the pair (a_t, o_t), and the evidence is revised in light of what came back, with passages or facts added, updated, or discarded. The run ends at step T, when the policy chooses to answer and generates y from q and E_T. The full sequence $\tau = (a_1, o_1, \ldots, a_T)$ is the system's *trajectory*.

Standard RAG is the simplest trajectory: one search with q itself, whose k results become the evidence, followed immediately by an answer. In an agentic system each of those choices is open: whether to search, with what query, and when to stop. If the director's identity is initially unknown, a trajectory for the second question might look like this:

t	Action a_t	Observation o_t	Evidence E_{t+1}
1	Search "*Oppenheimer* director"	Passages about the film	Nolan directed *Oppenheimer*
2	Search "Christopher Nolan education"	Biographical passages	+ Nolan studied at University College London
3	Answer "University College London"	—	—

The first search supplies the identity used to formulate the second query.

A good policy reaches a good answer without wasting effort. We capture this with a conceptual objective: the policy should maximize

$$\mathbb{E}_{\tau \sim \pi}\big[\, Q(y, E_T) - \lambda\, c(\tau) \,\big], \tag{3}$$

where Q scores the answer's correctness and grounding in the retained evidence, $c(\tau)$ is the trajectory's cost in searches, tokens, or time, and λ sets the trade-off between them. Each difficulty raised in Section 1 shows up in this objective. Every action adds to $c(\tau)$, so unnecessary searches carry a direct cost; a misleading passage that enters E_T can lower Q; and because the answer action sets T, stopping too early costs quality while stopping too late costs effort. Training with only a final reward poses a related problem. The reward approximates Q for the completed trajectory, so a failed run shows that something went wrong but not which action caused it.

2.3 The design space

The field arrived at this design space gradually (Figure 2). Early systems began interleaving search with reasoning, choosing each query from what earlier ones returned. One line of work then gave the policy more decisions, such as whether to retrieve at all and whether the evidence is adequate. Another changed how the policy is obtained, training it end to end and moving it into larger environments such as the open web. We compare the resulting systems along three dimensions and organize the rest of the survey around them.

Which decisions the policy makes We divide these decisions into two groups. The first governs acquisition, deciding what the system gets to observe: whether to retrieve, through which workflow, and when to stop (§3.1), how to phrase or decompose a query (§3.2), how to continue a search across steps (§3.3), and how to divide the search among several searchers or tools (§3.4). The second begins once observations arrive and governs how the evidence is handled.

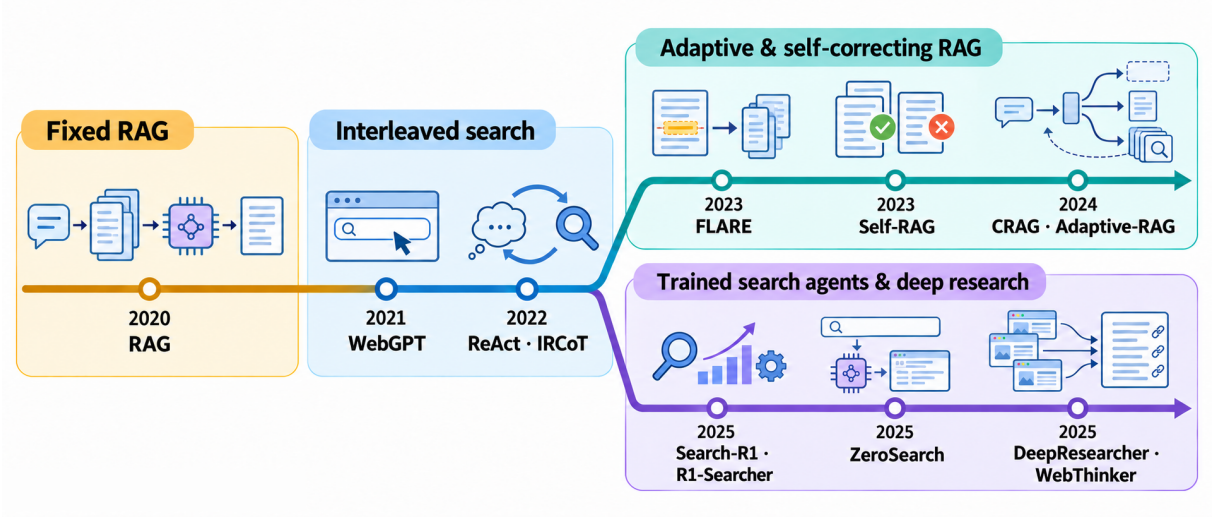


Figure 2: Agentic RAG grew from fixed retrieval through interleaved search and then split into two lines of work: giving the policy more decisions (top; §3 and §4) and training the policy end to end in larger environments (bottom; §5) [1–3, 6, 7, 9, 12, 13, 16, 26–29].

The system must judge whether passages are reliable and sufficient (§4.1), decide what to keep as the evidence grows (§4.2), and write an answer whose claims stay tied to their sources (§4.3). Few systems control all of these; most give one or two to the model and fix the rest.

How the policy is implemented The same loop can be driven in very different ways. The simplest policies are hand-written rules, such as retrieving only when a question concerns a rare entity [18]. More often the policy is a general-purpose language model steered by instructions and a few examples (e.g., ReAct [13]; §5.1), sometimes with a small trained component taking over one decision, such as a classifier that routes questions by complexity (Adaptive-RAG [3]) or a critic that grades retrieved passages (CRAG [7]). Increasingly, the language model itself is trained for the job, first by imitating good trajectories (e.g., FireAct [8]; §5.2) and more recently through reinforcement learning (e.g., Search-R1 [9]; §5.3). Real systems often mix these.

Where the policy acts The environment determines which actions exist and what they return, and therefore what an evaluation is actually measuring. A fixed document collection offers search over a known body of text with repeatable results, and most multihop question-answering benchmarks are built on one (§6.1). A *knowledge graph* instead exposes entities and relations, so actions follow links rather than issuing text queries [4]. The open web offers current sources and page navigation at the price of reproducibility: its contents change, and its searches add latency and cost [16]. Web information-seeking benchmarks (§6.2) and evaluations of long research reports (§6.3) have to work within these limits. Training and deployment environments also often differ, as when a policy learns against a simulated search engine before using a real one [28]. Because results depend on all of these conditions and not only on the policy, some analyses look inside the trajectory to ask what each additional step or resource actually contributes (§6.4).

3 Adaptive retrieval

A search is useful when it finds information needed to answer the question. *Adaptive retrieval* uses the current evidence to decide what to look for next, where to search, and when to stop. Additional searches help when they supply missing facts, but they can also repeat information, add distractions, or follow a mistaken assumption.

Table 1: Actions, decision signals, retained state, and search environments in representative acquisition methods.

Method	Action taken	Signal read	State kept	Environment
Adaptive-RAG [3]	Select no, single, or multi-step retrieval	Question complexity class	Initial workflow routing	Text corpus
FLARE [12]	Retrieve; regenerate next sentence	Low token probability in forecast sentence	Output prefix; temporary next sentence	Text passages
DRAGIN [30]	Retrieve; truncate and resume	Token entropy, attention, stopword mask	Generated context; attention-selected query	Text corpus
Rewrite-Retrieve-Read [31]	Rewrite search request(s)	Original task input	Rewrite–retrieve–read sequence	Web search
IRCoT [2]	Retrieve with latest reasoning sentence	Question, paragraphs, prior reasoning	Single reasoning chain; accumulated paragraphs	Text corpus
ITER-RETGEN [32]	Retrieve with question plus prior output	Previous full generation	Repeated full-answer generation	Text corpus
Search-o1 [33]	Search; refine returned documents	Reasoning prefix and current query	Reasoning chain; refined search observations	Web search
TreeSeeker [34]	Explore, exploit, or prune/return	Evidence, uncertainty, progress, failure cues	Subgoal trees; branch-local memory	Web search/pages
ToG [4]	Select relations/entities; extend paths	Question, KG neighbors, path sufficiency	Beam of connected triple paths	Knowledge graph
PoG [35]	Extend paths; backtrack to entities	Subgoal status and sufficiency reflection	Retrieved subgraph, paths, subgoal memory	Knowledge graph
Argus [5]	Dispatch searches to evidence gaps	Unsupported or contradicted claims; missing coverage	Navigator evidence–claim DAG; Searcher traces	Web search/pages

We organize this section around four decisions: whether to retrieve and where to search (§3.1); how to write or break down a query (§3.2); how to continue a search or revise earlier steps after reading the results (§3.3); and how to divide the work among simultaneous searches or separate searchers (§3.4). Together, these choices determine what information the system finds and how much time and computation it spends. Table 1 compares representative methods by what they do, what information guides their decisions, and what they retain for later steps.

3.1 Retrieval timing and routing

Before answering, a system has only the question and the model's own knowledge to judge whether it needs to search. For example, a model that already knows where *Oppenheimer*'s director studied may answer without retrieval. If it knows only that Nolan directed the film, it can search specifically for his education. Once drafting begins, however, particular claims can expose gaps that were not apparent from the question alone. Search results then let the system check whether those gaps have been filled or whether more evidence is needed. We group methods by these decision points: initial routing, retrieval during generation, and stopping after a search.

Routing from the question A router can save work by reserving multi-step search for questions that need it. Question-level *routers* such as Adaptive-RAG [3] choose among answering without retrieval, searching once, and searching over several steps. In its evaluation, this routing

retains much of the iterative solver's answer quality while taking less time, and it answers multihop questions better than retrieving just once.

Question difficulty alone cannot determine whether to retrieve. Rewriting a supplied passage may require substantial reasoning without outside information, while a simple question about a changing fact may require a search. A router can distinguish these cases using classifiers trained to predict the question's factual requirements, time sensitivity, the user's retrieval intent, and the model's ability to answer. A decision tree then turns those predictions into actions: search when external information is explicitly requested, skip tasks that need no factual knowledge, and otherwise retrieve for changing facts or predicted knowledge gaps (UAR [36]).

Choosing a retriever requires judging what its results will contribute to the answer. Training a router on both document-quality judgments and the correctness of answers generated from those documents gives it evidence for that choice (R3AG [37]). This connects retriever selection to the goal of producing a correct answer.

Triggers during generation Retrieval can instead respond to a particular claim as the model writes. One choice is to search using text already generated; another is to draft the next sentence and use it to identify what needs support. For example, FLARE [12] masks words with low generation probabilities in that draft, searches with the remaining text, and then rewrites the sentence using the results. The draft identifies what the model is about to claim, while masking keeps its uncertain guesses out of the query. Querying with the upcoming sentence works better in its evaluations than querying with the previous one, and the method improves on one-shot and periodic retrieval. The advantage comes from directing the search toward the next claim rather than repeatedly checking what has already been written.

Uncertainty alone does not tell the system whether a search will help. A model may be unsure about a word that contributes little to the answer. DRAGIN [30] extends uncertainty-based triggering by considering a token's importance and influence on later generation, while selecting query terms from the preceding context. Its ablations support both choices. Another approach estimates uncertainty by comparing the model's internal representations across sampled answers (SeaKR [38]). A policy can also choose how to retrieve, including semantic search, keyword search, or reading a whole paragraph (GRASP [39]). These approaches offer different information at different costs. Drafting generates extra text, comparing samples requires several generations, and some model interfaces expose neither token probabilities nor internal representations. A trigger is useful only if better answers or fewer searches justify the computation needed to run it.

Stopping after a search After reading search results, the system must check which facts are still missing. For the *Oppenheimer* question, identifying the director leaves the university unknown. Continuing after the supporting material has been found can also waste effort: trace analysis finds much unproductive search after discovery [40]. Yet a general judgment that the evidence looks sufficient does not reliably identify when to stop. Adding such a judge to a trained Search-R1 policy saves only a few searches, costs some accuracy, and adds computation of its own [19]. A more specific rule checks the evidence against each required step in the answer. HALT [20], for example, stops only once every required hop has support. Its search savings are larger when those requirements are supplied than when it must generate them. Choosing to stop fixes the final step T and the evidence E_T used to answer, so that choice depends on knowing what the question needs, not just judging that the collected passages look useful.

The cost of deciding Searching less does not always mean spending less. A system can probe the model's confidence before deciding to search, avoid some retrieval calls, and still use more tokens overall (BeyondUncertainty [17]). Nor is a useful signal necessarily easy to turn into a reliable decision. A method may correctly rank questions by how much they need retrieval, but it

still needs a cutoff for deciding which ones require a search. A cutoff that keeps searches within budget on one set of questions may fail to do so on another [41]. On RetrievalQA [42], directly asking the model whether it needs to retrieve can fail on questions that require outside facts, while methods that use a threshold depend strongly on its setting. Under explicit cost targets, simple fixed policies remain competitive, while a learned policy trained to reduce cost can do so by declining to answer [43]. Analyses find both unnecessary searches and cases where the model answers without searching when it needs evidence [44]. Evaluating a retrieval decision therefore requires counting the full trajectory cost $c(\tau)$, including the work needed to decide whether to search, and checking the quality of the resulting answer.

3.2 Query reformulation and decomposition

The user's question q and a useful search query do not always use the same words. Compare "Where did Christopher Nolan go to university?" with "Where did the director of *Oppenheimer* go to university?" Both request the same fact, but only the first names the person. It can be rewritten directly as "Christopher Nolan education." For the second question, that rewrite also requires identifying the director. We group methods by where they get the information used to build a query: (i) the original question, (ii) context already available in the task, and (iii) earlier search results. A system can also break a question into smaller questions, but it must still determine which of them can be searched immediately and which need an earlier answer.

Using the question Rewriting and expansion address different problems in matching a question to a passage. *Rewriting* selects keywords, removes conversational wording, or tries several ways of asking for the same fact. A learned rewriter can improve search while leaving the retriever and answer model fixed; the answer model still receives the original question [31]. Searching with several selected rewrites alongside the original query also avoids relying on a single formulation (DMQR-RAG [45]). *Expansion* adds terms that the question does not contain. Methods such as Query2doc [46] generate a short document that might answer the question and append it to the query to improve matching. This can suggest useful terms, but it also gives the retriever model-generated content to match against. If the expansion names Nolan as the director, the system is relying on a prediction until a source confirms it.

A rewriter can be trained using feedback from different parts of the retrieval pipeline: (i) how a reranker scores the retrieved passages, (ii) a combination of passage, ranking, and answer scores, or (iii) how well the answer model answers using the results [31, 47, 48]. Each signal favors a different property, so the resulting queries can differ. The effects can also continue beyond the first search. Better intermediate retrieval is associated with better final answers, and stronger retrievers can reduce the number of steps needed [49]. Conversely, changing a question's wording without changing its meaning can still alter later decisions about when to search and what to ask [50]. A rewrite therefore needs to be assessed through the evidence it retrieves and the searches that follow, as well as through how faithfully it expresses the original question.

Using task context Context can supply an identity without another search. If earlier dialogue has already named Nolan, "where did he study?" can become "Christopher Nolan education." In IterCQR [51], dialogue-based rewriting learns to recover that information from earlier turns. For semi-structured collections, documents and knowledge-graph relations can likewise supply context for expanding a query [52]. The common requirement is to identify the person or relation the user means. Adding context helps when it resolves that identity; more text alone can leave the query just as ambiguous.

Using retrieved observations A source can resolve an identity that neither the question nor its context supplies. The resulting query then rests on retrieved evidence, whereas expansion from the question alone may rely on a predicted identity. EviReform [53] combines such follow-ups with a separate search using the original question, so retrieval does not depend entirely on one reformulation. This preserves the broader request while making the missing fact easier to search for. However, the earlier source must have identified the right person: a more specific query can otherwise take the search further in the wrong direction.

Decomposing an information need Breaking a question into parts makes its requirements clearer, but does not make every part immediately searchable. The education subquestion can be planned before the director is known, but a query naming that person must wait for the identity. Retrieving for subquestions and then reranking their combined results improves over standard RAG on multihop tasks [54]. However, a passage about Nolan's education may match the education subquestion well while matching the original question poorly. The system must preserve such passages when combining the results.

How the subquestions are written also matters. A diagnostic study finds that evidence needed for later steps is identified best using reference subquestions that explicitly name the entities identified in earlier answers. Follow-ups generated from the original question alone do not help. Using the top retrieved passage to generate a follow-up helps mainly on two-hop tasks, and mainly when that passage is correct [55]. Decomposition works best when the system uses earlier answers to make later questions specific enough to search.

3.3 Iterative and structured search

After a search returns, the system must decide how to use the results. It can continue its current reasoning, rewrite a draft answer, reconsider an earlier conclusion, or try another search direction. These options matter when an early step is wrong. Suppose an early step identifies the wrong director of *Oppenheimer*. Later evidence may correct the identity, but the system must also reconsider conclusions drawn from searches about the misidentified person. We compare methods by how much a new observation o_t can change, from the next action a_{t+1} to an earlier conclusion that guides the search. Figure 3 shows how several designs retain earlier reasoning and evidence so that the system can decide what to do next.

Continuing a single chain Alternating reasoning with retrieval allows each result to guide the next search without planning every step in advance. In IRCoT [2], the latest reasoning sentence becomes the query, the gathered passages inform the next sentence, and a separate answer model reads the collected passages at the end. This reaches evidence that repeated searches with the original question can miss, but earlier reasoning sentences remain unchanged. How the results enter the chain is a separate choice: extracting information relevant to the current step works better than inserting raw retrieved pages (Search-o1 [33]). Both choices help the system use a result to continue reasoning. Neither, on its own, provides a way to reconsider the earlier inference that sent the search toward the wrong information.

Revising an answer or inference Revision can operate on a whole answer or on individual reasoning steps. Rewriting the complete answer lets a draft guide the next search without committing the system to its earlier wording. The question and previous answer can be used together as a query, followed by a fresh answer from the returned passages [32]. In that evaluation, two rounds match or improve model-judged accuracy over Self-Ask while using fewer model calls and passages. The draft can therefore reveal several facts worth checking without requiring a separate follow-up question for each one.

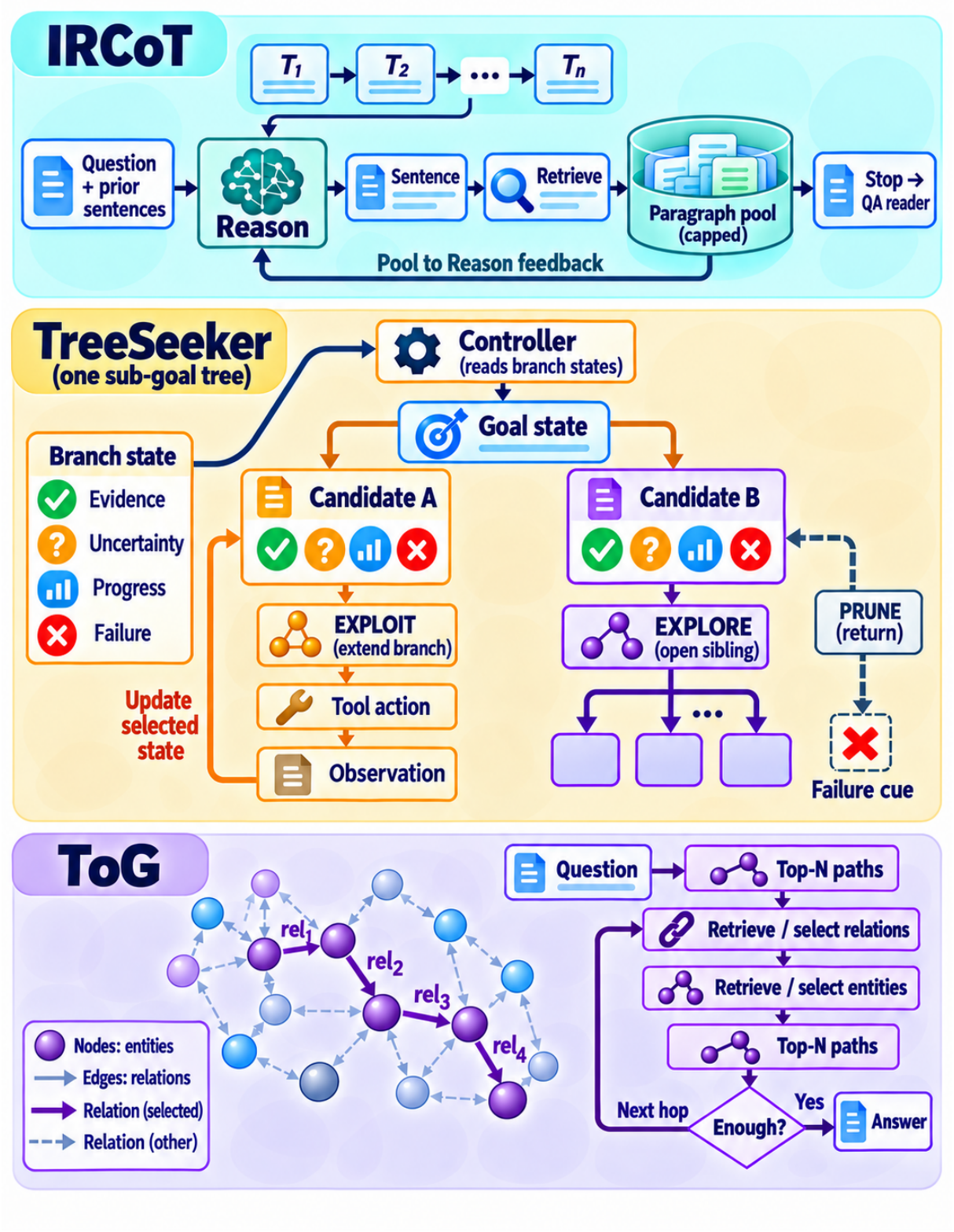


Figure 3: State and feedback in interleaved retrieval, branch-and-return search, and knowledge-graph traversal [2, 4, 34].

Other methods keep individual reasoning steps open to correction. Query–answer pairs can be verified against retrieved evidence and expanded into a reasoning tree [56]. Separating retrieved facts from the system's own inferences also lets later evidence change an earlier conclusion [57]. Whole-answer revision can change a broad account at once; step-level revision helps locate and repair the particular inference that sent the search astray.

Exploring alternative branches Keeping several search directions open provides another way to recover from an early mistake. The system can continue a promising direction, start an alternative, or return to an earlier choice when a branch fails. TreeSeeker [34] makes these choices using branch-specific evidence, uncertainty, progress, and signs of failure. Summaries retain what each branch found and why an unproductive direction was abandoned, while subgoals wait for any earlier answers they need. Removing exploration and pruning reduces success in its deep-search evaluations. Alternatives can also be selected through Monte Carlo Tree Search, with retrieval used to check and refine candidate reasoning steps [58]. Both designs spend effort on possibilities that a single chain would ignore. That effort is useful when checking alternatives prevents a plausible early mistake from controlling the rest of the search.

Searching structured sources A knowledge graph supplies possible search steps through relations between entities. One approach keeps several candidate paths and extends them by choosing relations and neighboring entities (Think-on-Graph [4]). This retains alternatives, but the current paths may still fail to supply a needed fact. PoG [35] extends path search by tracking subgoals and remembering the part of the graph already explored. It adjusts how many paths to explore and can return to earlier entities when the current paths are insufficient. Compared with the basic beam search, these additions improve answers while reducing model calls, token use, and latency. Retaining more search history can therefore save work when it helps the system return to a useful earlier choice instead of continuing an unsuccessful path.

Structure can also guide what the system reads before it begins exploring. The structure can specify relation sequences to find in a graph [59], ways to read tables and databases [60], or a reasoning tree for the question [61]. The benefit depends on whether that organization exposes useful evidence. In graph-based RAG, repeated retrieval helps most on questions needing a connecting fact when it brings the connecting documents to the front of the context; simply collecting more candidates adds noise and offers little benefit on simpler questions [62]. Extra retrieval helps when it finds missing evidence or makes that evidence easier to use. Enlarging the candidate pool alone may only add noise.

3.4 Coordinated retrieval

Some searches must wait for an earlier answer, while others can run at the same time. Consider "Compare the educational backgrounds of the directors of *Oppenheimer* and *Dune* (2021)." Evidence for the two directors can be gathered independently. Within each investigation, a search for the director's identity may be needed before a query naming that person can be written. Breaking down a question can reveal independent searches [54], while using earlier results to write follow-ups allows each investigation to continue [53]. We distinguish three arrangements: (i) one controller issuing several searches at once, (ii) separate searchers choosing their own search steps, and (iii) many workers using a shared record to coordinate their work. Running calls at the same time can reduce waiting; giving a task to a separate searcher also gives that searcher responsibility for deciding how to pursue it.

Concurrent calls from one policy Several search calls can run at once without giving each one a separate agent. A single controller writes the queries and reads their combined results

before deciding what to do next. This reduces sequential turns when the question contains independent subquestions (ParallelSearch [63]). Alternating batches with reasoning also lets one round of results determine the next batch [64]. Both arrangements keep planning with one controller; they differ in how much work can be prepared before results arrive. If a query needs a fact from another search, it must wait for that result and run in a later batch.

Delegating a search trajectory *Delegation* moves the choice of individual search steps to a separate searcher. The coordinator gives it a task, receives findings, and decides what still needs investigation. An evidence graph can make that last decision specific by linking claims to their support. For example, Argus [5] combines independent searchers' findings in such a graph and sends follow-up requests to claims that lack support or have conflicting evidence. Its searchers do not see the graph or one another, and they keep no state between requests. The coordinator therefore has to retain and compare their findings; delegation does not remove the need for a shared account of what the search has established.

The comparison with a standalone searcher also separates some of the benefits of coordination from those of concurrency. A coordinator with one searcher already improves results, and adding parallel searchers improves them further with more computation. Evidence tracking, targeted follow-ups, and concurrent execution all change in this arrangement. Its gains therefore support the combined design, but do not establish how much improvement comes from adding workers alone.

Maintaining shared progress With many searches underway, evidence and unfinished tasks must remain available to guide later work. A managed evidence store can identify what needs a follow-up [65], while nested delegation lets a large task be divided and replanned as results arrive [66]. Shared records of what is known and what still needs to be found can also guide collaborating or competing agents [67]. Keeping progress outside individual searchers serves two purposes: later requests can build on completed findings, and the coordinator can see which needs remain unanswered.

The same progress record can support different schedules. Waiting for a batch to finish makes every next assignment depend on its slowest worker. SearchOS [68] instead exposes evidence, answered and unanswered requirements, unfinished tasks, and past failures to a scheduling layer. That layer detects stalled work and assigns new tasks as workers become free, reducing delays compared with synchronized batches. The benefit here comes from using recorded progress to keep workers busy, instead of waiting for every task in a batch to finish.

Assigning different capabilities When searchers have access to different collections, choosing where to send a request becomes part of retrieval. Broadcasting every question to every specialist is one option; selecting specialists from summaries of their collections avoids requests to sources unlikely to help. The latter approach improves accuracy and latency in private-collection retrieval, with partial replies used to refine the unanswered request [69]. Roles can also differ by responsibility or tool access: a searcher supplies evidence while an answer writer requests further support [70], or agents use different tools and debate to guide their queries [71]. The useful division depends on whether the workers offer different access or make different decisions, rather than on their number alone.

Coordination also changes how existing evidence is used. Reusing a source across related claims avoids repeated searches [72], while verification over long multimodal documents can direct effort toward either retrieving more pages or reasoning further over what is already available [73]. These choices can improve an arrangement without increasing worker count. Taken together, the methods give several reasons to coordinate: different source access, stored evidence, targeted follow-ups, and less waiting. Establishing when separate searchers themselves justify their

overhead still requires comparison with one controller given the same total effort.

4 Evidence management

Finding a relevant passage does not guarantee a supported answer. The system must check what each source establishes (§4.1), keep useful evidence available as the search grows (§4.2), and connect the final claims to their sources (§4.3). These tasks continue after retrieval succeeds. A source may contradict an earlier finding, a summary may drop an important qualification, or a report may omit a required comparison even though each sentence it includes has support. Retrieval supplies an observation o_t, while evidence management determines how it changes the retained evidence E_{t+1} and contributes to the answer.

4.1 Evidence assessment and correction

When a passage leaves a claim unsupported, the system can seek more evidence or revise the claim. Suppose the draft answer says Nolan graduated from a university, but a biography says only that he "studied at" it. That wording supports attendance, but does not establish graduation. Finding another source that explicitly says he graduated would supply support for the stronger claim, but the source could still be mistaken. Resolving doubts about its accuracy requires checking its account against other evidence.

Passage assessment and generation critique Assessment can change either what the model writes or what sources it receives. Self-RAG [6] illustrates the first choice. As it generates answer segments, it emits special tokens judging passage relevance, claim support, and response utility. These judgments are learned from annotations supplied by a dedicated critic, and their weights affect which continuation is selected. Increasing the support weight favors better-supported segments. CRAG [7] instead evaluates retrieval before answering and changes the material passed to the answer model. It refines retrieved documents when the evaluator judges them correct, replaces them with web results when they appear incorrect, and uses both when the judgment is ambiguous. The labels describe estimated retrieval quality, rather than whether the final answer is true. Its evaluation improves on the Self-RAG model used as a baseline, and removing the ambiguous branch reduces accuracy. Scoring an answer and correcting its source set can therefore address different failures within the same system.

Filtering can also happen while documents are read, through explicit relevance assessment [74], sequential reading notes [75], or refinement of retrieved knowledge during reasoning [76]. Multimodal filtering can select candidates while preserving the raw inputs for final answering [77]. These choices change what reaches the model and in what form. However, filtering only for similarity to the original question can remove a passage needed for a later step [55]. Useful filtering must retain facts that connect the question to its answer, as well as passages that directly resemble the question.

Disagreement and falsification A system can compare conflicting accounts already available, or search deliberately for evidence against its draft. The first approach can contrast the model's own account with refined external evidence and produce a report of their disagreements [78]. Conflict reconciliation adds a small improvement beyond evidence refinement in that evaluation. FVA-RAG [79] takes the second approach: it treats the draft as a hypothesis, writes queries seeking contrary evidence, and uses a contradiction verifier to decide whether to revise it. This matters when a question contains a false premise that ordinary searches may repeat. On its fixed-corpus misconception tasks, falsification improves over prompted Self-RAG and CRAG implementations with matched retrieval and generation budgets. Reconciling sources can resolve

an observed disagreement; seeking counter-evidence can expose a mistake that the current sources have not challenged.

Explicit gaps and corrective feedback Feedback is most useful when it identifies what the system needs to do next. A checklist of required findings can separate established facts from missing support and send the gaps back to query refinement (FAIR-RAG [80]). Checking whether the reasoning follows from retrieved facts instead targets conclusions that need correction even when relevant evidence is present [81]. Diagnostics confirm that these failures are distinct: scientific multihop tasks in ChemKGMultiHopQA contain both missing-evidence cases and failures to combine sufficient evidence [82], while SeekBench's search-trace annotations distinguish grounding, recovery, and calibration errors [83]. In the checklist-based evaluation, repeated retrieval improves over one iteration, although the strongest system also changes how generation models are allocated. The comparison supports following up on explicit gaps, but does not assign the full improvement to the checklist alone. A single confidence score cannot tell the system whether to search again or reconsider an inference.

4.2 Context and memory

The model may see only part of what the system has stored (its *active context*). The history h_t and retained evidence E_t can extend beyond what a single model call receives. An external record can retain useful findings after they leave the active context. Recording unsuccessful directions alongside supporting facts helps later searches avoid repeating them [84], while a changing pool of evidence can guide both further retrieval and the decision to stop [85]. If a memory note keeps only the university's name, the answer model may no longer know whether the source established attendance or graduation. Retaining a link to the passage allows that distinction to be checked without keeping the full biography in context.

Compact state and source access A compact memory can serve different purposes. It may provide an overview that helps locate information in a long input (MemoRAG [86]), collect support and conflict for a particular answer candidate [87], or keep textual and graph-based memories updated together [88]. These representations preserve different things. An overview helps decide where to search, whereas a record tied to an answer helps check its claims. When a fact changes, the system must also identify which summaries and inferences still depend on the old version.

For example, ReTree [89] records where each source-linked fact entered a search tree. Each node contains a compact summary, facts introduced there, and a revision history; the model sees the path leading to its current node. If a new finding contradicts an earlier fact, the system first checks that the conflict concerns the same scope and time and that the new evidence is better supported. It then returns to the node that introduced the old fact, replaces that fact and source, regenerates the summary, and removes all descendant nodes before continuing. Compatible refinements and alternative candidates instead extend the current path. The check before replacement prevents a new mention from automatically overwriting earlier evidence.

This approach improves on replacing a fact in place, because later reasoning may still contain consequences of the old fact. Removing all descendants repairs that risk, but can also discard useful information that did not depend on the error. A bounded active context also does not bound the full storage cost: sources and passage mappings can continue to grow. Stable evidence identifiers connect final claims to the current fact, its source URL, and the original passage. The compact summary supports current reasoning, while those links allow a claim's source to be inspected after correction.

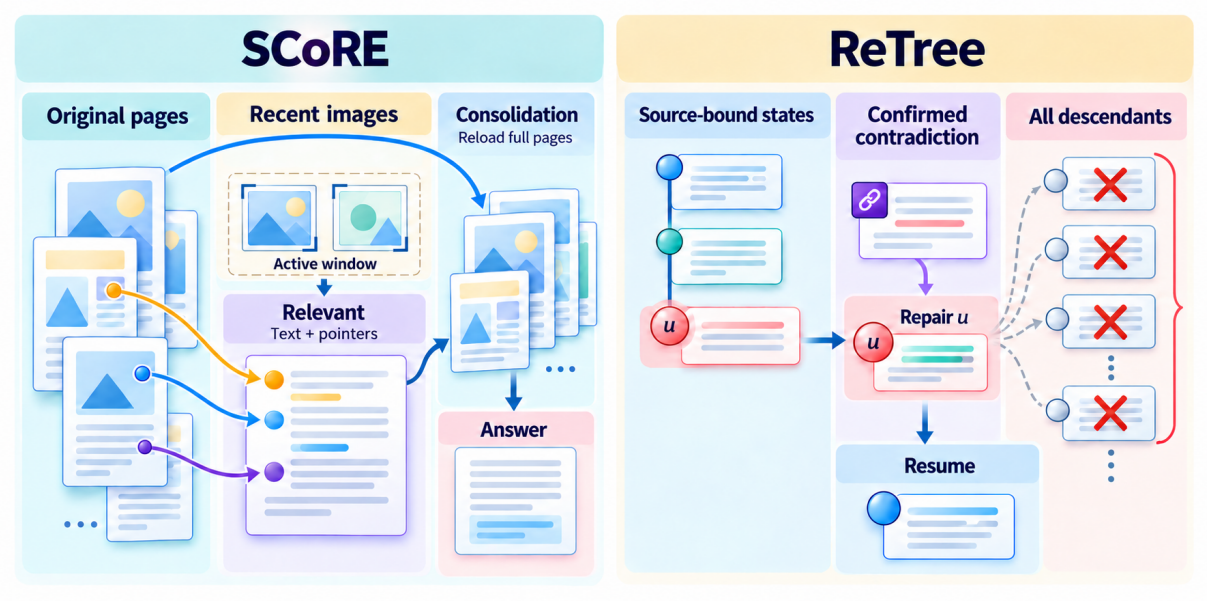


Figure 4: Source retention in SCoRE and state repair in ReTree, with separate evidence paths [89, 90].

Visual retention and reconstruction Visual evidence presents a further choice: store the relevant visual content, or preserve a way to load it again. SCoRE [90] illustrates the latter. It records selected textual observations with pointers to their original pages and, for image crops, bounding boxes. Recent visual inputs remain in a bounded window. Before final answering, the system reloads the referenced full-page images and organizes them into a consolidated context; claims identify the observations that support them. A visual observation excluded from the record disappears as the window advances. Selection therefore determines which original pages can be recovered for the answer.

VimRAG [91], by contrast, retains selected visual tokens in a multimodal memory graph and allocates their resolution according to node importance. Its comparisons favor keeping visual information over purely textual summaries. The difference is where access is preserved: source pointers allow images to be reconstructed later, while visual memory keeps selected content in stored state. Page-relevance estimates updated across exploration rounds provide another kind of persistent record [92]. Figure 4 contrasts visual source recovery with the correction of dependent reasoning state.

A system can also change what it shows the model without deleting its stored history. It can retrieve a query-specific subset of structured memories [93], or hide superseded values for current-state questions while retaining access to past values [94]. Reusing memory across a sequence of tasks extends this idea beyond the current request, with benefits depending on recurring task structure [95]. Memory for completing one search preserves its sources; memory across tasks must also determine which earlier experience applies to a new request.

4.3 Grounded synthesis

Writing an answer requires deciding which claims to include and whether the evidence supports them together. A response to the *Oppenheimer–Dune* comparison can accurately describe Nolan's education while omitting the other director. Even when both accounts are supported, the answer must bring them together to make the requested comparison. Checking the claims in y against the final evidence E_T is necessary, but it cannot establish that the answer has combined its facts correctly or addressed the full request.

Verified attribution and selective completion Citation verification checks whether a source supports the statement attached to it. A real paper or a passage on the right topic may still fail this test. In VeriCite [96], verification compares draft statements with extracted evidence, keeps supported statements, and attaches their sources; academic citation can also be planned around the intended claim and corrected when retrieved candidates fail to support it [97]. The first approach improves citation quality across the generation models evaluated, but its multihop answer improvements vary by model. Verifying individual fragments does not ensure that the model combines them into a correct answer.

Evidence sufficiency also depends on how support is combined. Taking the highest passage score can accept an answer when only one required step is supported. Averaging scores can hide a refuting passage among many neutral ones. SURE-RAG [98] instead aggregates passage-level support, refutation, and neutrality into answer-level coverage, disagreement, and uncertainty before choosing whether to answer or abstain. Missing evidence and contrary evidence remain separate outcomes, even when both prevent an answer. Keeping that distinction helps determine whether the system needs another source or must revise its current conclusion.

Combining verified support is necessary for biomedical findings with explicit citations [99] and for answers drawn from multiple reasoning branches [61]. Completion rules vary in how specifically they describe the required support. A policy can choose broadly between a grounded response and rejection [100], or require evidence for every expected step in a multihop answer (HALT [20]). In the latter evaluation, generated expected claims save fewer searches than reference supporting-fact claims. Checking every requirement helps only if those requirements correctly describe what the answer needs.

Incremental drafting and evolving outlines Research and writing can inform one another before the final answer is complete. Drafting, checking, and editing can be actions available during search, allowing a report to remain in place while the system investigates a particular gap (WebThinker [29]). In this arrangement, the main reasoning model directs the edits and an assistant executes them using the current report and stored documents. An outline offers another place for evidence to change the planned answer before prose is finalized.

WebWeaver [101], for instance, develops search and a citation-grounded outline together, then writes sections using retrieval from its memory bank. Sources leave active context after each section but remain stored. This improves citation accuracy and report insight over giving the full bank to a single writer in its ablation. AgentCPM-Report [102] extends outline revision into writing: after a drafter produces grounded content, a deepener adds subsections where arguments remain shallow or unresolved. The first design lets evidence revise the plan before section writing; the second also lets the written argument expose gaps in that plan. Both make organization responsive to findings rather than fixing it entirely from the initial request.

Claim-linked evidence integration An intermediate record of claims and sources can expose omissions before the report is written. DeepWeaver [103] uses blocks that connect claims and salient information to source fragments. It inspects unused fragments, creates additional blocks, merges them into the main structure, and generates the report block by block. Compared with generation directly from the full source pool, this preserves more fine-grained detail and covers more key arguments in its domain evaluation. Outline-guided refinement and citation relations offer another way to organize scientific synthesis [104]. Such records make a missing claim, unused fragment, or absent section available for a targeted edit. They also preserve a path from the finished prose back to the source text that supports it.

Table 2: Policy targets, supervision, feedback, experience environments, and transfer evaluations across retrieval training methods.

Method	Policy actions	Initialization / supervision	Feedback	Training environment	Transfer evaluated
WebGPT [26]	Search, navigate, quote, answer	GPT-3; human browser demonstrations	Human preference model; PPO	Live Bing browser	TruthfulQA questions
FireAct [8]	Reason, search, reflect, answer	GPT / Llama / CodeLlama; GPT-4 trajectories	Successful-trajectory imitation	Google snippets	HotpotQA → Bamboogle
Self-RAG [6]	Retrieve, critique, generate	Llama 2; GPT-4-distilled critic labels	Reflection-token and output likelihood	Offline retrieved passages	Zero-shot tasks; web and GTR inputs
Search-R1 [9]	Reason, query, answer	Qwen2.5 base / instruct; no trajectory SFT	Answer EM; PPO / GRPO	Wikipedia; E5	Unseen QA datasets
R1-Searcher [27]	Reason, query, answer	Qwen2.5 / Llama 3.1; no trajectory SFT	Search / format, then answer F1	Wikipedia; BGE	Unseen QA; Google with summaries
ReSearch [105]	Reason, query, answer	Qwen2.5 base / instruct; no trajectory SFT	Answer F1; format fallback; GRPO	Wikipedia; E5	MuSiQue → other multihop QA
ZeroSearch [28]	Reason, query, answer	Qwen2.5 / Llama 3.2; tuned simulator	Answer F1; REINFORCE / PPO / GRPO	LLM-generated documents	Simulator → Google; unseen QA
DeepResearcher [16]	Reason, search, browse, answer	Qwen2.5-Instruct; direct RL	Answer F1; format penalty; GRPO	Live web; reading agent	MuSiQue, Bamboogle, PopQA
ProRAG [106]	Reason, subquery, subanswer	Qwen3; teacher SFT; PRM-filtered SFT	Learned step reward + answer F1	Wikipedia; E5	PopQA, 2Wiki, Bamboogle
ASearcher [107]	Search, browse, summarize, answer	Qwen2.5 base / QwQ; direct RL	Format × F1 / answer judge; GRPO	Local Wikipedia / live web	Local → web; external summarizer

5 Learning retrieval policies

A system's search behavior can come from written rules and prompts (§5.1), demonstrations or preference feedback (§5.2), or rewards for its own attempts (§5.3). Each provides different guidance. Rules state what to do, *demonstrations* show a sequence of actions, and rewards tell the system which outcomes or intermediate steps were useful. We organize policy learning by these sources of behavior. The training target also matters: a router learns a search decision, a critic learns to assess evidence or actions, and a language-model policy learns to generate the interaction itself. Table 2 compares what is trained, what feedback it receives, and which changes in questions, tools, or observations are tested after training.

5.1 Prompted and programmed policies

A requirement can be written into the environment or left for a prompted model to follow. Rules for choosing documents, databases, or both can be refined from feedback while remaining available for inspection [108]. An execution rule can instead reject a particular action. Read-Gate [109], for example, returns a corrective observation when the controller tries to finish without reading a source. Its gains concentrate on runs that originally skipped reading; errors after reading reference evidence remain. This check requires the environment to record searching, reading, and

finishing as separate actions, but does not require a newly trained evaluator. Refining a routing rule changes which source the policy chooses; enforcing a read requirement changes which actions the environment accepts.

Demonstrations and tool design A prompt can show the model how actions should depend on earlier results, as in ReAct [13], while the available tools determine which decisions it can make. In A-RAG [110], keyword search, semantic search, and chunk reading offer different ways to inspect a collection. The model still has to choose effectively among them using capabilities it already possesses. Graph-navigation comparisons find that stronger tool-calling models locate scattered evidence more effectively than heuristic traversal, but gain less when the evidence is concentrated [111]. That flexibility also adds query-time model calls and tokens. Prompted control is most useful when the task needs choices that a fixed traversal rule cannot make cheaply.

Execution constraints and training prompts Resource controls can remain outside the policy's weights. At inference time, they can divide effort between search and answer revision [112], or state the remaining budget in prompts used to evaluate search and completion decisions [21]. These controls can accompany a trained retrieval policy as well as a prompted model. For example, explicit goal statements and reflection can themselves be trained with reinforcement learning [113]. Written controls and learned behavior can therefore contribute to the same trajectory.

Training also remains sensitive to how the prompt asks the model to act. A template requiring extensive intermediate reasoning can encourage repeated reasoning tags and unstable learning compared with direct search-and-answer decisions [114]. The prompt specifies the action pattern that training must reinforce. Adding more instructions does not necessarily give the policy more useful control if learning becomes dominated by producing the required form.

5.2 Supervision and preference optimization

Demonstrations show actions to imitate. *Preferences* compare alternatives and identify which one is better. The information available to either form of supervision depends on who produced it, what sources they saw, and whether the training examples include recovery from failed actions. A policy trained only on successful searches sees how a task can be completed, but not necessarily why another plausible search should be rejected.

Imitating trajectories and assessments Human browsing demonstrations can teach commands, while comparisons of completed answers can train a preference model, as in WebGPT [26]. The preference model can then guide policy training through proximal policy optimization (PPO), or select among sampled answers at inference time. The stronger rejection-sampling results use extra inference computation. The same feedback thus improves behavior at different stages, with different costs.

Model-generated demonstrations offer a more scalable source of complete trajectories. FireAct [8] filters successful trajectories from GPT-4 and fine-tunes smaller models on examples of reasoning, search, and reflection. Its smaller Llama models answer more accurately than few-shot ReAct prompting, while its GPT-3.5 agent takes substantially less inference time. It also improves over prompted GPT-3.5 on Bamboogle after training on HotpotQA with the same Google snippet interface; broader task experiments support adding multitask data. These comparisons show that training can absorb behavior otherwise carried by long prompt examples. They test transfer across questions and tasks while keeping the search interface unchanged.

Assessments can be imitated as well as actions. Self-RAG [6] uses GPT-4 labels to train a critic, which then annotates instruction–output examples for training the model's response and

assessment tokens. During data construction, retrieval and judging can inspect target answers and response segments that the deployed model will not have in advance. Training improves citation precision over retrieval-augmented Llama2-chat, and the trained model emits its own assessments at runtime. This remains supervised learning even when decoding later uses those assessments as scores. Some evaluation task families, including ASQA, are represented in training. A zero-shot evaluation prompt therefore does not imply that the model has received no relevant supervision.

Generating tasks also changes what a policy can learn. Synthetic web tasks can supply trajectories for supervised initialization before reinforcement learning (WebDancer [115]), or difficult information-seeking problems whose successful attempts are selected before policy optimization (WebSailor [116]). Such data do more than reproduce a teacher's search style. They determine which kinds of difficulty the initial policy encounters and which problems later exploration must solve.

Privileged information and self-distillation A teacher can provide targeted guidance using information unavailable at the student's current step. This can teach reflection and backtracking from a global view of the task [117], or convey plans and supporting facts through a multi-agent protocol [118]. An external teacher is not essential. SD-Search [119] uses the policy's own completed rollouts to give a teacher access to hindsight, then trains the policy to match the teacher's query-token distributions alongside outcome-based reinforcement learning. This improves over its outcome-reward baseline. However, feedback-conditioned self-distillation can also collapse into search templates unrelated to the question, even when the teacher answers better [120]. A fixed or slowly updated teacher stabilizes that study's training. Better-informed decisions help only if training transfers them without losing their dependence on the question.

Preferences over decisions Preferences can target different parts of the loop. At the first action, answering the same question with and without forced search supplies a direct comparison for routing [44]. Preference optimization improves on supervised routing, correcting either excessive or missed searches depending on the base model. Cases where both alternatives fail are excluded, so the labels teach which available solver to choose, rather than how to repair either solver.

Preferences over generated answers teach a different decision: whether to rely on retrieved information. Comparing answers with and without retrieval improves standard RAG in a single generation pass (RPO [121]), providing a learned evidence-use baseline without sequential search. Intermediate-action preferences instead compare candidate queries or other search decisions. RAG-Gym [122] uses model-annotated comparisons to train policies and critics at those points. Unlike a successful trajectory alone, these comparisons expose actions that were considered and rejected. The value of preference data therefore depends on where the comparison is made, not simply on whether preferences are used.

5.3 Reinforcement learning for retrieval

Reinforcement learning allows a policy to try searches that no demonstrator provided. A *final-answer reward* can score a whole trajectory τ using a reference answer, but does not identify which action a_t contributed to the result. *Intermediate rewards* attempt to supply that feedback before the run ends. Both rely on the actions the model can produce and the observations the environment returns. A reward may encourage useful evidence gathering, but it can also reward a correct answer reached without using the search at all. A reward based only on answer matching does not check the grounding component of $Q(y, E_T)$.

Outcome rewards and initialization Final-answer rewards can train search-interleaved generation without an initial stage of supervised trajectory imitation. Search-R1 [9] uses exact answer matching with PPO by default and also tests group relative policy optimization (GRPO), whereas ReSearch [105] uses answer F1 with GRPO and a fallback format reward when F1 is zero. Both still depend on reference answers for reward and on pretrained or instruction-tuned capabilities. R1-Searcher [27] separates learning to invoke search from learning to use it well: early rewards encourage search calls and valid formatting, then later rewards remove the invocation bonus and focus on answer F1 with a format penalty. Avoiding trajectory demonstrations does not remove the need for an initialization and reward that make useful actions learnable.

The trained component can also be narrower than the whole trajectory. A query rewriter can learn from a fixed answer model's feedback [31], while a full policy learns the generated search-and-answer sequence [9, 105]. These targets assign responsibility differently: the rewriter changes the request, while the full policy can also change when to search and what to answer. Held-out question evaluations show reuse across datasets with the original retrieval engine. A MuSiQue-trained ReSearch policy also improves over prompted IRCoT on other multihop datasets in the same Wikipedia environment. These results establish transfer across questions while the source of search feedback remains unchanged.

Generated actions and observations A retrieved passage is an observation o_t returned by the environment, rather than an action a_t generated by the policy. *Observation masking* preserves that distinction by excluding observation tokens from the policy loss while keeping them in context for later decisions [9, 27, 105]. Observation masking is also used in supervised trajectory training [115]. Queries, reasoning, and answers remain training targets. Masking teaches the policy to respond to a document without treating reproduction of that document as one of its actions.

Intermediate credit and evidence faithfulness A correct final answer does not reveal which search was useful. The model may already know the answer, so judging a query by the correctness of later completions can give it credit for evidence that contributed nothing. In RAG-Gym [122], such rollout-based process labels perform worse than outcome-only training, while model-based process supervision improves answering and retrieval recall. Another approach uses tree search to produce supervision for a learned step-reward model. ProRAG [106] uses that model to filter supervised refinement data, then trains with both step-level and final-answer feedback. Its ablations favor reward-guided refinement and the combination of local and trajectory-level advantages. Tree search supplies training supervision here; it is not required for every deployed answer. Intermediate feedback can help, but the procedure used to identify a useful step matters as much as how often the policy receives a score.

Step rewards also differ in what they call useful. They can reward whether search is needed at a reasoning step, with the bonus conditional on a correct answer and valid format (HiPRAG [123]), or reward whether reasoning can be traced to evidence and the answer to that reasoning [124]. The latter, evaluated by VERITAS, improves faithfulness measures over Search-R1. Merely rewarding documents that contain the reference answer string provides little benefit and sometimes reduces accuracy in a controlled study [125]. Finding that string does not establish that the document supports the reasoning. These comparisons favor checking the contribution or use of evidence over assuming that an answer-bearing retrieval was a helpful action.

The reward also interacts with the action format and optimization algorithm. In the same controlled study, format rewards help base models learn usable calls, while an initialization distilled for reasoning can struggle with tool syntax [125]. Search-R1++ [114] identifies a different failure: F1-based training can collapse into avoiding answers, and action penalties stabilize its tested recipe. Its comparisons favor REINFORCE over PPO and GRPO under those prompts

and rewards. PPO nevertheless succeeds in the Search-R1 setup. The results support choices within particular training conditions, rather than a ranking of algorithms independent of the reward and interface.

Fixed retrieval, simulation, and live experience A fixed Wikipedia retriever makes interactions repeatable, but the policy also learns how that engine responds. Changing the engine during both training and testing changes the evidence available at both stages [125]; Table 4 keeps that combined intervention explicit. Query-learning experiments show a more specific dependence: descriptive statements work better for Contriever, while explicit questions work better for Qwen3-Embedding [126]. These experiments reward the ranking of known relevant passages and use branching rollouts to examine intermediate rewrites. Query styles can transfer poorly between retrievers. Learning a good query therefore includes learning what a particular backend can match, not only expressing the user's intent clearly.

Simulation offers a way to reduce live search during training while retaining search-like observations. ZeroSearch [28] first tunes a document simulator on real query–document pairs, then freezes it for policy rollouts and gradually increases simulated noise. The simulator's prompts include the question and reference answer to control relevance. Preparation therefore uses real search experience and information unavailable to a deployed searcher, even though the policy rollouts avoid live API calls. The resulting policy is evaluated with real Google search. Increasing simulator capacity moves performance from parity with to above the real-engine training comparator, showing that rollout source and simulator quality both affect the result.

Live experience instead exposes the policy to search and crawling during optimization. DeepResearcher [16] combines GRPO with a reading agent, caching, and retries, improving over prompt-based and static-retrieval training baselines. The central policy chooses searching, browsing, and answering; the reading agent's extraction and continuation operations belong to its environment. Its rewards and evaluations concern short-answer questions, including held-out datasets. This differs from training on human preferences for answers with references [26]. Experience with web pages can teach verifiable fact-finding without directly optimizing the coverage and organization of a research report.

Long horizons and measured transfer Long searches create both a learning problem and an execution problem: different rollouts take different amounts of time. Asynchronous training, as used in ASearcher [107], lets short GRPO rollouts finish while longer interactions continue. Generated webpage summaries are trained with other generated tokens, connecting the compression of evidence to the eventual answer reward. In its web QwQ experiments, reinforcement learning improves accuracy over prompted QwQ on GAIA and xBench-DeepSearch, with the larger gain on the latter. Restricting turns reduces success. Additional turns help when the policy learns to use them and the training infrastructure can accommodate them.

Transfer tests change different parts of this system. The locally trained ASearcher policy is evaluated with web tools without further training, and a separate web-policy experiment replaces its summarizer with an external model. In R1-Searcher's local-to-web test, both the engine and webpage summarization change [27]. These tests go beyond new question datasets with an unchanged engine, but they do not all test the same capability. Tools, observation formats, summarization, and inference computation must be tracked separately to determine what the observed transfer establishes.

6 Evaluation

Evaluation measures whether a system answers the question, finds and uses supporting evidence, and completes the work within its resource budget. Answer scores, evidence scores, and resource

Table 3: Evidence access, outputs, and measurement scope of representative retrieval and research benchmarks.

Benchmark	Evidence access	Output	Outcome	Evidence / process	Resource	Outside measured scope
HotpotQA [127]	10 paragraphs; or full Wikipedia leads	Answer, support	Answer EM/F1; joint EM/F1	Supporting-sentence EM/F1	—	Search cost
MuSiQue [128]	20 supplied paragraphs	Answer, support	Answer F1; sufficiency-gated F1	Paragraph support F1	—	Search cost
MultiHop-RAG [129]	Fixed news collection	Short answer	Answer accuracy	MAP/MRR/Hit@K	—	Citation support
FRAMES [130]	Fixed Wikipedia index; oracle articles	Short answer	Judged accuracy	Reasoning-type breakdown	Retrieval-step sweeps	Citation support
BrowseComp-Plus [131]	Fixed web-derived corpus	Answer, citations	Judged accuracy; calibration	Evidence recall; citation P/R; nDCG	Calls; API cost	Claim entailment
BrowseComp [10]	Live web; no-tool baselines	Short answer	Judged accuracy; calibration	—	Browsing effort; samples	Citation support
WebWalkerQA [132]	Traverse from root URL	Short answer	Judged accuracy	Depth/source breakdown	Actions on successes	Failed-run cost
WideSearch [133]	Live web collection	Table	Exact-table success; row/item F1	—	Human time/ pages	Citation support
DeepResearch Bench [11]	Web-enabled agents	Report	RACE report quality	FACT citation accuracy; supported-pair count	—	Search trajectory
Dr. Bench [134]	Web-enabled agents	Report	Rubrics; topical focus	Recommended URL/host hits; link-use ratio	Tokens; score/ token	Claim entailment
ReportBench [135]	Web search; date cutoff	Survey	Reference P/R	Cited-statement match; uncited factuality	—	Writing quality
SummHay [136]	Synthetic fixed haystacks	Cited bullets	Insight coverage; joint score	Gold-document citation F1	—	Live-web acquisition
SeekBench [83]	QA search-agent traces	Annotated trace	Answer F1 (analysis)	Groundedness; recovery; sufficiency calibration	—	Report coverage

EM: exact match; P/R: precision/recall. Dashes indicate no measure in the corresponding comparison category, not absence of all supplementary analysis. Resource cells distinguish agent expenditure, controlled effort, and human annotation effort; these are not interchangeable costs. Evidence/process cells retain each benchmark's own measurement unit.

measurements capture these different parts of the task.

We group benchmarks by evidence access and required output. Fixed-evidence question answering measures answers and recovery of known support (§6.1). Web information seeking adds navigation, changing sources, and repeated attempts (§6.2). Research reports require coverage, supported claims, and coherent synthesis (§6.3). Each subsection defines the main measures before comparing how benchmarks use them. Table 3 summarizes their measurement scope. Process diagnostics then relate the final result to decisions and costs within the search (§6.4). Across these settings, comparisons need to state evidence access and resource budgets alongside the scores.

6.1 Question answering with fixed evidence

In *fixed-evidence question answering*, the system answers a question using a document collection selected in advance. It may receive passages with the question or search a larger collection, such as a saved version of Wikipedia. The system can search repeatedly and gather different passages,

but the collection it searches remains fixed.

Benchmarks in this family include questions that connect facts across Wikipedia articles (e.g., HotpotQA and MuSiQue [127, 128]) and across news articles (MultiHop-RAG [129]). Their annotations determine what can be checked. A reference answer allows answer scoring; marked supporting evidence also allows the evaluator to check what the system found. Support may be labeled as sentences, paragraphs, or structured reasoning facts [137]. These choices establish the task and the reference against which performance is measured.

Metrics *Exact match* (EM) gives one when the prediction matches the reference and zero otherwise. For answers, the comparison uses normalized text, ignoring differences such as capitalization and punctuation. For supporting evidence, it compares the selected set of evidence units with the annotated set. *Precision* P is the fraction of predicted items that match the reference. *Recall* R is the fraction of reference items recovered. *F1* combines them through the same calculation in both cases:

$$F_1 = \frac{2PR}{P + R},$$

with a score of zero when both precision and recall are zero.

Answer F1 compares tokens in the predicted and reference answers. *Support F1* compares the selected evidence with the annotated supporting sentences or paragraphs. The scores are computed for each question and then averaged [127, 128]. HotpotQA additionally reports joint scores that combine answer and support performance [127].[1]

Ranked retrieval adds a question about order: how early does relevant evidence appear? *Hit@K* is the fraction of questions with at least one relevant result among the first K results. *Mean reciprocal rank* (MRR) averages the reciprocal rank of the first relevant result, with zero when none appears within the evaluated cutoff. For example, a first relevant result at rank two contributes one-half. *Mean average precision* (MAP) considers the whole relevant set. For each question, *average precision* sums the precision at ranks containing a relevant result and divides by the number of reference relevant items; MAP averages those values over questions. MultiHop-RAG reports these ranking measures for retrieved news chunks [129]. Evidence-graph scores additionally check whether the connections between sources have been recovered [138].

Acquisition and use These measures distinguish finding evidence from answering with it. In visual document retrieval, for example, iterative search can recover target pages missed by direct query–page matching when a semantic connection is needed (VisDocAgentBench [139]). A better page-ranking score establishes a discovery benefit. Answer scores are needed to determine whether that benefit carries through to a generated answer. Even then, a correct answer y may come from supplied material or the model's own knowledge, so comparisons with no retrieval help establish what the retained evidence E_T contributed. Document labels based on contribution to the final answer can also track end-to-end performance better than conventional relevance labels [140].

Panel A of Table 4 examines that next step. It compares no retrieval, one-step BM25, and IRCoT while keeping the answer model and maximum context capacity fixed [2]. Searching the full Wikipedia paragraph corpus nearly doubles answer F1 over answering without evidence. Interleaving reasoning with search adds a further gain of nearly a fifth over one-step retrieval. The iterative method can spend more retrieval and generation calls to assemble its context, so the shared paragraph cap controls answer-model input rather than total effort.

[1] In HotpotQA, *joint EM* is one only when both the answer and supporting set match exactly. *Joint F1* first multiplies answer precision by support precision and answer recall by support recall, then applies the F1 formula to those two products.

Table 4: Matched comparisons of retrieval control, search environments, and evidence allocation.

A. IRCoT: Flan-T5-XXL; HotpotQA, 500 questions [2]

Acquisition	Answer F1 (0–100)
No retrieval	25.3 ± 0.3
One-step BM25	49.7 ± 0.5
IRCoT	59.1 ± 0.9

B. Qwen2.5-7B-Base, PPO; Wikipedia-18 [125]

Training and test engine	NQ EM	HotpotQA EM	Bamboogle EM
BM25	0.341	0.404	0.280
E5 HNSW	0.468	0.372	0.400
E5 exact	0.481	0.433	0.424

C. Qwen2.5-7B-Instruct; HotpotQA distractor, 5,000 questions [141]

Router / depth cap	EM (%)	F1 (%)	Latency (ms)
Heuristic / 1	46.1	53.9	5,426
Heuristic / 2	52.9	61.3	5,628
Heuristic / 5	53.2	61.6	5,642
Fixed hybrid RRF / 5	55.0	63.5	5,688

D. Qwen2.5-1.5B-Instruct; HotpotQA candidates, 2,000 questions [41]

Router	Deploy acc. (%)	Usage (%)	Exact-50 acc. (%)
Complexity features	31.8 ± 1.4	49.3	31.9 ± 1.3
Minimum confidence	33.3 ± 1.6	51.5	32.7 ± 0.9
BUR-linear	33.8 ± 1.1	51.6	32.7 ± 0.9

A: shared 15-paragraph cap; mean and standard deviation over three demonstration sets.
B: EM fractions; each row changes the engine in both training and evaluation.
C: mean wall-clock latency with one worker.
D: correctness is normalized exact match or token F1 > 0.80. Calibration-selected thresholds target 50% evidence use; Exact-50 selects the held-out top half by score.

The search engine also affects the comparison. Panel B holds the policy-learning algorithm and backbone fixed while changing the engine during both training and evaluation [125]. Exact E5 search improves EM over BM25 on all three datasets. Approximate E5 search improves Natural Questions and Bamboogle but falls below BM25 on HotpotQA. An average across datasets would hide that reversal. The result depends on the environment in which the policy learns and searches, as well as on the policy itself.

Evidence sufficiency and assembly Once the system has found the supporting sources, it still has to read and combine them correctly. Supplying those sources directly tests this remaining work. In FRAMES, a benchmark of questions requiring multiple Wikipedia sources, iterative retrieval closes much of the answer-accuracy gap between BM25 search and receiving all reference articles [130]. Numerical, tabular, and post-processing questions remain difficult even when the articles are supplied. Finding the right sources therefore resolves only part of the task.

The way evidence reaches the model can affect the answer too. With the number of passages held fixed, targeted replacement improves evidence precision and answer quality over the evaluated add-or-prune alternatives [142]. Iterative systems also outperform receiving all reference paragraphs at once on ChemKGMultiHopQA, a benchmark of scientific multihop questions [82]. This comparison changes both the reasoning procedure and the material shown at each step. It suggests that useful evidence must be assembled and interpreted, rather than merely included somewhere in the input.

These benefits need comparison with an appropriate fixed retrieval strategy. The strongest

strategy changes with the collection and question type: vector, graph, hybrid, iterative, and no-retrieval workflows exchange places across settings in RAGRouter-Bench [143]. Comparisons of RAG and GraphRAG likewise find complementary strengths and benefits from hybrid selection [144]. On STaRK, a retrieval benchmark over textual and relational knowledge bases [145], adding query profiles and trace hints with tools and data held fixed reduces ranking quality and produces repeated or empty-result calls [146]. More search control helps when it improves evidence selection or use enough to justify the added work.

Finally, a system needs to recognize when the available collection cannot support an answer. MuSiQue-Full pairs a question with two contexts, one containing the necessary support and one missing a required step [128]. The paired evaluation checks whether the system answers when it has support and recognizes when support is missing.[2] Other tests change what the evidence justifies through fictional news timelines [147], removal of direct support while retaining alternative logical paths [148], or restricted source views requiring explicit gap reports [149]. These tests reward answering from the evidence that is available and recognizing when it leaves the question unresolved.

6.2 Web information seeking

Web information seeking asks a system to answer by searching for sources and navigating their contents. The available pages can change between runs, and the search provider determines which results the system sees. Tasks range from finding one difficult fact to collecting a complete set of entities and their attributes. Evaluation must therefore specify both what counts as a complete answer and what access the system has to the web.

Benchmarks emphasize different parts of this work. Fact-seeking questions test whether the system can locate an obscure answer (BrowseComp and BrowseComp-ZH [10, 150]). Navigation tasks start from a given website and require following links (WebWalkerQA [132]), while broad collection tasks require complete tables (WideSearch [133]). Other tasks involve changing facts or false premises [151], or require finding and interpreting multimodal content (MMSearch [152]). These task differences determine which measures of success are useful.

Metrics For short answers, *judged accuracy* is the fraction of questions whose responses an evaluator accepts as matching the reference answer. The judgment can allow equivalent wording. *Confidence calibration* checks a different property: answers assigned 80% confidence should be correct about 80% of the time. Calibration-error scores summarize the discrepancy between confidence and observed accuracy [10].[3]

For table outputs, precision, recall, and F1 use the definitions in Section 6.1, with cells or rows as the counted items. *Item precision* is the fraction of predicted cells that match the reference; *item recall* is the fraction of reference cells recovered. *Row precision and recall* count a row as correct only when all its required cells match. *Item F1* and *row F1* apply the same F1 calculation to these respective counts. *Exact-table success* is the fraction of tasks whose entire table matches the reference [133].

Repeated runs require a separate aggregation rule. *Average success* is the fraction of individual attempts that succeed. *Pass@N* is the fraction of tasks solved at least once within N attempts. Average success describes reliability on one run; pass@N describes success within an attempt budget. These quantities can move differently even for the same tasks [153].

[2]The sufficiency-aware answer and support scores are zero if either answerability judgment in the pair is wrong. Otherwise, they retain the corresponding F1 score from the answerable instance.

[3]Confidence-calibration error depends on how predictions are grouped and how discrepancies are aggregated. BrowseComp reports this error without specifying its formula in the paper.

Live and fixed access A benchmark built from web questions can still provide a fixed collection. This makes it possible to check which supporting documents were retrieved or cited, using the evidence measures introduced in Section 6.1. BrowseComp-Plus provides such a collection and verified supporting documents [131].[4] Fixed access also permits repeatable tests of search sufficiency, knowledge use, and refusal [154].

Live-web difficulty can instead be controlled through question selection. Excluding questions solved by single-turn search systems yields questions that require more than those solvers can provide (InfoDeepSeek [155]). Comparing noisy search results with distractor-heavy context supplied directly separates difficulty in finding evidence from difficulty in reading it (SealQA and LongSeal [156]). The distinction matters when attributing a failed answer to search.

Search providers can also change the work required without greatly changing final accuracy. With the agent and page-fetch backend held fixed, Brave, Tavily, and Firecrawl achieve almost identical correctness on the evaluated SealQA-Hard subset [157]. Yet Brave returns supporting snippets for nearly twice as many questions. The providers return different snippet volumes and change subsequent page fetching. Answer accuracy therefore needs to be read alongside the evidence returned and the effort spent using it.

Repeated success and completeness Scoring individual facts and scoring completion can lead to very different assessments. WideSearch reports appreciable item F1 alongside very few perfectly completed tables [133]. A system can recover many correct cells while leaving most requested collections unfinished. Row and table scores reveal the missing combinations that an item score can hide.

More attempts can likewise increase the chance of eventual success without making each attempt more reliable. In HybridDeepResearch [153], a multi-agent arrangement can improve success across trials without consistently improving average trial success. Its mixed database and search tasks also test whether an entity found by one tool is preserved as a constraint for the next. The attempt budget and aggregation rule are needed to interpret the reported benefit.

Cost accounting must include failed attempts as well as successful ones. WebWalkerQA reports actions for successful executions only [132]. That statistic describes the effort needed when the system succeeds, but leaves the expenditure on failed runs uncounted. A system's total cost of completing web tasks depends on both.

6.3 Long-form research reports

A research report combines many claims into an answer to a broader request. It can contain accurate statements yet omit a required topic, or cover the requested topics while citing sources that do not support its claims. Evaluation needs an account of what the report should contain, evidence for checking its statements, and criteria for judging the report as a whole.

Benchmarks obtain these references in different ways. Academic survey tasks use an existing expert survey and its bibliography (ReportBench and DeepScholar-Bench [135, 158]). Broader research tasks use reference reports or expert-written criteria (DeepResearch Bench and Dr. Bench [11, 134]). Controlled summarization tasks place known insights across supplied documents (SummHay [136]). These choices determine which omissions, unsupported claims, and weaknesses in presentation can be checked.

[4] BrowseComp-Plus measures evidence recall over documents retrieved anywhere in the interaction. Its document-level citation precision and recall compare cited documents with the labeled relevant set. Its standalone retriever evaluation also reports *normalized discounted cumulative gain* (nDCG), which discounts relevance at lower ranks and divides by the score of an ideal ranking. Document relevance labels do not establish that a citation supports a particular claim.

Metrics *Content coverage* measures how much of the required content receives credit as present in the report. The reference may be a list of insights, topics, or task requirements; some protocols award partial credit. *Factual support* instead measures the fraction of generated facts for which a verifier finds supporting evidence. FActScore checks individual facts against a designated knowledge source [159], while SAFE uses search to verify facts and combines support with sensitivity to answer length [160]. Coverage counts expected content; factual support counts the claims the system actually makes.

Citation scores check the references attached to those claims. *Citation recall* is the fraction of statements supported by their attached references. *Citation precision* is the fraction of individual citations judged to contribute support [161, 162]. Their denominators differ: recall counts statements, while precision counts citations. A statement can require several sources, and an additional citation can be irrelevant even when the others support it.[5]

Bibliography precision and recall compare the generated reference list with an expert reference list. Here, the matching item is a cited paper: precision is the fraction of generated references matched to the expert list, and recall is the fraction of the expert list recovered [135]. These scores measure agreement on source selection. They do not check whether a reference supports the statement that cites it.

Report-quality scores apply a rubric to properties such as comprehensiveness, depth, organization, readability, and adherence to the request. The criteria, their weights, and any comparison with a reference report determine the score [11, 134].[6] These broader judgments complement the counts of content, facts, and citations.

Reference coverage and factual support Matching an expert bibliography provides a concrete test of literature selection, but ReportBench finds little additional reference recall even when reports cite very different numbers of papers [135]. Its prompts specify a date cutoff and exclude the source survey itself as a citation. Cited statements and uncited claims receive separate support and factuality checks. Recent academic exemplars similarly support separate tests of retrieval, synthesis, and verifiability (DeepScholar-Bench [158]). Agreement with an expert selection is useful evidence of coverage, while a different, well-supported synthesis can still be valuable. ReportBench leaves writing quality outside its assessment.

Source recommendations and claim support also need separate interpretation. Expert rubrics can identify trustworthy URLs or hosts and topical requirements (Dr. Bench [134]). Matching those recommendations shows that the system used the expected sources; it does not establish that their text supports a particular statement. Conversely, a supported statement can address the wrong topic. In DeepResearch Bench, the system with the highest effective-citation count does not have the highest citation accuracy [11]. Producing more supported material and making individual citations reliable are distinct achievements.

Factual support and citation quality can diverge when the system preserves content but loses the details needed for attribution. In ALCE's compression experiments, answer correctness improves while citation F1 falls on ASQA and ELI5; citation quality improves on QAMPARI [161]. The effect depends on the task. Checking generated facts against external evidence and checking the references actually attached to them therefore remain separate parts of evaluation.

[5] In ALCE, citation recall checks whether a statement's cited passages jointly entail it. Citation precision additionally excludes a citation when it neither supports the statement alone nor adds necessary support to the other citations. DeepResearch Bench's FACT uses a different unit: unique statement–URL pairs. Its *citation accuracy* is the supported fraction within each report, averaged across reports; *effective citations* are the mean number of supported pairs per task.

[6] DeepResearch Bench's RACE weights comprehensiveness, depth, instruction-following, and readability for each task. It scores the target and reference reports against the criteria, then divides the target score by their summed scores. The resulting score is relative to that reference.

Known insights and human judgments A controlled list of expected insights makes omissions directly observable. SummHay uses this design to compare retrieval with full-context processing [136]. Its joint score rewards covering an insight and citing the documents that contain it.[7] Retrieval raises citation and joint scores in the reported comparisons, while its effect on coverage varies across models. Even the strongest oracle-retrieval result falls below estimated human joint performance. The known content list lets completeness and attribution be examined together.

Human readers can assess qualities beyond such a list. Experienced editors evaluating STORM's articles find better organization and breadth than outline-driven RAG, while identifying neutrality and unsupported associations as remaining problems [163]. Fixed search snapshots with report judging provide another way to control source access (DeepResearchGym [164]). Known insights, expert reading, and frozen sources each make different parts of report quality observable.

Automated judges need validation against the judgment they are intended to replace. RACE improves agreement with expert report comparisons over direct scoring [11], while SummHay compares automated coverage judgments with human ratings [136]. Such agreement supports the corresponding measurement. Position, length, and self-preference biases in language-model judges [165] motivate checking how a particular rubric behaves. Agreement on report preference or individual facts does not by itself establish that the relevant literature has been covered.

The cost of evaluation must also be distinguished from the cost of research. Dr. Bench records agent tokens and report metadata, whereas DeepResearch Bench's evaluator-cost comparison concerns the judge. A cheaper judge can make evaluation more affordable without reducing the resources the research agent uses to produce its report.

6.4 Process diagnostics and robustness

A final answer shows the outcome of a search, but not which decisions helped produce it or where the process went wrong. *Process diagnostics* examine the sequence of searches, observations, and reasoning steps. They can distinguish failing to find evidence from failing to read or use evidence that was already available. Resource measurements add the work spent reaching the answer.

These evaluations use different kinds of comparison. Budget experiments vary the amount or allocation of search. Annotated traces identify unsupported reasoning and unsuccessful recovery. Fault-injection tests introduce a known mistake and check whether a diagnosis locates it. *Robustness tests* change the retrieved material and observe how the system's later decisions and answer respond. Each requires a measure tied to the part of the process being tested.

Metrics *Search-call counts* measure search API invocations. *Token counts* measure model input, output, or both, which must be stated. *Latency* is elapsed time; *monetary cost* applies the stated prices to model calls, tools, or source access. These are different measures of $c(\tau)$. Concurrent calls can reduce latency without reducing their number, so a budget comparison needs both a named resource and the answer score achieved within it.

Evidence usage is the fraction of questions sent to retrieval. A *usage-budget violation* occurs when that fraction exceeds the target. *Budget calibration* selects a decision threshold intended to keep usage within that target on new questions [41]. This concerns allocation of retrieval, whereas confidence calibration concerns agreement between stated confidence and answer correctness (§6.2).

Trace scores measure behavior within a run. SeekBench's *reasoning quality index* is the

[7] Each insight receives full, partial, or zero coverage credit. Citation F1 compares cited documents with the gold document set for covered insights. The joint score multiplies coverage by citation F1 for each insight, then averages across all required insights, including omitted insights with zero credit.

fraction of reasoning steps grounded in retrieved evidence, averaged first within traces and then across traces. Its *evidence recovery function* is the fraction of traces that reach clear, sufficient evidence or a correct answer by a given turn. Its *calibration error* measures how often answering decisions depart from answering only with clear, sufficient evidence [83].[8]

Diagnosis and robustness scores use their own target events. *Exact-hop accuracy* measures the fraction of still-failing injected traces for which the diagnosed step matches the injected fault [166]. *Context preference rate* measures the fraction of responses that follow the supplied context, including when that context has been altered [167]. For report poisoning, *PRISM* measures the weighted fraction of extracted report claims that match an attacker's narrative, with weights determined by claim type [168]. The counted event must be specified before a score can establish successful diagnosis or resistance to an attack.

Useful depth and calibrated allocation Increasing a search budget shows whether more work improves the answer. Panel C of Table 4 reports answer EM, F1, and mean wall-clock latency as the step cap increases [141]. Most of the answer gain arrives between one and two steps; further steps add little. The pipeline decomposes the question upfront and searches ten supplied paragraphs, with no language-model call between retrieval steps. Decomposition and final synthesis dominate runtime, so deeper retrieval changes latency only slightly. A fixed combination of retrieval rankings also outperforms the entity-based heuristic router. The useful comparison is each complete policy's answer quality at its measured cost.

Allocating a budget introduces a further question: which inputs should receive the available searches? Panel D compares a threshold chosen on calibration data with selecting exactly half the held-out questions by score [41]. BUR-linear has a small apparent accuracy advantage over minimum-confidence routing under the deployable thresholds, but both tie at exactly 50% usage. Ranking the held-out set tests allocation at a fixed rate. A threshold chosen in advance must also maintain that rate on new inputs.

The decision itself belongs in the cost account. Complexity features require little work, minimum-confidence routing requires an initial answer, and BUR-linear requires both initial generation and a probe retrieval. Confidence probing can therefore avoid searches while adding generation cost [17]. Sources may carry prices too: access-friction experiments show that larger source budgets do not uniformly improve answers [169]. This adds a constraint beyond passage relevance in collections such as MS MARCO [170]. Budget-controlled studies can vary retrieval strategy, depth, and the allowance for completing the answer separately [21]. Counting search, decision overhead, and synthesis makes clear where a policy saves resources and where it spends them.

Locating acquisition and reasoning failures Trace annotations make it possible to locate failures relative to the evidence available at each step. SeekBench uses expert annotations to distinguish unsupported reasoning, recovery behavior, and decisions about when to answer [83]. Analysis on fixed BrowseComp-Plus documents finds both retrieval gaps and wrong answers after relevant documents have been found, with much unproductive search occurring after discovery [40]. Paired trajectories further distinguish skipping inspection from failing after reading reference evidence [109]. These failures call for different repairs; another search cannot resolve every mistake made after discovery.

A useful diagnosis should help the system recover. RePAIR improves mean answer F1 over standard RAG and RAG-Critic while using fewer counted auxiliary retrieval operations, although gains vary across datasets [171]. Because the comparison changes training as well as error

[8] SeekBench weights the deviation between the observed answer rate and the ideal decision within each annotated evidence state, then averages over traces. This penalizes both answering with inadequate evidence and withholding an answer despite good evidence. It assesses evidence-based answering decisions, not confidence percentages.

categories, it supports the combined repair system. Fault injection provides a more direct test of localization: a known mistake is introduced, subsequent steps are rerun, and the diagnosed step is compared with the injected one (AgenticRAG-FP [166]). Traces that recover are evaluated separately, since they no longer have a final failure to attribute. Counterfactual repair recovers a localization signal lost by passive coverage-based diagnosis, although deeper failed-case samples are small.

Other diagnostics test whether retrieval-quality estimates predict final answers [49], equivalent question wordings change the search [50], or added context representations change the result [172]. A late visible error may originate in an earlier decision. The point of tracing it is to identify where changing the process could improve the result.

Conflicting and malicious observations Changing the retrieved material tests whether the system can use evidence under less reliable conditions. Noise tests separate different kinds of irrelevant or misleading material [173]. Controlled changes to answer-bearing context show how responses shift between the supplied account and the model's prior answer. In ClashEval, larger perturbations and stronger prior confidence reduce preference for the context [167]. Visual diffusion-model experiments show why discovery and answering still need separate scores: adding pages improves answer-page recall, while conflicting sources reduce answer accuracy [174].

Malicious sources can also change the search itself. In FORGE, coordinated poisoned documents steer follow-up planning and introduce attacker-aligned claims into research reports [168]. Its report-contamination score tracks this influence across extracted claims. The 25-query attack set and smaller defense subset provide a limited empirical base, but expose a failure specific to the search loop: an early observation can influence which evidence the policy seeks next. Robustness tests therefore need to examine both the resulting claims and the search decisions that produced them.

7 Open problems

Finding more evidence does not resolve every difficulty in agentic RAG. A system can retrieve the right source and still misread it, correct one fact while retaining conclusions based on its old version, or spend its remaining budget on searches that do not improve the answer. The comparisons above leave three connected problems unresolved: feedback must identify the right correction (§7.1); the remaining evidence needs must guide how effort is spent (§7.2); and these decisions must remain reliable when questions, tools, and sources change (§7.3).

7.1 Reliable evidence feedback

Feedback must distinguish missing evidence from an incorrect inference or unresolved source disagreement. As Sections 4.1 and 6.4 show, these failures require different actions even when they produce similar answer confidence. The judgment also changes the states that the system reaches next. A judge may stop some searches, request more evidence in others, and later assess the results of its own earlier decisions. Its confidence must remain reliable across those changes. Recognizing apparently sufficient evidence does not ensure that stopping preserves answer accuracy [19]. The system still needs to establish that its sources jointly support the answer, including requirements that its own assessment may have missed.

Making missing support affect completion [100] and reconciling conflicting accounts before writing [78] address different parts of this problem. Disagreement can reflect a mistaken source, a difference in scope, or facts that changed over time. A compact summary may preserve a statement while dropping the date or qualification needed to distinguish those cases. Source-linked memory allows the system to inspect the original again, but does not decide which source

should overturn a claim. Reliable correction needs both access to that detail and a judgment about what the apparent conflict means.

Feedback providers can also be trained to address a policy's current mistakes. CAFE [175] alternates policy learning with critic refinement on recent trajectories. Matching later agents with critics from the same training stage works better in its checkpoint comparisons, suggesting that useful feedback depends on how the policy is changing. Another approach links helpfulness judgments to stable evidence identifiers and trains sensitivity to controlled changes in evidence (Proof-of-Use [176]). The first adapts corrective guidance to the policy; the second tests whether a judgment responds to the observations it claims to assess. Neither resolves correlated mistakes between an agent and critic, particularly when they share parameters or encounter sources that agree on a false premise. Feedback must still detect an error that both the current reasoning and the retrieved material reinforce.

7.2 Efficient long-horizon retrieval

A search that keeps finding similar documents is not necessarily ready to stop. Failures after discovery show that having the evidence and using it remain separate problems [40]. Conversely, many relevant documents can still leave one connecting fact missing. Coverage, document novelty, and query signals can guide a choice among continuing, intervening, and stopping [177]. However, the system needs an accurate account of what remains uncovered, and computing that account also costs effort. A fixed turn limit avoids those judgments but cannot adapt to the difference between a missing fact and an unresolved inference.

Allocation also needs to cover what happens after retrieval. Search-time effort and answer-time revision can be controlled separately [112], but memory repair and delegation introduce further demands. Correcting a fact may require discarding dependent reasoning, including useful content that must then be recovered (ReTree [89]). The system must choose between recovering that content and exploring another direction. Delegation likewise spends resources on assigning work, exchanging findings, and integrating results. The comparisons in Section 3.4 do not isolate those contributions from the additional computation supplied by more workers. Counting search calls alone cannot guide all these choices.

Execution changes their costs as well. A cache-aware executor can manage retrieval waits and schedule generation [178]. The cost also depends on how interrupted generations resume. Two policies with the same call count may therefore consume different resources depending on how their actions are executed. It remains unclear how a policy should respond when these costs change while preserving access to complementary sources. Decisions about search depth, memory repair, and concurrent work need to account for the cost of completing the answer, rather than treating every additional call as the same expense.

7.3 Generalization and credible evaluation

A policy must adapt both what it asks and how it reads the reply. Keeping the policy π fixed does not keep the interaction fixed: a new backend can return different observations o_t for the same search action. Retriever-specific query learning shows that a formulation useful for one backend may transfer poorly to another [126]. Comparisons of static and agentic retrieval also show that follow-up queries change which retriever strengths matter [179]. The challenge is to adapt search to a new backend without losing reliable evidence assessment or stopping behavior. Success on new question datasets with an unchanged engine does not establish that broader transfer.

Language, domain, and modality can change what evidence is usable. The English and Chinese tests in RGB [180] examine rejection, information integration, and robustness to counterfactual content. The visual cases in Sections 4.2 and 6.1 add another requirement: finding a page does not

ensure that its answer-bearing content survives reading or compression. These capabilities need to transfer together. Simulator-to-real deployment offers one direction (§5.3), but simulations must represent more than successful retrieval. Failed searches, conflicting sources, and changes in observation format remain important tests of their fidelity.

Transfer claims also depend on how improvement is measured. Fixed collections [131] and source snapshots [164] make access reproducible, while deployment still encounters changing web content. Evaluation can introduce its own shortcuts: candidate sentences can be filtered using evaluator prompts, or generation can be guided by reference content later used for scoring [181]. Improvements must therefore be checked under changes in sources and evaluators, as well as under repeated use of one benchmark. Human agreement on a support judgment or coverage rating, discussed in Section 6.3, validates that local measurement. It does not establish that the research is complete or that the same judgment remains valid in an unfamiliar domain.

8 Conclusion

Agentic RAG allows a system to change its search as it learns what a question requires. Across the methods reviewed, this flexibility helps recover facts that an initial query misses and investigate problems revealed by earlier results. But successful retrieval is only part of a successful answer. If a summary drops an important qualification or a report combines facts incorrectly, more retrieval may leave the underlying error untouched. The value of adaptive search therefore depends on how the system carries evidence through to the final answer.

Training and evaluation must account for these dependencies. What a policy learns depends on the tools and observations available during training, so success on new questions does not establish that it will work with a different search engine. Higher answer accuracy also needs to be assessed against the resources used to achieve it. Counting search calls alone misses the cost of deciding, reading, and revising. Evaluation should follow evidence from retrieval to the answer, checking whether the system used it correctly and addressed the full request. Further progress requires feedback that leads to the right next action: another search for a missing fact, a correction to faulty reasoning, or a decision to answer once the evidence supports a complete response.

References

[1] Patrick Lewis, Ethan Perez, Aleksandra Piktus, et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems, 2020.

[2] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions. arXiv preprint arXiv:2212.10509, 2023.

[3] S Jeong, J Baek, S Cho, SJ Hwang, and JC Park. Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity. arXiv preprint arXiv:2403.14403, 2024.

[4] Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Sai Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, H. Shum, and Jian Guo. Think-on-Graph: Deep and Responsible Reasoning of Large Language Model on Knowledge Graph. International Conference on Learning Representations, 2023.

[5] Zhen Zhang, Liangcai Su, Zhuo Chen, Xiang Lin, Haotian Xu, Simon Shaolei Du, Kaiyu Yang, Bo An, Lidong Bing, and Xinyu Wang. Argus: Evidence Assembly for Scalable Deep Research Agents. arXiv preprint arXiv:2605.16217, 2026.

[6] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. arXiv preprint arXiv:2310.11511, 2023.

[7] SQ Yan, JC Gu, Y Zhu, and ZH Ling. Corrective Retrieval Augmented Generation. arXiv preprint arXiv:2401.15884, 2024.

[8] Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. FireAct: Toward Language Agent Fine-tuning. arXiv preprint arXiv:2310.05915, 2023.

[9] Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning. arXiv preprint arXiv:2503.09516, 2025.

[10] Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. BrowseComp: A Simple Yet Challenging Benchmark for Browsing Agents. arXiv preprint arXiv:2504.12516, 2025.

[11] Mingxuan Du, Benfeng Xu, Chiwei Zhu, Xiaorui Wang, and Zhendong Mao. DeepResearch Bench: A Comprehensive Benchmark for Deep Research Agents. arXiv preprint arXiv:2506.11763, 2025.

[12] Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active Retrieval Augmented Generation. arXiv preprint arXiv:2305.06983, 2023.

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

[14] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language Models Can Teach Themselves to Use Tools. NeurIPS, 2023.

[15] Zhengliang Shi, Yiqun Chen, Haitao Li, et al. Deep Research: A Systematic Survey. arXiv preprint arXiv:2512.02038, 2025.

[16] Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. DeepResearcher: Scaling Deep Research via Reinforcement Learning in Real-world Environments. Conference on Empirical Methods in Natural Language Processing, 2025.

[17] Chandan Kumar Sah, Li Zhang, and Xiaoli Lian. Beyond Self-Knowledge: Propagating Uncertainty Across Reasoning and Retrieval in LLMs. arXiv preprint arXiv:2607.25600, 2026.

[18] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust LMs: Investigating effectiveness of parametric and non-parametric memories. ACL, 2023.

[19] Weimeng Luo. When Should Multi-Round RAG Stop? Structured Stopping Judgments and Retrieval Reduction in Search-R1. arXiv preprint arXiv:2608.13237, 2026.

[20] Daeyoung Roh and Donghee Han. HALT: Verification-Aware Stopping for Retrieval-Augmented Search Agents. arXiv preprint arXiv:2608.02009, 2026.

[21] Kyle McCleary and James Ghawaly. Quantifying the Accuracy and Cost Impact of Design Decisions in Budget-Constrained Agentic LLM Search. arXiv preprint arXiv:2603.08877, 2026.

[22] 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.

[23] Jintao Liang, Gang Su, Huifeng Lin, You Wu, Rui Zhao, and Ziyue Li. Reasoning RAG via System 1 or System 2: A Survey on Reasoning Agentic Retrieval-Augmented Generation for Industry Challenges. IJCNLP-AACL, 2025.

[24] Yangning Li, Weizhi Zhang, Yuyao Yang, et al. Towards Agentic RAG with Deep Reasoning: A Survey of RAG-Reasoning Systems in LLMs. arXiv preprint arXiv:2507.09477, 2025.

[25] Hao Yu, Aoran Gan, Kai Zhang, Shiwei Tong, Qi Liu, and Zhaofeng Liu. Evaluation of Retrieval-Augmented Generation: A Survey. arXiv preprint arXiv:2405.07437, 2024.

[26] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, et al. WebGPT: Browser-assisted question-answering with human feedback. arXiv preprint arXiv:2112.09332, 2022.

[27] Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-Searcher: Incentivizing the Search Capability in LLMs via Reinforcement Learning. arXiv preprint arXiv:2503.05592, 2025.

[28] Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. ZeroSearch: Incentivize the Search Capability of LLMs without Searching. arXiv preprint arXiv:2505.04588, 2025.

[29] Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yutao Zhu, Yongkang Wu, Ji-Rong Wen, and

Zhicheng Dou. WebThinker: Empowering Large Reasoning Models with Deep Research Capability. arXiv preprint arXiv:2504.21776, 2025.

[30] Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. DRAGIN: Dynamic Retrieval Augmented Generation based on the Information Needs of Large Language Models. arXiv preprint arXiv:2403.10081, 2024.

[31] Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query Rewriting for Retrieval-Augmented Large Language Models. arXiv preprint arXiv:2305.14283, 2023.

[32] Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. Enhancing Retrieval-Augmented Large Language Models with Iterative Retrieval-Generation Synergy. arXiv preprint arXiv:2305.15294, 2023.

[33] 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. Conference on Empirical Methods in Natural Language Processing, 2025.

[34] Zhuofan Shi, Mingzhe Ma, Lu Wang, et al. TreeSeeker: Tree-Structured Trial, Error, and Return in Deep Search. arXiv preprint arXiv:2606.11662, 2026.

[35] L Chen, P Tong, Z Jin, Y Sun, J Ye, and H Xiong. Plan-on-Graph: Self-Correcting Adaptive Planning of Large Language Model on Knowledge Graphs. arXiv preprint arXiv:2410.23875, 2024.

[36] Qinyuan Cheng, Xiaonan Li, Shimin Li, Qin Zhu, Zhangyue Yin, Yunfan Shao, Linyang Li, Tianxiang Sun, Hang Yan, and Xipeng Qiu. Unified Active Retrieval for Retrieval Augmented Generation. arXiv preprint arXiv:2406.12534, 2024.

[37] Tong Zhao, Yutao Zhu, Yucheng Tian, and Zhicheng Dou. R3AG: Retriever Routing for Retrieval-Augmented Generation. arXiv preprint arXiv:2604.22849, 2026.

[38] Z Yao, W Qi, L Pan, S Cao, L Hu, W Liu, L Hou, and J Li. SeaKR: Self-aware Knowledge Retrieval for Adaptive Retrieval Augmented Generation. arXiv preprint arXiv:2406.19215, 2024.

[39] Varun Gandhi, Jaewook Lee, Shantanu Todmal, Franck Dernoncourt, Ryan Rossi, Zichao Wang, and Andrew Lan. GRASP: GRanularity-Aware Search Policy for Agentic RAG. arXiv preprint arXiv:2607.10463, 2026.

[40] Qi Liu, Jiaxin Mao, Fengbin Zhu, and Tat-Seng Chua. Diagnosing Search Behavior and Failure Modes in Long-Horizon Search Agents. arXiv preprint arXiv:2608.01913, 2026.

[41] Pin Qian, Su Wang, Chong Peng, Junxian You, Lifei Liu, Haoran Yu, Yihang Chen, and Xiaochong Jiang. When Should Active RAG Retrieve? A Budget-Aware Evaluation of Utility, Calibration, and Cost. arXiv preprint arXiv:2607.24010, 2026.

[42] Zihan Zhang, Meng Fang, and Ling Chen. RetrievalQA: Assessing Adaptive Retrieval-Augmented Generation for Short-form Open-Domain Question Answering. arXiv preprint arXiv:2402.16457, 2024.

[43] Bharath Nunepalli. SLO-Conditioned Action Routing for Retrieval-Augmented Generation: Objective Ablation and Failure Modes. arXiv preprint arXiv:2601.00841, 2025.

[44] Minho Kim. When Should LLMs Search? Counterfactual Supervision for Search Routing. arXiv preprint arXiv:2607.05752, 2026.

[45] Zhicong Li, Jiahao Wang, Zhishu Jiang, Hangyu Mao, Zhongxia Chen, Jiazhen Du, Yuanxing Zhang, Fuzheng Zhang, Di Zhang, and Yong Liu. DMQR-RAG: Diverse Multi-Query Rewriting for RAG. arXiv preprint arXiv:2411.13154, 2024.

[46] Liang Wang, Nan Yang, and Furu Wei. Query2doc: Query Expansion with Large Language Models. arXiv preprint arXiv:2303.07678, 2023.

[47] Shengyu Mao, Yong Jiang, Boli Chen, Xiao Li, Peng Wang, Xinyu Wang, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. RaFe: Ranking Feedback Improves Query Rewriting for RAG. arXiv preprint arXiv:2405.14431, 2024.

[48] Yujing Wang, Hainan Zhang, Liang Pang, Binghui Guo, Hongwei Zheng, and Zhiming Zheng. MaFeRw: Query Rewriting with Multi-Aspect Feedbacks for Retrieval-Augmented Large Language Models. arXiv preprint arXiv:2408.17072, 2024.

[49] Fangzheng Tian, Jinyuan Fang, Debasis Ganguly, Zaiqiao Meng, and Craig Macdonald. Am I on the Right Track? What Can Predicted Query Performance Tell Us about the Search Behaviour of Agentic RAG. arXiv preprint arXiv:2507.10411, 2025.

[50] Yunah Jang, Megha Sundriyal, Kyomin Jung, and Meeyoung Cha. How You Ask Matters! Adaptive RAG Robustness to Query Variations. arXiv preprint arXiv:2604.10745, 2026.

[51] Yunah Jang, Kang il Lee, Hyunkyung Bae, Hwanhee Lee, and Kyomin Jung. IterCQR: Iterative Conversational Query Reformulation with Retrieval Guidance. arXiv preprint arXiv:2311.09820, 2023.

[52] Yu Xia, Junda Wu, Sungchul Kim, Tong Yu, Ryan A. Rossi, Haoliang Wang, and Julian McAuley. Knowledge-Aware Query Expansion with Large Language Models for Textual and Relational Retrieval. arXiv preprint arXiv:2410.13765, 2024.

[53] Xinlong Xu and Yoshua Y. Li. EviReform: Evidence-Guided Query Reformulation for Multi-Hop Graph Retrieval. arXiv preprint arXiv:2608.13006, 2026.

[54] Paul J. L. Ammann, Jonas Golde, and Alan Akbik. Question Decomposition for Retrieval-Augmented Generation. arXiv preprint arXiv:2507.00355, 2025.

[55] Randhir Kumar. Verification Without Sufficiency: Per-Chunk Filtering Fails on Multi-Hop RAG, and Decomposition Repairs It. arXiv preprint arXiv:2608.00585, 2026.

[56] Shicheng Xu, Liang Pang, Huawei Shen, Xueqi Cheng, and Tat-Seng Chua. Search-in-the-Chain: Interactively Enhancing Large Language Models with Search for Knowledge-intensive Tasks. arXiv preprint arXiv:2304.14732, 2023.

[57] Liang Xiao, Wen Dai, Shuai Chen, Bin Qin, Chongyang Shi, Haopeng Jing, and Tianyu Guo. Retrieval-Augmented Generation by Evidence Retroactivity in LLMs. arXiv preprint arXiv:2501.05475, 2025.

[58] J Jiang, J Chen, J Li, R Ren, S Wang, WX Zhao, et al. RAG-Star: Enhancing Deliberative Reasoning with Retrieval Augmented Verification and Refinement. arXiv preprint arXiv:2412.12881, 2024.

[59] Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Reasoning on Graphs: Faithful and Interpretable Large Language Model Reasoning. International Conference on Learning Representations, 2023.

[60] Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. StructGPT: A General Framework for Large Language Model to Reason over Structured Data. arXiv preprint arXiv:2305.09645, 2023.

[61] Songeun Lee, Kyungjin Min, Injae Na, Suyeong Lee, Chiyoung Kim, and Woohwan Jung. A Tree-based RAG Framework for Evidence-Intensive QA via Adaptive Planning and Topology-Aware Evidence Gathering. arXiv preprint arXiv:2609.04981, 2026.

[62] Kai Guo, Xinnan Dai, Shenglai Zeng, Harry Shomer, Haoyu Han, Yu Wang, and Jiliang Tang. Beyond Static Retrieval: Opportunities and Pitfalls of Iterative Retrieval in GraphRAG. arXiv preprint arXiv:2509.25530, 2025.

[63] Shu Zhao, Tan Yu, Anbang Xu, Japinder Singh, Aaditya Shukla, and Rama Akkiraju. ParallelSearch: Train your LLMs to Decompose Query and Search Sub-queries in Parallel with Reinforcement Learning. arXiv preprint arXiv:2508.09303, 2025.

[64] Dayoon Ko, Jihyuk Kim, Haeju Park, Sohyeon Kim, Dahyun Lee, Yongrae Jo, Gunhee Kim, Moontae Lee, and Kyungjae Lee. Hybrid Deep Searcher: Scalable Parallel and Sequential Search Reasoning. arXiv preprint arXiv:2508.19113, 2025.

[65] Xincheng You, Qi Sun, Neha Bora, Huayi Li, Shubham Goel, Kang Li, and Sean Culatana. Orchestrating Specialized Agents for Trustworthy Enterprise RAG. arXiv preprint arXiv:2601.18267, 2026.

[66] Lingyong Yan, Can Xu, Yukun Zhao, et al. DuMate-DeepResearch: An Auditable Multi-Agent System with Recursive Search and Rubric-Grounded Reasoning. arXiv preprint arXiv:2606.07299, 2026.

[67] Seyoung Song. Knowledge-Aware Iterative Retrieval for Multi-Agent Systems. arXiv preprint arXiv:2503.13275, 2025.

[68] Yuyao Zhang, Junjie Gao, Zhengxian Wu, et al. SearchOS-V1: Towards Robust Open-Domain Information-Seeking Agent Collaboration. arXiv preprint arXiv:2607.15257, 2026.

[69] Feijie Wu, Zitao Li, Fei Wei, Yaliang Li, Bolin Ding, and Jing Gao. Talk to Right Specialists: Iterative Routing in Multi-agent Systems for Question Answering. arXiv preprint arXiv:2501.07813,

2025.

[70] Jaewan Park, Solbee Cho, and Jay-Yoon Lee. Divide and Cooperate: Role-Decomposed Multi-Agent LLM Training with Cross-Agent Learning Signals. arXiv preprint arXiv:2606.10684, 2026.

[71] Seyeon Jeong, Yeonjun Choi, JongWook Kim, and Beakcheol Jang. Tool-MAD: A Multi-Agent Debate Framework for Fact Verification with Diverse Tool Augmentation and Adaptive Retrieval. arXiv preprint arXiv:2601.04742, 2026.

[72] Yupeng Cao, Chengyang He, Yangyang Yu, Ping Wang, and K. P. Subbalakshmi. MERMAID: Memory-Enhanced Retrieval and Reasoning with Multi-Agent Iterative Knowledge Grounding for Veracity Assessment. arXiv preprint arXiv:2601.22361, 2026.

[73] Hao Zhang, Longrong Yang, Lunhao Duan, Ziyang Wang, Qing-Guo Chen, and Shanshan Zhao. D2-ScaleAgent: Dual-Dimensional Scaling for Long Document Understanding. arXiv preprint arXiv:2608.16417, 2026.

[74] Y Wang, R Ren, J Li, WX Zhao, J Liu, and JR Wen. REAR: A Relevance-Aware Retrieval-Augmented Framework for Open-Domain Question Answering. arXiv preprint arXiv:2402.17497, 2024.

[75] W Yu, H Zhang, X Pan, K Ma, H Wang, and D Yu. Chain-of-Note: Enhancing Robustness in Retrieval-Augmented Language Models. arXiv preprint arXiv:2311.09210, 2023.

[76] Yaorui Shi, Sihang Li, Chang Wu, Zhiyuan Liu, Junfeng Fang, Hengxing Cai, An Zhang, and Xiang Wang. Search and Refine During Think: Facilitating Knowledge Refinement for Improved Retrieval-Augmented Reasoning. arXiv preprint arXiv:2505.11277, 2025.

[77] Shuwen Yang, Anran Wu, Xingjiao Wu, Luwei Xiao, Tianlong Ma, Cheng Jin, and Liang He. Progressive Evidence Refinement for Open-domain Multimodal Retrieval Question Answering. arXiv preprint arXiv:2310.09696, 2023.

[78] Juan Chen, Baolong Bi, Wei Zhang, Jingyan Sui, Xiaofei Zhu, Yuanzhuo Wang, Lingrui Mei, and Shenghua Liu. Rethinking All Evidence: Enhancing Trustworthy Retrieval-Augmented Generation via Conflict-Driven Summarization. arXiv preprint arXiv:2507.01281, 2025.

[79] Mayank Ravishankara. FVA-RAG: Falsification-Verification Alignment for Mitigating Sycophantic Hallucinations. arXiv preprint arXiv:2512.07015, 2025.

[80] MA Asl and M Asgari. FAIR-RAG: Faithful Adaptive Iterative Refinement for Retrieval-Augmented Generation. arXiv preprint arXiv:2510.22344, 2025.

[81] Jiaqi Wei, Hao Zhou, Xiang Zhang, Di Zhang, Zijie Qiu, Wei Wei, Jinzhe Li, Wanli Ouyang, and Siqi Sun. Retrieval is Not Enough: Enhancing RAG Reasoning through Test-Time Critique and Optimization. arXiv preprint arXiv:2504.14858, 2025.

[82] Mahdi Astaraki, Mohammad Arshi Saloot, Ali Shiraee Kasmaee, Hamidreza Mahyar, and Soheila Samiee. When Iterative RAG Beats Ideal Evidence: A Diagnostic Study in Scientific Multi-hop Question Answering. arXiv preprint arXiv:2601.19827, 2026.

[83] Jiaqi Shao, Yuxiang Lin, Munish Prasad Lohani, Yufeng Miao, and Bing Luo. Do LLM Agents Know How to Ground, Recover, and Assess? A Benchmark for Epistemic Competence in Information-Seeking Agents. arXiv preprint arXiv:2509.22391, 2025.

[84] Qi Dong, Ziheng Lin, and Ning Ding. Stateful Evidence-Driven Retrieval-Augmented Generation with Iterative Reasoning. arXiv preprint arXiv:2604.14170, 2026.

[85] L Li, D Shi, J Zhou, X Wei, M Yang, S Jin, and S Yang. Retrieval Feedback Memory Enhancement Large Model Retrieval Generation Method. arXiv preprint arXiv:2508.17862, 2025.

[86] Hongjin Qian, Zheng Liu, Peitian Zhang, Kelong Mao, Defu Lian, Zhicheng Dou, and Tiejun Huang. MemoRAG: Boosting Long Context Processing with Global Memory-Enhanced Retrieval Augmentation. The Web Conference, 2024.

[87] Xinkang Li, Rong Jiang, Xin Song, Ye Wang, Yue Han, and Changjian Li. STEC: Evidence Compression for Deep Search in Open-domain Multi-Hop QA. arXiv preprint arXiv:2607.10795, 2026.

[88] Hieu Man and Thien Huu Nguyen. Co-Evolving Graph and Text Memory for Training-Free Multi-Hop Question Answering. arXiv preprint arXiv:2607.23278, 2026.

[89] Aijun Yang, Qianxue Guo, Ziyi Huang, Yuxuan Chen, Shiyou Qian, and Jian Cao. Self-Correcting Long-Horizon Search Agents via Tree-Structured Memory. arXiv preprint arXiv:2608.10676, 2026.

[90] Yucheng Shen, Lingyong Yan, Jiulong Wu, Shuaiqiang Wang, Jianmin Wu, Dawei Yin, and Min Cao. Navigating Sparse Evidence: Agentic Visual RAG via Explicit Context Selection and Consolidation. arXiv preprint arXiv:2609.15800, 2026.

[91] Qiuchen Wang, Shihang Wang, Yu Zeng, et al. VimRAG: Navigating Massive Visual Context in Retrieval-Augmented Generation via Multimodal Memory Graph. arXiv preprint arXiv:2602.12735, 2026.

[92] Hanshu Yao, Janfeng Zhong, Niu Lian, and Jinpeng Wang. DocMemo: Dynamic Evidence Discovery via Probabilistic Memory-Guided Retrieval for Multi-Modal Document Understanding. arXiv preprint arXiv:2608.07067, 2026.

[93] Beidi Zhao, Yaoqi Chen, Yuru Feng, et al. MESA:Task-Adaptive Multi-Structure Evidence Selection for Long-Horizon Agent Memory. arXiv preprint arXiv:2608.10108, 2026.

[94] Yuhang Li and Yuchen Li. What Should an Agent Forget? Separating What Is Stored from What Is Used. arXiv preprint arXiv:2609.10263, 2026.

[95] Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, et al. Evo-Memory: Benchmarking LLM Agent Test-time Learning with Self-Evolving Memory. arXiv preprint arXiv:2511.20857, 2025.

[96] Haosheng Qian, Yixing Fan, Jiafeng Guo, Ruqing Zhang, Qi Chen, Dawei Yin, and Xueqi Cheng. VeriCite: Towards Reliable Citations in Retrieval-Augmented Generation via Rigorous Verification. arXiv preprint arXiv:2510.11394, 2025.

[97] Yuyang Huang, Bobo Li, Jiajia Song, Yuzhe Ding, Chong Teng, Fei Li, and Donghong Ji. ReCite: Agentic Reasoning for Faithful Citation. arXiv preprint arXiv:2609.09156, 2026.

[98] Jingxi Qiu, Zeyu Han, and Cheng Huang. SURE-RAG: Sufficiency and Uncertainty-Aware Evidence Verification for Selective Retrieval-Augmented Generation. arXiv preprint arXiv:2605.03534, 2026.

[99] Yiqing Zhang, Xiaozhong Liu, and Fabricio Murai. PubMed Reasoner: Dynamic Reasoning-based Retrieval for Evidence-Grounded Biomedical Question Answering. arXiv preprint arXiv:2603.27335, 2026.

[100] Yibo Zhao, Jiapeng Zhu, Zichen Ding, and Xiang Li. GRACE: Reinforcement Learning for Grounded Response and Abstention under Contextual Evidence. arXiv preprint arXiv:2601.04525, 2026.

[101] Z Li, X Guan, B Zhang, S Huang, H Zhou, S Lai, M Yan, et al. WebWeaver: Structuring Web-Scale Evidence with Dynamic Outlines for Open-Ended Deep Research. arXiv preprint arXiv:2509.13312, 2025.

[102] Yishan Li, Wentong Chen, Yukun Yan, et al. AgentCPM-Report: Interleaving Drafting and Deepening for Open-Ended Deep Research. arXiv preprint arXiv:2602.06540, 2026.

[103] Xujia Wang, Yizhe Zhang, Bin Xu, Lei Hou, and Juanzi Li. DeepWeaver: Bridging the Evidence Synthesis Gap in Open-Ended Question Answering. arXiv preprint arXiv:2608.18988, 2026.

[104] Hang Ding, Yilun Zhao, Tiansheng Hu, Manasi Patwardhan, and Arman Cohan. SciRAG: Adaptive, Citation-Aware, and Outline-Guided Retrieval and Synthesis for Scientific Literature. arXiv preprint arXiv:2511.14362, 2025.

[105] Mingyang Chen, Tianpeng Li, Haoze Sun, et al. ReSearch: Learning to Reason with Search for LLMs via Reinforcement Learning. arXiv preprint arXiv:2503.19470, 2025.

[106] Zhao Wang, Ziliang Zhao, and Zhicheng Dou. ProRAG: Process-Supervised Reinforcement Learning for Retrieval-Augmented Generation. arXiv preprint arXiv:2601.21912, 2026.

[107] Jiaxuan Gao, Wei Fu, Minyang Xie, Shusheng Xu, Chuyi He, Zhiyu Mei, Banghua Zhu, and Yi Wu. Beyond Ten Turns: Unlocking Long-Horizon Agentic Search with Large-Scale Asynchronous RL. arXiv preprint arXiv:2508.07976, 2025.

[108] Haoyue Bai, Haoyu Wang, Shengyu Chen, Zhengzhang Chen, Lu-An Tang, Wei Cheng, Haifeng Chen, and Yanjie Fu. Learning to Route: A Rule-Driven Agent Framework for Hybrid-Source Retrieval-Augmented Generation. arXiv preprint arXiv:2510.02388, 2025.

[109] Daeyoung Roh and Donghee Han. Before Reasoning Can Fail: Pre-Evidence Procedural Failures in Agentic RAG. arXiv preprint arXiv:2608.02011, 2026.

[110] Mingxuan Du, Benfeng Xu, Chiwei Zhu, Shaohan Wang, Pengyu Wang, Xiaorui Wang, and Zhendong Mao. A-RAG: Scaling Agentic Retrieval-Augmented Generation via Hierarchical Retrieval Interfaces. arXiv preprint arXiv:2602.03442, 2026.

[111] Andrea Volpini and Elie Raad. RLM-on-KG: Heuristics First, LLMs When Needed: Adaptive

Retrieval Control over Mention Graphs for Scattered Evidence. arXiv preprint arXiv:2604.17056, 2026.

[112] Zhengru Fang, Senkang Forest Hu, Zhonghao Chang, Yu Guo, Yihang Tao, Hongyao Liu, Mengzhe Ruan, Jun Huang, and Yuguang Fang. Inference-Time Budget Control for LLM Search Agents. arXiv preprint arXiv:2605.05701, 2026.

[113] Daocheng Fu, Jianbiao Mei, Licheng Wen, et al. RE-Searcher: Robust Agentic Search with Goal-oriented Planning and Self-reflection. arXiv preprint arXiv:2509.26048, 2025.

[114] Yinuo Xu, Shuo Lu, Jianjie Cheng, Meng Wang, Qianlong Xie, Xingxing Wang, Ran He, and Jian Liang. How to Train Your Deep Research Agent? Prompt, Reward, and Policy Optimization in Search-R1. arXiv preprint arXiv:2602.19526, 2026.

[115] Jialong Wu, Baixuan Li, Runnan Fang, et al. WebDancer: Towards Autonomous Information Seeking Agency. arXiv preprint arXiv:2505.22648, 2025.

[116] Kuan Li, Zhongwang Zhang, Huifeng Yin, et al. WebSailor: Navigating Super-human Reasoning for Web Agent. arXiv preprint arXiv:2507.02592, 2025.

[117] Zhixin Zhang, Xinke Jiang, Zhibang Yang, Weixuan Xu, Guohong Qiu, Xu Chu, Junfeng Zhao, and Yasha Wang. LoongReflect: Boosting Long-Horizon Reflection in Search Agents via Global Perspective Distillation. arXiv preprint arXiv:2608.11967, 2026.

[118] Junlin Liu, Jiangwang Chen, Zixin Song, Shuaiyu Zhou, Chunji Lv, Hank Wu, Kailin Jiang, Jinyang Wu, Bohan Yu, and Chenxi Zhou. From Proprietary to Open-Source: Bridging the Distribution Gap via Multi-Agent Protocol Distillation in Agentic Search. arXiv preprint arXiv:2607.24280, 2026.

[119] Yufei Ma, Zihan Liang, Ben Chen, Zhipeng Qian, Huangyu Dai, Lingtao Mao, Xuxin Zhang, Chenyi Lei, and Wenwu Ou. SD-Search: On-Policy Hindsight Self-Distillation for Search-Augmented Reasoning. arXiv preprint arXiv:2605.18299, 2026.

[120] Fan Yang, Rui Meng, and Yuxin Wen. Why Does Feedback-Augmented Self-Distillation Fail to Improve Retrieval-Interleaved Search Agents? arXiv preprint arXiv:2607.17558, 2026.

[121] SQ Yan and ZH Ling. RPO: Retrieval Preference Optimization for Robust Retrieval-Augmented Generation. arXiv preprint arXiv:2501.13726, 2025.

[122] Guangzhi Xiong, Qiao Jin, Xiao Wang, et al. Supervising the search process produces reliable and generalizable information-seeking agents. arXiv preprint arXiv:2502.13957, 2025.

[123] Peilin Wu, Mian Zhang, Kun Wan, Wentian Zhao, Kaiyu He, Xinya Du, and Zhiyu Chen. HiPRAG: Hierarchical Process Rewards for Efficient Agentic Retrieval Augmented Generation. arXiv preprint arXiv:2510.07794, 2025.

[124] Zhichao Xu, Zongyu Wu, Yun Zhou, et al. Beyond Correctness: Rewarding Faithful Reasoning in Retrieval-Augmented Generation. arXiv preprint arXiv:2510.13272, 2025.

[125] Bowen Jin, Jinsung Yoon, Priyanka Kargupta, Sercan O. Arik, and Jiawei Han. An Empirical Study on Reinforcement Learning for Reasoning-Search Interleaved LLM Agents. arXiv preprint arXiv:2505.15117, 2025.

[126] Ruifeng Yuan, Chaohao Yuan, David Dai, Yu Rong, Hong Cheng, Hou Pong Chan, and Chenghao Xiao. Understanding the Behaviors of Environment-aware Information Retrieval. arXiv preprint arXiv:2606.16817, 2026.

[127] 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. EMNLP, 2018.

[128] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop Questions via Single-hop Question Composition. Transactions of the Association for Computational Linguistics, 2022.

[129] Y Tang and Y Yang. MultiHop-RAG: Benchmarking Retrieval-Augmented Generation for Multi-Hop Queries. arXiv preprint arXiv:2401.15391, 2024.

[130] Satyapriya Krishna, Kalpesh Krishna, Anhad Mohananey, S. Schwarcz, Adam Stambler, Shyam Upadhyay, and Manaal Faruqui. Fact, Fetch, and Reason: A Unified Evaluation of Retrieval-Augmented Generation. North American Chapter of the Association for Computational Linguistics, 2024.

[131] Zijian Chen, Xueguang Ma, Shengyao Zhuang, et al. BrowseComp-Plus: A More Fair and Transparent

Evaluation Benchmark of Deep-Research Agent. arXiv preprint arXiv:2508.06600, 2025.

[132] Jialong Wu, Wenbiao Yin, Yong Jiang, et al. WebWalker: Benchmarking LLMs in Web Traversal. arXiv preprint arXiv:2501.07572, 2025.

[133] Ryan Wong, Jiawei Wang, Junjie Zhao, et al. WideSearch: Benchmarking Agentic Broad Info-Seeking. arXiv preprint arXiv:2508.07999, 2025.

[134] Yang Yao, Yixu Wang, Yuxuan Zhang, et al. Dr. Bench: A Multidimensional Evaluation for Deep Research Agents, from Answers to Reports. arXiv preprint arXiv:2510.02190, 2025.

[135] Minghao Li, Ying Zeng, Zhihao Cheng, Cong Ma, and Kai Jia. ReportBench: Evaluating Deep Research Agents via Academic Survey Tasks. arXiv preprint arXiv:2508.15804, 2025.

[136] Philippe Laban, Alexander R. Fabbri, Caiming Xiong, and Chien-Sheng Wu. Summary of a Haystack: A Challenge to Long-Context LLMs and RAG Systems. arXiv preprint arXiv:2407.01370, 2024.

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

[138] Minjun Zhu, Yixuan Weng, Shizhu He, Kang Liu, and Jun Zhao. Towards Graph-hop Retrieval and Reasoning in Complex Question Answering over Textual Database. arXiv preprint arXiv:2305.14211, 2023.

[139] Lexiang Hu, Yanzhao Zhang, Mingxin Li, Dingkun Long, Yikang Li, Fuwei Zhang, Yisen Wang, and Zhouchen Lin. VisDocAgentBench: Benchmarking Agents for Visually Rich Document Retrieval. arXiv preprint arXiv:2608.17889, 2026.

[140] Alireza Salemi and Hamed Zamani. Evaluating Retrieval Quality in Retrieval-Augmented Generation. arXiv preprint arXiv:2404.13781, 2024.

[141] Sheroz Shaikh. Dissecting Agentic RAG: A Component Ablation for Multi-Hop QA with a Local 7B Model. arXiv preprint arXiv:2606.21553, 2026.

[142] Moshe Lahmy and Roi Yozevitch. Replace, Don't Expand: Mitigating Context Dilution in Multi-Hop RAG via Fixed-Budget Evidence Assembly. arXiv preprint arXiv:2512.10787, 2025.

[143] Z Wang, X Zhu, S Lin, H Xue, M Guo, and Y Zhang. RAGRouter-Bench: A Dataset and Benchmark for Adaptive RAG Routing. arXiv preprint arXiv:2602.00296, 2026.

[144] Haoyu Han, Li Ma, Yu Wang, et al. RAG vs. GraphRAG: A Systematic Evaluation and Key Insights. arXiv preprint arXiv:2502.11371, 2025.

[145] S Wu, S Zhao, M Yasunaga, K Huang, K Cao, Q Huang, et al. STaRK: Benchmarking LLM Retrieval on Textual and Relational Knowledge Bases. arXiv preprint arXiv:2404.13207, 2024.

[146] Partha Sarathi Purkayastha. When Thinking Before Retrieval Hurts: TraceBound Diagnostics for Adaptive Knowledge-Graph Retrieval. arXiv preprint arXiv:2607.24800, 2026.

[147] M Glockner, X Jiang, LFR Ribeiro, I Gurevych, et al. NeoQA: Evidence-based Question Answering with Generated News Events. arXiv preprint arXiv:2505.05949, 2025.

[148] Dongzhuoran Zhou, Yuqicheng Zhu, Xiaxia Wang, Hongkuan Zhou, Yuan He, Jiaoyan Chen, Steffen Staab, and Evgeny Kharlamov. What Breaks Knowledge Graph based RAG? Benchmarking and Empirical Insights into Reasoning under Incomplete Knowledge. arXiv preprint arXiv:2508.08344, 2025.

[149] Krti Tallam. Partial Evidence Bench: Benchmarking Authorization-Limited Evidence in Agentic Systems. arXiv preprint arXiv:2605.05379, 2026.

[150] Peilin Zhou, Bruce Leon, Xiang Ying, et al. BrowseComp-ZH: Benchmarking Web Browsing Ability of Large Language Models in Chinese. arXiv preprint arXiv:2504.19314, 2025.

[151] Tu Vu, Mohit Iyyer, Xuezhi Wang, et al. FreshLLMs: Refreshing Large Language Models with Search Engine Augmentation. Annual Meeting of the Association for Computational Linguistics, 2023.

[152] Dongzhi Jiang, Renrui Zhang, Ziyu Guo, et al. MMSearch: Benchmarking the Potential of Large Models as Multi-modal Search Engines. arXiv preprint arXiv:2409.12959, 2024.

[153] Ruofan Wu, Peiran Xu, Xiaolong Li, et al. Benchmarking Hybrid Deep Research Across Database Querying and Web Search. arXiv preprint arXiv:2609.09410, 2026.

[154] Maojia Song, Renhang Liu, Xinyu Wang, Yong Jiang, Pengjun Xie, Fei Huang, Jingren Zhou, Dorien Herremans, and Soujanya Poria. Demystifying deep search: a holistic evaluation with hint-free

multi-hop questions and factorised metrics. arXiv preprint arXiv:2510.05137, 2025.

[155] Yunjia Xi, Jianghao Lin, Menghui Zhu, et al. InfoDeepSeek: Benchmarking Agentic Information Seeking for Retrieval-Augmented Generation. arXiv preprint arXiv:2505.15872, 2025.

[156] Thinh Pham, Nguyen Nguyen, Pratibha Zunjare, Weiyuan Chen, Yu-Min Tseng, and Tu Vu. SealQA: Raising the Bar for Reasoning in Search-Augmented Language Models. arXiv preprint arXiv:2506.01062, 2025.

[157] Sriram Selvam and Anneswa Ghosh. Equal Accuracy, Unequal Evidence: Search APIs as Decision Surfaces for Tool-Using Agents. arXiv preprint arXiv:2607.10198, 2026.

[158] Liana Patel, Negar Arabzadeh, Harshit Gupta, Ankita Sundar, Ion Stoica, Matei Zaharia, and Carlos Guestrin. DeepScholar-Bench: A Live Benchmark and Automated Evaluation for Generative Research Synthesis. arXiv preprint arXiv:2508.20033, 2025.

[159] Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. FactScore: Fine-grained atomic evaluation of factual precision in long form text generation. arXiv preprint arXiv:2305.14251, 2023.

[160] Jerry Wei, Chengrun Yang, Xinying Song, et al. Long-form factuality in large language models. arXiv preprint arXiv:2403.18802, 2024.

[161] Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. Enabling Large Language Models to Generate Text with Citations. arXiv preprint arXiv:2305.14627, 2023.

[162] Nelson F. Liu, Tianyi Zhang, and Percy Liang. Evaluating Verifiability in Generative Search Engines. arXiv preprint arXiv:2304.09848, 2023.

[163] Yijia Shao, Yucheng Jiang, Theodore A. Kanell, Peter Xu, Omar Khattab, and Monica S. Lam. Assisting in Writing Wikipedia-like Articles From Scratch with Large Language Models. arXiv preprint arXiv:2402.14207, 2024.

[164] João Coelho, Jingjie Ning, Jingyuan He, et al. DeepResearchGym: A Free, Transparent, and Reproducible Evaluation Sandbox for Deep Research. arXiv preprint arXiv:2505.19253, 2025.

[165] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, et al. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. arXiv preprint arXiv:2306.05685, 2023.

[166] Lauren Pothuru. When Failures Propagate: Causal Failure Attribution in Agentic Retrieval-Augmented Generation. arXiv preprint arXiv:2608.20627, 2026.

[167] Kevin Wu, Eric Wu, and James Zou. ClashEval: Quantifying the tug-of-war between an LLM's internal prior and external evidence. Advances in Neural Information Processing Systems, 2024.

[168] Yue Pan, Ziheng Zhang, Junxiang Lei, Changhao Jia, Qingyi Si, and Hongcheng Guo. FORGE: Research-Trajectory Hijacking Attacks on Deep Research Agents. arXiv preprint arXiv:2607.04718, 2026.

[169] Mingyan Wu, Han Yang, Omer Ben-Porat, and Yftah Ziser. When Knowledge Is Not Free: Cost-Aware Evidence Selection in Retrieval-Augmented Generation. arXiv preprint arXiv:2606.02245, 2026.

[170] Payal Bajaj, Daniel Campos, Nick Craswell, et al. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. arXiv preprint arXiv:1611.09268, 2016.

[171] Gongbo Zhang, Yifan Peng, and Chunhua Weng. Improving Retrieval-Augmented Generation without Taxonomy-based Error Categorization. arXiv preprint arXiv:2605.18772, 2026.

[172] Saber Zerhoudi, Michael Granitzer, and Jelena Mitrovic. Metadata, Structure, or Strategy? A Decomposition of RAG Context Enrichment. arXiv preprint arXiv:2606.29645, 2026.

[173] F Fang, Y Bai, S Ni, M Yang, X Chen, and R Xu. Enhancing Noise Robustness of Retrieval-Augmented Language Models with Adaptive Adversarial Training. arXiv preprint arXiv:2405.20978, 2024.

[174] Jiankun Wang, Yisen Gao, Ziwei Zhang, Xingcheng Fu, Jiaxin Bai, and Chen Gao. Does More Retrieved Evidence Help Visual Retrieval-Augmented Generation with Diffusion Language Models? arXiv preprint arXiv:2608.07006, 2026.

[175] Boyang Liu, Senjie Jin, Peixin Wang, et al. CAFE: Self-Improving Search Agents Need Co-Evolving Feedback. arXiv preprint arXiv:2608.24794, 2026.

[176] Shengjie Ma, Chenlong Deng, Jiaxin Mao, Jiadeng Huang, Teng Wang, Junjie Wu, Changwang Zhang, and Jun Wang. Proof-of-Use: Mitigating Tool-Call Hacking in Deep Research Agents. arXiv

preprint arXiv:2510.10931, 2025.

[177] Heydar Soudani, Elizabeth Lingg, Faegheh Hasibi, and Navid Rekabsaz. When Deep Research Agents Stagnate: Enhancing Reasoning with Retrieval-Aware Agent Control. arXiv preprint arXiv:2608.15191, 2026.

[178] Tiannuo Yang, Zebin Yao, Bowen Jin, Lixiao Cui, Yusen Li, Gang Wang, and Xiaoguang Liu. Demystifying and Enhancing the Efficiency of Large Language Model Based Search Agents. arXiv preprint arXiv:2505.12065, 2025.

[179] Yilun Zhao, Jinbiao Wei, Tingyu Song, Siyue Zhang, Chen Zhao, and Arman Cohan. Rethinking Reasoning-Intensive Retrieval: Evaluating and Advancing Retrievers in Agentic Search Systems. arXiv preprint arXiv:2605.04018, 2026.

[180] Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. Benchmarking Large Language Models in Retrieval-Augmented Generation. arXiv preprint arXiv:2309.01431, 2023.

[181] Laura Dietz, Bryan Li, Eugene Yang, Dawn Lawrie, William Walden, and James Mayfield. Insider Knowledge: How Much Can RAG Systems Gain from Evaluation Secrets? arXiv preprint arXiv:2601.13227, 2026.