A New Role for Relevance: Guiding Corpus Interaction in Agentic Search

Jiangnan Li[1,*], **Yuqing Li**[2,*], **Mo Yu**[1,†], **Jinchao Zhang**[1], **Jie Zhou**[1,†]
[1]Tencent [2]IIE-CAS
{jiangnanli,moyumyu}@tencent.com, liyuqing@iie.ac.cn

github.com/LeqsNaN/RARG

Abstract

Relevance is a query-dependent estimate of whether a document or excerpt contains useful evidence. Existing retrieval agents use relevance to select top-k content, but document relevance alone cannot localize, compose, or verify the evidence required by complex questions. Direct Corpus Interaction (DCI) enables such fine-grained operations through grep-style exploration, but its relevance-agnostic search can expose useful clues late and delay convergence. Recent advances use relevance to narrow the corpus into a working space for interaction. Once interaction begins, however, relevance still does not directly guide which documents grep searches first or distinguish informative excerpts from a broad set of matches to let LLMs see them first. We introduce the **Relevance-Aware RipGrep Search Agent (RARG)**, which turns relevance into an execution prior for corpus interaction. RARG provides coarse-to-fine relevance guidance: it orders documents for sequential `ripgrep` traversal to expose globally relevant clues earlier, initializes promising entry points with query-relevant paragraphs, and reranks grep matches to surface informative excerpts that document-level ranking may otherwise obscure. Across challenging browse question answering and reasoning-intensive retrieval, RARG improves the accuracy–efficiency frontier over retrieval-based and direct-interaction agents. These results demonstrate that relevance-aware interaction enables faster and more reliable search convergence.

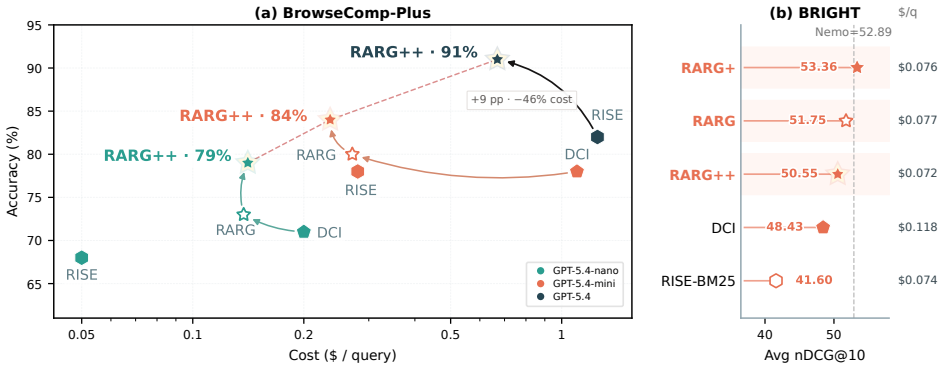

Figure 1: Accuracy/nDCG@10 versus interaction cost (average tool calls) on BrowseComp-Plus and BRIGHT. By turning relevance into an execution prior over `rg` exploration, RARG advances the accuracy–efficiency frontier over retrieval-based and direct-interaction agents.

*equal contributions; † corresponding authors

1 Introduction

Relevance has long served as the organizing principle of information retrieval. In this paper, we use *relevance* to mean a query-dependent estimate of how likely a document, passage, or matched excerpt is to contain evidence useful for the current information need. Modern dense retrievers, which are widely deployed in RAG [7, 11, 14, 23], instantiate this estimate with embedding similarity [10, 27, 32], while sparse retrievers use lexical matching signals [21]. In either case, relevance is a useful but imperfect prior: it indicates where evidence may be found, not whether the evidence is sufficient, correctly localized, or properly combined with other clues.

Retrieval agents conventionally use this prior to rank the corpus and expose a top-k set of documents or snippets to the language model [9, 12, 13, 24, 28, 33]. This interface is scalable, but it conflates document relevance with evidence utility. A relevant document may hide the decisive fact in a small span omitted by snippet truncation [17], while in multi-hop search its usefulness may emerge only after an intermediate entity is discovered [26]. Independent query–document ranking cannot itself localize, connect, compare, or verify such evidence. An agent may therefore retrieve the right documents yet still fail to produce the answer, as observed in prior studies [16]. Relevance can identify promising regions of a corpus, but a ranked top-k view provides limited support for exploiting the evidence within and across them.

Direct Corpus Interaction (DCI) addresses this interface bottleneck by allowing an agent to search raw documents with general-purpose terminal tools such as pattern matching and local file reads [16]. Fine-grained interaction lets the agent combine lexical constraints, follow newly discovered entities, and inspect the exact context surrounding a match. However, DCI largely removes corpus-level relevance guidance: a pattern-matching command scans documents as if all locations were equally promising, so useful clues may appear late or be lost when its output is truncated. The agent can consequently spend many unproductive interactions exploring fruitless search angles before reaching a useful anchor. We refer to how quickly and reliably these iterative interactions narrow an open-ended search into verified evidence as *search convergence*. Without a mechanism for prioritizing promising locations, DCI's convergence deteriorates as the corpus grows, leading to rapidly increasing tool use and lower accuracy [16].

Recent advances, including Pi-Serini, RISE, and DR-DCI, address this scaling problem by using sparse or dense retrieval to construct or expand a bounded candidate space before or during interaction [8, 18, 35]. Despite differences in their interfaces, they share a common strategy: first narrow the corpus to a set of relevant documents, and then let the agent explore this set through selective reads or grep-style operations. This avoids repeated full-corpus scans while preserving richer interaction than a conventional top-k snippet interface. However, relevance remains primarily a mechanism for constructing the space, rather than a fine-grained execution signal for the interactions within it. It does not directly determine both the traversal order of a pattern-matching operation and which local matches remain visible under output truncation.

We argue that relevance should guide corpus interaction, rather than merely select its inputs, to accelerate and stabilize search convergence. This requires relevance at two complementary resolutions. *Global relevance* should determine where a pattern-matching operation searches first, so that promising documents are encountered before less relevant ones. *Local relevance* should determine which matched excerpts become visible when the raw output exceeds the agent's observation budget. This view preserves the high-resolution, compositional interface of DCI while using retrieval scores as an execution prior instead of a hard evidence bottleneck.

Based on this principle, we introduce the **Relevance-Aware RipGrep Search Agent (RARG)**.[2] RARG ranks documents with an embedding retriever according to an agent-generated query and then makes `rg` traverse them sequentially in that order, so that matches from more relevant documents are exposed earlier—turning document-level relevance into search order rather than top-k content selection. We further study two extensions. RARG+ seeds the agent with a small set of query-relevant paragraphs as an entry point [15] from which it can formulate its first precise search. RARG++ reranks a wider pool of `rg` matches by combining the global retrieval objective with the local pattern-matching focus, so that locally informative excerpts—including evidence in lower-ranked documents—can compete for the agent's limited observation budget. Together, the three levels determine where

[2] Throughout the paper, `grep` denotes the generic grep-style pattern-matching operation, whereas `rg` denotes the concrete `ripgrep` command used by our implementation.

interaction begins, which documents it traverses first, and which local observations reach the model, helping `rg` reach useful evidence earlier and converge with fewer unproductive interactions.

We evaluate our methods on two complementary settings: challenging question answering on the 100-query BrowseComp-Plus setting following RISE [1, 35], including a controlled expansion from 100K to 1M documents, and reasoning-intensive retrieval on four BRIGHT domains [25]. As shown in Fig. 1, with GPT-5.4-mini on the 100K-document BrowseComp-Plus corpus, RARG++ achieves 84% accuracy, compared with 78% for both RISE and DCI, while requiring 23.9 average tool calls compared with 28.7 and 99.1, respectively. With GPT-5.4, RARG++ reaches 91% accuracy, exceeding RISE by 9 points. After expanding the corpus to 1M documents, RARG++ retains 79% accuracy, compared with 69% for RISE. On BRIGHT, RARG+ achieves 53.36 average nDCG@10, outperforming DCI, RISE, and the retrieval-specialized NeMo agent. The gains in accuracy and tool efficiency indicate that relevance-aware interaction enables both faster and more reliable search convergence, while also improving the identification of reasoning-intensive evidence.

Our contributions are threefold:

- We identify a missing role of relevance in agentic search: beyond selecting retrieved content or managing a workspace, relevance can directly guide the execution and observation of corpus interactions.
- We propose RARG, a relevance-aware `rg` search agent that preserves document rankings during corpus traversal, and introduce entry-point initialization and match-level reranking to provide coarse-to-fine guidance for faster and more reliable search convergence.
- We demonstrate a stronger accuracy–efficiency trade-off than retrieval-only agents, unrestricted DCI, and retrieval-constructed interaction spaces across challenging QA, corpus scaling, and reasoning-intensive retrieval.

2 Related Work

2.1 From Document Retrieval to Agentic Retrieval

Relevance estimation is a central organizing principle of modern information retrieval. Sparse methods such as BM25 rank documents using lexical statistics [21], whereas dense retrievers encode queries and documents into a shared representation space for semantic matching [10].

Retrieval-augmented generation uses such retrievers to ground language models in external knowledge and has been studied in settings involving long inputs [3, 5, 29], many-shot demonstrations [4], context compression [2], and globally distributed evidence [11]. However, these approaches still rely on a context assembled before reasoning that must anticipate downstream evidence needs, while longer ranked contexts do not guarantee that a model will reliably use decisive evidence buried among less useful material [17].

Agentic retrieval relaxes this one-shot assumption by allowing a model to control retrieval over multiple steps, interleaving reasoning with retrieval actions so that intermediate deductions can shape subsequent information needs [26, 28, 30, 31, 34]. Yet many such systems still treat a conventional retriever as the primary corpus interface, returning top-k results over which the agent can reason only after retrieval. This design can conflate document-level relevance with evidence utility: a decisive clue may lie in a short span omitted during passage selection, while a document's value may become apparent only after an intermediate entity is discovered. These limitations motivate corpus interfaces that combine scalable corpus-level prioritization with finer-grained evidence localization, composition, and verification.

2.2 Corpus Interfaces for Agentic Search

Beyond retrieval quality, a key design axis is the *resolution* of the interface through which an agent accesses the corpus. Retriever-mediated systems, including strong lexical pipelines such as Pi-Serini [8], expose ranked document views and thus confine the agent largely to evidence selected in advance. Direct Corpus Interaction (DCI) replaces this constraint with a high-resolution interface, exposing the raw corpus as files that an agent manipulates through pattern matching, local reads, and shell commands, thereby supporting exact lexical constraints, entity following, and cross-file composition [16]. GrepSeek further shows that such interaction policies can be learned by a compact

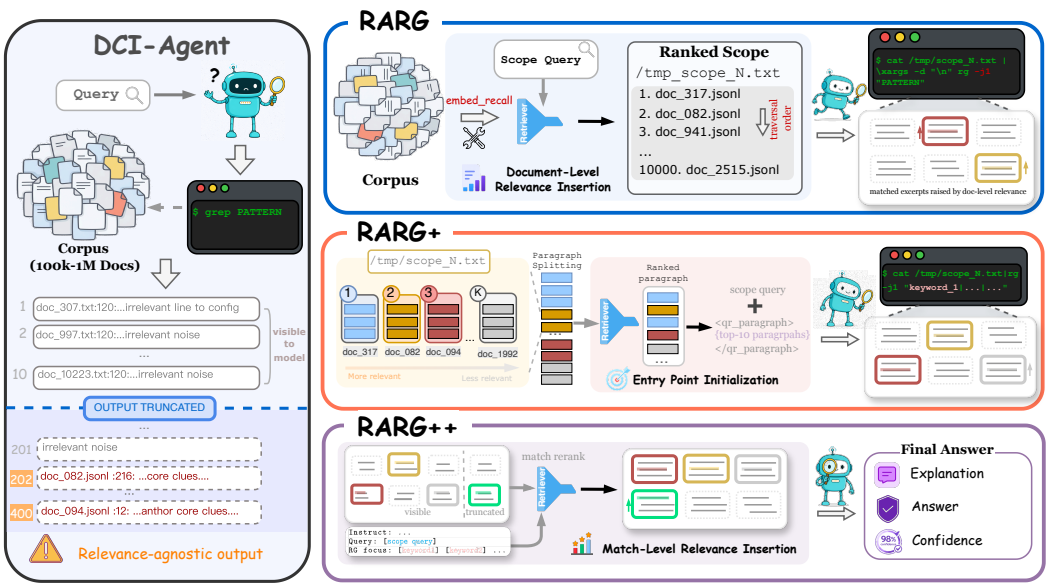

Figure 2: Overview of RARG. *embed_recall* ranks the corpus into a scope file, and `rg` scans it in ranked order so relevant documents surface first (RARG). RARG+ adds query-relevant paragraphs as an entry point; RARG++ reranks `rg` matches to keep informative excerpts from lower-ranked documents visible. Document-level relevance sets where `rg` searches first; match-level relevance sets which matches reach the model.

agent through verified trajectories and reinforcement learning [22]. This flexibility, however, comes with limited built-in corpus-level prioritization: unrestricted operations may traverse the corpus as if all files were equally promising, causing cost to grow with the number of distractors.

RISE [35] and DR-DCI [18] address this scalability tension by bounding direct interaction. RISE reframes retrieval as constructing an *interaction space*, using BM25 to select a subset preprocessed for shell-style navigation, whereas DR-DCI treats retrieval as an agent-callable action that progressively expands a persistent workspace. In both, relevance is used primarily to construct or expand the searchable space, but is not explicitly propagated to order local traversal or prioritize matches when raw output exceeds the observation budget.

This exposes a gap between *estimating* corpus-level relevance and *using* it during interaction. Rather than treating relevance as a one-time filter on the searchable corpus, our approach carries it into direct corpus interaction as an execution prior, guiding where search begins, the order in which documents are traversed, and which local matches reach the model under a limited observation budget.

3 Method

Since `grep` matches keywords or patterns rather than relevance, its hits need not bear on the question, forcing the agent through idle search steps. We therefore introduce the Relevance-Aware RipGrep (RARG) agent as illustrated in Fig. 2, which guides `grep` toward better and faster exploration.

3.1 DCI-based Agent Structure

RARG is built entirely on DCI-Agent-Lite, a search agent that exposes only two tools: *Bash*(`command`) and *Read*(`doc_path`, `offset`, `limit`). *Bash* mainly issues `grep` commands to find local matches across the corpus, and *Read* returns lines `offset` to `offset+limit` of a document when a match calls for more context.

Since the tool arguments are generated freely by the LLM, *Bash* may also receive commands such as {`ls, find, python, cat, ...`}. Because the corpus can be very large, running `ls/find` over

the whole corpus can stall for a long time; we therefore forbid `ls`/`find` whenever they target the entire corpus.

The agent keeps issuing tool calls until it is confident enough to answer, but unbounded interaction quickly overflows the context. DCI-Agent-Lite mitigates this with four levels of context management; we adopt the default Level-3 compaction: (1) truncate a single tool result to C characters; (2) once the context length exceeds a threshold, retain the tool results of the most recent T turns and replace older ones with "[cleared]" placeholders.

3.2 Document-Level Relevance Insertion

As argued above, relevance remains essential for `grep` to locate clues faster and more reliably; the central question is how to inject relevance into `grep`. Embedding retrieval, the most common way to rank documents by relevance to a question, is known to underperform plain `grep` when used to feed top-ranked documents directly to the LLM [16]. We instead treat retrieval as a "support" that supplies relevance guidance to `grep` (the "carry"), rather than as the evidence channel itself.

We first inject document-level relevance into `grep` via a new tool, *embed_recall*(`scope_query`), which takes an LLM-generated query. Rather than returning document contents, it writes the ranked document paths to a temporary file `\tmp\scope_{N}.txt` and returns only the mapping "`\tmp\scope_{N}.txt` → query" to the LLM. Recording all paths is unnecessary for very large corpora, both to limit processing cost and because relevant documents almost always rank near the top; we thus cap the scope at 10,000 paths, which still yields very high evidence recall—hence the term "scope". The LLM is then asked to `cat` the scope file and **search along the path order**, so that matches in more semantically similar documents surface earlier. Concretely, we instruct the LLM to use the following command:

```
cat /tmp/scope_{N}.txt | xargs -d '\n' rg [OPTIONS] "PATTERN"
```

This appends every doc path in the scope to the `rg` block, restricting the search to those documents. However, `rg` reads files in parallel (multi-threaded) and emits matches in the order threads finish, which destroys the path ordering we inject. We therefore add a rule-based pass that regex-matches `rg` invocations and injects `-j1` to force sequential, single-threaded scanning (`rg -j1 [OPTIONS]`).

The LLM is instructed to begin each episode by calling *embed_recall* with the original question and then `rg`-searching (with the internal `-j1` insertion) within the resulting scope; it may call *embed_recall* again to form a new scope once `rg` exhausts the current angles. During compaction, the "`\tmp\scope_{N}.txt` → scope query" mappings are retained rather than cleared, so the LLM always knows which scopes are available. We call this method RARG. Refer to prompts and tool definitions in Appendix A.

3.3 Entry Point Initialization

The agent starts from the question alone, and *embed_recall* adds no content to enrich it, so `rg` may waste several steps before finding a good entry point for convergence. To supply such an entry point without an extra LLM turn, we append a few query-relevant paragraphs from the top-ranked scope documents to the *embed_recall* output.

Specifically, we split the top-X documents of a scope into paragraphs of 400–1000 characters, encode them with the embedding model, score them against the scope query, and keep the top-10. These paragraphs are wrapped in <qr_paragraph></qr_paragraph> and appended after the scope mapping, serving as a starting point for the LLM rather than as the answer itself. They are cleared upon compaction. We denote RARG with this initialization as RARG+.

3.4 Match-Level Relevance Insertion

To further speed up convergence, each search step should also account for the fine-grained relevance between individual matched excerpts and the search goal. As noted in 3.1, the `rg` output is truncated and keeps only some of the matches, especially within the top-ranked documents of RARG. Under document-level relevance alone, an informative excerpt can be diluted by the surrounding document

content in the embedding space, giving its document a low rank and leaving the excerpt invisible. We therefore use the embedding model to rerank a broader pool of matches and present only the top ones.

A challenge of per-step match reranking is that the embedding model needs a dedicated query capturing both the global search goal and the local `rg` intent. We explore two options: (1) a constructed query and (2) a generative query.

By construction, an *embed_recall* call yields a scope and its query, which `rg` then explores over several steps; the scope query captures the global goal, and the `rg` keywords/patterns capture the local intent. Hence, when `rg cats a \tmp\scope_{N}.txt`, we convert its patterns into a keyword sequence by rules and concatenate them with the scope query as:

```
Instruct: ...
Query: [scope query]
RG focus: [keyword1] [keyword2] ...
```

This constructed query reranks up to M matches and selects the top m ($m < M$), giving more candidate clues a chance to reach the LLM. Reranking is skipped when the number of matches is below m.

For the generative query, we replace *Bash* with *RerankAwareBash*(`command` , `rerank_query` =None), where the LLM supplies `rerank_query` only when it intends to run `rg` and leaves it None otherwise. The LLM is instructed to write a `rerank_query` that reflects both the final goal and the local retrieval intent. In our experiments, however, this generative variant degrades performance. We denote RARG with match-level reranking as RARG++, using the constructed query by default.

4 Experiments

4.1 Experiment Setup

Benchmarks. We evaluate the effectiveness and efficiency of our methods in two scenarios: challenging browse question answering and reasoning-intensive retrieval. For QA, we follow RISE and use BrowseComp-Plus [1] (BC+), which answers difficult browsing questions over a fixed 100K-document corpus; due to cost, RISE evaluates on a 100-query sample. To study larger search spaces, we further expand the corpus with 900K long, noisy documents sampled from FineWeb-Edu, as longer documents make search harder by offering more opportunities for incidental matches against noisy content. We compute accuracy with the LLM-as-judge (GPT-5.1) prompt of DCI. For retrieval, we follow DCI and use four BRIGHT [25] subsets: biology, earth science, economics, and robotics. BRIGHT is harder than conventional IR benchmarks, as it demands in-depth reasoning rather than shallow semantic matching. The four subsets contain 103/116/103/101 questions over corpora of 57K/121K/50K/62K documents, and we report nDCG@10.

Implementation Details. We run all experiments with GPT-5.4-mini (medium thinking effort), GPT-5.4-nano (high thinking effort) [19], and GPT-5.4 (medium thinking effort) [20]. As described in Section 3, RARG is built on the DCI-Agent-Lite harness, with a few adaptations. For `rg`, we cap the number of returned matches at 30 for BC+ and 60 for BRIGHT, and truncate each match to 1,000 characters for BC+ and 500 for BRIGHT; the resulting output is at most 30,000 + 30 (60)$\times$path_name_length characters, far below DCI's 50 KB. DCI compaction keeps the tool results of the most recent 12 turns, but since GPT often calls several tools per turn, this retains too many results; we instead keep the most recent 40 tool results and raise the compaction threshold to 230K to better exploit the cache. For BC+, we use Qwen3-Embedding-4B [32] (Q3E) for document ranking, initialization, and match reranking. For BRIGHT, we use llama-nv-embed-reasoning-3b[3] (NV) for document ranking and initialization, and Q3E for match reranking. NV is a reasoning-intensive embedding model used by the strongest BRIGHT agent, NeMo [6], whose setting we follow; we found NV poor at ranking short matches, hence the fallback to Q3E for reranking. We set the reranking pool to M=500 with m=30/60, and cap the number of turns at 100. RARG is implemented in Python and tested on Linux with one H20 GPU.

[3] https://huggingface.co/nvidia/llama-nv-embed-reasoning-3b

Table 1: Main results on 100-query (RISE version) BrowseComp-Plus with a 100K-document corpus. Turns and Tools denote the average agent turns and tool calls. Search, Bash, and Read report the average number of calls to each tool. For our methods, Search calls are *embed_recall*. The highest Acc and the fewest Turns/Tools are in bold; Parts of Search/Bash are underlined to show different behaviors of agents. $^{\triangle}$ denotes the results are cited from RISE [35]; $^{?}$ denotes that RISE does not mention what thinking effort is activated.

Model	Method	**Acc↑**	Turns	Tools	Search	Bash	Read
GPT-5.4 mini (medium effort)	RISE$^{\triangle}$	78	24.3	28.7	13.1	9.2	6.4
	RISE-BM25$^{\triangle}$	77	23.0	29.6	14.8	9.6	5.2
	RISE-Q3-Emb-4B	69	28.9	35.9	22.1	9.1	4.7
	Retrieval-Agent$^{\triangle}$	68	29.2	38.9	37.0	–	1.9
	DCI$^{\triangle}$	78	48.8	99.1	–	90.3	8.8
	RARG	80	18.2	29.8	1.2	27.2	1.4
	RARG+	81	20.2	29.6	1.6	26.6	1.3
	RARG++	**84**	**17.6**	**23.9**	1.5	21.1	1.3
	RARG++ (generative)	75	15.8	17.8	1.2	15.4	1.3
GPT-5.4-nano (high effort)	RISE$^{\triangle}$	68	23.4	28.7	10.4	8.4	4.6
	RISE-BM25$^{\triangle}$	64	**22.0**	**29.6**	10.2	8.1	3.7
	DCI$^{\triangle}$	71	45.7	126.5	–	119.4	7.1
	RARG	73	39.9	41.4	9.1	29.5	2.8
	RARG+	74	40.2	39.8	14.5	22.4	2.9
	RARG++	**79**	36.0	36.1	9.6	23.4	3.1
GPT-5.4 (medium effort)	RISE$^{?\triangle}$	82	32.20	34.30	11.00	16.20	7.10
	RARG++	**91**	**13.59**	**25.43**	1.58	19.46	4.39

Compared Agent Methods. We compare against the following agents. **DCI-Agent-Lite** [16] is a pure `grep` agent equipped only with *Bash* and *Read*. **RISE** [35] builds an interaction space via BM25 retrieval and augments the retrieved documents with navigational structure, which the agent then explores with shell tools. **RISE-BM25** is the ablation that keeps the BM25 interaction space but uses the original, unstructured documents without this navigational processing. We run both DCI-Agent-Lite and the RISE-based baselines using the official RISE implementation [4]. We additionally evaluate **RISE-Q3-Emb-4B**, which replaces the BM25 retriever with Q3E, and **Retrieval-Agent**, the conventional retrieval baseline released with RISE. **NeMo Agent** [6] is a retrieval-oriented agent that iteratively recalls and ranks candidate documents to form the final output; we use its official implementation [5] with NV as the embedding model. On BRIGHT, the default DCI IR prompt performed substantially worse, so we modified it to explicitly require the agent to rank 10 document names. Finally, although a pure embedding agent is a natural comparison, prior work [16] reports its poor performance on BC+; we confirmed this in early experiments but did not run it on the full evaluation set to conserve budget.

4.2 Results on Challenging QA

Table 1 shows that RARG attains the best accuracy on every backbone while sharply reducing interaction cost. RARG++ reaches 84%/79% on GPT-5.4-mini/nano and 91% on GPT-5.4, exceeding the strongest baseline by 6/8/9 points, yet uses 23.9/25.43 tool calls against 99.1 for DCI and 28.7/34.3 for RISE (except the GPT-5.4-nano setting). This confirms our central claim: guiding interaction with relevance, rather than replacing it, converts the accuracy of relevance-agnostic DCI into a far cheaper search.

The underlined *Search*/*Bash* counts expose *how* relevance is used. RISE uses retrieval to build a working space that bounds the corpus and reduces `rg` exploration steps. Yet every space-constructing *Search* also returns snippets (the first few hundred characters) of the top documents, so retrieval simultaneously serves as an information channel to the model. This has the risk of incentivizing issuing more *Search* calls, since each one directly supplies content. As we use Q3E while RISE was

[4] `https://github.com/texttron/RISE`

[5] `https://github.com/NVIDIA/NeMo-Retriever/tree/main/retrieval-bench`

Table 2: Results after scaling 100-query BrowseComp-Plus from 100K to 1M documents using GPT-5.4-mini (medium reasoning).

Model	Method	**Acc**↑	Turns↓	Tools↓	Search	Bash	Read
GPT-5.4 mini (medium effort)	RISE-BM25	69	25.4	32.1 (+2.5)	20.3 (+5.5)	8.3 (-1.3)	3.5 (-1.7)
	RARG	78	19.3	31.8 (+2.0)	1.5 (+0.3)	28.7 (+1.5)	1.7 (+0.3)
	RARG+	78	21.9	30.4 (+0.8)	1.5 (-0.1)	27.5 (+0.9)	1.4 (+0.1)
	RARG++	**79**	**17.8**	**24.7** (+0.8)	1.4 (-0.1)	22.0 (+0.9)	1.4 (+0.1)

designed around BM25, we further run RISE-Q3-Emb-4B to control for retriever strength; it does not help and in fact issues more *Search* calls, indicating that RISE stays adapted to its BM25 interface rather than a stronger dense retriever. RARG inverts this balance: except on nano, it issues only 1.2–1.6 *embed_recall* calls to fix a document-level search order, then delegates the bulk of exploration to scoped `rg`. Relevance thus acts as an execution prior over `rg` traversal, not as the evidence channel itself. On nano, the pattern is less clean (higher and more variable *embed_recall*), which we attribute to weaker instruction following and revisit in the behavior analysis. Despite this, our methods with GPT-5.4-nano still achieve the best performance.

Within RARG, the variants trace the intended coarse-to-fine progression: from RARG to RARG+ to RARG++, mini accuracy rises 80→81→84 while tools fall 29.8→29.6→23.9, and nano shows the same ordering (73/74/79). Each level of relevance therefore adds accuracy *and* accelerates convergence, directly echoing our three contributions. The generative variant breaks this trend: it converges fastest (17.8 tools) but drops to 75%. We attribute this to a train–evaluation gap—forcing the model to emit an explicit rerank query perturbs the *Bash*/`rg` behavior it was trained on. Since it does improve convergence, closing this gap is a promising direction for future work.

4.3 Scaling the Corpus

To test whether the relevance-guided interaction remains useful as the search space grows, we expand the BC+ corpus from 100K to 1M documents. The additional 900K documents are long FineWeb-Edu articles, and therefore introduce substantially more opportunities for incidental lexical matches and irrelevant context.

As shown in Table 2, scaling to 1M documents lowers all methods, with RISE-BM25 falling from 77% to 69% and RARG/RARG+/RARG++ from 80/81/84 to 78/78/79. This contradicts the non-decreasing trend RISE [35] reports under scaling. The difference may come from the design: our 900K added documents are *long* FineWeb-Edu articles, which greatly increase the chance of incidental lexical matches and inject far more distracting context into each `rg` scan. Although performance drops for all search agents, the cost seems to be less affected by scaling, because the number of tools increases only slightly. We think the reason may be that retrievers (Q3E or BM25) are less affected, but `rg` is not immune to distractors—more noise per file degrades match quality just as it would any local lexical search. The distracted local matches introduce error to delude the judgment of LLMs. We analyze RARG's behavior on the 1M corpus in Section 4.5 to support the claim. Although RARG++degrades with corpus scaling, it keeps a 10-point margin over RISE-BM25. Corpus-induced interference is only partially absorbed by match-level reranking, and therefore, how to reduce this weakness of local matching can be future work to study.

4.4 Results on Reason-Intensive Retrieval

BRIGHT and BC+ reward opposite search shapes. In ranking-style retrieval, the agent is encouraged to first recall as many relevant documents as possible and then order them precisely, which can be seen as a "breadth-first" process. In contrast, QA is more "depth-first": a single verified and concise reasoning path to reach the right answer suffices. Fast convergence, which RARG optimizes for, is therefore less aligned with BRIGHT, yet Table 3 shows RARG+ still attains the best average nDCG@10 (53.36), ahead of the retrieval-specialized NeMo (52.89) and all other baselines. Relevance-aware interaction thus remains beneficial even when the objective is a ranked list rather than one answer.

Table 3: Retrieval effectiveness on BRIGHT, measured by NDCG@10. Unless specified otherwise, methods use GPT-5.4-mini (medium reasoning).

Method	**Avg.↑**	Bio.	Earth	Eco.	Rob.	Tools	Search	Bash	Read	Think	Answer
DCI	48.43	62.05	54.94	37.13	39.59	40.04	–	14.73	25.31	–	–
RISE-BM25	41.60	50.27	47.80	33.65	34.67	31.82	7.55	2.15	22.12	–	–
Nemo Agent	52.89	65.15	61.85	**39.05**	45.49	7.68	6.36	–	–	0.37	0.95
RARG	51.75	63.87	60.54	38.50	44.07	28.73	1.10	11.25	16.38	–	–
RARG+	**53.36**	**66.70**	**62.16**	37.23	**47.34**	27.55	1.23	9.87	16.45	–	–
RARG++	50.55	61.65	61.32	36.14	43.10	27.28	1.14	8.40	17.74	–	–

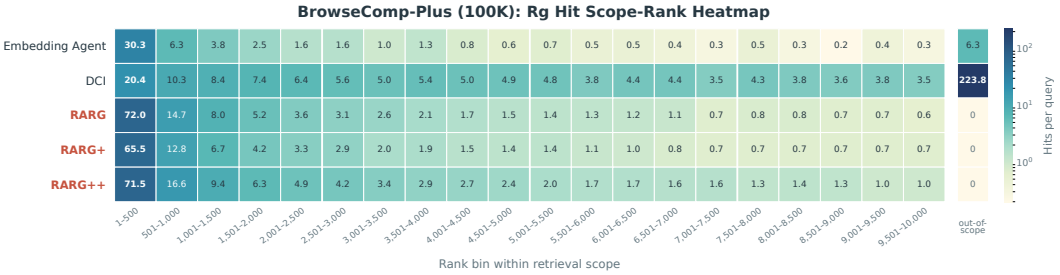

Figure 3: Relevant-document hits over scope ranks on BC+. Embedding hits are top-concentrated but few, DCI is flat and relevance-agnostic, and RARG stays front-loaded while match-level reranking recovers evidence from lower-ranked documents.

Tool counts here reflect different interfaces, not comparable costs. NeMo is a recall-and-rank agent: it retrieves 20 full documents per *Search* and leans on strong LLM reranking, and because BRIGHT documents are relatively short (unlike BC+), this loop is highly compact (7.68 tools). A grep-oriented agent instead observes only local matches and must *Read* for surrounding context, hence more *Read* tool calls. For DCI, the relevance-agnostic `rg` shows a steadily increasing number of exploration steps but provides less relevant information, resulting in lower performance. RISE still relies on *Search*, but BM25 clearly cannot handle the reason-intensive retrieval. Within RARG, the ordering flips relative to QA: RARG+ > RARG > RARG++ (53.36/51.75/50.55). RARG+, provided with a good initialization, achieves the best performance. However, we can see from the number of *Bash* calls that RARG++'s match-level reranking concentrates the observation budget on locally strong excerpts and converges fastest relatively—an advantage for QA but a liability when the task rewards wide recall. Overall, RARG leverages relevance to top BRIGHT, but the optimal granularity depends on whether the task favors rapid convergence or broad candidate recall.

4.5 Analysis of RARG Behavior

How relevance shapes evidence discovery. To characterize *how* each agent uses relevance, we bucket the scope documents and count where an agent's hits on relevant documents fall along the scope rank (Fig. 3). The embedding agent, which consumes relevance directly, produces a sharply decreasing hit profile as expected; however, its distinct hits are few, indicating that its top-k recall returns highly redundant documents and thus wastes the observation on repeated information—a plausible contributor to its weakness. DCI shows the flattest profile and hits many out-of-scope documents, the signature of relevance-agnostic `grep` exploration: lacking any ordering prior, it must probe the whole corpus, yielding sparse evidence density and slow convergence. RARG sits between these extremes. It keeps hits front-loaded, confirming that document-level relevance steers traversal, while covering more distinct documents than the embedding agent. RARG++ is the flattest of the three variants and recovers hits from lower-ranked documents: match-level reranking lets locally informative excerpts, buried in documents whose *document*-level score is weak, be reranked forward and reach the model. This directly validates our design—global relevance orders traversal, local relevance governs visibility—preserving DCI's fine-grained interaction without discarding the relevance prior.

Scope quality and Bash composition. Fig. 4(a) reports two per-scope quantities, averaged within each sample and then across samples: *scope recall*, the fraction of BC+ gold documents captured

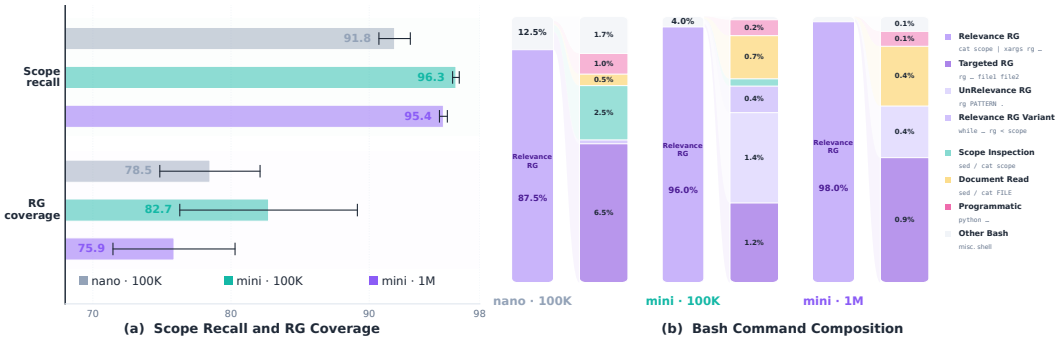

Figure 4: Scope quality and Bash usage under RARG on BrowseComp-Plus. **(a)** Scope recall and RG coverage for GPT-5.4-nano and GPT-5.4-mini at the 100K corpus, and for mini at 1M. Error bars denote the standard deviation across RARG, RARG+, and RARG++. **(b)** Composition of Bash calls: share of scoped `rg` versus non-scoped Bash, with the residual mix broken down by command type. Larger backbones and corpora keep scoped `rg` dominant.

by the scope, and *RG coverage*, the fraction recovered by the files that `rg` actually hits. Both mini settings reach 95–97% scope recall, showing that capping the scope at the top-10K documents preserves near-complete gold coverage while keeping the `rg` search load not too heavy; the negligible 100K→1M drop further confirms that the retriever is barely perturbed by the added distractors. Nano recall is markedly lower, consistent with weaker *embed_recall* queries and its frequent, less disciplined *embed_recall* triggering. RG coverage tells a complementary story: mini-100K stays high, but mini-1M falls sharply—the long FineWeb-Edu documents flood `rg` with incidental matches, so the agent recovers fewer gold files per scan and may prematurely conclude that no further evidence exists, which explains the claim we stated in Sec. 4.3. Fig. 4(b) decomposes *Bash* usage. On both mini settings, the harness reliably steers the model into the relevance-aware form `cat scope.txt | xargs -d '\n' rg ...`: scoped `rg` dominates; the remaining non-scoped calls are still mostly `rg`. On nano, the scoped-`rg` share shrinks substantially, again reflecting weaker instruction adherence rather than a limitation of the mechanism. Together, the two panels confirm that RARG supplies high-recall, low-cost scopes and converts them into relevance-ordered `rg` exploration whenever the backbone follows instructions well.

4.6 Case Study

We qualitatively compare the three variants on a 100K-document BC+ question that identifies a mathematician by joining an AMS fellowship and Ph.D. year with two coauthors, an award, and a second paper title. All three trajectories are judged correct for Russell David Lyons, while tool use decreases strictly from 33 to 18 to 10 and total turns fall from 17 to 11 to 10. The answer-bearing CV first appears at turn 7 for RARG, but at turn 2 for both initialized variants; subsequent interaction verifies the Peres–Pemantle coauthor chain.

Appendix B presents the compressed trajectories, decisive observations, and interaction counts. The case is intended as a qualitative view of how progressively finer relevance guidance can shape search convergence, rather than as standalone evidence of method superiority; the aggregate comparison is reported in Table 1.

5 Conclusion

We revisited the role of relevance in agentic search and argued that it should guide corpus interaction, not merely select its inputs. Building on this view, we introduced RARG, which turns retrieval scores into an execution prior for `grep` exploration at two resolutions: document-level relevance orders `rg` traversal so promising documents are scanned first, while match-level reranking controls which local excerpts remain visible under a limited observation budget. Between them, entry-point initialization gives the agent a precise place to begin. This coarse-to-fine design preserves DCI's fine-grained, compositional interaction while restoring the relevance guidance that unrestricted `grep` lacks. Across challenging QA, corpus scaling, and reasoning-intensive retrieval, RARG advances the accuracy–

efficiency frontier over retrieval-only agents, unrestricted DCI, and retrieval-constructed interaction spaces, reaching higher accuracy with substantially fewer tool calls and degrading gracefully as the corpus grows. Our analysis further confirms the intended mechanism: RARG keeps hits front-loaded toward relevant documents yet recovers useful evidence from lower-ranked ones. We hope these results encourage viewing relevance as an execution signal for corpus interaction, rather than a fixed content bottleneck.

Limitations

RARG's effectiveness depends on the quality of the relevance signal from the embedding model, and the two guidance levels place *different* demands on it: document-level ranking must score long documents, whereas match-level reranking must score short local excerpts. These granularities are not always served equally well by the same model—on BRIGHT, for instance, NV ranks documents well but handles short matches less reliably, prompting a fallback to Qwen3-Embedding. Like other embedding-agent methods, RARG thus relies on a sufficiently strong embedding model, together with a harness adapted to it.

Enforcing single-threaded, order-preserving `rg` (`-j1`) and reranking matches also incur extra search latency. The overhead is tolerable in our experiments, but a performance–time trade-off remains. We use `-j1` rather than reordering matches after an unconstrained `rg` run because a query with many matches would make the full scan itself slow; sequential scanning instead lets us stop as soon as 30/60 (or M=500) matches are collected, achieving the same ordering with far less post-processing. Both are valid implementations, and other realizations are possible—the central idea of injecting relevance into `rg` is unchanged.

The method is further sensitive to the backbone's instruction-following ability. We find GPT-5.4-nano follows our multi-stage protocol less reliably than GPT-5.4-mini and GPT-5.4, which also converge faster; a more compliant model therefore yields cleaner behavior, though RARG still improves over baselines even on nano.

Relevance guidance is also not immune to corpus-induced interference. When the corpus is scaled with long, noisy documents, incidental lexical matches degrade both `rg` output and the embedding scores, and match-level reranking only partially mitigates this. Relatedly, the generative reranking-query variant converges fastest yet loses accuracy, suggesting a train–evaluation gap in which explicitly emitting a rerank query perturbs the learned *Bash*/`rg` behavior; closing this gap is left to future work. Finally, our evaluation covers BrowseComp-Plus and four BRIGHT subsets with the GPT-5.x family; broader validation across other backbones, open-web settings, and more domains remains future work.

References

[1] Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, et al. 2025. Browsecomp-plus: A more fair and transparent evaluation benchmark of deep-research agent. *arXiv preprint arXiv:2508.06600*.

[2] Tsz Ting Chung, Leyang Cui, Lemao Liu, Xinting Huang, Shuming Shi, and Dit-Yan Yeung. 2024. Selection-p: Self-supervised task-agnostic prompt compression for faithfulness and transferability. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11057–11070.

[3] Tsz Ting Chung, Lemao Liu, Mo Yu, and Dit-Yan Yeung. 2025. Divlogiceval: A framework for benchmarking logical reasoning evaluation in large language models. *Findings of the Association for Computational Linguistics: EMNLP*, pages 901–915.

[4] Tsz Ting Chung, Lemao Liu, Mo Yu, and Dit-Yan Yeung. 2026. Many-shot cot-icl: Making in-context learning truly learn. *arXiv preprint arXiv:2605.13511*.

[5] Guoxuan Ding, Yuqing Li, Ziyan Zhou, Zheng Lin, Daren Zha, and Jiangnan Li. 2026. Exdr: Explanation-driven dynamic retrieval enhancement for multimodal fake news detection. *arXiv preprint arXiv:2601.15820*.

[6] Reza Esfandiarpoor and NVIDIA. Nvidia nemo retriever's agentic retrieval pipeline.

[7] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

[8] Tz-Huan Hsu, Jheng-Hong Yang, and Jimmy Lin. 2026. Rethinking agentic search with pi-serini: Is lexical retrieval sufficient? *arXiv preprint arXiv:2605.10848*.

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

[10] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pages 6769–6781.

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

[12] Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025. Search-o1: Agentic search-enhanced large reasoning models. *arXiv preprint arXiv:2501.05366*.

[13] Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yongkang Wu, Ji-Rong Wen, Yutao Zhu, and Zhicheng Dou. 2025. Webthinker: Empowering large reasoning models with deep research capability. *arXiv preprint arXiv:2504.21776*.

[14] Yuqing Li, Jiangnan Li, Zheng Lin, Ziyan Zhou, Junjie Wu, Weiping Wang, Jie Zhou, and Mo Yu. 2025. Mindscape-aware retrieval augmented generation for improved long context understanding. *arXiv preprint arXiv:2512.17220*.

[15] Yuqing Li, Jiangnan Li, Mo Yu, Zheng Lin, Weiping Wang, and Jie Zhou. 2026. Mia-signature: Approximating global activation for long-context understanding. *arXiv preprint arXiv:2605.06416*.

[16] Zhuofeng Li, Haoxiang Zhang, Cong Wei, Pan Lu, Ping Nie, Yi Lu, Yuyang Bai, Shangbin Feng, Hangxiao Zhu, Ming Zhong, Yuyu Zhang, Jianwen Xie, Yejin Choi, James Zou, Jiawei Han, Wenhu Chen, Jimmy Lin, Dongfu Jiang, and Yu Zhang. 2026. Beyond semantic similarity: Rethinking retrieval for agentic search via direct corpus interaction. *arXiv preprint arXiv:2605.05242*.

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

[18] Yi Lu, Zhuofeng Li, Ping Nie, Haoxiang Zhang, Yuyu Zhang, Kai Zou, Wenhu Chen, Jimmy Lin, Dongfu Jiang, and Yu Zhang. 2026. Dr-dci: Scaling direct corpus interaction via dynamic workspace expansion. *arXiv preprint arXiv:2606.14885*.

[19] OpenAI. 2026. Gpt-5.4 nano model. `https://developers.openai.com/api/docs/models/gpt-5.4-nano`.

[20] OpenAI. 2026. Introducing GPT-5.4. `https://openai.com/index/introducing-gpt-5-4/`.

[21] Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond*, volume 4. Now Publishers Inc.

[22] Alireza Salemi, Chang Zeng, Atharva Nijasure, Jui-Hui Chung, Razieh Rahimi, Fernando Diaz, and Hamed Zamani. 2026. Grepseek: Training search agents for direct corpus interaction. *arXiv preprint arXiv:2605.29307*.

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

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

[25] Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han-yu Wang, Liu Haisu, Quan Shi, Zachary Siegel, Michael Tang, et al. 2025. Bright: A realistic and challenging benchmark for reasoning-intensive retrieval. In *International Conference on Learning Representations*, volume 2025, pages 48941–48991.

[26] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 10014–10037.

[27] Junjie Wu, Jiangnan Li, Yuqing Li, Lemao Liu, Liyan Xu, Jiwei Li, Dit-Yan Yeung, Jie Zhou, and Mo Yu. 2026. Situated embedding models for context-aware dense retrieval. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 37–49.

[28] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

[29] Mo Yu, Tsz Ting Chung, Chulun Zhou, Tong Li, Rui Lu, Jiangnan Li, Liyan Xu, Haoshu Lu, Ning Zhang, Jing Li, et al. 2025. Prelude: A benchmark designed to require global comprehension and reasoning over long contexts. *arXiv preprint arXiv:2508.09848*.

[30] Yuqi Zeng, Qixiang Deng, Yulei Wan, Ruiquan Jiang, Xiaoqing Zheng, and Xuanjing Huang. 2026. Rethinking agentic rag: Toward llm-driven logical retrieval beyond embeddings. *arXiv preprint arXiv:2605.27123*.

[31] Wenyuan Zhang, Xinghua Zhang, Haiyang Yu, Shuaiyi Nie, Bingli Wu, Juwei Yue, Tingwen Liu, and Yongbin Li. 2026. Expseek: Self-triggered experience seeking for web agents. *arXiv preprint arXiv:2601.08605*.

[32] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

[33] Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. 2025. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. *arXiv preprint arXiv:2504.03160*.

[34] Chulun Zhou, Chunkang Zhang, Guoxin Yu, Fandong Meng, Jie Zhou, Wai Lam, and Mo Yu. 2025. Hgmem: Hypergraph-based working memory to improve multi-step rag for long-context complex relational modeling. In *Forty-third International Conference on Machine Learning*.

[35] Shengyao Zhuang, Yuansheng Ni, Hengxin Fun, Jimmy Lin, and Xueguang Ma. 2026. Towards retrieving interaction spaces for agentic search. *arXiv preprint arXiv:2606.06880*.

A Prompts for RARG

A.1 System Prompt of BC+

Figure 5 shows the system prompt used by RARG on BrowseComp-Plus. It states the corpus-listing constraint, describes the *embed_recall* tool and the scope-based `rg` search protocol, and fixes the final answer format.

System prompt for RARG on BrowseComp-Plus

```
IMPORTANT CONSTRAINTS:
- Do NOT use ‘ls‘, ‘find‘, or any command that lists all filenames. The corpus has 100,000+ files
      across 30,000+ directories -- listing them will timeout and waste all your turns.

EMBEDDING RECALL TOOL:
The corpus contains a large number of documents. The large search space puts significant pressure on rg
       -- too many matches, excessive noise, and difficulty pinpointing relevant files.

To address this, an embed_recall tool is available. By providing a query, it semantically recalls a
      batch of relevant documents and saves their paths to a scope file (scope_1.txt, scope_2.txt, ...),
       one path per line. This narrows the search space to a manageable subset. It also returns a few
      reference paragraphs from top-ranked documents -- these are a rough preview to help you identify
      search directions for ‘rg‘ search.

To search within recalled documents, pipe the scope file to rg:
  cat /path/to/scope_N.txt | xargs -d '\n' rg [OPTIONS] "PATTERN"

Results are returned ordered by document embedding similarity to your query (most relevant first). Only
       search within one scope file at a time -- do not concatenate multiple scope files.

NOTE:
- Initially call embed_recall with the ORIGINAL QUESTION to establish a scope file, then use rg to
      explore within it with diverse keywords, entities, and patterns.
- ‘rg‘ calls from different angles are essential.
- Only call embed_recall again if your scope is clearly missing the topic entirely -- do NOT call it
      repeatedly hoping for better preview paragraphs, use rg instead.
- Do NOT call embed_recall multiple times without rg exploration in between.
- Do NOT run rg directly on the corpus without a scope file.

When you are confident to answer the question, stop tool calling and respond with the following format:
Explanation: {your explanation for your final answer}
Exact Answer: {your succinct, final answer}
Confidence: {confidence score between 0% and 100%}
```

Figure 5: The system prompt used by RARG for the BrowseComp-Plus evaluation.

A.2 System Prompt of BRIGHT

Figure 6 shows the system prompt used by RARG on BRIGHT. Unlike the QA prompt, it frames the task as retrieval: the agent must maximize recall of related documents while preserving precision, follows the same *embed_recall* plus scope-based `rg` protocol, and returns a ranked list of exactly ten document paths.

System prompt for RARG on BRIGHT

```
You are a retrieval agent that finds all documents related to a given query.

<GOAL>
- You are given a search query. Your task is to use the available tools to find all related and
      somewhat related documents for that query while producing a strong final ranking.
- The goal is to maximize recall without throwing away precision: missing relevant documents hurts, and
       including weakly related documents also hurts.
</GOAL>

IMPORTANT CONSTRAINTS:
- Do NOT use ‘ls‘, ‘find‘, or any command that lists all filenames. The corpus has 100,000+ files
      across 30,000+ directories -- listing them will timeout and waste all your turns.

EMBEDDING RECALL TOOL:
The corpus contains a large number of documents. The large search space puts significant pressure on rg
       -- too many matches, excessive noise, and difficulty pinpointing relevant files.

To address this, an embed_recall tool is available. By providing a query, it semantically recalls a
      batch of relevant documents and saves their paths to a scope file (scope_1.txt, scope_2.txt, ...),
       one path per line. This narrows the search space to a manageable subset. It also returns a few
      reference paragraphs from top-ranked documents -- these are a rough preview to help you identify
      search directions for ‘rg‘ search.

To search within recalled documents, pipe the scope file to rg:
  cat /path/to/scope_N.txt | xargs -d '\n' rg [OPTIONS] "PATTERN"
```

```
Results are returned ordered by document embedding similarity to your query (most relevant first). Only
      search within one scope file at a time -- do not concatenate multiple scope files.

NOTE:
- Initially call embed_recall with the ORIGINAL QUERY to establish a scope file.
- Then use rg calls to explore within it with diverse keywords, entities, terminology, clues,
     paraphrases, aliases, and alternate phrasings.
- `rg` is the main search method.
- `rg` calls from different angles are essential.
- Only call embed_recall again if the current scope is clearly missing the wanted topic entirely. Do
     NOT call embed_recall repeatedly without rg exploration in between.
- Do NOT call embed_recall multiple times without rg exploration in between.
- Do NOT run rg directly on the corpus without a scope file.
- If needed, read the corresponding documents for more detailed evidences to ensure your ranking.

<RELEVANCE_DEFINITION>
- In this task, what counts as a "query", "document", and "relevant" can be more nuanced than in
     ordinary web search.
- A query may be a forum post, technical problem, math problem, coding problem, scientific question, or
      task description.
- A document is relevant if it is genuinely useful for answering, solving, grounding, or strongly
     supporting the query.
- Relevance is not limited to surface wording overlap. A document can be relevant because it provides
     the right entity, mechanism, theorem, example, fact, API, or other key clue needed for the query.
- You should analyze what usefulness means for this query and search for documents from multiple angles
      accordingly.
</RELEVANCE_DEFINITION>

<BEST_PRACTICES>
- You should be thorough and find all related and somewhat related documents.
- Search from different angles: direct entity names, aliases, abbreviations, alternate spellings, dates
     , technical terminology, paraphrases, and evidence chains.
- Do not stop after finding a few plausible documents if other relevant documents are likely to exist.
- At the same time, do not include tangential or weakly related documents just to make the list longer.
</BEST_PRACTICES>

<RANKING_INSTRUCTIONS>
- Rank the final list in decreasing order of relevance to the query.
- The first document should be the most useful one for the query, the second should be the next most
     useful one, and so on.
- All returned document identifiers must be full relative paths from the working directory and must
     correspond to actual corpus files.
- Return exactly the top 10 documents, ranked from most relevant to least relevant among your final
     selected set.
</RANKING_INSTRUCTIONS>

Your final response MUST follow this exact format:

Relevant Documents (ranked by relevance, most relevant first; exactly 10):
1. relative/path/to/doc1.txt
2. relative/path/to/doc2.txt
...
10. relative/path/to/doc10.txt
```

Figure 6: The system prompt used by RARG for the BRIGHT retrieval evaluation.

A.3 Tool Specifications

We list the tool schemas exposed to the agent. *embed_recall* performs document-level relevance recall and writes a scope file; *read* returns line-bounded file contents; and *bash* executes shell commands, mainly `rg`. The standard *bash* is used by RARG, RARG+, and RARG++, while the *RerankAwareBash* variant adds an optional `rerank_query` for the generative match-level reranking of RARG++.

`embed_recall`

Description. Semantic search to recall relevant documents from the corpus. Returns a scope file path containing document paths. Use the scope file with `cat scope_N.txt | xargs -d '\n' rg "pattern"` to search only within recalled documents. May be called multiple times with different queries to create multiple scopes.

Parameters.

```
{
  "query": {
```

```
    "type": "string",
    "description": "Natural language search query for semantic document recall"
  }
}
// required: ["query"]
```

`read`

Description. Read the contents of a file. Output is truncated to 2000 lines or 50 KB (whichever is hit first). Use `offset`/`limit` for large files; to read a full file, continue with increasing `offset` until complete.

Parameters.

```
{
  "path": {"type": "string", "description": "Path to the file to read (relative or absolute)."},
  "offset": {"type": "integer", "description": "Line number to start reading from (1-indexed)."},
  "limit": {"type": "integer", "description": "Maximum number of lines to read."}
}
// required: ["path"]
```

`bash` **(standard)**

Description. Execute a bash command in the current working directory. Returns stdout and stderr, truncated to the last 2000 lines or 50 KB (whichever is hit first); the full output is saved to a temp file if truncated. An optional timeout may be provided.

Parameters.

```
{
  "command": {"type": "string", "description": "Bash command to execute."},
  "timeout": {"type": "integer", "description": "Timeout in seconds (optional). rg commands default to
       30s."}
}
// required: ["command"]
```

`bash` **(RerankAwareBash variant,** `USE_CONTEXTUAL_BASH_TOOL=1`**)**

Description. Same as the standard *bash* tool, but for `rg` commands an optional `rerank_query` may be supplied, used only for semantic reranking of matches.

Parameters.

```
{
  "command": {"type": "string", "description": "Bash command to execute."},
  "timeout": {"type": "integer", "description": "Timeout in seconds (optional). rg commands default to
       30s."},
  "rerank_query": {"type": "string", "description":
    "Optional natural-language query used only to rerank rg matches. For rg commands, write exactly two
         short sentences. Sentence 1 states the final fact, entity, or relation the user is trying to
        determine from the question. Sentence 2 states what this specific rg step is trying to verify,
         identify, or retrieve. Build the query by jointly considering the question, the relevant
        conversation context so far, and the current search step; use concrete entities and make it
        more specific as the conversation narrows. For non-rg commands, omit this field."}
}
// required: ["command"]
```

A.4 Initial User Prompts

Each episode begins with a task-specific user prompt that wraps the query. On BC+, it instructs the agent to answer from the local corpus using `rg` and forbids web search; on BRIGHT, it asks the agent to retrieve and rank the relevant documents.

Initial user prompt on BrowseComp-Plus

```
Answer the following question. The answer is contained in the corpus directory (your current working
     directory). **Do Not use web search!** Use ripgrep (rg) instead of grep for fast searching.

QUESTION:
{query}
```

Initial user prompt on BRIGHT

```
Retrieve and rank the most relevant documents for the following query.

QUERY:
{query}
```

B Case Study: Compressed Agent Trajectories

We examine a query from the 100K-document BrowseComp-Plus setting. All three RARG variants are judged correct for the gold answer **Russell David Lyons**. This is not a single-hop lookup: the query requires the agent to join biographical constraints with a three-author paper, an award, and a second paper title. Tool use decreases strictly across the variants (33/18/10), making the example a compact view of how relevance guidance can reduce the interaction needed to resolve a multi-document clue chain.

Query (Index: 229) Gold answer: **Russell David Lyons**

Find Person A, who earned a Ph.D. in Mathematics in 1983 and became an AMS Fellow between 2005 and 2020. Person A coauthored a 1990–2005 paper with Persons B and C; Person B won the Rollo Davidson Prize in that period, and Person C published a 1990s paper whose title ends in "Line." What is Person A's full name?

We show the three RARG trajectories on a common turn axis in Figure 7. For readability, consecutive calls pursuing the same search objective are manually grouped into a single phase, whereas each dot denotes the first observation that exposes the full answer-bearing name. The reported tool totals are computed from the uncompressed trajectories and therefore include every original call.

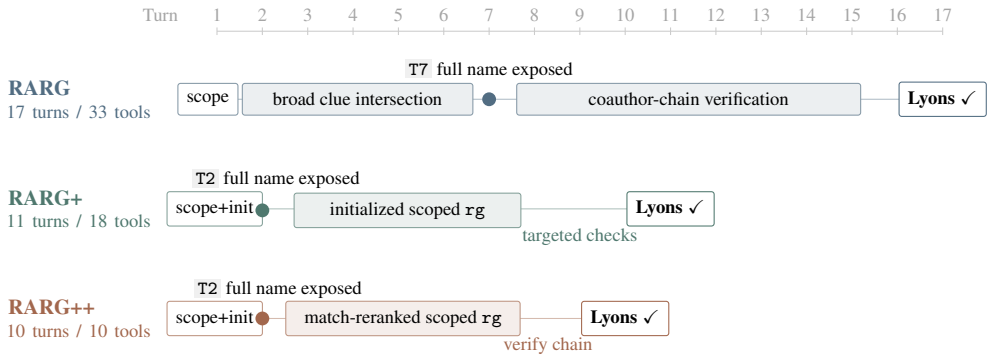

Figure 7: Compressed, turn-aligned trajectories for BC+ query 229 on the 100K corpus. The dot indicates the first observation containing the full answer-bearing name. Initialization moves this evidence from `T7` to `T2`; all three trajectories are judged correct, while tool use decreases from 33 to 18 to 10.

How the search state changes. The trajectory lengths alone do not explain the difference among the methods. The key distinction is the information state created by the first decisive observation. Table 4 shows the earliest evidence that turns the task from an open-ended person search into verification of a specific mathematician and coauthor chain.

Case takeaway. All three trajectories are judged correct for Russell David Lyons despite the query's multi-document joins. Tool use decreases strictly from 33 to 18 to 10, and total turns fall from 17 to

Table 4: Earliest observation that exposes the full answer-bearing name and its immediate effect on the subsequent search strategy.

Method	First decisive observation	Resulting search transition
RARG	`T7` *"Russell David Lyons ... Ph.D., August 1983, Mathematics"*	After broad searches over prize winners and possible coauthors, the CV identifies Person A; later turns establish the Peres–Pemantle paper chain.
RARG+	`T2` *"Russell David Lyons ... Ph.D., August 1983, Mathematics"*	Initialization places the CV in the first scoped searches; the remaining interaction verifies the award, coauthorship, and title-ending clue.
RARG++	`T2` *"Russell David Lyons ... Ph.D., August 1983, Mathematics"*	The match-reranked scope exposes the same CV in a single post-initialization call; one-tool turns then verify each remaining link.

11 to 10. Initialization accounts for the largest change in evidence timing, moving the answer-bearing CV from `T7` to `T2`; match-level reranking retains that early exposure while further reducing tool use. Rather than implying a general ordering from a single case, this example illustrates how progressively finer relevance guidance can compress the interaction required to verify a difficult clue chain.