ACARE: AGENT-CONTEXT-AWARE RERANKER FOR ITERATIVE SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

In agentic search, conventional rerankers score candidate documents against an agent-generated sub-query, without considering the original question or the agent's reasoning trace. We introduce the Agent-Context-Aware Reranker (ACARe), a 4B decoder-only LLM that reranks documents using this context when available. Across three benchmarks, ACARe improves dataset-averaged nDCG@10 by 0.9–2.0 percentage points over the base reranker, with fewer searches for all five agents, each from a different LLM family. In answer generation across two agents on ClimbMix, ACARe raises accuracy by 8.2–9.1 percentage points and reduces searches relative to no reranking, while matching or exceeding the base reranker's mean accuracy with higher evidence precision. On BrowseComp-Plus, ACARe helps smaller agents close more of the nDCG@10 gap to larger agents without reranking: 51–74%, compared with 26–46% using the base reranker. We will release the model, training data, and code upon acceptance.

1 INTRODUCTION

Search agents answer complex questions by searching for evidence over multiple steps (Yao et al., 2023; Jin et al., 2025; Tongyi DeepResearch Team et al., 2025; Li et al., 2026). Given a user's question, an agent issues a search query, reads the returned documents, and decides what to search for next. We call each agent-generated search query a *sub-query*. A reranker scores the retrieved documents to prioritize the evidence the agent sees. Conventional rerankers score each document against only the sub-query, which is often keyword-like (Meng et al., 2026) and omits the broader task. A document can therefore match a sub-query without helping answer the user's original question.

The original question states the agent's goal, while its available reasoning trace explains what evidence is needed next. Neither requires additional generation. Some first-stage retrievers (Z. Chen et al., 2026a; H. Chen et al., 2026) use this context, but deploying them requires local access to the document corpus. A second-stage reranker, however, refines candidates from any source, supporting lexical and dense retrieval as well as search APIs exposing document text without corpus access, where context-aware first-stage retrieval is infeasible.

We introduce the Agent-Context-Aware Reranker (ACARe), a 4B decoder-only LLM fine-tuned from Qwen3-Reranker-4B (Zhang et al., 2025), a compact, strong open-weight reranker. We use the sub-query and agent context to score candidates independently, which avoids the sequential sliding-window passes that listwise rerankers require for long candidate lists (Sun et al., 2023; Pradeep et al., 2023). To train ACARe, we combine new agent trajectories with original and re-mined DR-Synth (Z. Chen et al., 2026a) labels. Re-mining changes candidates but preserves sub-queries and traces. We incorporate independent context dropout to support missing fields and proprietary agents that do not expose reasoning traces. We also show that ACARe's benefits combine with context-aware first-stage retrieval (Section 5.2), complementing existing retrievers. In summary, our contributions are:

- **A context-aware LLM reranker for agentic document search.** To our knowledge, we introduce the first pointwise LLM reranker trained for search-agent sub-queries using agent context. Our context formulation and context dropout let one checkpoint rerank with the original question, reasoning trace, both, or neither, including for proprietary agents that hide their reasoning.

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

- **Expanded training data.** We extend DR-Synth with original questions, re-mined candidates, and a new rollout, creating 16,601 labeled instances for context-aware reranking.
- **Retrieval and answer-generation gains.** Across three benchmarks, four first-stage retrievers plus API-based search, and two harnesses, ACARe improves dataset-averaged nDCG@10 over the base reranker by 0.9–2.0 percentage points (pp), with fewer searches for all five main agents. In end-to-end answer generation on ClimbMix (Diao et al., 2025), it raises accuracy by 8.2–9.1 pp over no reranking and matches or exceeds the base reranker's mean accuracy. In the BCP context control, ACARe's context formulation drives most of its gain over the base reranker even with the base model unchanged, while training yields the best mean on every metric.

2 Background and related work

Retrieval for search agents. Search agents interleave reasoning and external actions; deep-research systems extend this pattern to sustained search and evidence synthesis (Yao et al., 2023; Jin et al., 2025; Tongyi DeepResearch Team et al., 2025; Li et al., 2026). Fixed-corpus benchmarks such as BrowseComp-Plus enable reproducible evaluation of these interactions (Z. Chen et al., 2026b). Unlike a user question, an agent's sub-query is generated within a trajectory and may omit information already present in the conversation. Meng et al. (2026) find that these queries often consist of keywords, phrases, and exact-match expressions, differing from the natural-language questions commonly used to train neural rankers. This motivates retrieval methods that use the agent's context.

Context-aware first-stage retrieval. AgentIR embeds the reasoning trace with the sub-query (Z. Chen et al., 2026a), while ITER also incorporates the original question and preceding sub-queries (H. Chen et al., 2026). Both require local access to the document corpus, whereas ACARe reranks returned documents.

Session and conversational search. Context-aware ranking predates search agents: session-search models rank documents using a user's earlier in-session search activity (Ahmad et al., 2019), and conversational search resolves context-dependent questions by rewriting them or by conditioning retrieval and reranking on dialogue history (Dalton et al., 2020; Anantha et al., 2021; Qu et al., 2020). ACARe instead conditions a model-generated, often keyword-like sub-query on the agent's original question and its own reasoning trace, with one checkpoint handling either field's absence.

Reasoning-intensive retrieval. ReasonIR, RaDeR, and ReasonEmbed train dense retrievers (Shao et al., 2025; Das et al., 2025; J. Chen et al., 2026). Their focus is improving one-shot neural retrieval rather than interpreting sub-queries within an evolving agent trajectory.

Retrieval learning from agent feedback. LRAT learns from the agent's browsing actions, unbrowsed candidates, and post-browse reasoning (Zhou et al., 2026). Agentic-R combines local query–passage relevance with global answer correctness to measure passage utility and alternates agent and retriever optimization (Liu et al., 2026b). Critic-R uses post-retrieval reasoning to refine queries and converts feedback from successful and failed refinements into retriever supervision (Alam et al., 2026). CoSearch jointly trains an agent and a generative ranker using rewards that combine ranking quality with trajectory outcomes (Zeng et al., 2026). These systems use downstream agent behavior to train the retrieval tool, sometimes jointly with the agent. ACARe focuses on a different requirement: using the original question and available reasoning trace to interpret each sub-query at ranking time, with one checkpoint supporting all four context modes.

Reranking in agentic search. Pointwise and generative listwise rerankers refine first-stage candidates (Zhang et al., 2025; Sun et al., 2023; Pradeep et al., 2023). Instruction-aware retrieval includes a reranker conditioned on human-written task descriptions (Asai et al., 2023). Sharifymoghaddam & Lin (2026) use the search agent's own LLM at low reasoning effort as a listwise reranker; medium-effort search with reranking can outperform high-effort search without reranking in retrieval quality and answer accuracy at lower effective token cost. Meng et al. (2026) report retrieval and answer-quality gains from reranking, generally with fewer searches, and test query-only and reasoning-conditioned sub-query rewriting. SetR selects passage sets that satisfy a query's information re-

quirements (Lee et al., 2025), while Rank4Gen learns selection and ordering from downstream generation preferences (Fan et al., 2026). Generative ranking and selection add autoregressive decoding overhead; long candidate lists can also require sequential sliding-window passes (Sun et al., 2023; Pradeep et al., 2023). Concurrent to our work, RubricRanker (Liu et al., 2026a) trains a generative listwise selector with search rubrics, supporting reasoning-conditioned search and question-only RAG. ACARe instead scores documents independently and trains one checkpoint for four context modes, including question-only context when traces are hidden.

3 ACARe Methodology

3.1 Problem formulation

We consider an agent that gathers evidence for an original question Q through a ReAct-style loop (Yao et al., 2023). Its output is either a ranked document list for an upstream orchestrator or an evidence-grounded answer for a deep-search task. Both settings share the trajectory

$$\mathcal{S} = (Q, \tau_1, a_1, o_1, \ldots, \tau_N, a_N, o_N, y), \tag{1}$$

where N is the number of tool-interaction turns, a_t is an action, o_t its observation, and τ_t the preceding reasoning text, when available. At turn t, the agent conditions its generation on the history $h_t = (Q, \tau_1, a_1, o_1, \ldots, \tau_{t-1}, a_{t-1}, o_{t-1})$. A search action contains sub-queries $q_{t,1}, \ldots, q_{t,n_t}$; retrieval and reranking supply the documents in o_t. Sub-queries in the same turn share τ_t, and we write q_t for an individual sub-query when its within-turn index is unnecessary.

Let $\mathcal{D}_{\text{seen}}$ denote the documents returned during the trajectory. The terminal generation conditioned on h_{N+1} has one of two forms:

$$y = \begin{cases} (\mathrm{id}(d_{\pi(1)}), \ldots, \mathrm{id}(d_{\pi(K)})), & \text{retrieval-only,} \\ (w_1, \ldots, w_L), & \text{answer generation,} \end{cases} \tag{2}$$

where $\mathrm{id}(d)$ is a document ID, π orders K distinct documents from $\mathcal{D}_{\text{seen}}$, and $(w_1, \ldots, w_L)$ is an evidence-grounded answer. Our reranker supports both outputs; the main experiments evaluate retrieval-only output, and a transfer experiment evaluates answer generation (Section 6.4).

Conventional rerankers score candidates against q_t alone, although sub-queries may omit the search intent or constraints in Q. We include available agent context to recover this information.

3.2 Agent context

We use two context fields: the original question Q and the reasoning trace τ_t preceding the current search. The former is fixed throughout the session; the latter is the text the agent has already produced, when exposed by its interface. We do not make an additional model call to elicit a trace. Context therefore requires no additional agent generation, although including it increases the reranker's input length. We define four input modes: $\mathrm{M0}$ contains q_t alone; $\mathrm{M1}$ contains (q_t, τ_t); $\mathrm{M2}$ contains (q_t, Q); and $\mathrm{M3}$ contains (q_t, τ_t, Q). The instruction and query block depend on the mode, while the system message and document field are shared (Figure 4). In $\mathrm{M0}$, the prompt is identical to the base Qwen3-Reranker prompt. Following Z. Chen et al. (2026a), we use the current trace rather than concatenating traces from earlier turns.

The decoder-only LLM jointly processes the sub-query, retained context, and one candidate document. Relevance can therefore depend on the available agent context, without requiring each document to answer the whole question. Context availability determines the inference mode. Agents that expose their reasoning can use $\mathrm{M3}$ when Q is also available. Agents that do not expose reasoning can use $\mathrm{M2}$; deployments without either field use $\mathrm{M0}$. The inference-context ablation in Section 6.2 compares these modes for one agent and benchmark.

3.3 Multi-task training over context modes

ACARe is initialized from Qwen3-Reranker-4B and fine-tuned as a single checkpoint supporting all four modes. During training, the reasoning trace and original question are independently dropped with probability 0.5. A dropped or unavailable field is omitted, and the prompt uses the instruction

corresponding to the remaining fields. When both fields are available, each mode has probability 0.25. Missing source fields can change the effective mode distribution. Dropout decisions are deterministic functions of the seed, epoch, field, and query, ensuring consistent masks across workers.

Each training instance is a sub-query with its available context, one positive document, and retrieved negatives. The same sampled context is used for all documents in its group. The model scores each query–document pair using the difference between the next-token logits for `yes` and `no`. Following Gao et al. (2021), we use localized contrastive estimation (LCE) rather than pointwise binary cross-entropy to train the reranker to distinguish a positive document from retrieved negatives. LCE applies cross-entropy over the group's scores with temperature 0.5. At inference, a sigmoid converts the logit difference to a relevance score without changing document order. Training inputs are capped at 5,120 tokens so that eight query–document pairs (one training group), model state, and activations fit within each GPU's 80 GB memory; inference uses an 8,192-token cap. Appendix A gives the exact objective and optimization settings.

A single checkpoint thus supports both agentic and one-shot retrieval. In the one-shot evaluation (Table 1), it uses $\mathrm{M0}$, so comparisons with the base reranker measure the effect of fine-tuning without additional inference context.

3.4 Training data

Training requires relevance labels for an agent's sub-queries rather than only for the original question. We extend the DR-Synth construction of Z. Chen et al. (2026a) to retain Q alongside each sub-query and its reasoning trace. Starting from questions with gold answers and supporting documents, an agent generates search trajectories. For each sub-query, an oracle reranker receives the retrieved candidates, gold supporting documents, original question, and gold answer. Its ranking supplies positive and negative documents for that sub-query. A separate judge filters trajectories by comparing the agent's final answer with the gold answer.

We combine three label sets from two rollouts. We retain the original DR-Synth labels and retrieve and label new candidates for its recorded sub-queries using Nemotron-3-Embed-8B[1] as a second retriever. Re-mining changes the supervision documents but preserves the sub-queries and reasoning traces. We also generate a new rollout using that second retriever, with the oracle's top-ranked documents returned to the agent during search. The pool contains 16,601 instances, versus 5,238 in DR-Synth. We select the data mixture and hyperparameters on a validation split grouped by original question, then train the final checkpoint on the full pool. Appendix A specifies the models, candidate construction, filtering, and size of each label set.

4 Experimental setup

We use Harness A for iterative document retrieval and Harness B for end-to-end answer generation. Both evaluation harnesses differ from the AgentIR harness used to collect the training trajectories (Appendix A). Our retrieval evaluations cover BrowseComp-Plus (BCP; Z. Chen et al., 2026b), BRIGHT (Su et al., 2025), and Total Recall QA (TRQA; Rafiee et al., 2026), using 200, 1,384, and 569 questions, respectively. One-shot retrieval and Harness A report nDCG@10 and Recall@10 on the final document ranking; Harness A additionally measures search calls, LLM turns, and agent API cost per query. Harness B reports end-to-end answer accuracy, Recall@all, Precision@all, and searches per query on BrowseComp-Plus$_{\mathrm{CM}}$ (Sharifymoghaddam et al., 2026) and BCP (Section 6.4). In all our cost calculations, we exclude retrieval and reranker serving costs: the neural retrievers and rerankers used in the main agentic experiments are relatively small, locally hosted 4B–8B models, whose serving costs we assume are negligible compared with agent LLM API costs (Appendix E).

BM25 provides the lexical baseline. AgentIR-4B is designed for agentic retrieval and achieved the highest downstream accuracy among the first-stage retrievers compared in its BrowseComp-Plus deep-search evaluation (Z. Chen et al., 2026a). Qwen3-Embedding-4B (Zhang et al., 2025) is its base model, providing a general-purpose dense baseline. Nemotron-3-Embed-8B represents a

[1] `https://huggingface.co/nvidia/Nemotron-3-Embed-8B-BF16`

Table 1: **One-shot retrieval.** Δ: change from no reranking. Group (e): BM25 on BCP, Nemotron-3-Embed-8B on BRIGHT, and AgentIR-4B on TRQA.

Retriever	**nDCG@10** (Δ)				**Recall@10** (Δ)			
+**Reranker**	BCP	BRIGHT	TRQA	Avg.	BCP	BRIGHT	TRQA	Avg.
(a1) BM25	9.7 (–)	14.1 (–)	12.4 (–)	12.0 (–)	10.3 (–)	16.4 (–)	7.7 (–)	11.5 (–)
(a2) + Jina-Reranker-v3	17.2 (+7.5)	19.4 (+5.3)	15.8 (+3.4)	17.4 (+5.4)	16.4 (+6.1)	21.7 (+5.3)	10.6 (+2.9)	16.3 (+4.8)
(a3) + Qwen3-Reranker-4B	21.3 (+11.6)	21.6 (+7.5)	17.3 (+4.9)	20.1 (+8.1)	19.1 (+8.8)	24.4 (+8.0)	11.7 (+4.0)	18.4 (+6.9)
(a4) + ACARe-4B (M0)	**23.3** (+13.6)	**22.9** (+8.8)	**19.1** (+6.7)	**21.8** (+9.8)	**20.6** (+10.3)	**25.4** (+9.0)	**12.2** (+4.5)	**19.4** (+7.9)
(b1) Qwen3-Embedding-4B	12.9 (–)	21.7 (–)	9.8 (–)	14.8 (–)	14.7 (–)	26.3 (–)	7.1 (–)	16.0 (–)
(b2) + Jina-Reranker-v3	23.6 (+10.7)	21.8 (+0.1)	14.3 (+4.5)	19.9 (+5.1)	26.1 (+11.4)	25.5 (–0.8)	10.6 (+3.5)	20.7 (+4.7)
(b3) + Qwen3-Reranker-4B	31.3 (+18.4)	26.3 (+4.6)	15.9 (+6.1)	24.5 (+9.7)	31.1 (+16.4)	32.2 (+5.9)	11.2 (+4.1)	24.8 (+8.8)
(b4) + ACARe-4B (M0)	**35.9** (+23.0)	**28.6** (+6.9)	**18.3** (+8.5)	**27.6** (+12.8)	**35.7** (+21.0)	**34.4** (+8.1)	**12.7** (+5.6)	**27.6** (+11.6)
(c1) AgentIR-4B	26.7 (–)	22.5 (–)	17.6 (–)	22.3 (–)	30.2 (–)	27.6 (–)	11.8 (–)	23.2 (–)
(c2) + Jina-Reranker-v3	31.7 (+5.0)	22.1 (–0.4)	17.4 (–0.2)	23.7 (+1.4)	35.7 (+5.5)	26.4 (–1.2)	12.5 (+0.7)	24.9 (+1.7)
(c3) + Qwen3-Reranker-4B	44.6 (+17.9)	27.3 (+4.8)	19.9 (+2.3)	30.6 (+8.3)	45.5 (+15.3)	33.0 (+5.4)	14.3 (+2.5)	30.9 (+7.7)
(c4) + ACARe-4B (M0)	**48.7** (+22.0)	**29.4** (+6.9)	**22.1** (+4.5)	**33.4** (+11.1)	**49.7** (+19.5)	**34.9** (+7.3)	**15.2** (+3.4)	**33.3** (+10.1)
(d1) Nemotron-3-Embed-8B	28.0 (–)	25.3 (–)	14.4 (–)	22.6 (–)	31.3 (–)	30.2 (–)	10.2 (–)	23.9 (–)
(d2) + Jina-Reranker-v3	30.8 (+2.8)	22.1 (–3.2)	16.0 (+1.6)	22.9 (+0.3)	33.2 (+1.9)	26.2 (–4.0)	11.9 (+1.7)	23.7 (–0.2)
(d3) + Qwen3-Reranker-4B	43.1 (+15.1)	27.0 (+1.7)	18.5 (+4.1)	29.5 (+6.9)	46.2 (+14.9)	32.7 (+2.5)	13.5 (+3.3)	30.8 (+6.9)
(d4) + ACARe-4B (M0)	**50.7** (+22.7)	**29.6** (+4.3)	**21.6** (+7.2)	**34.0** (+11.4)	**53.6** (+22.3)	**35.0** (+4.8)	**15.2** (+5.0)	**34.6** (+10.7)
(e1) *Mixed first stage*	9.7 (–)	25.3 (–)	17.6 (–)	17.5 (–)	10.3 (–)	30.2 (–)	11.8 (–)	17.4 (–)
(e2) + Qwen3-Reranker-4B	21.3 (+11.6)	27.0 (+1.7)	19.9 (+2.3)	22.7 (+5.2)	19.1 (+8.8)	32.7 (+2.5)	14.3 (+2.5)	22.0 (+4.6)
(e3) + ACARe-4B (M0)	**23.3** (+13.6)	**29.6** (+4.3)	**22.1** (+4.5)	**25.0** (+7.5)	**20.6** (+10.3)	**35.0** (+4.8)	**15.2** (+3.4)	**23.6** (+6.2)

leading retrieval model: it ranked first on the multilingual Retrieval Embedding Benchmark (RTEB)[2] in the July 2026 release report.[3] Strong first-stage retrievers leave less headroom for reranking. One-shot tests compare no reranking, Jina-Reranker-v3 (0.6B) (Wang et al., 2025), Qwen3-Reranker-4B, and ACARe-4B on the same candidates. In this setting, ACARe uses the context-free mode M0. Unless noted, Qwen3-Reranker-4B receives only the sub-query in its native prompt.

Harness A connects an LLM agent to a search tool, manages iterative search, and records the ranked document IDs submitted through `final_results`; Appendix B explains the harness and serving details. Each search normally retrieves 100 candidates, reranks them when enabled, and returns ten unseen documents.

The main Harness A experiments use five LLMs as agents: two open-weight models, Nemotron-3 Ultra and MiniMax M3, and three proprietary models, Gemini 3.6 Flash, GPT-5.6 Terra, and Claude Sonnet 5. Throughout this paper, Nemotron-3 Ultra denotes the NVFP4 version. The open-weight agents expose reasoning traces and use M3; the proprietary agents use M2, which requires only the sub-query and original question. The three smaller proprietary models in the size–cost experiment (Section 6.3) also use M2.

Each configuration is evaluated five times on a fixed question sample, except the Harness B studies, which use six runs. We average quality metrics over benchmark subsets, then average the three benchmarks equally. Tables report means and sample standard deviations across runs; quality metrics use a 0–100 scale. Differences and percentage changes are computed before rounding, so they may differ from calculations using displayed values. Bold identifies the best mean within each comparison group. Symbols † and ∗ mark statistically significant differences from no reranking and Qwen3-Reranker-4B, respectively. Unmarked results may be nonsignificant or untested; Appendix I specifies the tested comparisons and correction families.

5 Main Results

5.1 One-shot retrieval

Table 1 compares rerankers across four first-stage retrievers and three datasets in one-shot retrieval: each original question is searched once, without agent-generated follow-up queries. Results com-

[2] `https://huggingface.co/blog/rteb`

[3] `https://huggingface.co/blog/nvidia/nemotron-3-embed-wins-rteb`

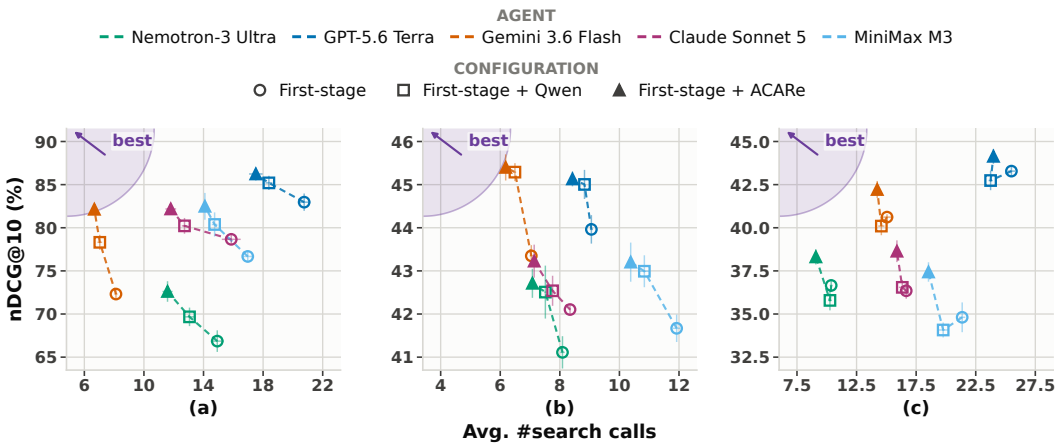

Figure 1: **Agentic retrieval quality versus search effort**: upper left is better. Panels: (a) BCP, (b) BRIGHT, (c) TRQA; axis ranges differ. Dashed lines connect each agent's reranking configurations. Qwen denotes Qwen3-Reranker-4B. Results: Appendix D.

bine BCP's two qrel sets and BRIGHT's and TRQA's subsets, with equal weight across datasets. Appendix C provides detailed results and standard deviations.

ACARe achieves the highest nDCG@10 and Recall@10 across all retriever–dataset combinations. Dataset-averaged nDCG@10 gains are 9.8–12.8 pp over no reranking and 1.7–4.5 pp over Qwen3-Reranker-4B. Both models use the same context-free prompt, so the gains reflect fine-tuning rather than additional inference context. Improvements over no reranking are largest on BCP (13.6–23.0 pp), versus 4.3–8.8 pp on BRIGHT and TRQA. Jina-Reranker-v3 improves BM25 but reduces BRIGHT Recall@10 for all three dense retrievers, by up to 4.0 pp.

For the subsequent agentic experiments, we use one retriever per dataset to limit cost: BM25 on BCP, Nemotron-3-Embed-8B on BRIGHT, and AgentIR-4B on TRQA (group e). For each dataset, we choose the retriever with which ACARe provides the smallest one-shot nDCG@10 improvement over no reranking. The agentic evaluation therefore uses the least favorable first-stage settings for ACARe under this one-shot criterion. We omit Jina-Reranker-v3, which consistently trails both other rerankers, and compare ACARe with no reranking and Qwen3-Reranker-4B throughout the remaining experiments.

5.2 Agentic Retrieval

Retrieval quality and search effort. Using BM25 on BCP, Nemotron-3-Embed-8B on BRIGHT, and AgentIR-4B on TRQA, ACARe improves retrieval quality while reducing search effort across all five agents. Over Qwen3-Reranker-4B, dataset-averaged nDCG@10 and Recall@10 rise by 0.9–2.0 pp each, with 0.3–1.0 fewer searches per query (2.0–10.0%; Table 2). Quality gains and LLM-turn reductions are significant for all five agents (Appendix I). Against no reranking, ACARe gains 1.8–4.5 nDCG@10 pp and saves 1.0–2.4 searches per query.

Figure 1 shows higher mean nDCG@10 than the base reranker in all 15 agent–dataset combinations and fewer searches in 14. Eight retain significant gains in at least one quality metric after correction. On BCP and BRIGHT, searches fall by 0.3–1.5 per query (4.2–11.3%) for every agent, with nine of ten reductions statistically significant. Nemotron-3-Embed-8B's training includes synthetic data generated from the BRIGHT corpus, which may leave less room for agentic search and reranking to improve quality. Replacing it with the general-purpose Qwen3-Embedding-4B increases ACARe's gain over the base reranker from 0.2 to 0.9 nDCG@10 pp for Nemotron-3 Ultra (Appendix H).

On TRQA, adding the context-blind base reranker on top of AgentIR-4B lowers mean nDCG@10 for four agents, whereas ACARe improves all five. Context-blind reranking can thus erode a context-aware retriever's ranking quality; ACARe instead builds on it. Over the base reranker, nDCG@10 gains are significant for all five agents and Recall@10 gains for four. Searches fall

Table 2: **Agentic retrieval**, averaged equally over BCP, BRIGHT, and TRQA, using BM25, Nemotron-3-Embed-8B, and AgentIR-4B, respectively. Each group starts without reranking, followed by the two rerankers. Per-benchmark results: Appendix D.

Agent LLM	nDCG@10↑	Recall@10↑	Avg. per query stats		
+Reranker			Searches↓	LLM turns↓	Cost (¢)↓
(a1) Nemotron-3 Ultra	48.2 (±0.4)	46.3 (±0.3)	11.1 (±0.1)	12.2 (±0.1)	18.8 (±0.2)
(a2) + Qwen3-Reranker-4B	49.3 (±0.5)†	48.1 (±0.6)†	10.3 (±0.1)†	11.3 (±0.1)†	17.1 (±0.6)†
(a3) + ACARe-4B (M3)	**51.2** (±0.6)†*	**49.5** (±0.5)†*	**9.2** (±0.1)†*	**10.3** (±0.1)†*	**15.0** (±0.6)†*
(b1) MiniMax M3	51.1 (±0.3)	50.2 (±0.2)	16.8 (±0.1)	8.8 (±0.1)	6.5 (±0.5)
(b2) + Qwen3-Reranker-4B	52.5 (±0.5)†	52.1 (±0.4)†	15.1 (±0.2)†	8.0 (±0.1)†	5.9 (±0.4)
(b3) + ACARe-4B (M3)	**54.4** (±0.6)†*	**53.4** (±0.6)†*	**14.3** (±0.1)†*	**7.7** (±0.1)†*	**5.8** (±0.2)
(c1) Gemini 3.6 Flash	52.1 (±0.3)	49.5 (±0.2)	10.1 (±0.1)	11.1 (±0.1)	13.9 (±0.6)
(c2) + Qwen3-Reranker-4B	54.6 (±0.2)†	52.6 (±0.3)†	9.4 (±0.1)†	10.4 (±0.1)†	**12.9** (±0.5)
(c3) + ACARe-4B (M2)	**56.6** (±0.1)†*	**54.5** (±0.2)†*	**9.0** (±0.1)†*	**10.0** (±0.1)†*	13.2 (±0.5)
(d1) GPT-5.6 Terra	56.7 (±0.4)	54.2 (±0.4)	18.4 (±0.2)	4.2 (±0.0)	28.2 (±0.6)
(d2) + Qwen3-Reranker-4B	57.6 (±0.2)†	54.9 (±0.1)†	17.0 (±0.1)†	4.0 (±0.0)†	**25.0** (±0.2)†
(d3) + ACARe-4B (M2)	**58.5** (±0.4)†*	**55.9** (±0.3)†*	**16.6** (±0.1)†*	**3.9** (±0.0)†*	25.2 (±0.6)†
(e1) Claude Sonnet 5	52.4 (±0.2)	51.5 (±0.2)	13.6 (±0.2)	7.3 (±0.1)	36.5 (±0.9)
(e2) + Qwen3-Reranker-4B	53.1 (±0.2)†	52.0 (±0.2)†	12.3 (±0.1)†	6.6 (±0.1)†	31.3 (±0.3)†
(e3) + ACARe-4B (M2)	**54.7** (±0.2)†*	**53.4** (±0.2)†*	**11.6** (±0.2)†*	**6.3** (±0.1)†*	**30.3** (±0.8)†

for four agents, significantly for Ultra and M3; Terra's small increase is nonsignificant. Across all 15 agent–dataset combinations, ACARe improves mean nDCG@10 by 0.9–9.8 pp and reduces searches by 4.7–25.6% over no reranking.

Agent API cost. Both rerankers reduce mean agent API cost for every agent relative to no reranking: by 5.1–20.1% with ACARe and 7.6–14.2% with Qwen3-Reranker-4B. ACARe additionally saves 12% over Qwen3-Reranker-4B for Nemotron-3 Ultra, a statistically significant reduction. For the other four agents, the difference between rerankers is not statistically significant across five runs. Repeated document tokens and variable implicit caching can partially obscure savings from fewer searches (Appendix E). Section 6.1 examines whether fewer searches also shorten end-to-end runs.

6 Analysis

6.1 Latency and reranking depth

To measure the latency benefit of reduced search effort, we compare reranking depths of 50 and 100 with Nemotron-3 Ultra on BCP using BM25 (Figure 2). Against Qwen3-Reranker-4B, ACARe reduces mean end-to-end latency by 7–8% while improving nDCG@10. The latency reduction at depth 50 and quality gains at depth 100 are statistically significant. Both rerankers remain slower than no reranking. Appendix F gives the configuration and detailed results.

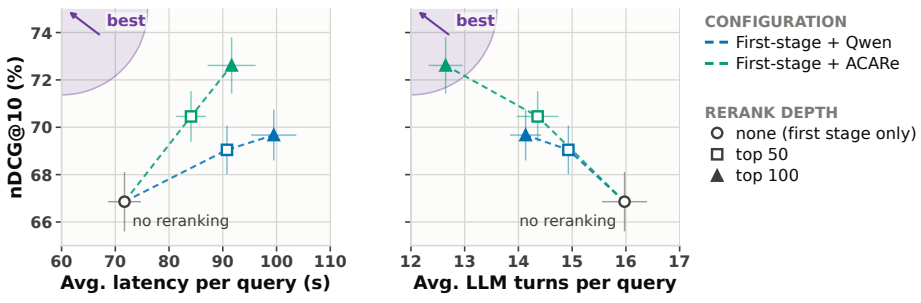

Figure 2: **Reranking depth on BCP** with Nemotron-3 Ultra and BM25. nDCG@10 versus latency (left) and LLM turns (right), averaged per query; upper left is better. Values: Table 14.

Table 3: **Inference context** on BCP with Nemotron-3 Ultra and BM25. Tests use this study's correction family (Appendix I); marks may differ for shared runs in Tables 11 to 13.

Reranker, context	nDCG@10↑	Recall@10↑	Avg. per query stats		
			Searches↓	LLM turns↓	Cost (¢)↓
No reranker	66.9 (±1.3)	68.6 (±1.0)	14.9 (±0.4)	16.0 (±0.4)	35.0 (±1.8)
Qwen3-Reranker-4B	69.7 (±1.1)†	72.2 (±1.5)†	13.1 (±0.3)†	14.1 (±0.3)†	30.8 (±0.8)†
ACARe-4B (M0)	70.8 (±0.9)†	73.4 (±1.3)†	12.7 (±0.4)†	13.8 (±0.4)†	33.0 (±2.1)
ACARe-4B (M1)	71.9 (±0.6)†*	74.9 (±0.6)†	12.2 (±0.3)†*	13.2 (±0.3)†*	31.9 (±1.3)
ACARe-4B (M2)	72.1 (±1.6)†	74.8 (±1.8)†	12.0 (±0.3)†*	13.0 (±0.3)†*	31.3 (±1.2)†
ACARe-4B (M3)	**72.6** (±1.2)†*	**75.2** (±0.9)†*	**11.6** (±0.3)†*	**12.6** (±0.3)†*	**25.9** (±1.4)†*

6.2 Effect of inference context

We evaluate the same ACARe checkpoint in modes M0–M3 on BCP with Nemotron-3 Ultra and BM25 (Table 3). In M0, ACARe achieves 70.8 nDCG@10, compared with 69.7 for Qwen3-Reranker-4B. The mean rises to 71.9 in M1, 72.1 in M2, and 72.6 in M3. Compared with M0, M3 reduces searches from 12.7 to 11.6 per query and achieves the highest mean Recall@10 and lowest mean API cost among the four modes. In M3, all five metrics improve statistically significantly over Qwen3-Reranker-4B and no reranking. These results assess inference-time context for one checkpoint, not training with versus without context.

6.3 Small vs. large agents

We compare three pairs of smaller and larger agents on BCP with BM25: GPT-5.6 Luna and Terra, Gemini 3.5 Flash-Lite and 3.6 Flash, and Claude Haiku 4.5 and Sonnet 5. Small and large denote the compared model tiers; exact parameter counts are not disclosed. Figure 3 compares quality and cost.

Relative to their larger counterparts without reranking, Luna, Flash-Lite, and Haiku close 73%, 74%, and 51% of the nDCG@10 gap with ACARe, versus 46%, 46%, and 26% with Qwen3-Reranker-4B. When both agents in each pair use ACARe, the smaller agents incur 8%, 28%, and 38% of their larger counterparts' API cost, with nDCG@10 scores 4.6, 13.4, and 15.5 pp lower, respectively (Appendix G).

All six agents improve mean nDCG@10 over Qwen3-Reranker-4B with ACARe. Haiku gains 6.3 pp, while Flash-Lite and Flash each gain approximately 3.8 pp. Both rerankers lower mean API cost for every agent; Appendix G gives detailed results and significance tests.

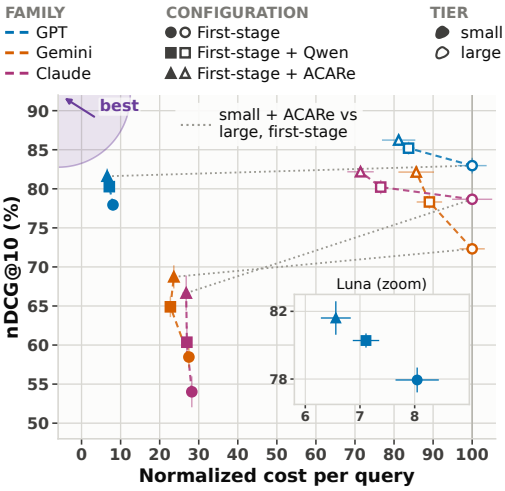

Figure 3: **Small vs. large agents on BCP** with BM25. Pairs (small/large): GPT-5.6 Luna/Terra; Gemini 3.5 Flash-Lite/3.6 Flash; Claude Haiku 4.5/Sonnet 5. Qwen: Qwen3-Reranker-4B; ACARe: ACARe-4B. Cost is relative to the large agent without reranking (100); up and left is better. Inset: Luna zoom. See Table 15.

6.4 Transfer to end-to-end answer generation

We evaluate Harness B on 57 BrowseComp-Plus$_{\text{CM}}$ questions using BM25 API retrieval over ClimbMix's 553 million documents (Diao et al., 2025). GPT-5.6 Luna and Nemotron-3.5 Lightning use no reranking, Qwen3-Reranker-4B, or ACARe-4B. ACARe uses M3 for Lightning and M2 for Luna, which exposes no reasoning trace. We omit monetary cost: Lightning switched between

Table 4: **Harness B answer generation** on BrowseComp-Plus$_{\mathrm{CM}}$ (BM25 API).

Agent LLM +**Reranker**	**Answer accuracy** (%) ↑	**Recall@all** (%) ↑	**Precision@all** (%) ↑	**Searches** / query ↓
(a1) GPT-5.6 Luna	62.9 (±2.8)	9.3 (±0.6)	20.4 (±2.0)	28.0 (±2.1)
(a2) + Qwen3-Reranker-4B	**71.1** (±2.4)	9.3 (±0.4)	21.3 (±1.1)	24.8 (±2.6)
(a3) + ACARe-4B (M2)	**71.1** (±2.1)†	**9.9** (±0.9)	**24.8** (±0.9)$^{\dagger *}$	**22.8** (±1.0)†
(b1) Nemotron-3.5 Lightning	10.2 (±0.7)	1.4 (±0.5)	5.8 (±0.6)	11.1 (±0.8)
(b2) + Qwen3-Reranker-4B	18.7 (±5.0)	1.6 (±0.2)	6.8 (±0.4)	11.5 (±2.1)
(b3) + ACARe-4B (M3)	**19.3** (±3.5)†	**2.1** (±0.6)	**9.4** (±1.7)†	**10.8** (±0.8)

API providers and local serving under heavy throttling; Luna's separate vLLM reranker servers had different loads, so queuing delays changed agent-side prompt-cache reuse (Appendix J).

Across six runs (Table 4), ACARe significantly improves accuracy and precision over no reranking for both agents, with the fewest mean searches. With Luna, it also significantly reduces searches and matches Qwen3's mean accuracy with significantly higher precision. Lightning's mean accuracy nearly doubles (10.2% to 19.3%); its limited capacity, about 3B active parameters, may partly explain its low accuracy and recall (Appendix J).

We assess context and fine-tuning with Lightning on a separate stratified sample of 200 BCP questions, using six runs per configuration (Appendix J.2). With Qwen3-Reranker-4B's weights fixed, mean accuracy rises from 48.2% in M0 to 54.8% in M2 and 55.9% in M3; ACARe in M3 reaches 57.6%. Recall and precision average gold and evidence qrel scores and rise as searches fall at each step. ACARe significantly improves all four metrics over Qwen3 in M0 ($p \leq 0.002$), and recall, precision, and searches over Qwen3 in M2. With identical M3 inputs, ACARe leads on all four means, adding 1.7 pp accuracy and 1.0 pp precision; these final gains are not significant after correction.

7 Conclusion

Search-agent sub-queries can omit the goal and reasoning needed to judge document relevance. We introduce ACARe, to our knowledge the first pointwise LLM reranker trained to use this context for search-agent sub-queries. Fine-tuned from Qwen3-Reranker-4B, ACARe improves mean retrieval quality across three benchmarks and five LLM families while reducing search effort. On TRQA, it improves context-aware AgentIR's results for all five agents, whereas Qwen3-Reranker-4B lowers mean nDCG@10 for four. On BCP, smaller agents gain at least as much as larger ones over no reranking, closing 51–74% of the nDCG@10 gap to larger agents without reranking, versus 26–46% with the base. The benefits extend to answer generation through a BM25 API over 553 million documents. A BCP context control shows that ACARe's context formulation also benefits the unchanged base reranker, while the trained checkpoint achieves the best means on every metric. These results highlight ACARe as a promising lightweight reranker whose independent, parallelizable scoring makes deeper reranking practical to explore.

8 Limitations and future work

We expand the training data through an agent rollout with a second retriever, while retaining the original DR-Synth seed questions, agent LLM, and oracle reranker. Varying these components, the corpus, and the retrievers and rerankers used for negative mining could broaden supervision and further improve reranking effectiveness.

We evaluate on two existing deep-search datasets with fixed corpora, BCP and TRQA. The base-reranker context control uses one agent on BCP. Evaluation on additional such datasets as they become available and with live-web search APIs remains future work.

Our cost analysis covers agent API usage, assuming agent LLMs are substantially larger and costlier than retrievers and rerankers. In our size study, smaller agents benefit more from a higher-quality retrieval tool, but further reducing their size toward the 4B reranker could make reranking costs non-negligible. Measuring total serving costs would clarify this effectiveness–cost tradeoff.

ETHICS STATEMENT

ACARe is fine-tuned from Qwen3-Reranker-4B and may inherit biases and other limitations of the base model. Training examples are mined using an oracle reranker, whose judgments may introduce biases into the training dataset. Fine-tuning for relevance does not ensure that highly ranked documents are accurate, fair, or safe. Downstream applications should account for these risks when using the retrieved evidence.

AI-USE STATEMENT

We used Claude and OpenAI Codex to draft and polish some of the manuscript sections, identify and examine related literature, assist with code development, and check analyses. LLMs also generated synthetic training trajectories, provided relevance supervision, and judged answers, as described in Appendices A and J. The authors thoroughly reviewed and vetted every section of the manuscript through multiple rounds of revision, including AI-assisted text, claims, citations, and analyses. The authors retain full responsibility for the final manuscript and all reported results.

REPRODUCIBILITY STATEMENT

Data generation and training. Section 3 and Appendix A specify the data sources, seed questions and corpus, rollout agents and retrievers, oracle reranker and answer judge, and prompts used to construct supervision. We describe re-mining, filtering, positive and negative selection, and the size of each component of the final training pool. The paper also provides the four context templates, context-dropout procedure, exact scoring and loss functions, optimization settings, sequence limits, training hardware, and the validation protocol used to select hyperparameters and the data mixture. Together, these details document the construction of the training data and the fine-tuning procedure.

Evaluations and analyses. Sections 4 and 6 and Appendix B specify benchmark versions and evaluated samples, relevance labels, retriever and reranker configurations, agent models and reasoning settings, and search and output depths. We document harness behavior, tool interfaces, context limits, stopping and recovery rules, and prompt sources; the Harness B agent and judge prompts are reproduced from retained execution logs. Detailed per-dataset and analysis results appear in Appendices C, D, F to H, and J. Appendix E explains token-based cost accounting and the included charges. All reported evaluations aggregate at least five runs; Appendix I defines the aggregation, variability estimates, statistical tests, and correction families, and distinguishes tested from descriptive comparisons. Exact API responses may vary with generation randomness and model updates.

We will release the trained checkpoint, training data, and all model-training and evaluation code upon acceptance to support independent reproduction and extension of this work.

REFERENCES

Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. Context attentive document ranking and query suggestion. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 385–394, Paris, France, 2019. Association for Computing Machinery.

Md Zarif Ul Alam, Alireza Salemi, and Hamed Zamani. Critic-R: Improving agentic search using instruction-tuned retrievers with natural language introspective feedback. *arXiv:2606.00590*, 2026.

Raviteja Anantha, Svitlana Vakulenko, Zhucheng Tu, Shayne Longpre, Stephen Pulman, and Srinivas Chappidi. Open-domain question answering goes conversational via question rewriting. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 520–534, Online, 2021. Association for Computational Linguistics.

Akari Asai, Timo Schick, Patrick Lewis, Xilun Chen, Gautier Izacard, Sebastian Riedel, Hannaneh Hajishirzi, and Wen-tau Yih. Task-aware retrieval with instructions. In *Findings of the Association*

for Computational Linguistics: ACL 2023, pp. 3650–3675, Toronto, Canada, 2023. Association for Computational Linguistics.

Yoav Benjamini and Daniel Yekutieli. The control of the false discovery rate in multiple testing under dependency. *Annals of Statistics*, 29(4):1165–1188, 2001.

Haodong Chen, Shuai Wang, Yu Yin, Shengyao Zhuang, Guido Zuccon, and Teerapong Leelanupab. ITER: Interaction-aware retrieval for agentic search. *arXiv:2608.27912*, 2026.

Jianlyu Chen, Junwei Lan, Chaofan Li, Defu Lian, and Zheng Liu. ReasonEmbed: Enhanced text embeddings for reasoning-intensive document retrieval. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1203–1221, San Diego, California, United States, 2026. Association for Computational Linguistics.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Jimmy Lin, Akari Asai, and Victor Zhong. AgentIR: Reasoning-aware retrieval for deep research agents. *arXiv:2603.04384*, 2026a.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Sahel Sharifymoghaddam, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Hosna Oyarhoseini, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and Jimmy Lin. BrowseComp-Plus: A fair and disentangled evaluation benchmark for deep search agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22349–22370, San Diego, California, United States, 2026b. Association for Computational Linguistics.

Jeffrey Dalton, Chenyan Xiong, Vaibhav Kumar, and Jamie Callan. CAsT-19: A dataset for conversational information seeking. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1985–1988, Virtual Event, China, 2020. Association for Computing Machinery.

Debrup Das, Sam O'Nuallain, and Razieh Rahimi. RaDeR: Reasoning-aware dense retrieval models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 19970–19997, Suzhou, China, 2025. Association for Computational Linguistics.

Shizhe Diao, Yu Yang, Yonggan Fu, Xin Dong, Dan Su, Markus Kliegl, Zijia Chen, Peter Belcak, Yoshi Suhara, Hongxu Yin, Mostofa Patwary, Yingyan (Celine) Lin, Jan Kautz, and Pavlo Molchanov. Nemotron-CLIMB: Clustering-based iterative data mixture bootstrapping for language model pre-training. In *Advances in Neural Information Processing Systems*, volume 38, pp. 29096–29124, 2025.

Yongqi Fan, Yuxiang Chu, Zhentao Xia, Xiaoyang Chen, Jie Liu, Haijin Liang, Jin Ma, Ben He, Yingfei Sun, Jie Zhai, Dezhi Ye, and Tong Ruan. Rank4Gen: RAG-preference-aligned document set selection and ranking. *arXiv:2601.11273*, 2026.

Luyu Gao, Zhuyun Dai, and Jamie Callan. Rethink training of BERT rerankers in multi-stage retrieval pipeline. In *Advances in Information Retrieval*, pp. 280–286, Virtual Event, 2021. Springer International Publishing.

Tz-Huan Hsu, Jheng-Hong Yang, and Jimmy Lin. Rethinking agentic search with Pi-Serini: Is lexical retrieval sufficient? *arXiv:2605.10848*, 2026.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Ö. Arık, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling (COLM)*, Montreal, Canada, 2025. OpenReview.net.

Dahyun Lee, Yongrae Jo, Haeju Park, and Moontae Lee. Shifting from ranking to set selection for retrieval augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17606–17619, Vienna, Austria, 2025. Association for Computational Linguistics.

Zhuofeng Li, Dongfu Jiang, Xueguang Ma, Haoxiang Zhang, Ping Nie, Yuyu Zhang, Kai Zou, Jianwen Xie, Yu Zhang, and Wenhu Chen. OpenResearcher: A fully open pipeline for long-horizon deep research trajectory synthesis. *arXiv:2603.20278*, 2026.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2356–2362, Virtual Event, Canada, 2021. Association for Computing Machinery.

Wenhan Liu, Yu Lu, Qiaolin Xia, Hui Xu, Tong Zhao, Jian Xi, Yutao Zhu, Haijin Liang, Haibo Shi, Hao Wang, and Zhicheng Dou. Training documents reranker with search rubrics for deep research agent. *arXiv:2608.03527*, 2026a.

Wenhan Liu, Xinyu Ma, Yutao Zhu, Yuchen Li, Daiting Shi, Dawei Yin, and Zhicheng Dou. Agentic-R: Learning to retrieve for agentic search. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 15987–16005, San Diego, California, United States, 2026b. Association for Computational Linguistics.

Chuan Meng, Litu Ou, Sean MacAvaney, and Jeff Dalton. Revisiting text ranking in deep research. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 3006–3016, Melbourne, Australia, 2026. Association for Computing Machinery.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. RankVicuna: Zero-shot listwise document reranking with open-source large language models. *arXiv:2309.15088*, 2023.

Chen Qu, Liu Yang, Cen Chen, Minghui Qiu, W. Bruce Croft, and Mohit Iyyer. Open-retrieval conversational question answering. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 539–548, Virtual Event, China, 2020. Association for Computing Machinery.

Mahta Rafiee, Heydar Soudani, Zahra Abbasiantaeb, Mohammad Aliannejadi, Faegheh Hasibi, and Hamed Zamani. Total Recall QA: A verifiable evaluation suite for deep research agents. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 3365–3373, Melbourne, Australia, 2026. Association for Computing Machinery.

Rulin Shao, Rui Qiao, Varsha Kishore, Niklas Muennighoff, Xi Victoria Lin, Daniela Rus, Bryan Kian Hsiang Low, Sewon Min, Wen-tau Yih, Pang Wei Koh, and Luke Zettlemoyer. ReasonIR: Training retrievers for reasoning tasks. In *Second Conference on Language Modeling (COLM)*, Montreal, Canada, 2025. OpenReview.net.

Sahel Sharifymoghaddam and Jimmy Lin. Rerank before you reason: Analyzing reranking tradeoffs through effective token cost in deep search agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 25874–25886, San Diego, California, United States, 2026. Association for Computational Linguistics.

Sahel Sharifymoghaddam, Lingwei Gu, Yijun Ge, and Jimmy Lin. Projecting BrowseComp-Plus onto ClimbMix: Toward more realistic corpora for agentic search. *arXiv:2608.20317*, 2026.

Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han-yu Wang, Haisu Liu, Quan Shi, Zachary S. Siegel, Michael Tang, Ruoxi Sun, Jinsung Yoon, Sercan O. Arik, Danqi Chen, and Tao Yu. BRIGHT: A realistic and challenging benchmark for reasoning-intensive retrieval. In *The Thirteenth International Conference on Learning Representations (ICLR 2025)*, Singapore, 2025. OpenReview.net.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is ChatGPT good at search? Investigating large language models as reranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 14918–14937, Singapore, 2023. Association for Computational Linguistics.

Zhengwei Tao, Jialong Wu, Wenbiao Yin, Junkai Zhang, Baixuan Li, Haiyang Shen, Kuan Li, Liwen Zhang, Xinyu Wang, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. WebShaper: Agentically data synthesizing via information-seeking formalization. *arXiv:2507.15061*, 2025.

Tongyi DeepResearch Team, Baixuan Li, Bo Zhang, Dingchu Zhang, Fei Huang, Guangyu Li, Guoxin Chen, Huifeng Yin, Jialong Wu, Jingren Zhou, Kuan Li, Liangcai Su, Litu Ou, Liwen Zhang, Pengjun Xie, Rui Ye, Wenbiao Yin, Xinmiao Yu, Xinyu Wang, Xixi Wu, Xuanzhong Chen, Yida Zhao, Zhen Zhang, Zhengwei Tao, Zhongwang Zhang, Zile Qiao, Chenxi Wang, Donglei Yu, Gang Fu, Haiyang Shen, Jiayin Yang, Jun Lin, Junkai Zhang, Kui Zeng, Li Yang, Hailong Yin, Maojia Song, Ming Yan, Minpeng Liao, Peng Xia, Qian Xiao, Rui Min, Ruixue Ding, Runnan Fang, Shaowei Chen, Shen Huang, Shihang Wang, Shihao Cai, Weizhou Shen, Xiaobin Wang, Xin Guan, Xinyu Geng, Yingcheng Shi, Yuning Wu, Zhuo Chen, Zijian Li, and Yong Jiang. Tongyi DeepResearch technical report. *arXiv:2510.24701*, 2025.

Feng Wang, Yuqing Li, and Han Xiao. jina-reranker-v3: Last but not late interaction for listwise document reranking. *arXiv:2509.25085*, 2025.

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

Hansi Zeng, Liam Collins, Bhuvesh Kumar, Neil Shah, and Hamed Zamani. CoSearch: Joint training of reasoning and document ranking via reinforcement learning for agentic search. *arXiv:2604.17555*, 2026.

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

Yuqi Zhou, Sunhao Dai, Changle Qu, Liang Pang, Jun Xu, and Ji-Rong Wen. Learning to retrieve from agent trajectories. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2544–2555, Melbourne, Australia, 2026. Association for Computing Machinery.

A TRAINING DETAILS

Table 5 summarizes the training configuration. All training-input truncation removes excess tokens from the end, retaining the beginning of each field. Figure 4 gives the prompt templates for the context modes defined in Section 3.2.

Model development and final training. During model development, we held out 10% of the training pool as a validation set, assigning all instances of each original question to a single side of the split. We selected hyperparameters and the training-data mixture using LCE loss, accuracy (the positive ranked first within its group), and mean reciprocal rank (MRR) on the validation split. With the recipe fixed, we trained ACARe-4B once on the full pool without a validation holdout.

Table 5: **Training configuration** for ACARe-4B. The data pool is described in Appendix A; the context modes are defined in Section 3.2, and their prompts are shown in Figure 4.

Setting	Value
Model	
Initialization	Qwen3-Reranker-4B (Zhang et al., 2025)
Fine-tuning	full, bf16; no adapters and no new parameters
Scoring head	pretrained LM head; score $\sigma(\ell_{\text{yes}} - \ell_{\text{no}})$
Data	
Training pool	16,601 instances over 54,838 distinct passages
Labelings	three over two rollouts (5,104 + 5,219 + 6,278)
Group	1 positive + 7 retrieved negatives, mined for the same turn
In-batch negatives	none
Context dropout	τ_t and Q dropped independently, $p = 0.5$ each
Objective	
Loss	localized contrastive; cross-entropy over the group, positive at index 0
Temperature	0.5, applied to logit differences
Sequence	
Maximum length	5,120 tokens
Per-field caps	Q 2,048, τ_t 512, q_t 256; the document takes the remainder
Optimization	
Distributed training	FSDP2 with activation checkpointing
Optimizer	AdamW, weight decay 0.01
Learning rate	5×10^{-6}, 20 warm-up steps, then cosine decay
Gradient clipping	global norm 1.0
Batch size	32 global; 1 group per device step, gradient accumulation 4
Epochs	1, i.e. about 519 optimizer steps
Seed	42, for both data order and context dropout
Hardware	
GPUs	8× A100 80 GB
Wall clock	about 10 hours

System message (identical in all four modes)

Judge whether the Document meets the requirements based on the Query and the Instruct provided. Note that the answer can only be "yes" or "no".

M0 — no context
⟨Instruct⟩: Given a web search query, retrieve relevant passages that answer the query
⟨Query⟩: {sub_query}
⟨Document⟩: {$document$}

M1 — reasoning trace
⟨Instruct⟩: Given a user generated web search query and the user's reasoning trace that motivated the query (when available), retrieve relevant passages that answer the query in light of the reasoning context
⟨Query⟩: #Reasoning Trace: {$reasoning_trace$}
#Query: {sub_query}
⟨Document⟩: {$document$}

M2 — original question
⟨Instruct⟩: Given a user generated query issued during an iterative research session and the original question (global query) that motivated the session, retrieve relevant passages that answer the query in light of the original question
⟨Query⟩: #Original Question (global query): {$global_query$}
#Query: {sub_query}
⟨Document⟩: {$document$}

M3 — reasoning trace and original question
⟨Instruct⟩: Given a user generated query, the original question (global query), and the user's reasoning trace (when available) that motivated the query, retrieve relevant passages that answer the query in light of the original question and reasoning context
⟨Query⟩: #Original Question (global query): {$global_query$}
#Reasoning Trace: {$reasoning_trace$}
#Query: {sub_query}
⟨Document⟩: {$document$}

Figure 4: The four context modes ACARe is trained to score in. Only the instruction and the contents of the query block change; the system message and the document field are the same throughout.

Rollouts and supervision. We use 500 WebShaper questions (Tao et al., 2025) with gold answers and supporting documents. Both rollouts use the AgentIR harness[4] over the same approximately 1.1M-document corpus. The rollout agent is Tongyi-DeepResearch-30B-A3B; GLM-4.6 is the oracle reranker. The original DR-Synth rollout uses Qwen3-Embedding-8B (Z. Chen et al., 2026a), and our new rollout uses Nemotron-3-Embed-8B. Evaluation uses neither the rollout agent nor its oracle; Harnesses A and B also differ from the rollout harness.

The oracle ranks up to 50 candidates per sub-query, including prepended gold supporting documents, using the original question and gold answer. Its top document supplies the positive and its seven lowest-ranked documents supply distinct negatives; its top five documents are returned to the rollout agent. Nemotron-3 Ultra judges final-answer correctness. With the same harness, agent, and oracle, the new rollout has 276 correct trajectories out of 500 (55.2%), compared with 249 (49.8%) in the original DR-Synth rollout. This is an observed difference between the two rollouts, without repeated-rollout estimates of variability. Only successful trajectories contribute training instances. Figures 5 to 7 reproduce the AgentIR rollout, oracle, and judge prompts (Z. Chen et al., 2026a).

Filtering and training pool. We exclude New York Times passages, at the publisher's request, before constructing the final pool. For the original DR-Synth labels, removing rows with an excluded passage in any position reduces 5,238 instances to 5,104. For re-mined labels, we remove rows with an excluded positive and replace excluded negatives with other ranked candidates. This removes 19 re-mined rows, leaving 5,219; thus the original and re-mined labels retain different numbers of instances from the same 249 trajectories.

Together with 6,278 instances from the 276 new trajectories, the pool contains 16,601 instances over 54,838 distinct passages. Every instance retains its original question. In the re-mined component, 15.7% of instances have malformed reasoning traces that we treat as missing; these instances remain usable in modes without a trace.

Scoring and objective. Let x_t denote the prompt containing the sub-query q_t and the retained context fields. For a candidate document d, define the logit difference and inference score as

$$z_\theta(x_t, d) = \ell_\theta(\text{yes} \mid x_t, d) - \ell_\theta(\text{no} \mid x_t, d), \qquad s_\theta(x_t, d) = \sigma(z_\theta(x_t, d)) \,. \tag{3}$$

Here ℓ_θ is the next-token logit at the final non-padding position. For a training group $D_t = (d_t^+, d_{t,1}^-, \ldots, d_{t,m}^-)$, localized contrastive estimation (LCE) (Gao et al., 2021) minimizes

$$\mathcal{L}_{\text{LCE}}(x_t, D_t) = -\log \frac{\exp\big(z_\theta(x_t, d_t^+)/T\big)}{\exp\big(z_\theta(x_t, d_t^+)/T\big) + \sum_{j=1}^{m} \exp\big(z_\theta(x_t, d_{t,j}^-)/T\big)}, \qquad T = 0.5. \tag{4}$$

The loss uses the raw logit differences before sigmoid and is averaged over training groups. Each group contains one positive at index 0 and seven distinct negatives mined for the same sub-query; no in-batch negatives are used. The temperature applies only to the training objective. The sigmoid preserves document ordering at inference.

[4] https://github.com/texttron/AgentIR

A.1 ROLLOUT AND SUPERVISION PROMPTS

You are a deep research assistant. Your core function is to conduct thorough, multi-source investigations into any topic. You must handle both broad, open-domain inquiries and queries within specialized academic fields. For every request, synthesize information from credible, diverse sources to deliver a comprehensive, accurate, and objective response. When you have gathered sufficient information and are ready to provide the definitive response, you must enclose the entire final answer within <answer></answer> tags.

 # Tools

You may call the search tool one or multiple times to assist with the user query.

You are provided with the search tool, its signature within <tools></tools> XML tags:
<tools>
{"type": "function", "function": {"name": "search", "description": "Perform Google web searches then returns a string of the top search results. Accepts a single query.", "parameters": {"type": "object", "properties": {"query": {"type": "string", "description": "The search query."}}, "required": ["query"]}}}
</tools>

For each search tool call, return a json object with function name and arguments within <tool_call></tool_call> XML tags:
<tool_call>
{"name": "search", "arguments": <args-json-object>}
</tool_call>

Current date:

Figure 5: Rollout-agent prompt from the AgentIR release (Z. Chen et al., 2026a). Its template names Google web search, but the handler calls the configured corpus retriever. AgentIR appends the execution date in YYYY-MM-DD format to the last line.

System message
You are an intelligent assistant that can rank passages to best help an user in correctly answering a question. You will be given a complex question, and its correct answer. A user, who is trying to solve the complex question, issues a simpler probing web query, which retrieved some passages. You are to rank these passages based on their relevancy to the user's probing query, while prioritizing their usefulness in helping the user to correctly answer the complex question.

Prefix
A user is trying to answer a complex question: {question}. The correct answer is: {correct_answer}. A user, trying to solve the complex question, issued this probing web query: {query}, which retrieved {num} passages. I will provide you with these {num} passages, each indicated by a numerical identifier []. Rank the passages based on their relevance to the probing query, as well as their usefulness in helping the user to correctly answer the complex question.

Body (one line per candidate)
[{rank}] {candidate}

Suffix
Complex Question: {question}.
Correct Answer: {correct_answer}.
Probing Query: {query}.
Rank the {num} passages above based on their relevance to the probing query, as well as their usefulness in helping the user to correctly answer the complex question. All the passages should be included and listed using identifiers, in descending order of relevance. The output format should be [] > [], e.g., [2] > [1], Answer concisely and directly and only respond with the ranking results, do not say any word or explain.

Figure 6: Oracle listwise prompt from the AgentIR release (Z. Chen et al., 2026a), assembled from a system message, a prefix, one body line per candidate, and a suffix. Both the agent and oracle receive the seed question; only the oracle receives its gold answer. {num} is at most 50 in our mining.

Judge whether the following [response] to [question] is correct or not based on the precise and unambiguous [correct_answer] below.

[question]: {question}

[response]: {response}

[correct_answer]: {correct_answer}

Your judgment must be in the format and criteria specified below:

extracted_final_answer: The final exact answer extracted from the [response].

[correct_answer]: Repeat the [correct_answer] given above.

reasoning: Explain why the extracted_final_answer is correct or incorrect based on [correct_answer], in the context of this [question]. You should judge whether the extracted_final_answer is semantically equivalent to [correct_answer], allowing the extracted_final_answer to be string variations of [correct_answer]. You should also allow the extracted_final_answer to be more precise or verbose than [correct_answer], as long as its additional details are correct. Do not comment on any background to the problem, do not attempt to solve the problem, do not argue for any answer different than [correct_answer], focus only on whether the answers are semantically equivalent.

correct: Answer 'yes' if extracted_final_answer matches the [correct_answer] given above, or is within a small margin of error for numerical problems. Answer 'no' otherwise, i.e. if there is any inconsistency, ambiguity, non-equivalency, or if the extracted answer is incorrect.

confidence: The extracted confidence score between 0% and 100% from [response]. Put 100 if there is no confidence score available.

Figure 7: Judge prompt adapted from the AgentIR release (Z. Chen et al., 2026a), with spelling and typographic corrections. It decides whether a trajectory may contribute to the training pool. Only its `correct` field is used: all turns from trajectories judged `yes` enter the training pool; trajectories judged `no` are excluded.

B Experimental setup details

Table 6 reports benchmark sizes and shared evaluation settings. For one-shot retrieval and Harness A, samples are drawn with seed 42 and capped at 200 questions per subset. Each BCP run is evaluated against both gold and evidence qrels; BRIGHT includes all 1,384 questions; TRQA contributes 569 questions across its three subsets. We will release the selection seed and the complete list of selected queries with the evaluation code so the evaluation samples can be reproduced exactly.

Benchmarks. BCP (Z. Chen et al., 2026b) evaluates multi-hop questions over a fixed corpus of 100,195 documents. For its 830 questions, evidence qrels identify documents needed to answer the question, while gold qrels identify those that also contain the final answer. Each run produces one ranking, which we score against both qrel sets and average; these are not separate experiments. Following the original BCP paper's setup, we retain the first 512 tokens of each document for evaluation, limiting both reranker input and the document text retained in the agent conversation.

BRIGHT (Su et al., 2025) contains 1,384 questions in twelve domains, each with its own corpus. Corpus sizes range from 7,894 documents for Pony to 413,932 for LeetCode. Relevance often requires reasoning beyond lexical matching. The number of relevant documents also varies by domain, from about two per question in the theorem and code subsets to twenty in Pony.

TRQA (Rafiee et al., 2026) contains 2,327 test questions across Wiki1, Wiki2, and synthetic e-commerce. Each requires a single numerical answer aggregating relevant entities, such as a sum, average, or count, so missing evidence can change the answer. We use the later released `wiki_partial` corpus (5.9M passages) for both Wikipedia subsets, alongside the 3.3M-passage e-commerce corpus.[5] Unlike BCP, BRIGHT and TRQA have no separate 512-token document cap; model sequence limits still apply.

[5] https://huggingface.co/datasets/mahtaa/trqa

Table 6: **Evaluation configuration.** Corpus size is the number of documents; BRIGHT gives a range because each domain has its own corpus. **Sampled** is the number of queries actually evaluated, and **Retriever** the fixed first stage used in the agentic experiments. BCP uses two qrel sets (gold and evidence) to score the same run. Only BCP applies separate document truncation.

	BrowseComp-Plus (Z. Chen et al., 2026b)	**BRIGHT (Su et al., 2025)**	**Total Recall QA (Rafiee et al., 2026)**
Corpus size	100,195	7,894–413,932	3.3M–5.9M
Questions	830	1,384	2,327
Sampled	200	1,384	569
Subsets	1	12 domains	3 (2 Wikipedia, 1 e-commerce)
Retriever	BM25	Nemotron-3-Embed-8B	AgentIR-4B
Doc. truncation	512 tokens	none	none
Shared across benchmarks			
Retrieval depth	top 100 candidates per call		
Rerank depth	all 100 (Section 6.1 is the only experiment that varies this)		
Model input cap	8,192 tokens, retriever and reranker alike		
Documents per search	$k = 10$ by default, unseen documents only		
Query sample	drawn with seed 42, capped at 200 queries per subset		
Runs per configuration	5; tables report mean $\pm$ sample standard deviation		
Significance test	See Appendix I		
Maximum trajectory length	100 turns		
Concurrency	2 queries per run		

Retrieval and reranking. On BRIGHT and TRQA, BM25 uses Pyserini's default parameter values, $k_1 = 0.9$ and $b = 0.4$ (Lin et al., 2021). On BCP, both one-shot and agentic experiments deliberately use $k_1 = 16$ and $b = 1$, the long-document setting identified by the grid search of Hsu et al. (2026), rather than the defaults. In one-shot retrieval, AgentIR-4B receives the original question and the literal string `Empty` for absent reasoning. In agentic retrieval, it also receives the current reasoning trace when available. Other retrievers use only the query.

Inference input limits. Before prompt assembly, ACARe caps the original question at 2,048 tokens, an available reasoning trace at 512 tokens, and the sub-query at 256 tokens in both evaluation harnesses. All input truncation removes excess tokens from the end, retaining the beginning of each field or document. Dense-retriever inputs and the combined inputs to ACARe and Qwen3-Reranker-4B are capped at 8,192 tokens.

Harness A. Each trajectory starts with ten documents retrieved for the original question and reranked when reranking is enabled. The agent then follows a ReAct-style loop (Yao et al., 2023), using `search` (dense) or `keyword_search` (BM25), which replace the upstream `retrieve` tool. Figures 8 to 10 reproduce the public upstream system prompt and tool descriptions. The optional `think` tool was disabled in all runs.

A search retrieves and, when enabled, reranks 100 candidates unless the experiment varies the depth. It returns the highest-ranked k unseen documents, with $k = 10$ by default. Previously seen documents are listed by identifier rather than repeating their content in the tool response. Agents can change k: across the evaluated agents and benchmarks, searches return 9.4–10.3 new documents on average, except for GPT-5.6 Terra on BCP (11.8).

A turn may contain multiple search calls. Nemotron-3 Ultra and Gemini 3.6 Flash average approximately one per turn, MiniMax M3 and Claude Sonnet 5 approximately two, and GPT-5.6 Terra more than four. The trajectory is limited to 100 turns. The search agent selects and ranks documents from its accumulated evidence, then submits their IDs through `final_results`. This tool records the supplied ranking and terminates the search. We compute nDCG@10 and Recall@10 on the first ten submitted IDs, or on the recovered ranking if the search agent fails. Across the five agents, three reranking configurations, and five runs on all benchmarks, 0.145% of query executions fail before recovery. The most common failures are output-token limits, followed by failed LLM calls and turn limits. On failure, the harness invokes a separate selection agent using the same LLM to review all documents retrieved throughout the search and select a ranked subset. If selection fails, the harness

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

<Goal>
You are given a search query and a list of documents retrieved for that query. Your task is to write new queries and use the given search tool to find *ALL* the related and somewhat related documents to the given query (i.e., maximize recall).
If the user's query is a question, you should not answer the question yourself. Instead, you should find the related documents for the given query.
</Goal>

<RELEVANCE_DEFINITION>
- You should be careful, in the context of this task, what it means to be a "query", "document", and "relevant" can sometimes be very complex and might not follow the traditional definition of these terms in standard information retrieval.
- In standard retrieval, a query is usually a user question (like a web search query), the document is some sort of content that provides information (e.g., a web page), and these two are considered relevant if the document provides information that answers the user's query.
- However, in our setting, this could be different. Here are some examples:
* the query is a programming problem and documents are programming language syntax references. A document is relevant if it contains the reference for the programming syntax used for solving the problem.
* both query and documents are descriptions programming problems and a query and document are relevant if the same approach is used to solve them.
* the query is a math problem and documents are theorems. Relevant documents (theorems) are the ones that are useful for solving the math problem.
* the query and document are both math problems. A query and a document are relevant if the same theorem is used for solving them.
* the query is a task description (e.g., for an API programmer) and documents are descriptions of available APIs. Relevant documents (e.g., APIs) are the ones needed for completing the task.
- This is not an exhaustive list. These are just some examples to show you the complexity of queries, documents, and the concept of relevance in this task.
- Note that even here, the relevant documents are still the ones that are useful for a user who is searching for the given query. But the relation is more nuanced.
- You should analyze the query and some of the available documents. And then reason about what could be a meaningful definition of relevance in this case, and what the user could be looking for.
- Moreover, sometimes, the query could be even a prompt that is given to a Large Language Model (LLM) and the user wants to find the useful documents for the LLM that help answering/solving this prompt.
</RELEVANCE_DEFINITION>

Figure 8: Harness A system prompt (part 1): task and extended relevance definition.

uses reciprocal rank fusion (RRF) over the accumulated retrieval rankings. After recovery, no query execution is left without a final ranking.

<WORKFLOW>
- You are given a retrieval tool, powered by a dense embedding model, that takes a text query and returns the most similar documents.
- As explained above, reason and figure out what the meaning of relevance is in this case, and what could be relevant and useful information for the given query.
- You can call the search tool multiple times.
- Search for related documents to the user's query from different angles.
- If needed, revise your search queries based on the documents you find in previous steps.
- Once you are confident that you have found all the related and somewhat related documents and there are no more related documents in the corpus, call the "final_results" tool to finish the task.
- When calling "final_results", you must select exactly the 10 most relevant documents among all documents you have retrieved.
- When calling the "final_results" tool, the list of documents must be sorted in the decreasing level of relevance to the query. I.e., the first document is the most relevant to the query, the second document is the second most relevant to the query, and so on.
</WORKFLOW>

<BEST_PRACTICES>
- You should be thorough and find all related and somewhat related documents.
- The goal is to increase the **Recall** of your search attempt. So, if multiple documents are relevant to the given query, you should find and report all of them even if only a subset of them is enough for answering the query.
- **TIP**: you can look at the list of documents retrieved using the original query and think what other queries you can use to find the potentially related documents that are missing in these results.
</BEST_PRACTICES>

Figure 9: Harness A system prompt (part 2): workflow and search guidance, with initial documents enabled and a final list of ten documents.

Agent serving and sampling. Table 7 lists agent routes, reasoning controls, context windows, token prices, and token-priced usage totals for the reported experiments. Harness A agents, including Luna in the size study, are accessed through an NVIDIA NIM gateway using LiteLLM.[6] In Harness B, Luna uses OpenAI directly, while Lightning uses OpenRouter and local serving; the GPT-5.5 answer judge also uses OpenAI directly. The completion limit for Claude Sonnet 5 is 16,384 tokens because the default of 4,096 truncated its responses. Other agents retain their default completion limits; Table 7 lists each agent's reasoning setting. Each Harness A run processes two questions concurrently.

[6] https://docs.litellm.ai/

retrieve

Search for documents related to a query using dense retrieval.

Parameters (source descriptions):

query (string, required): The search query.

top_k (integer, optional, default 10): Number of documents to retrieve.

final_results

Signals the completion of the search process for the current query.

Use this tool when:
- You have found all the relevant documents to the query.
- Despite several attempts, you cannot find good documents for the given query.

The message should include:
- A brief summary of your exploration and the results
- Explanation if the search was unsuccessful

When reporting the selected document IDs, make sure:
- the list of document IDs is sorted in the decreasing level of relevance to the query. I.e., the first document in the list is the most relevant to the query, the second is the second most relevant to the query, and so on.
- You must choose exactly 10 document IDs when calling this function.

The successful_search field should be set to true if you believed you have found the most relevant documents to the user's query, and false otherwise. And partial if it is in between.

Parameters (source descriptions):

message (string, required): A message for the user to explain why you think you found all the related documents and there is no related document is missing. Also, include a short description of your exploration process. If your attempts to find related documents were unsuccessful, explain why.

doc_ids (array of string, required): List of document IDs that are relevant to the user's query sorted descending by their level of relevance to the user's query. I.e., the first document is the most relevant to the query, the second is the second most relevant to the query, and so on.

search_successful (string, required, values true, false, partial): Whether you managed to find all the related documents to the query.

Figure 10: Harness A retrieval and final-ranking tool descriptions and parameter schemas. Each search retrieves 100 candidates and returns ten unseen documents by default; the agent can change the return count. The final ranking contains ten documents.

Table 7: **Agent configuration, token prices, and token-priced costs.** B denotes Harness B. Prices are USD per million tokens; – denotes no supplied cache rate or charge. Totals are computed before rounding. Totals count retained runs once, including both Harness B studies and their answer judging. Lightning includes local-inference estimates; see Appendix E.

Agent configuration

Agent LLM	**Provider**	**Model identifier**	**Reasoning**	**Context**
Nemotron-3 Ultra	NIM	`nvidia/nemotron-3-ultra-nvfp4`	high	262K
MiniMax M3	NIM	`minimaxai/minimax-m3`	high	512K
Gemini 3.6 Flash	NIM	`google/gemini-3.6-flash`	high	1M
GPT-5.6 Terra	NIM	`openai/gpt-5.6-terra`	high	1M
Claude Sonnet 5	NIM	`anthropic/claude-sonnet-5`	adaptive	1M
GPT-5.6 Luna	NIM (size study); OpenAI (B)	`openai/gpt-5.6-luna`	high	1M
Gemini 3.5 Flash-Lite	NIM	`google/gemini-3.5-flash-lite`	high	1M
Claude Haiku 4.5	NIM	`anthropic/claude-haiku-4-5`	fixed budget	200K
Nemotron-3.5 Lightning	OpenRouter; local serving	`nvidia/nemotron-3.5-lightning`	high	262K

LLM	**USD per million tokens**				**Token-priced**
	Input	**Cache read**	**Cache write**	**Output**	**cost (USD)**
Nemotron-3 Ultra	0.6	0.12	–	2.4	7,495.87
MiniMax M3	0.3	0.06	–	1.2	1,511.12
Gemini 3.6 Flash	0.75	0.075	–	3.75	3,732.40
GPT-5.6 Terra	2.00	0.2	2.5	12.00	5,861.11
Claude Sonnet 5	2.00	0.2	2.5	10.00	7,533.48
GPT-5.6 Luna	0.2	0.02	0.25	1.2	152.69
Gemini 3.5 Flash-Lite	0.3	0.03	–	2.5	101.21
Claude Haiku 4.5	1.00	0.1	1.25	5.00	523.35
Nemotron-3.5 Lightning	0.08	0.04	–	0.2	123.92
GPT-5.5 (judge; OpenAI)	5.00	0.5	–	30.00	64.21
Total					27,099.35

Table 8: **One-shot retrieval on BCP.** Avg. combines the gold and evidence qrel sets.

Retriever	**nDCG@10**			**Recall@10**		
+Reranker	Gold	Evidence	Avg.	Gold	Evidence	Avg.
(a1) BM25	8.9 (±0.0)	10.4 (±0.0)	9.7 (±0.0)	10.3 (±0.0)	10.3 (±0.0)	10.3 (±0.0)
(a2) + Jina-Reranker-v3	16.3 (±0.0)	18.0 (±0.0)	17.2 (±0.0)	17.1 (±0.0)	15.8 (±0.0)	16.4 (±0.0)
(a3) + Qwen3-Reranker-4B	20.1 (±0.0)	22.4 (±0.0)	21.3 (±0.0)	19.9 (±0.0)	18.3 (±0.0)	19.1 (±0.0)
(a4) + ACARe-4B (M0)	**21.6** (±0.1)	**25.0** (±0.0)	**23.3** (±0.1)	**20.8** (±0.0)	**20.3** (±0.0)	**20.6** (±0.0)
(b1) Qwen3-Embedding-4B	12.8 (±0.0)	13.1 (±0.0)	12.9 (±0.0)	15.9 (±0.0)	13.5 (±0.0)	14.7 (±0.0)
(b2) + Jina-Reranker-v3	23.7 (±0.0)	23.6 (±0.0)	23.6 (±0.0)	29.4 (±0.0)	22.8 (±0.0)	26.1 (±0.0)
(b3) + Qwen3-Reranker-4B	31.5 (±0.0)	31.0 (±0.0)	31.3 (±0.0)	34.4 (±0.0)	27.8 (±0.0)	31.1 (±0.0)
(b4) + ACARe-4B (M0)	**35.7** (±0.0)	**36.0** (±0.0)	**35.9** (±0.0)	**39.5** (±0.0)	**31.8** (±0.0)	**35.7** (±0.0)
(c1) AgentIR-4B	25.0 (±0.0)	28.4 (±0.0)	26.7 (±0.0)	32.0 (±0.0)	28.5 (±0.0)	30.2 (±0.0)
(c2) + Jina-Reranker-v3	31.8 (±0.0)	31.6 (±0.0)	31.7 (±0.0)	40.5 (±0.0)	30.9 (±0.0)	35.7 (±0.0)
(c3) + Qwen3-Reranker-4B	44.4 (±0.0)	44.7 (±0.0)	44.6 (±0.0)	50.3 (±0.0)	40.7 (±0.0)	45.5 (±0.0)
(c4) + ACARe-4B (M0)	**48.5** (±0.0)	**48.9** (±0.0)	**48.7** (±0.0)	**54.7** (±0.0)	**44.7** (±0.0)	**49.7** (±0.0)
(d1) Nemotron-3-Embed-8B	28.0 (±0.0)	28.0 (±0.0)	28.0 (±0.0)	34.8 (±0.0)	27.7 (±0.0)	31.3 (±0.0)
(d2) + Jina-Reranker-v3	31.0 (±0.0)	30.5 (±0.0)	30.8 (±0.0)	37.2 (±0.0)	29.2 (±0.0)	33.2 (±0.0)
(d3) + Qwen3-Reranker-4B	44.6 (±0.0)	41.6 (±0.0)	43.1 (±0.0)	53.7 (±0.1)	38.8 (±0.0)	46.2 (±0.0)
(d4) + ACARe-4B (M0)	**52.0** (±0.0)	**49.4** (±0.0)	**50.7** (±0.0)	**61.5** (±0.0)	**45.6** (±0.0)	**53.6** (±0.0)

C PER-DATASET ONE-SHOT RESULTS

Tables 8 to 10 give the one-shot results by subset, using the row identifiers from Table 1. BCP's average combines gold and evidence labels; BRIGHT and TRQA averages weight their twelve and three subsets equally. The main table averages benchmarks equally before rounding.

No-reranker and Jina-Reranker-v3 results are identical across all five runs. The two 4B rerankers have small run-to-run variation, with standard deviations of at most 0.7 pp. A displayed standard deviation of 0.0 can represent a value below 0.05 pp.

Table 9: **One-shot retrieval on BRIGHT.** Avg. is the macro-average over twelve subsets.

Retriever +**Reranker**	Bio.	Earth	Econ.	Psy.	Rob.	StackO.	Sust.	Leet.	Pony	AoPS	TQA-Q	TQA-T	Avg.
nDCG@10													
(a1) BM25	17.6 (±0.0)	28.7 (±0.0)	15.5 (±0.0)	14.4 (±0.0)	12.3 (±0.0)	17.8 (±0.0)	15.8 (±0.0)	26.1 (±0.0)	5.0 (±0.0)	6.1 (±0.0)	8.0 (±0.0)	1.7 (±0.0)	14.1 (±0.0)
(a2) + Jina-Reranker-v3	22.8 (±0.0)	35.9 (±0.0)	**21.8** (±0.0)	29.1 (±0.0)	20.3 (±0.0)	18.6 (±0.0)	22.7 (±0.0)	20.4 (±0.0)	**12.4** (±0.0)	8.1 (±0.0)	10.8 (±0.0)	**9.3** (±0.0)	19.4 (±0.0)
(a3) + Qwen3-Reranker-4B	24.1 (±0.1)	37.5 (±0.0)	18.5 (±0.0)	28.6 (±0.1)	25.5 (±0.0)	**28.6** (±0.0)	22.1 (±0.0)	33.4 (±0.0)	10.1 (±0.0)	10.0 (±0.0)	**12.4** (±0.0)	8.9 (±0.0)	21.6 (±0.0)
(a4) + ACARe-4B (M0)	**27.4** (±0.0)	**41.6** (±0.1)	21.5 (±0.0)	**30.0** (±0.0)	**27.3** (±0.1)	27.2 (±0.1)	**23.8** (±0.1)	**33.6** (±0.0)	11.0 (±0.0)	**10.5** (±0.0)	**12.4** (±0.1)	8.6 (±0.0)	**22.9** (±0.0)
(b1) Qwen3-Embedding-4B	17.0 (±0.0)	34.7 (±0.0)	16.5 (±0.0)	23.2 (±0.0)	11.7 (±0.0)	15.9 (±0.0)	16.7 (±0.0)	36.2 (±0.0)	1.2 (±0.0)	**9.7** (±0.0)	35.9 (±0.0)	**41.3** (±0.0)	21.7 (±0.0)
(b2) + Jina-Reranker-v3	24.2 (±0.0)	37.9 (±0.0)	23.8 (±0.0)	27.3 (±0.0)	17.3 (±0.0)	15.6 (±0.0)	25.8 (±0.0)	19.1 (±0.0)	4.7 (±0.0)	5.3 (±0.0)	26.4 (±0.0)	34.0 (±0.0)	21.8 (±0.0)
(b3) + Qwen3-Reranker-4B	29.8 (±0.0)	37.1 (±0.0)	20.4 (±0.0)	26.7 (±0.0)	19.8 (±0.0)	26.5 (±0.0)	23.5 (±0.2)	39.8 (±0.0)	**6.6** (±0.0)	9.1 (±0.0)	39.2 (±0.0)	36.6 (±0.1)	26.3 (±0.0)
(b4) + ACARe-4B (M0)	**34.6** (±0.0)	**42.8** (±0.1)	**25.1** (±0.1)	**30.0** (±0.1)	**22.8** (±0.1)	**27.2** (±0.0)	**26.6** (±0.0)	**40.2** (±0.0)	6.3 (±0.0)	8.6 (±0.0)	**40.8** (±0.1)	38.3 (±0.1)	**28.6** (±0.0)
(c1) AgentIR-4B	24.5 (±0.0)	**43.9** (±0.0)	17.9 (±0.0)	24.5 (±0.0)	14.2 (±0.0)	17.6 (±0.0)	13.8 (±0.0)	37.5 (±0.0)	2.8 (±0.0)	8.2 (±0.0)	36.0 (±0.0)	28.6 (±0.0)	22.5 (±0.0)
(c2) + Jina-Reranker-v3	26.5 (±0.0)	37.4 (±0.0)	23.6 (±0.0)	29.3 (±0.0)	18.9 (±0.0)	16.9 (±0.0)	24.3 (±0.0)	18.8 (±0.0)	5.2 (±0.0)	5.5 (±0.0)	26.4 (±0.0)	32.2 (±0.0)	22.1 (±0.0)
(c3) + Qwen3-Reranker-4B	30.2 (±0.1)	37.8 (±0.2)	18.9 (±0.0)	27.6 (±0.0)	20.4 (±0.0)	**28.9** (±0.0)	23.0 (±0.0)	39.3 (±0.0)	**14.5** (±0.1)	**9.4** (±0.0)	**40.4** (±0.1)	37.1 (±0.0)	27.3 (±0.0)
(c4) + ACARe-4B (M0)	**36.7** (±0.1)	43.4 (±0.1)	**23.7** (±0.0)	**29.6** (±0.0)	**23.1** (±0.0)	28.7 (±0.0)	**26.1** (±0.1)	**40.6** (±0.0)	**14.5** (±0.0)	9.1 (±0.1)	**40.4** (±0.1)	**37.2** (±0.0)	**29.4** (±0.0)
(d1) Nemotron-3-Embed-8B	33.8 (±0.0)	41.4 (±0.0)	23.8 (±0.0)	31.1 (±0.0)	20.1 (±0.0)	19.7 (±0.0)	22.3 (±0.0)	30.6 (±0.0)	3.5 (±0.0)	**10.3** (±0.0)	36.4 (±0.0)	30.1 (±0.0)	25.3 (±0.0)
(d2) + Jina-Reranker-v3	25.2 (±0.0)	37.3 (±0.0)	23.3 (±0.0)	27.8 (±0.0)	20.4 (±0.0)	16.0 (±0.0)	25.2 (±0.0)	19.2 (±0.0)	6.5 (±0.0)	5.8 (±0.0)	27.5 (±0.0)	30.5 (±0.0)	22.1 (±0.0)
(d3) + Qwen3-Reranker-4B	30.5 (±0.1)	37.3 (±0.0)	21.8 (±0.0)	28.4 (±0.0)	22.1 (±0.1)	27.0 (±0.0)	22.9 (±0.0)	37.9 (±0.0)	13.8 (±0.0)	10.0 (±0.0)	35.8 (±0.0)	36.5 (±0.3)	27.0 (±0.0)
(d4) + ACARe-4B (M0)	**37.0** (±0.1)	**43.3** (±0.0)	**25.9** (±0.1)	**32.0** (±0.0)	**25.4** (±0.0)	**27.3** (±0.1)	**25.9** (±0.1)	**39.1** (±0.0)	**14.1** (±0.1)	8.8 (±0.0)	**37.7** (±0.0)	**38.7** (±0.1)	**29.6** (±0.0)
Recall@10													
(a1) BM25	20.7 (±0.0)	33.9 (±0.0)	15.1 (±0.0)	17.0 (±0.0)	16.3 (±0.0)	23.3 (±0.0)	20.1 (±0.0)	30.5 (±0.0)	2.7 (±0.0)	5.4 (±0.0)	9.4 (±0.0)	2.6 (±0.0)	16.4 (±0.0)
(a2) + Jina-Reranker-v3	27.4 (±0.0)	39.6 (±0.0)	22.0 (±0.0)	33.5 (±0.0)	25.3 (±0.0)	25.1 (±0.0)	**27.2** (±0.0)	26.1 (±0.0)	**6.5** (±0.0)	8.1 (±0.0)	10.6 (±0.0)	**9.2** (±0.0)	21.7 (±0.0)
(a3) + Qwen3-Reranker-4B	27.9 (±0.1)	43.6 (±0.0)	24.3 (±0.0)	33.2 (±0.0)	29.5 (±0.0)	**32.0** (±0.0)	25.3 (±0.0)	**39.6** (±0.0)	4.6 (±0.1)	10.2 (±0.0)	**12.8** (±0.1)	**9.2** (±0.0)	24.4 (±0.0)
(a4) + ACARe-4B (M0)	**31.1** (±0.1)	**46.0** (±0.0)	**25.8** (±0.0)	**35.6** (±0.0)	**31.2** (±0.0)	**32.0** (±0.0)	26.0 (±0.2)	39.0 (±0.0)	5.1 (±0.0)	**10.7** (±0.1)	12.4 (±0.2)	**9.2** (±0.0)	**25.4** (±0.0)
(b1) Qwen3-Embedding-4B	20.3 (±0.0)	40.8 (±0.0)	19.7 (±0.0)	29.3 (±0.0)	15.3 (±0.0)	22.9 (±0.0)	22.3 (±0.0)	43.1 (±0.0)	1.1 (±0.0)	**10.2** (±0.0)	39.0 (±0.0)	**51.4** (±0.0)	26.3 (±0.0)
(b2) + Jina-Reranker-v3	28.7 (±0.0)	42.3 (±0.0)	26.2 (±0.0)	30.6 (±0.0)	19.7 (±0.0)	21.1 (±0.0)	**31.6** (±0.0)	21.9 (±0.0)	2.7 (±0.0)	6.7 (±0.0)	30.5 (±0.0)	44.0 (±0.0)	25.5 (±0.0)
(b3) + Qwen3-Reranker-4B	35.7 (±0.1)	44.4 (±0.0)	26.1 (±0.0)	33.4 (±0.0)	23.9 (±0.0)	34.6 (±0.0)	28.1 (±0.0)	53.4 (±0.0)	**3.6** (±0.0)	9.9 (±0.0)	45.3 (±0.0)	47.9 (±0.7)	32.2 (±0.1)
(b4) + ACARe-4B (M0)	**40.3** (±0.0)	**47.5** (±0.0)	**32.9** (±0.1)	**36.2** (±0.0)	**25.1** (±0.0)	**36.5** (±0.0)	31.4 (±0.0)	**53.5** (±0.0)	3.5 (±0.0)	9.4 (±0.0)	**46.2** (±0.1)	50.7 (±0.3)	**34.4** (±0.0)
(c1) AgentIR-4B	29.7 (±0.0)	46.3 (±0.0)	20.3 (±0.0)	30.9 (±0.0)	19.6 (±0.0)	27.3 (±0.0)	20.4 (±0.0)	46.9 (±0.0)	2.1 (±0.0)	9.4 (±0.0)	41.5 (±0.0)	36.2 (±0.0)	27.6 (±0.0)
(c2) + Jina-Reranker-v3	32.8 (±0.0)	41.1 (±0.0)	24.3 (±0.0)	**34.2** (±0.0)	22.7 (±0.0)	24.7 (±0.0)	29.3 (±0.0)	23.1 (±0.0)	3.6 (±0.0)	5.5 (±0.0)	31.1 (±0.0)	44.6 (±0.0)	26.4 (±0.0)
(c3) + Qwen3-Reranker-4B	36.4 (±0.3)	44.5 (±0.1)	24.5 (±0.0)	33.9 (±0.0)	25.4 (±0.0)	38.0 (±0.0)	29.1 (±0.1)	**53.1** (±0.2)	7.5 (±0.0)	9.9 (±0.0)	46.0 (±0.1)	47.9 (±0.0)	33.0 (±0.0)
(c4) + ACARe-4B (M0)	**44.1** (±0.1)	**48.7** (±0.2)	**31.5** (±0.0)	**34.2** (±0.0)	**25.6** (±0.0)	**39.5** (±0.0)	**31.2** (±0.2)	51.8 (±0.0)	**7.7** (±0.0)	**10.0** (±0.0)	**46.4** (±0.3)	**48.0** (±0.0)	**34.9** (±0.0)
(d1) Nemotron-3-Embed-8B	39.1 (±0.0)	45.3 (±0.0)	27.2 (±0.0)	35.3 (±0.0)	23.9 (±0.0)	29.0 (±0.0)	29.5 (±0.0)	39.3 (±0.0)	2.2 (±0.0)	10.1 (±0.0)	39.8 (±0.0)	41.9 (±0.0)	30.2 (±0.0)
(d2) + Jina-Reranker-v3	28.7 (±0.0)	40.7 (±0.0)	24.0 (±0.0)	32.9 (±0.0)	24.0 (±0.0)	21.5 (±0.0)	**33.6** (±0.0)	22.8 (±0.0)	4.0 (±0.0)	5.9 (±0.0)	32.6 (±0.0)	43.5 (±0.0)	26.2 (±0.0)
(d3) + Qwen3-Reranker-4B	37.5 (±0.4)	44.4 (±0.0)	28.9 (±0.0)	36.1 (±0.0)	26.7 (±0.0)	35.2 (±0.0)	27.6 (±0.0)	**48.5** (±0.0)	7.4 (±0.0)	**11.3** (±0.1)	40.9 (±0.0)	**48.3** (±0.6)	32.7 (±0.1)
(d4) + ACARe-4B (M0)	**43.8** (±0.2)	**48.3** (±0.1)	**34.9** (±0.0)	**39.7** (±0.0)	**28.8** (±0.0)	**36.5** (±0.1)	30.9 (±0.1)	47.9 (±0.1)	**7.5** (±0.0)	9.9 (±0.0)	**43.1** (±0.0)	**48.3** (±0.1)	**35.0** (±0.0)

Table 10: **One-shot retrieval on TRQA.** Avg. is the macro-average over three subsets.

Retriever	**nDCG@10**				**Recall@10**			
+Reranker	Wiki1	Wiki2	E-comm.	Avg.	Wiki1	Wiki2	E-comm.	Avg.
(a1) BM25	20.8 (±0.0)	8.0 (±0.0)	8.3 (±0.0)	12.4 (±0.0)	15.9 (±0.0)	5.5 (±0.0)	1.8 (±0.0)	7.7 (±0.0)
(a2) + Jina-Reranker-v3	27.4 (±0.0)	11.6 (±0.0)	8.2 (±0.0)	15.8 (±0.0)	20.9 (±0.0)	8.9 (±0.0)	2.1 (±0.0)	10.6 (±0.0)
(a3) + Qwen3-Reranker-4B	29.7 (±0.0)	13.1 (±0.0)	9.1 (±0.0)	17.3 (±0.0)	22.8 (±0.0)	9.8 (±0.0)	2.4 (±0.0)	11.7 (±0.0)
(a4) + ACARe-4B (M0)	**32.4** (±0.0)	**14.5** (±0.0)	**10.5** (±0.1)	**19.1** (±0.0)	**23.8** (±0.0)	**10.2** (±0.0)	**2.7** (±0.0)	**12.2** (±0.0)
(b1) Qwen3-Embedding-4B	19.6 (±0.0)	1.8 (±0.0)	7.9 (±0.0)	9.8 (±0.0)	18.1 (±0.0)	1.2 (±0.0)	2.0 (±0.0)	7.1 (±0.0)
(b2) + Jina-Reranker-v3	28.9 (±0.0)	6.0 (±0.0)	7.9 (±0.0)	14.3 (±0.0)	24.9 (±0.0)	4.7 (±0.0)	2.2 (±0.0)	10.6 (±0.0)
(b3) + Qwen3-Reranker-4B	33.0 (±0.1)	6.5 (±0.0)	8.1 (±0.0)	15.9 (±0.0)	27.1 (±0.1)	4.6 (±0.1)	2.1 (±0.0)	11.2 (±0.0)
(b4) + ACARe-4B (M0)	**37.5** (±0.1)	**7.6** (±0.0)	**9.8** (±0.1)	**18.3** (±0.0)	**30.2** (±0.1)	**5.3** (±0.0)	**2.5** (±0.0)	**12.7** (±0.0)
(c1) AgentIR-4B	32.0 (±0.0)	11.1 (±0.0)	9.8 (±0.0)	17.6 (±0.0)	24.2 (±0.0)	8.6 (±0.0)	2.6 (±0.0)	11.8 (±0.0)
(c2) + Jina-Reranker-v3	30.4 (±0.0)	12.2 (±0.0)	9.5 (±0.0)	17.4 (±0.0)	25.6 (±0.0)	9.3 (±0.0)	2.6 (±0.0)	12.5 (±0.0)
(c3) + Qwen3-Reranker-4B	36.4 (±0.1)	14.1 (±0.0)	9.1 (±0.0)	19.9 (±0.0)	29.3 (±0.1)	11.1 (±0.0)	2.6 (±0.0)	14.3 (±0.0)
(c4) + ACARe-4B (M0)	**40.2** (±0.0)	**15.5** (±0.1)	**10.6** (±0.1)	**22.1** (±0.0)	**31.3** (±0.0)	**11.6** (±0.0)	**2.7** (±0.0)	**15.2** (±0.0)
(d1) Nemotron-3-Embed-8B	27.0 (±0.0)	4.8 (±0.0)	**11.6** (±0.0)	14.4 (±0.0)	24.1 (±0.0)	3.4 (±0.0)	**3.2** (±0.0)	10.2 (±0.0)
(d2) + Jina-Reranker-v3	30.1 (±0.0)	10.3 (±0.0)	7.6 (±0.0)	16.0 (±0.0)	24.9 (±0.0)	8.8 (±0.0)	1.9 (±0.0)	11.9 (±0.0)
(d3) + Qwen3-Reranker-4B	35.0 (±0.0)	11.6 (±0.0)	9.0 (±0.1)	18.5 (±0.0)	28.7 (±0.1)	9.4 (±0.0)	2.5 (±0.1)	13.5 (±0.0)
(d4) + ACARe-4B (M0)	**40.2** (±0.1)	**13.5** (±0.0)	11.1 (±0.1)	**21.6** (±0.1)	**32.4** (±0.2)	**10.4** (±0.1)	2.9 (±0.0)	**15.2** (±0.1)

Table 11: **Agentic retrieval on BCP**, with BM25 as the first-stage retriever.

Agent LLM	**nDCG@10↑**	**Recall@10↑**	**Avg. per query stats**		
+Reranker			**Searches↓**	**LLM turns↓**	**Cost (¢)↓**
(a1) Nemotron-3 Ultra	66.9 (±1.3)	68.6 (±1.0)	14.9 (±0.4)	16.0 (±0.4)	35.0 (±1.8)
(a2) + Qwen3-Reranker-4B	69.7 (±1.1)	72.2 (±1.5)†	13.1 (±0.3)†	14.1 (±0.3)†	30.8 (±0.8)†
(a3) + ACARe-4B (M3)	**72.6** (±1.2)†*	**75.2** (±0.9)†	**11.6** (±0.3)†*	**12.6** (±0.3)†*	**25.9** (±1.4)†*
(b1) MiniMax M3	76.7 (±0.7)	79.8 (±0.4)	17.0 (±0.4)	9.1 (±0.2)	9.5 (±1.2)
(b2) + Qwen3-Reranker-4B	80.4 (±1.4)†	84.0 (±1.3)†	14.7 (±0.2)†	8.1 (±0.1)†	8.7 (±0.7)
(b3) + ACARe-4B (M3)	**82.5** (±1.6)†	**86.0** (±1.4)†	**14.1** (±0.2)†*	**7.9** (±0.1)†*	**8.4** (±0.5)
(c1) Gemini 3.6 Flash	72.3 (±0.6)	73.7 (±0.6)	8.1 (±0.1)	9.1 (±0.1)	13.7 (±0.4)
(c2) + Qwen3-Reranker-4B	78.3 (±0.9)†	80.4 (±0.9)†	7.0 (±0.2)†	8.0 (±0.2)†	12.2 (±0.4)†
(c3) + ACARe-4B (M2)	**82.1** (±0.5)†*	**84.6** (±0.5)†*	**6.7** (±0.1)†*	**7.7** (±0.1)†*	**11.7** (±0.6)†
(d1) GPT-5.6 Terra	83.0 (±1.0)	85.6 (±0.9)	20.8 (±0.4)	4.5 (±0.0)	47.2 (±1.7)
(d2) + Qwen3-Reranker-4B	85.2 (±0.8)	87.4 (±0.8)	18.4 (±0.2)†	4.2 (±0.1)†	39.5 (±0.7)†
(d3) + ACARe-4B (M2)	**86.2** (±0.8)†	**88.7** (±0.8)†	**17.5** (±0.5)†	**4.1** (±0.1)†	**38.3** (±2.0)†
(e1) Claude Sonnet 5	78.7 (±0.6)	82.1 (±0.5)	15.9 (±0.6)	8.9 (±0.3)	63.8 (±3.3)
(e2) + Qwen3-Reranker-4B	80.2 (±0.9)	83.1 (±0.9)	12.7 (±0.2)†	7.4 (±0.1)†	48.9 (±0.8)†
(e3) + ACARe-4B (M2)	**82.2** (±0.6)†	**85.2** (±0.6)†*	**11.8** (±0.4)†*	**7.0** (±0.2)†	**45.6** (±2.2)†

D PER-BENCHMARK AGENTIC RESULTS

Tables 11 to 13 report results underlying Table 2, with BM25 on BCP, Nemotron-3-Embed-8B on BRIGHT, and AgentIR-4B on TRQA. Each agent is evaluated without reranking and with each 4B reranker. Aggregation and testing follow Appendix I.

The benchmark breakdown shows effects obscured by overall averages. In particular, Qwen3-Reranker-4B lowers mean TRQA nDCG@10 for four agents, while ACARe improves all five. Document selection and caching also affect API cost (Appendix E).

Table 12: **Agentic retrieval on BRIGHT**, with Nemotron-3-Embed-8B as the first-stage retriever.

Agent LLM	**nDCG@10↑**	**Recall@10↑**	**Avg. per query stats**		
+Reranker			**Searches↓**	**LLM turns↓**	**Cost (¢)↓**
(a1) Nemotron-3 Ultra	41.1 (±0.4)	45.4 (±0.4)	8.1 (±0.1)	9.1 (±0.1)	9.2 (±0.5)
(a2) + Qwen3-Reranker-4B	42.5 (±0.6)†	**47.1** (±0.6)†	7.5 (±0.0)†	8.5 (±0.0)†	**8.7** (±0.5)
(a3) + ACARe-4B (M3)	**42.7** (±0.3)†	**47.1** (±0.6)†	**7.1** (±0.0)†*	**8.1** (±0.0)†*	**8.7** (±0.5)
(b1) MiniMax M3	41.7 (±0.3)	46.4 (±0.5)	11.9 (±0.1)	6.6 (±0.1)	3.7 (±0.2)
(b2) + Qwen3-Reranker-4B	43.0 (±0.4)†	48.1 (±0.3)†	10.8 (±0.2)†	6.1 (±0.1)†	**3.5** (±0.2)
(b3) + ACARe-4B (M3)	**43.2** (±0.5)†	**48.2** (±0.4)†	**10.4** (±0.1)†*	**5.9** (±0.0)†	3.7 (±0.2)
(c1) Gemini 3.6 Flash	43.3 (±0.2)	48.1 (±0.5)	7.0 (±0.1)	8.0 (±0.0)	8.3 (±0.4)
(c2) + Qwen3-Reranker-4B	45.3 (±0.2)†	50.3 (±0.2)†	6.5 (±0.0)†	7.5 (±0.0)†	**8.1** (±0.3)
(c3) + ACARe-4B (M2)	**45.4** (±0.3)†	**51.0** (±0.4)†	**6.2** (±0.0)†*	**7.2** (±0.0)†*	8.3 (±0.3)
(d1) GPT-5.6 Terra	44.0 (±0.3)	48.9 (±0.3)	9.1 (±0.1)	3.0 (±0.0)	**11.8** (±0.1)
(d2) + Qwen3-Reranker-4B	45.0 (±0.3)†	49.7 (±0.5)	8.8 (±0.0)†	**2.9** (±0.0)†	**11.8** (±0.0)
(d3) + ACARe-4B (M2)	**45.1** (±0.2)†	**50.4** (±0.2)†	**8.4** (±0.0)†*	**2.9** (±0.0)†*	12.4 (±0.1)†*
(e1) Claude Sonnet 5	42.1 (±0.1)	47.2 (±0.4)	8.3 (±0.0)	5.2 (±0.0)	**16.9** (±0.3)
(e2) + Qwen3-Reranker-4B	42.5 (±0.3)	47.3 (±0.5)	7.8 (±0.1)†	4.9 (±0.0)†	17.2 (±0.2)
(e3) + ACARe-4B (M2)	**43.2** (±0.4)†	**48.4** (±0.5)†	**7.1** (±0.0)†*	**4.7** (±0.0)†*	17.1 (±0.2)

Table 13: **Agentic retrieval on TRQA**, with AgentIR-4B as the first-stage retriever.

Agent LLM	**nDCG@10↑**	**Recall@10↑**	**Avg. per query stats**		
+Reranker			**Searches↓**	**LLM turns↓**	**Cost (¢)↓**
(a1) Nemotron-3 Ultra	36.7 (±0.3)	24.9 (±0.2)	10.4 (±0.1)	11.4 (±0.1)	12.2 (±0.9)
(a2) + Qwen3-Reranker-4B	35.8 (±0.6)	24.9 (±0.4)	10.3 (±0.2)	11.3 (±0.2)	11.8 (±0.6)
(a3) + ACARe-4B (M3)	**38.3** (±0.4)†*	**26.1** (±0.4)†*	**9.1** (±0.1)†*	**10.1** (±0.1)†*	**10.4** (±0.5)
(b1) MiniMax M3	34.8 (±0.9)	24.3 (±0.6)	21.4 (±0.3)	10.6 (±0.2)	6.3 (±0.5)
(b2) + Qwen3-Reranker-4B	34.1 (±0.4)	24.4 (±0.4)	19.8 (±0.4)†	9.9 (±0.2)†	5.5 (±0.5)
(b3) + ACARe-4B (M3)	**37.4** (±0.6)†*	**26.1** (±0.1)†*	**18.5** (±0.3)†*	**9.3** (±0.2)†*	**5.4** (±0.4)
(c1) Gemini 3.6 Flash	40.6 (±0.2)	26.8 (±0.2)	15.1 (±0.5)	16.0 (±0.5)	19.8 (±1.3)
(c2) + Qwen3-Reranker-4B	40.1 (±0.5)	27.0 (±0.3)	14.6 (±0.3)	15.6 (±0.3)	**18.3** (±0.9)
(c3) + ACARe-4B (M2)	**42.2** (±0.5)†*	**28.0** (±0.4)†*	**14.2** (±0.3)	**15.2** (±0.2)	19.6 (±0.7)
(d1) GPT-5.6 Terra	43.3 (±0.3)	27.9 (±0.5)	25.5 (±0.3)	5.0 (±0.0)	25.7 (±0.3)
(d2) + Qwen3-Reranker-4B	42.7 (±0.6)	27.7 (±0.4)	**23.7** (±0.2)†	**4.8** (±0.0)†	**23.8** (±0.1)†
(d3) + ACARe-4B (M2)	**44.1** (±0.4)*	**28.6** (±0.3)	24.0 (±0.3)†	**4.8** (±0.0)†	25.0 (±0.3)†*
(e1) Claude Sonnet 5	36.3 (±0.5)	25.2 (±0.5)	16.7 (±0.1)	7.8 (±0.1)	28.7 (±0.4)
(e2) + Qwen3-Reranker-4B	36.5 (±0.4)	25.6 (±0.2)	16.3 (±0.3)	7.4 (±0.1)†	**27.8** (±0.3)
(e3) + ACARe-4B (M2)	**38.6** (±0.6)†*	**26.5** (±0.4)†*	**15.9** (±0.2)†	**7.3** (±0.1)†	28.1 (±0.5)

E COST ANALYSIS

We report the agent LLM's API cost per query. Retrieval and reranker serving costs are excluded from all experiments, including those in the Analysis section (Section 6). Reported savings concern agent API charges, not total system cost; excluded costs may matter more for cheaper agents.

Costs are computed from recorded token usage using the rates in Table 7, with reasoning tokens counted as output. Public list prices are as of September 2026. Lightning uses the experiment-recorded rates for its OpenRouter route: \$0.08 for input, \$0.04 for cache reads, and \$0.20 for output per million tokens. Local inference is valued at the same rates, as a token-priced estimate. GPT-5.5 judging uses recorded OpenAI rates of \$5 for input and \$30 for output per million tokens; its published cache-read rate[7] is \$0.50. No cache usage was recorded for the judge. MiniMax M3 has no recorded cache writes, so its cost calculation requires no cache-write rate. Because the conversation is resubmitted at each turn, a document can incur input charges repeatedly after it is first retrieved. Fewer searches and shorter trajectories can therefore reduce cost, but the number of calls alone does not determine the savings.

The token-priced totals in Table 7 sum each recorded run's per-query cost multiplied by its number of questions, plus retained answer-judge usage for both Harness B studies. Runs reused across studies are counted once using their source-result paths. These totals exclude omitted experiments, unrecorded attempts, training costs, and judging outside those studies. Harness B contributes 36 ClimbMix runs and 30 BCP context-control runs (Appendices J.1 and J.2), totaling 8,052 question executions. Its recorded usage is valued at \$50.30 for GPT-5.6 Luna, \$123.92 for Nemotron-3.5 Lightning, and \$64.21 for 7,802 GPT-5.5 judgments through OpenAI. The remaining 250 executions incur no judge cost. The Lightning total includes local inference valued at API token rates, so the combined total is a token-priced accounting estimate. Harness B cost comparisons are omitted because serving conditions affected billed cost: Lightning runs switched between API providers and local serving under heavy throttling, and each Luna reranking configuration ran on its own vLLM server under a different load, so queuing delays changed agent-side prompt-cache reuse. Search counts are reported throughout the agentic evaluations as a measure of retrieval effort.

Two additional factors affect the cost of a trajectory. First, document length and retrieval timing determine how much text appears in later prompts: a long document retrieved early contributes more repeated input than one retrieved near the end. Second, caching determines how much of that input receives a discounted rate. With implicit caching, cache availability can vary with endpoint workload and cache eviction, changing the billed cost even for similar token volumes. The reranker influences which documents are returned, but does not control the provider's cache. Our measurements do not isolate the contributions of caching and document timing to cost variance.

Averaged over benchmarks, ACARe uses significantly fewer search calls than Qwen3-Reranker-4B for every agent, but has significantly lower API cost only for Nemotron-3 Ultra, by approximately 12%. Sonnet has a 3% mean saving that does not remain significant after correction. The other three agents' cost differences are also nonsignificant, including MiniMax M3's 0.069-cent saving ($p = 1.000$; Table 17). These five runs do not establish equal costs.

Benchmark-specific outcomes are more mixed. Relative to Qwen3-Reranker-4B, ACARe has higher mean API cost in seven cases: Nemotron-3 Ultra, MiniMax M3, Gemini 3.6 Flash, and GPT-5.6 Terra on BRIGHT, and Gemini 3.6 Flash, GPT-5.6 Terra, and Claude Sonnet 5 on TRQA. Only Terra's two increases remain significant after correction (Table 17). MiniMax M3's cost differences are nonsignificant on all three benchmarks ($p = 0.969$ on BRIGHT; $p = 1.000$ on BCP and TRQA). Thus the aggregate quality and search-effort gains should not be interpreted as uniform cost improvements. Search calls provide a direct measure of retrieval effort, while API cost measures the combined effect of trajectory length, document content, and provider pricing.

F LATENCY AND RERANKING DEPTH

Table 14 examines whether reduced search effort translates into lower end-to-end latency for Nemotron-3 Ultra on BCP with BM25. Each search retrieves and reranks d candidates, for

[7] https://developers.openai.com/api/docs/models/gpt-5.5

$d \in \{50, 100\}$; the no-reranking baseline retrieves 100. Output depth remains ten unseen documents by default. The reranker runs on one A100 GPU, with two questions processed concurrently. Latency is elapsed time per query, including retrieval, reranking, LLM calls, and any recovery, rather than total run time divided by the number of questions.

Table 14: **Reranking depth** on BCP with Nemotron-3 Ultra and BM25. Each depth compares the two rerankers with the shared no-reranker baseline; row definitions are in Appendix F. Tests use this study's correction family (Appendix I); marks may differ for shared runs in Tables 11 to 13.

		Depth 50		Depth 100	
	No reranker	**Qwen3-Reranker-4B**	**ACARe-4B** (M3)	**Qwen3-Reranker-4B**	**ACARe-4B** (M3)
Quality					
nDCG@10↑	66.9 (±1.3)	69.0 (±1.0)	**70.5** (±1.1)†	69.7 (±1.1)†	**72.6** (±1.2)$^{\dagger *}$
Recall@10↑	68.6 (±1.0)	71.3 (±1.3)†	**72.5** (±1.5)†	72.2 (±1.5)†	**75.2** (±0.9)$^{\dagger *}$
Avg. per query stats					
Searches↓	14.9 (±0.4)	13.9 (±0.1)†	**13.3** (±0.4)†	13.1 (±0.3)†	**11.6** (±0.3)$^{\dagger *}$
LLM turns↓	16.0 (±0.4)	14.9 (±0.1)†	**14.4** (±0.4)†	14.1 (±0.3)†	**12.6** (±0.3)$^{\dagger *}$
Cost (¢)↓	35.0 (±1.8)	33.1 (±0.7)	**31.9** (±2.0)	30.8 (±0.8)†	**25.9** (±1.4)$^{\dagger *}$
End-to-end latency (s)↓	**71.7** (±3.1)	90.8 (±1.0)†	84.1 (±2.8)$^{\dagger *}$	99.5 (±4.2)†	91.6 (±4.5)†

Both rerankers increase latency relative to no reranking. Compared with Qwen3-Reranker-4B, ACARe reduces mean end-to-end latency from 90.8 to 84.1 s at depth 50 ($p = 0.025$) and from 99.5 to 91.6 s at depth 100 ($p = 0.110$). Corresponding nDCG@10 values increase from 69.0 to 70.5 ($p = 0.299$) and from 69.7 to 72.6 ($p = 0.025$). At depth 100, Recall@10 also improves from 72.2 to 75.2 ($p = 0.038$). Thus the depth-50 latency reduction and depth-100 gains in both quality metrics are significant under the 36-comparison correction defined in Appendix I. Relative to no reranking, depth 100 adds approximately 0.29 nDCG@10 pp per second of added latency for ACARe, compared with 0.10 for Qwen3-Reranker-4B.

ACARe reduces mean LLM turns by 0.6 at depth 50 ($p = 0.140$) and 1.5 at depth 100 ($p = 0.001$). These observations are consistent with shorter trajectories reducing end-to-end work, although they do not isolate the contribution of each component to latency.

G SMALL VS. LARGE AGENTS

Table 15 reports five-run results on BCP with BM25 and reranking depth 100 for three pairs: GPT-5.6 Luna/Terra, Gemini 3.5 Flash-Lite/3.6 Flash, and Claude Haiku 4.5/Sonnet 5. ACARe uses the original-question mode M2 for all six agents. Relative cost is API cost divided by the mean API cost of the larger model without reranking in the same family. This fixed normalization also scales the reported standard deviations; the baseline mean is 100%, but its standard deviation can be nonzero.

ACARe improves mean nDCG@10 over no reranking by 3.3–12.6 pp, and over Qwen3-Reranker-4B by 1.0–6.3 pp. The gains over no reranking are significant for all six agents under the reported tests. After correction across the 51 size-study comparisons, nDCG@10 gains over Qwen3-Reranker-4B are significant for Flash-Lite ($p = 0.019$), Flash ($p = 0.002$), Haiku ($p = 0.014$), and Sonnet ($p = 0.036$). Recall@10 gains remain significant for Flash-Lite, Flash, Haiku, and Sonnet. Within each family, the smaller agent gains at least as much as the larger one relative to no reranking. These are comparisons between deployed model tiers, not isolated changes in parameter count; the Gemini and Claude pairs also differ in version or generation.

Let S_0 and L_0 denote the smaller and larger agents' mean nDCG@10 without reranking, and let S_r denote the smaller agent's mean with reranker r. Gap closure relative to the larger agent without reranking is $(S_r - S_0)/(L_0 - S_0)$. Qwen3-Reranker-4B closes 46%, 46%, and 26% of the GPT, Gemini, and Claude gaps, respectively; ACARe closes 73%, 74%, and 51%. For ACARe, the remaining GPT and Gemini gaps are 1.4 pp ($p = 0.331$) and 3.6 pp ($p = 0.025$); the nonsignificant GPT difference does not establish equal performance.

Table 15: **Small versus large agents** on BCP with BM25. Relative cost is normalized to the larger agent without reranking in each family. Tests use this study's correction family (Appendix I); marks may differ for shared runs in Tables 11 to 13.

Agent LLM	**nDCG@10↑**	**Recall@10↑**	**Avg. per query stats**		**Rel. cost (%)**
+Reranker			**Searches↓**	**Cost (¢)↓**	
(a1) GPT-5.6 Luna (small)	77.9 (±0.7)	81.3 (±0.7)	19.0 (±0.5)	3.8 (±0.2)	8.0 (±0.4)
(a2) + Qwen3-Reranker-4B	80.3 (±0.4)	82.7 (±0.4)	17.0 (±0.5)	3.4 (±0.1)	7.1 (±0.2)
(a3) + ACARe-4B (M2)	**81.6** (±1.0)†	**84.5** (±1.3)†	**15.4** (±0.5)$^{\dagger *}$	**3.1** (±0.1)†	**6.6** (±0.3)†
(b1) GPT-5.6 Terra (large)	83.0 (±1.0)	85.6 (±0.9)	20.8 (±0.4)	47.2 (±1.7)	100.0 (±3.7)
(b2) + Qwen3-Reranker-4B	85.2 (±0.8)	87.4 (±0.8)	18.4 (±0.2)	39.5 (±0.7)	83.7 (±1.6)
(b3) + ACARe-4B (M2)	**86.2** (±0.8)†	**88.7** (±0.8)†	**17.5** (±0.5)†	**38.3** (±2.0)†	**81.2** (±4.3)†
(c1) Gemini 3.5 Flash-Lite (small)	58.5 (±1.4)	59.9 (±1.4)	6.4 (±0.1)	3.8 (±0.1)	27.5 (±1.0)
(c2) + Qwen3-Reranker-4B	64.9 (±1.3)	66.6 (±0.7)	**5.2** (±0.1)	**3.1** (±0.1)	**22.7** (±0.4)
(c3) + ACARe-4B (M2)	**68.7** (±1.5)$^{\dagger *}$	**71.0** (±1.4)$^{\dagger *}$	5.3 (±0.1)†	3.2 (±0.1)†	23.6 (±0.5)†
(d1) Gemini 3.6 Flash (large)	72.3 (±0.6)	73.7 (±0.6)	8.1 (±0.1)	13.7 (±0.4)	100.0 (±3.2)
(d2) + Qwen3-Reranker-4B	78.3 (±0.9)	80.4 (±0.9)	7.0 (±0.2)	12.2 (±0.4)	89.1 (±3.3)
(d3) + ACARe-4B (M2)	**82.1** (±0.5)$^{\dagger *}$	**84.6** (±0.5)$^{\dagger *}$	**6.7** (±0.1)$^{\dagger *}$	**11.7** (±0.6)†	**85.7** (±4.6)†
(e1) Claude Haiku 4.5 (small)	54.0 (±2.0)	57.1 (±2.1)	15.3 (±0.6)	18.0 (±0.8)	28.2 (±1.2)
(e2) + Qwen3-Reranker-4B	60.4 (±2.1)	63.5 (±2.1)	14.3 (±0.6)	17.2 (±0.7)	27.0 (±1.1)
(e3) + ACARe-4B (M2)	**66.6** (±2.2)$^{\dagger *}$	**70.5** (±2.2)$^{\dagger *}$	**14.0** (±0.4)†	**17.1** (±0.6)	**26.8** (±1.0)
(f1) Claude Sonnet 5 (large)	78.7 (±0.6)	82.1 (±0.5)	15.9 (±0.6)	63.8 (±3.3)	100.0 (±5.1)
(f2) + Qwen3-Reranker-4B	80.2 (±0.9)	83.1 (±0.9)	12.7 (±0.2)	48.9 (±0.8)	76.6 (±1.3)
(f3) + ACARe-4B (M2)	**82.2** (±0.6)$^{\dagger *}$	**85.2** (±0.6)$^{\dagger *}$	**11.8** (±0.4)$^{\dagger *}$	**45.6** (±2.2)†	**71.4** (±3.4)†

Using the larger agent with ACARe as the reference replaces L_0 with L_{ACARe}, its mean nDCG@10 with ACARe. Against this fixed target, adding ACARe to the smaller agent closes 44%, 43%, and 45% of its initial gap. With ACARe on both sides, the smaller agents' nDCG@10 scores are 4.6, 13.4, and 15.5 pp lower, at 8.1%, 27.6%, and 37.5% of their larger counterparts' API costs. Compared with neither agent using reranking, the gaps shrink by 8%, 3%, and 37%, respectively. These comparisons are descriptive.

Every agent has lower mean API cost with reranking than without it. With ACARe, the larger agents' relative costs are 81.2%, 85.7%, and 71.4% for Terra, Flash, and Sonnet; the smaller agents' costs are 6.6%, 23.6%, and 26.8%. The reduction from Haiku's no-reranking cost is not significant ($p = 0.388$). No API cost difference between the two rerankers remains significant after correction.

The larger agent with ACARe remains the most effective configuration in each family, with nDCG@10 of 86.2, 82.1, and 82.2. The small-agent results therefore describe a lower-cost alternative with some loss of quality, under the cost accounting in Appendix E.

H Retriever headroom on BRIGHT

The main BRIGHT experiments show small nDCG@10 differences between ACARe and Qwen3-Reranker-4B (Table 17). Nemotron-3-Embed-8B's training includes synthetic query–document pairs generated using BRIGHT as a seed corpus.[8] This corpus-specific supervision may reduce the room for further gains from agentic search and reranking. The BRIGHT results with this retriever therefore involve benchmark-seeded training data; the comparison below tests a different retriever and training recipe. We examine this possibility by replacing Nemotron-3-Embed-8B with the general-purpose Qwen3-Embedding-4B (Zhang et al., 2025) for the Nemotron-3 Ultra agent. The agent, question sample, five run indices, search interface, and reranking depth remain fixed. Both use BRIGHT; the Nemotron rows repeat the main results.

[8] `https://huggingface.co/nvidia/Nemotron-3-Embed-8B-BF16`

Table 16: **Retriever headroom on BRIGHT** with Nemotron-3 Ultra: Nemotron-3-Embed-8B, used in the main experiments, versus Qwen3-Embedding-4B.

Retriever	nDCG@10↑	Recall@10↑	Avg. per query stats		
+Reranker			Searches↓	LLM turns↓	Cost (¢)↓
(a1) Nemotron-3-Embed-8B	41.1 (±0.4)	45.4 (±0.4)	8.1 (±0.1)	9.1 (±0.1)	9.2 (±0.5)
(a2) + Qwen3-Reranker-4B	42.5 (±0.6)	**47.1** (±0.6)	7.5 (±0.0)	8.5 (±0.0)	**8.7** (±0.5)
(a3) + ACARe-4B (M3)	**42.7** (±0.3)	**47.1** (±0.6)	**7.1** (±0.0)*	**8.1** (±0.0)*	**8.7** (±0.5)
(b1) Qwen3-Embedding-4B	40.1 (±0.4)	44.2 (±0.6)	8.3 (±0.1)	9.3 (±0.1)	10.7 (±0.1)
(b2) + Qwen3-Reranker-4B	41.6 (±0.3)	45.9 (±0.5)	7.6 (±0.1)	8.6 (±0.1)	**10.0** (±0.4)
(b3) + ACARe-4B (M3)	**42.5** (±0.3)*	**47.0** (±0.4)*	**7.2** (±0.1)*	**8.2** (±0.1)*	10.3 (±0.4)

Without reranking, agentic search improves nDCG@10 over one-shot retrieval by 15.8 pp with Nemotron-3-Embed-8B and 18.4 pp with Qwen3-Embedding-4B (Tables 1 and 16). Thus, agentic search provides substantial gains with both retrievers, with a larger gain for Qwen3-Embedding-4B.

With Nemotron-3-Embed-8B, ACARe exceeds Qwen3-Reranker-4B by 0.2 nDCG@10 pp ($p = 1.000$), and both have rounded Recall@10 of 47.1 ($p = 1.000$). With Qwen3-Embedding-4B, the differences increase to 0.9 nDCG@10 pp ($p = 0.022$) and 1.1 Recall@10 pp ($p = 0.033$). The gain over no reranking also increases, from 1.6 to 2.4 nDCG@10 pp.

Switching to Qwen3-Embedding-4B lowers ACARe's mean nDCG@10 from 42.7 to 42.5 ($p = 1.000$), compared with reductions of 0.9 pp for Qwen3-Reranker-4B ($p = 0.122$) and 1.0 without reranking ($p = 0.033$). For the base reranker, Recall@10 also falls by 1.2 pp ($p = 0.044$). These observed differences suggest that ACARe is less sensitive to the retriever change in this setting. The larger gains with Qwen3-Embedding-4B are consistent with the headroom explanation, but this comparison does not isolate the effect of training data or establish a significant retriever–reranker interaction.

ACARe uses approximately 0.4 fewer searches per query than Qwen3-Reranker-4B with either retriever ($p < 0.001$). Cost differences between the rerankers are not significant in either configuration (both adjusted $p = 1.000$).

I AGGREGATION AND STATISTICAL TESTS

In the retrieval evaluations, BCP quality metrics average the gold and evidence label sets; BRIGHT and TRQA quality metrics are macro-averages over twelve and three subsets, respectively. Search calls, LLM turns, and API costs are pooled over the questions within each benchmark. Overall results average the three benchmark-level values with equal weight. Tables report means and sample standard deviations over five runs (six for Harness B); figures use the same means and standard deviations.

We compare configurations with two-sided Welch t-tests on their run-level values and apply Benjamini–Yekutieli (BY) correction (Benjamini & Yekutieli, 2001) within each comparison family. BY controls the false discovery rate, the expected fraction of false rejections among rejected hypotheses, under arbitrary dependence between tests, provided the individual p-values are valid. All reported p-values are adjusted, with significance defined as $p < 0.05$. Welch tests allow unequal variances but assume independent runs and approximately normal run-level means. With only five or six runs, these assumptions are difficult to assess. The tests assess whether differences are consistent across repeated executions of the same questions. They do not assess generalization to other questions. A nonsignificant result means that the repeated runs provide insufficient evidence of a difference, not that the systems perform equally.

Main comparisons. The aggregate family contains 75 tests: all three pairwise comparisons among no reranking, Qwen3-Reranker-4B, and ACARe, for five agents and five metrics (nDCG@10, Recall@10, searches, LLM turns, and API cost). The benchmark family contains the corresponding 225 tests over three benchmarks. Thus claims based on either quality metric account for testing both metrics. False discovery rate control applies within each specified family, not across the entire

paper. A comparison reused in multiple studies can therefore receive different adjusted p-values and significance marks, even when its run-level values are identical.

Table 17 reports ACARe minus Qwen3-Reranker-4B and the adjusted p-values. Quality differences are pp on the 0–100 scale; cost differences are cents per query. Negative differences indicate fewer calls or lower cost. The same $*$ symbol marks significant differences in this table; negative cost differences favor ACARe. Bold is reserved for best means in result tables.

All five agents retain significant aggregate gains in both quality metrics, fewer searches, and fewer LLM turns. Aggregate cost is significantly lower only for Nemotron-3 Ultra. In the benchmark breakdown, nDCG@10 gains remain significant for Ultra and Flash on BCP and for all five agents on TRQA. Recall@10 gains remain significant for Flash and Sonnet on BCP and for Ultra, M3, Flash, and Sonnet on TRQA.

Analysis comparisons. The latency and reranking-depth study (Appendix F), small-agent study (Appendix G), and retriever-headroom study (Appendix H) each have a separate correction family. The latency and depth study includes 36 tests: all three pairwise comparisons among the no-reranking baseline and two rerankers at each of two depths on six outcomes in Table 14: nDCG@10, Recall@10, searches, LLM turns, API cost, and end-to-end latency. The size study includes 51 tests: ACARe against each baseline for six agents on nDCG@10, Recall@10, searches, and cost (48 tests), plus the three nDCG@10 comparisons between a smaller agent with ACARe and its larger counterpart without reranking. The retriever-headroom study includes 16 tests: the two rerankers on five metrics under each retriever (ten tests), plus both quality metrics across retrievers for each of the three reranking configurations (six tests). The context study (Section 6.2) has 45 tests: each of four ACARe modes against both baselines, plus Qwen3-Reranker-4B against no reranking, on five metrics. Comparisons between ACARe context modes remain descriptive. One-shot results are descriptive (Appendix C).

Harness B ClimbMix comparisons. We compare ACARe with both baselines on answer accuracy, Recall@all, Precision@all, and searches for both agents, forming one pooled 16-test BY correction family (Section 6.4). Each metric is averaged over all 57 questions before testing across six runs. Every such comparison is retained regardless of the observed difference. Comparisons among baselines remain descriptive; monetary cost is excluded from these tests.

Base-reranker context control. The separate Harness B BCP study (Appendix J.2) has its own 16-test BY family: ACARe against no reranking and three Qwen3-Reranker-4B configurations (subquery only, original question added, and original question plus reasoning added), on answer accuracy, Recall@all, Precision@all, and searches. Recall and precision each average gold and evidence qrel scores equally. Metrics average all 200 questions within each of six runs. All sixteen comparisons are retained. Comparisons among the base-reranker configurations are descriptive.

Table 17: **ACARe-4B versus Qwen3-Reranker-4B.** Cells show the mean difference followed by the adjusted p-value; quality differences are pp, and cost differences are cents per query.

Agent LLM, ACARe mode	BCP	BRIGHT	TRQA	Overall
Δ *nDCG@10 (points)*				
Nemotron-3 Ultra (M3)	+2.9 (0.035)*	+0.2 (1.000)	+2.5 (0.001)*	+1.9 (0.005)*
MiniMax M3 (M3)	+2.1 (0.430)	+0.2 (1.000)	+3.4 (<0.001)*	+1.9 (0.006)*
Gemini 3.6 Flash (M2)	+3.8 (0.002)*	+0.1 (1.000)	+2.1 (0.003)*	+2.0 (<0.001)*
GPT-5.6 Terra (M2)	+1.0 (0.546)	+0.1 (1.000)	+1.4 (0.027)*	+0.9 (0.022)*
Claude Sonnet 5 (M2)	+2.0 (0.052)	+0.7 (0.155)	+2.1 (0.006)*	+1.6 (<0.001)*
Δ *Recall@10 (points)*				
Nemotron-3 Ultra (M3)	+3.0 (0.061)	−0.0 (1.000)	+1.3 (0.012)*	+1.4 (0.023)*
MiniMax M3 (M3)	+2.0 (0.391)	+0.2 (1.000)	+1.7 (0.006)*	+1.3 (0.037)*
Gemini 3.6 Flash (M2)	+4.3 (0.001)*	+0.7 (0.082)	+1.0 (0.019)*	+2.0 (<0.001)*
GPT-5.6 Terra (M2)	+1.3 (0.311)	+0.6 (0.367)	+0.8 (0.082)	+0.9 (0.009)*
Claude Sonnet 5 (M2)	+2.1 (0.032)*	+1.0 (0.077)	+0.9 (0.033)*	+1.3 (<0.001)*
Δ *search calls (per query)*				
Nemotron-3 Ultra (M3)	−1.5 (0.002)*	−0.4 (<0.001)*	−1.2 (<0.001)*	−1.0 (<0.001)*
MiniMax M3 (M3)	−0.7 (0.006)*	−0.5 (0.019)*	−1.2 (0.014)*	−0.8 (<0.001)*
Gemini 3.6 Flash (M2)	−0.4 (0.039)*	−0.3 (<0.001)*	−0.3 (0.501)	−0.3 (0.002)*
GPT-5.6 Terra (M2)	−0.9 (0.077)	−0.4 (<0.001)*	+0.2 (0.933)	−0.3 (0.006)*
Claude Sonnet 5 (M2)	−0.9 (0.025)*	−0.6 (<0.001)*	−0.5 (0.200)	−0.7 (0.002)*
Δ *LLM turns (per query)*				
Nemotron-3 Ultra (M3)	−1.5 (0.001)*	−0.4 (<0.001)*	−1.2 (<0.001)*	−1.0 (<0.001)*
MiniMax M3 (M3)	−0.2 (0.019)*	−0.2 (0.077)	−0.5 (0.015)*	−0.3 (<0.001)*
Gemini 3.6 Flash (M2)	−0.4 (0.044)*	−0.3 (<0.001)*	−0.4 (0.455)	−0.3 (0.001)*
GPT-5.6 Terra (M2)	−0.1 (0.221)	−0.1 (<0.001)*	+0.0 (1.000)	−0.1 (0.011)*
Claude Sonnet 5 (M2)	−0.4 (0.068)	−0.3 (<0.001)*	−0.1 (0.969)	−0.3 (0.014)*
Δ *cost (¢/query)*				
Nemotron-3 Ultra (M3)	−4.87 (0.006)*	+0.02 (1.000)	−1.35 (0.068)	−2.07 (0.004)*
MiniMax M3 (M3)	−0.28 (1.000)	+0.19 (0.969)	−0.12 (1.000)	−0.07 (1.000)
Gemini 3.6 Flash (M2)	−0.46 (1.000)	+0.18 (1.000)	+1.33 (0.241)	+0.35 (1.000)
GPT-5.6 Terra (M2)	−1.20 (1.000)	+0.67 (<0.001)*	+1.22 (0.005)*	+0.23 (1.000)
Claude Sonnet 5 (M2)	−3.30 (0.196)	−0.13 (1.000)	+0.30 (1.000)	−1.04 (0.210)

J HARNESS B: END-TO-END ANSWER GENERATION

J.1 TRANSFER TO CLIMBMIX

The ClimbMix study in Section 6.4 evaluates GPT-5.6 Luna and Nemotron-3.5 Lightning on all 57 BrowseComp-Plus$_{\text{CM}}$ questions over ClimbMix's 553 million documents (Sharifymoghaddam et al., 2026). Harness B exposes `search` and `read_document` tools through a remote Pyserini BM25 REST API (Lin et al., 2021). Search returns five excerpts capped at 2,000 characters (`max_doc_length`); reranking selects these from the top 100 candidates, while document reads request full text. Both agents use high reasoning effort and a two-hour query timeout. Each agent runs six times with each of three retrieval configurations: no reranking, Qwen3-Reranker-4B, and ACARe-4B.

Luna exposes no reasoning trace, so ACARe uses $\mathrm{M2}$. Lightning exposes its trace, enabling ACARe's $\mathrm{M3}$. Lightning has about 30B total parameters and 3B active parameters.[9] Qwen3-Reranker-4B uses only the sub-query with each candidate.

Before reranking, candidate excerpts are further capped at 300 words for both ACARe and Qwen3-Reranker-4B. This limit affects reranker inputs, not the excerpts returned to the agent.

The gold-answer judge is GPT-5.5 with low reasoning effort, accessed directly through OpenAI. All 57 questions contribute to every run's metrics, including 19 Lightning executions that exhausted the 32K-token output budget during reasoning without producing a final answer; these count as incorrect while retaining their retrieval and efficiency measurements. The 36 runs contain 2,052 question executions, with no recorded reranker failures.

For each question, let S be the distinct documents surfaced throughout the trajectory and R the positively labeled documents. Recall@all is $|S \cap R|/|R|$; Precision@all is $|S \cap R|/|S|$, or zero if S is empty. Both deduplicate documents across the trajectory and are macro-averaged over questions within each run. Unlabeled documents count as nonrelevant. Lightning runs switched between API providers and local serving under heavy throttling. For Luna, each reranking configuration ran on a separate vLLM server under a different load; the resulting queuing delays changed agent-side prompt-cache reuse, so billed cost depended on serving conditions as well as trajectories. We therefore compare search counts rather than monetary costs. Token-priced usage is retained only for accounting in Appendix E; local-inference estimates do not measure serving expenditure. Table 4 reports six-run means and sample standard deviations. Table 18 compares ACARe with each baseline on answer accuracy, Recall@all, Precision@all, and searches, using two-sided run-level Welch tests and one pooled 16-test BY correction family across both agents (Appendix I); bold identifies adjusted $p < 0.05$.

Table 18: **Harness B: adjusted p-values for ACARe versus each baseline.** BM25 API retrieval on BrowseComp-Plus$_{\text{CM}}$.

Comparison	Accuracy	Recall@all	Precision@all	Searches
GPT-5.6 Luna, ACARe-4B ($\mathrm{M2}$*)*				
vs. No reranker	**0.007**	0.847	**0.019**	**0.014**
vs. Qwen3-Reranker-4B	1.000	0.847	**0.007**	0.635
Nemotron-3.5 Lightning, ACARe-4B ($\mathrm{M3}$*)*				
vs. No reranker	**0.016**	0.401	**0.024**	1.000
vs. Qwen3-Reranker-4B	1.000	0.635	0.098	1.000

[9] `huggingface.co/nvidia/NVIDIA-Nemotron-3.5-Lightning-30B-A3B-BF16`

J.2 BASE-RERANKER CONTEXT CONTROL ON BCP

We test the effect of context on the unchanged Qwen3-Reranker-4B. Nemotron-3.5 Lightning runs in Harness B on 200 BCP questions, sampled with seed 42 and stratified by one-shot BM25 recall and evidence-document count. This sample differs from Harness A's. We will release the query IDs and sampling code. Six runs per configuration yield 30 runs and 6,000 executions.

Table 19: **BCP context control** with Nemotron-3.5 Lightning and BM25 in Harness B. $*$: significant difference from sub-query-only Qwen3-Reranker-4B.

Reranker	Input	Accuracy ↑	Recall@all ↑	Precision@all ↑	Searches ↓
No reranker	–	42.2 (±1.6)	44.3 (±1.3)	11.8 (±0.5)	10.8 (±0.8)
Qwen3-Reranker-4B	q_t (M0)	48.2 (±2.3)	54.2 (±1.4)	14.4 (±0.6)	10.1 (±0.6)
Qwen3-Reranker-4B	q_t, Q (M2)	54.8 (±2.3)	61.2 (±1.2)	19.2 (±0.6)	9.3 (±0.5)
Qwen3-Reranker-4B	q_t, τ_t, Q (M3)	55.9 (±0.7)	63.9 (±0.9)	20.1 (±0.8)	8.7 (±0.4)
ACARe-4B (M3)	q_t, τ_t, Q (M3)	**57.6** (±1.8)$^{\dagger *}$	**65.1** (±1.4)$^{\dagger *}$	**21.2** (±0.5)$^{\dagger *}$	**8.4** (±0.3)$^{\dagger *}$

The configurations are no reranking, Qwen3-Reranker-4B in M0, M2, and M3, and ACARe in M3. The contextual Qwen3 arms use ACARe's prompts with unchanged weights; both M3 arms therefore use identical context fields and prompts. Local BM25 ($k_1 = 16$, $b = 1$) searches the BCP corpus, returning five of 100 candidates. Document prefixes are capped at 512 tokens for search previews, reads, and reranking; reranker preprocessing also applies the shared 300-word cap. Lightning uses high reasoning effort with a 32K-token completion budget and a two-hour timeout.

We use the gold-answer judge and metric definitions in Appendix J.1. Recall@all and Precision@all each average gold and evidence qrel scores equally per question. Table 19 reports means and sample standard deviations on the shared 0–100 quality scale. Table 20 reports all 16 BY-adjusted, two-sided Welch tests of ACARe against four alternatives on four metrics, using six run-level means. Baseline comparisons remain descriptive. Monetary cost is excluded because runs switched between API providers and local serving; retained agent and judge usage is included in Appendix E.

All 200 questions remain in every metric. Of 231 unsuccessful executions, 174 exhaust the 32K output budget, 48 stop without a final answer, seven time out, and two encounter context-length API errors. These receive zero accuracy while retaining retrieval and efficiency measurements. Counts are 37, 61, 58, 51, and 24 for no reranking, Qwen3 M0, M2, M3, and ACARe M3, respectively. Timeout counts are 1, 3, 1, 1, and 1 in that order; the API errors affect only Qwen3 M2.

All four means improve from Qwen3 M0 to M2, M3, and finally ACARe M3. Accuracy rises from 48.2% to 54.8%, 55.9%, and 57.6%; recall from 54.2% to 61.2%, 63.9%, and 65.1%; and precision from 14.4% to 19.2%, 20.1%, and 21.2%, while searches fall from 10.1 to 9.3, 8.7, and 8.4. ACARe significantly improves all four metrics over Qwen3 M0 ($p \leq 0.002$), and recall, precision, and searches over M2. Against Qwen3 M3, it gains 1.7 pp accuracy, 1.2 pp recall, and 1.0 pp precision, with 0.2 fewer searches; these differences are not significant after correction ($p = 0.277$, 0.415, 0.135, and 0.890). Thus ACARe's input formulation benefits the unchanged base, while the trained checkpoint achieves the best means on all four metrics.

Table 20: **BCP context control: adjusted p-values** with Nemotron-3.5 Lightning and BM25 in Harness B. Bold: adjusted $p < 0.05$.

ACARe-4B (M3) vs.	Input	Accuracy	Recall@all	Precision@all	Searches
No reranker	–	**<0.001**	**<0.001**	**<0.001**	**0.002**
Qwen3-Reranker-4B	q_t (M0)	**<0.001**	**<0.001**	**<0.001**	**0.002**
Qwen3-Reranker-4B	q_t, Q (M2)	0.155	**0.002**	**0.001**	**0.049**
Qwen3-Reranker-4B	q_t, τ_t, Q (M3)	0.277	0.415	0.135	0.890

J.3 SHARED AGENT AND JUDGE PROMPTS

Figures 11 and 12 reproduce the shared agent-loop user prompt and gold-answer judge prompt. The Harness B judge differs from the rollout judge in Figure 7: it requires JSON output, handles acceptable answer alternatives and missing final answers, and does not instruct the judge to extract confidence. Answer accuracy uses its Boolean `correct` field; missing final answers receive zero.

You are a research and retrieval agent using only the provided tools.

Workflow:
1. Use search with a concise raw query string based on the original question.
2. Prefer short lexical searches over long natural-language rewrites.
3. Inspect the ranked hits returned directly by search before rewriting the query.
4. If a promising candidate document appears, inspect it with read_document.
5. Follow the read_document schema and any continuation guidance returned by the tool.
6. Use search refinements only when they add a genuinely new clue from what you already saw.
7. Every call to search and read_document must include reason as the first argument. Keep it specific, under 100 words, and focused on the clue, gap, candidate, or ranking issue.
8. Satisfy every requested output below; use the same research pass for all outputs.
9. As soon as you have enough evidence and candidates, stop using tools and answer in plain assistant text.

Answer output:
- Produce a concise answer supported by the documents you found.
- Cite supporting docids inline when possible.
- Keep Exact Answer directly responsive to the question.

Your final response must contain exactly these sections in this order:
Answer:
Explanation: {your explanation for your final answer. Cite supporting docids inline in square brackets [] at the end of sentences when possible, for example [123].}
Exact Answer: {your succinct, final answer}
Confidence: {your confidence score between 0% and 100%}

Do not include any text before, after, or between those sections beyond the requested fields and ranked document lines.

If you later receive a user steer telling you to submit now, stop using tools immediately and answer right away with the exact final response format below. Do not do more research after that steer.

Question: {question}

Figure 11: Harness B agent-loop user prompt for the answer-generation experiments, with the benchmark question replaced by {question}. All retrieval configurations share this two-tool workflow.

You are an evaluation judge.

Your job is to determine whether the response's final answer is semantically equivalent to the known correct answer.
Do not solve the question yourself.
Do not use outside knowledge.
Focus only on whether the response's final answer matches the correct answer.
Allow harmless wording differences, equivalent formatting, and added correct detail.
The correct-answer field may be a JSON array of acceptable alternatives; matching any one alternative is correct.
For numerical answers, allow small formatting differences and obvious equivalent forms.
If the response does not contain a final answer you can extract, set extracted_final_answer to null and correct to false.

Return exactly one JSON object and nothing else.
Do not wrap the JSON in markdown or code fences.
Use this exact schema:

```
{
"extracted_final_answer": string | null,
"correct_answer": string,
"reasoning": string,
"correct": boolean,
"confidence": number
}
```

Requirements:
- confidence must be a number between 0 and 100
- correct must be true or false
- repeat the provided correct answer exactly in correct_answer
- reasoning must explain only whether the extracted final answer matches the correct answer

Question: {question}

Response:
{response}

Correct answer: {correct_answer}

Figure 12: Harness B reference-based answer-judge prompt, used with GPT-5.5 at low reasoning effort. Braced input placeholders replace the question, final agent response, and reference answer.