FROM DELIVERY TO STATEFUL EXPLORATION: RETHINKING THE INDEX FOR AGENTIC SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in agentic search have given large language model (LLM) agents finer control over corpus exploration. However, search interfaces often return matching passages even when feedback about the candidate set would suffice for the next decision, coupling candidate refinement with source-text exposure. We propose INDEXACT, an interface for Index-Native Corpus Interaction that separates candidate-set refinement from text inspection. Agents construct and manipulate persistent candidate sets through lexical conditions and set operations over an inverted index, receiving reusable state references and statistics such as candidate counts rather than matching passages. This feedback guides further refinement, while separately requested passages provide new clues or evidence that can inform subsequent operations on retained candidate sets. Experiments on five benchmarks spanning agentic search and multi-hop question answering show that INDEXACT outperforms the evaluated baselines on each benchmark. On BrowseComp-Plus, it also achieves higher evidence coverage with a smaller average live context than terminal-based corpus interfaces, and maintains answer accuracy as the corpus expands. Further analyses suggest that informative refinement feedback and state reuse support continued evidence discovery, while shorter contexts or fewer search steps alone do not ensure better performance. [CODE].

1 INTRODUCTION

Search enables large language model (LLM) agents to explore external corpora by submitting queries, inspecting results, and deciding what to do next (Yao et al., 2023; Jin et al., 2025). Conventional retrieval systems make this process efficient by using an index to retrieve and rank candidate documents (Lewis et al., 2020; Karpukhin et al., 2020) and returning selected results as text. An agent can use these results to gather evidence, discover new terms, and revise its queries (Trivedi et al., 2023; Shao et al., 2023). The search interface therefore shapes not only which documents the agent encounters, but also what information enters its context as it decides how to continue.

Recent work has expanded agents' control over how searches are constructed and revised. Direct Corpus Interaction (DCI) (Li et al., 2026b) gives agents direct access to raw corpora and general-purpose tools, demonstrating the value of applying exact constraints and inspecting local context during search and verification. Such control makes observations more important, as each search outcome informs the agent's next action. This raises a complementary question: what feedback helps agents decide their next action, and when does that decision require reading document text?

During exploration, an agent may need to assess the effect of a search condition without yet reading the matching content. For example, when searching for reports about an event, the agent may first want to know how much adding a location narrows the candidate set, or whether it leaves no matches. Statistical information such as candidate counts (Tanin et al., 2000) can inform whether to retain or revise that condition (Yang et al., 2026) without requiring the agent to inspect the reports. When a search interface returns matching text at each step, however, the agent receives content even when feedback about the candidate set would suffice for the immediate decision. As such passages accumulate, they occupy the agent's context even when the current search decision depends primarily on information about the candidate set. Extending the index beyond result selection to expose candidate-set feedback without automatically returning matching text would let agents refine candidate sets and request relevant passages when needed to discover new clues or verify evidence.

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.

We introduce INDEXACT, an interface for Index-Native Corpus Interaction that separates candidate-set refinement from text inspection. Agents construct and refine candidate sets by applying lexical conditions and set operations over the index. Each new set is returned as a reusable reference with statistics such as its size, rather than matching text. With the returned reference and statistics, the agent can refine the set further, combine it with a retained set, or request relevant passages. Retaining intermediate sets also allows the agent to revisit an earlier search state and explore another direction.

INDEXACT keeps intermediate candidate sets available for further exploration without automatically exposing their contents, letting agents choose what information to inspect at each step. Agents can use index feedback to assess the effect of a search condition and inspect local context when they need new clues or supporting evidence. Reading and refinement can alternate throughout exploration: information learned from a passage can guide subsequent operations on retained candidate sets. By connecting reusable search states, index-based feedback, and agent-directed reading, INDEXACT gives agents control over both how their search develops and when source text enters their context.

We evaluate INDEXACT on five benchmarks spanning agentic search and multi-hop question answering. INDEXACT achieves the highest task performance on all five, and our agentic-search evaluation shows that it achieves higher evidence coverage with less live context than terminal-based corpus interfaces. Our analyses suggest that effective context use depends on supporting continued exploration, rather than simply minimizing context length or shortening search trajectories. Making the effects of search conditions on candidate sets visible and keeping intermediate candidates available for reuse help agents revise their searches, combine earlier results, and seek further evidence. This support allows candidate exploration to continue without automatically introducing matching passages at each refinement step. Together, these findings demonstrate that index-native corpus interaction can support effective evidence discovery while keeping source-text exposure selective.

The main contributions of our work are summarized as follows:

- We analyze what enters agent context and when source text enters during evidence discovery, motivating interfaces that support candidate exploration without automatic source-text delivery.
- We propose INDEXACT, an interface for Index-Native Corpus Interaction that lets agents construct, refine, and reuse candidate states through index-based operations, observe their effects through statistical feedback, and inspect source text when needed.
- We demonstrate the effectiveness of INDEXACT across agentic search and multi-hop question answering, and show that informative refinement feedback and reusable candidate states support continued evidence discovery while keeping source-text exposure selective.

2 RELATED WORK

Agentic and Long-Horizon Search. Agentic search extends retrieval into multi-turn interaction by interleaving reasoning and retrieval (Yao et al., 2023; Trivedi et al., 2023). Recent work develops learned multi-turn search policies (Song et al., 2025; Jin et al., 2025; Sun et al., 2026) and longer investigations that combine repeated search, evidence inspection, and synthesis (Li et al., 2025). Retrieved information therefore serves as evidence for answering and as an observation guiding subsequent actions. As search trajectories lengthen, context-management methods reorganize working memory or prune stale observations to manage accumulating observations and intermediate reasoning (Lu et al., 2026b; Zhang et al., 2026). Complementing work on managing accumulated interaction histories, we study what information corpus-access operations return to agents as observations.

Retrieval Interfaces for Agentic Search. Corpus interfaces determine the form and granularity of information entering agent context. Recent interfaces give agents finer control over retrieval and evidence access: Interact-RAG (Hui et al., 2026) exposes retrieval strategies and scope, PI-SERINI (Hsu et al., 2026) separates cached-ranking browsing from document reading, and SIEVE (Wang et al., 2026) combines Boolean candidate selection, result-card inspection, and selective section fetching. PI-SERINI and SIEVE therefore avoid automatically exposing full documents, using excerpts or result cards to support decisions about whether and where additional source text should be read. INDEXACT extends this separation to the feedback returned by candidate-set refinement, allowing agents to assess refinement results and operate on retained candidate sets without automatically receiving matching passages.

Direct Corpus Interaction. DCI exposes raw corpora through a general-purpose terminal instead of a retriever, enabling composable exploration and localized evidence inspection (Li et al., 2026b). However, full-corpus interaction becomes increasingly costly and noisy as corpora grow. RISE and DR-DCI use retrieval to maintain bounded document workspaces for local interaction (Zhuang et al., 2026; Lu et al., 2026a), while RARG (Li et al., 2026a) carries relevance into corpus interaction by prioritizing traversal and matched evidence; these approaches keep retrieved candidates available outside the model context but conduct subsequent fine-grained exploration over materialized documents or file scopes. INDEXACT retains fine-grained evidence access while making candidate sets persistent operands for index-based refinement and composition: these operations return reusable references and statistics rather than source text, which agents inspect separately when needed.

3 PRELIMINARY ANALYSIS: CONTEXT USE DURING EVIDENCE DISCOVERY

Before introducing INDEXACT, we first examine how existing search agents use their context while searching for evidence. Our analysis addresses two questions:

- **Question I:** What constitutes the context introduced during agentic search?
- **Question II:** When is non-evidence source text introduced during evidence discovery?

Agents and Evaluation Setting. We analyze context use under two representative corpus-access paradigms on BrowseComp-Plus (BC+) (Chen et al., 2025), a fixed-corpus benchmark for iterative search and evidence gathering. Retrieval Agent (BM25), a ReAct-style agent (Yao et al., 2023) based on the official BC+ implementation, uses ranked BM25 results. DCI-Agent-Lite (Li et al., 2026b) replaces the retriever with general-purpose commands such as `grep` and `read` for fine-grained corpus access. We evaluate both agents on the same fixed 100-question subset using GPT-5.4-nano (OpenAI, 2026) with high reasoning effort. Measurement details appear in Appendix C.2.

3.1 ANALYSIS I: WHAT CONSTITUTES THE AGENT'S CONTEXT?

Setup. We analyze text retained in each trajectory's final logged model request, excluding the original question, system instructions, and tool specifications. We group source text by provenance: gold documents, other annotated evidence documents, and documents outside the annotated evidence set. *Other* includes all remaining interaction text, such as agent-generated messages and non-source content returned by tools. For each agent, we report the average token composition across trajectories.

Corpus observations constitute a substantial share of the retained context. Figure 1 shows the final retained context composition for Retrieval Agent (BM25) and DCI-Agent-Lite. Source-text observations account for 85.0% and 62.3% of the retained context, respectively. Notably, text from sources outside the annotated evidence set alone accounts for 74.4% of the context in Retrieval Agent (BM25) and 52.0% in DCI-Agent-Lite. The composition alone does not establish that these observations are unnecessary: text from unannotated sources may reveal new entities, help reject candidates, or provide clues for subsequent searches. We next examine when this text is observed during evidence discovery.

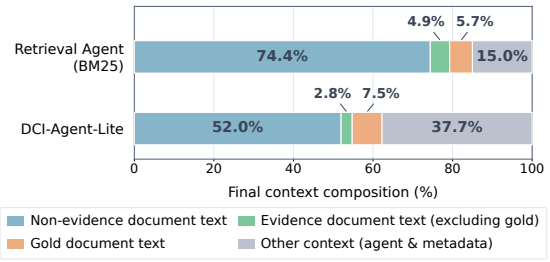

Figure 1: **Final context composition.** Mean per-trajectory token shares for both agents.

3.2 ANALYSIS II: WHEN IS NON-EVIDENCE TEXT INTRODUCED?

Setup. Using the same runs, we focus on trajectories that eventually reach an annotated evidence source. For each trajectory, we identify the first model request that includes source text from an annotated evidence document. We then compare the amount of deduplicated non-evidence source text observed up to and including this request with that observed in subsequent requests. For each trajectory, we compute the corresponding shares and report their average for each agent.

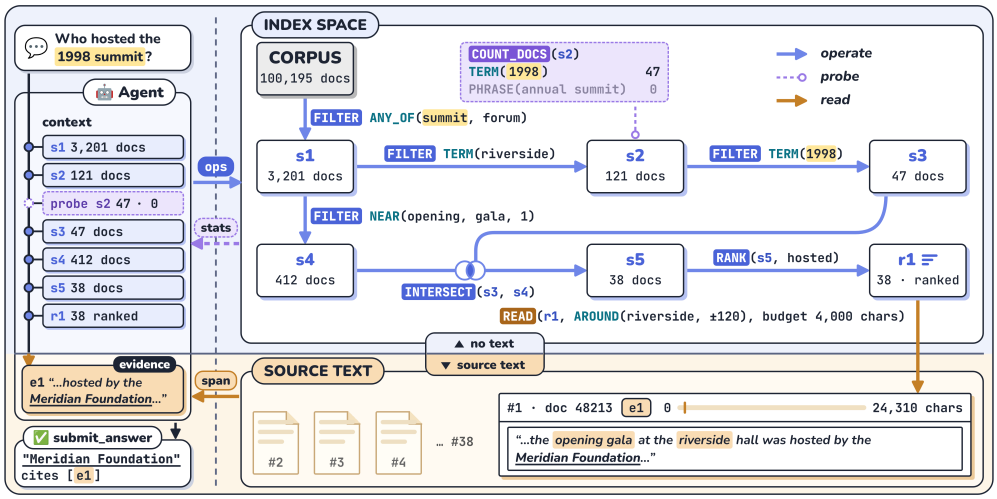

Figure 3: Overview of INDEXACT. Candidate exploration is performed over reusable states in the index, where operations return state references and statistical feedback without source text. The agent explicitly reads selected source-text regions when it needs new clues or evidence verification.

Non-evidence exposure around evidence discovery. DCI-Agent-Lite and Retrieval Agent (BM25) observe, on average, 68.58% and 42.07% of their non-evidence source text, respectively, up to and including the first model request containing source text from an annotated evidence document (Figure 2). This contrast may reflect how the two access paradigms support continued search. Retrieval Agent (BM25) can use discovered clues to refine subsequent queries, but each search still exposes retriever-selected text, which may include non-evidence sources. DCI-Agent-Lite can instead use discovered clues to guide targeted searches and local inspection. Its smaller post-evidence share of non-evidence text is consistent with this fine-grained mode of corpus interaction. The same pattern also points to initial evidence discovery as an important stage for managing source-text exposure, even with fine-grained corpus access.

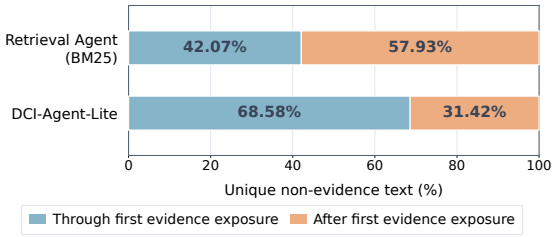

Figure 2: **When non-evidence source text is observed.** Mean per-trajectory token shares.

3.3 INSIGHTS FROM PRELIMINARY ANALYSIS

Our analyses examine *what constitutes the agent's context* and *when source text is observed during search*. In particular, a substantial share of non-evidence source text is observed while searching for the first evidence source. Corpus-access design should therefore consider not only how to provide information effectively, but also how to support the search for evidence.

Rather than reducing exploration or restricting necessary reading, we focus on using context effectively to support exploration and the use of evidence. To this end, we separate decisions about candidate exploration and refinement from decisions about introducing source text into context. Candidates remain available in the index for exploration, while source text is read selectively to discover new clues or verify evidence. We aim to reduce the source-text exposure incurred when testing search hypotheses, leaving more context available for necessary reading and further exploration.

4 INDEXACT

We introduce INDEXACT, an interface for Index-Native Corpus Interaction that provides agents with reusable candidate states and operations to manipulate and inspect them. Agents construct candidates through index-based search and set operations, and reuse the results in subsequent exploration. They can prioritize candidates, inspect statistical feedback, and read selected source-text regions when needed. The operation grammar and detailed execution rules are provided in Appendix A.

4.1 Overview of IndexAct Interaction

IndexAct groups candidate documents into a state that can be manipulated throughout subsequent exploration. Its basic form, `SetRef`, is an unordered set state that records which documents are candidates. The initial candidate state `CORPUS`, containing the entire corpus, serves as the starting point for exploration. State-producing operations create new states rather than overwriting existing ones, and the agent uses these states as inputs to subsequent operations. It can therefore apply different conditions to earlier candidates or recombine results obtained through different searches.

As shown in Figure 3, the agent selects a candidate state and its next action based on the question and previous observations. Candidate-state operations construct candidates by evaluating search conditions through the inverted index, combine candidate sets, and determine inspection order (Section 4.2). State-relative statistics inform subsequent search decisions through candidate-set size and the effects of conditions (Section 4.3). Fine-grained source access lets the agent read around matching locations or within specified regions when needed to examine clues in context (Section 4.4).

Throughout this process, candidate states remain in the execution environment. The agent can reuse the same state as input to subsequent exploration while selectively inspecting information about its candidates through statistics or source text. Continued use of a candidate state therefore does not require reading all of its documents. Multiple operations determined in advance can be executed together through `EXECUTE`, and the agent selects its next action based on the returned observations.

4.2 Candidate-State Operations

Candidate-state operations address three decisions: which documents to retain, how to combine results, and which documents to inspect first. The operations draw on Boolean retrieval and set algebra for filtering and set composition (Codd, 1972), positional indexing for phrase and proximity search (Manning et al., 2008), and term-based ranking (Robertson & Zaragoza, 2009). The design aims to let agents draw on prior knowledge of search and data manipulation to construct searches.

Condition filtering. `FILTER` creates a new state containing documents that satisfy a condition within an agent-selected set. The input condition combines the following expressions. Boolean operators (`AND`, `OR`, and `NOT`) combine required clues, alternatives, and exclusions. `TERM` specifies term occurrence. Positional conditions constrain the order and spacing of clues within a document, using `PHRASE` for ordered adjacency and `NEAR` for proximity. `ANY_OF` matches locations where any of its alternatives, such as a term or phrase, occurs. Boolean and positional conditions distinguish requiring clues to occur in the same document from requiring them to occur near each other.

Let h be the input state reference, S_h its document set, p a condition, and $\mathcal{I}$ the inverted index. Then

$$S_{h'} = F_{\mathcal{I}}(S_h, p) = \{d \in S_h \mid d \models_{\mathcal{I}} p\}, \tag{1}$$

where h' references the resulting state and $d \models_{\mathcal{I}} p$ means that document d satisfies condition p.

Set composition. The agent combines set states through intersection (`INTERSECT`), union (`UNION`), and difference (`DIFFERENCE`). Results for different clues can be retained separately, inspected independently, and recombined as needed. For example, the agent can retain candidates shared by multiple clues, merge candidates found through aliases, or exclude identified distractors.

Ranking. `RANK` determines the inspection order without removing candidates, producing a ranked state, `RankedRef`, containing the same documents. The agent specifies ranking terms (`TERM`) and weights. For each document in the candidate set, the executor computes each term's BM25 contribution and sums these contributions using the agent-specified weights to determine the ranking.

This separation allows the agent to distinguish requirements from preferences. Entity names or phrase and proximity relations that the agent considers mandatory can be used as filter conditions, while auxiliary terms that suggest relevance but are unsuitable as requirements can guide ranking. Documents lacking those auxiliary terms remain candidates; only their inspection order changes.

Ranked-state operations. For a ranked state, `TOPK` selects up to the top k candidates, while `RESTRICT` retains documents also in a specified set, preserving their order. For subsequent filtering or set composition, `AS_SET` produces a new unordered set state containing the same documents.

4.3 State-Relative Statistical Feedback

Statistical feedback helps assess the effects of search conditions without reading candidate source text, informing whether to revise a condition, broaden the scope, or inspect source evidence. Result-size feedback has long supported exploratory query formulation (Tanin et al., 2000), and recent agentic retrieval uses corpus statistics to validate proposed search vocabulary (Yang et al., 2026). The current feedback consists of candidate-set sizes and counts of documents matching each condition. Statistical observations do not automatically include document lists or source-text passages.

State statistics. Creating a state also returns its candidate count. By comparing counts before and after an operation, the agent can determine whether the result is empty, how much a condition reduces the candidates, or how much combining results expands the set of candidate documents.

Condition probing. `COUNT_DOCS` evaluates a condition within a retained candidate state and returns the number of matching documents without registering a new candidate state. Comparing match counts for individual clues and their combinations helps the agent identify and adjust restrictive conditions. Testing the same condition on the current candidates and on `CORPUS` reveals whether matches exist beyond the current scope, helping it decide whether to broaden the search. The agent can then choose which results to retain as candidate states for subsequent exploration.

4.4 Fine-Grained Evidence Access

Fine-grained source access lets the agent examine how the clues are used in their original passages and assess whether the context surrounding a lexical match supports the intended interpretation.

Source-text reading. The agent calls `READ` on a candidate state or an individual document and selects text using its region argument. For ranked states, reading follows their established order.

To read around matches, the agent uses `AROUND` as the region argument and specifies an anchor expression and the context extent before and after each selected match. The executor uses the index to locate matches and return surrounding source text. For example, the agent can reuse a `NEAR` expression from candidate search as the anchor to examine what relationship the nearby clues express. Changing the anchor allows the agent to investigate other clues within the same candidate state.

The agent uses `DOCUMENT` as the region argument to read a full document, or `RANGE` to read a specified region in an individual document. Read results identify the source document and passage location, so the agent can inspect adjacent context or revisit the passage without locating it again.

5 Results

5.1 Experimental Setup

Benchmarks. We evaluate IndexAct on all 830 questions in BrowseComp-Plus (BC+) (Chen et al., 2025), our primary benchmark for agentic search. We additionally evaluate multi-hop QA on HotpotQA (Yang et al., 2018), the answerable version of MuSiQue (Trivedi et al., 2022), 2WikiMultiHopQA (2Wiki) (Ho et al., 2020), and Bamboogle (Press et al., 2023). Following the evaluation scale used in prior multi-hop QA studies (Trivedi et al., 2023; Jeong et al., 2024), we use fixed 500-question subsets of HotpotQA, MuSiQue, and 2Wiki, and evaluate all 125 Bamboogle questions. The same question subsets are used across all methods. BC+ uses its released corpus. For the four QA benchmarks, all methods use the full Wikipedia-18 corpus (Karpukhin et al., 2020), containing approximately 20M passages, as the underlying corpus. More details are provided in Appendix B.1.

Baselines. To evaluate the index-native candidate manipulation and selective reading enabled by IndexAct, we compare against methods spanning fine-grained corpus interaction and selective evidence access. We include DCI-Agent-Lite (Li et al., 2026b), DR-DCI (Lu et al., 2026a), and RARG (Li et al., 2026a) as terminal-based alternatives for corpus exploration. We also include Pi-Serini (Hsu et al., 2026) and Sieve (Wang et al., 2026) to compare against interfaces that already separate retrieval from selective evidence access. As conventional agentic-search baselines, we include Retrieval Agent (BM25) and Retrieval Agent (Dense), which use the same ReAct-style

Table 1: End-to-end comparison of search agents on BrowseComp-Plus and multi-hop QA benchmarks. Bold and underlined entries mark the best and second-best in each column, respectively.

	BrowseComp-Plus			MuSiQue	Hotpot	2Wiki	Bamb.
Method	Acc. ↑	Evi. Cov. ↑	Avg. Ctx. ↓	Acc. ↑	Acc. ↑	Acc. ↑	Acc. ↑
Standard Search Agents							
Retrieval Agent (BM25)	30.5	36.5	31,243	37.8	68.4	63.4	68.8
Retrieval Agent (Dense)	48.2	54.4	24,543	<u>44.4</u>	69.2	<u>69.0</u>	<u>70.4</u>
Agentic Corpus Interfaces							
PI-SERINI	62.7	55.4	**10,835**	41.0	68.4	67.2	68.0
SIEVE	53.0	40.8	15,931	40.2	68.6	59.2	68.0
DCI-Agent-Lite	63.0	43.6	30,161	42.0	68.6	63.6	63.2
DR-DCI	67.2	46.6	22,435	42.6	<u>70.4</u>	66.8	64.8
RARG	<u>69.4</u>	<u>57.2</u>	27,896	43.4	68.2	67.0	64.0
INDEXACT	**73.5**	**60.7**	<u>13,960</u>	**46.2**	**71.8**	**69.2**	**71.2**

search–read loop (Yao et al., 2023) and differ only in the retrieval backend. More detailed descriptions of each baseline are provided in Appendix B.2.

Implementation details. We use GPT-5.4 nano with high reasoning effort across the compared interfaces. We run INDEXACT in a lightweight Pi-based agent harness (Zechner & Pi Contributors, 2026) and expose its operations through structured tool calls. The backend is implemented as a Java service over Apache Lucene, with positional postings for lexical and proximity operations. Candidate-set states are maintained server-side and exposed to the agent through reusable references. We impose a 300-turn limit per question and a 30-second timeout per tool call. Dense retrieval uses Qwen3-Embedding-8B (Zhang et al., 2025) on BC+ and E5-base-v2 (Wang et al., 2022) on Wikipedia-18. Additional implementation and resource details are provided in Appendix B.

Metrics. We report Accuracy on all benchmarks. For each question, an LLM judge determines whether the final answer from the agent's response is semantically equivalent to the reference answer. On BC+, we additionally report Average Live Context, the per-query mean number of dynamic-history tokens supplied to decision-making model calls after truncation or compaction, excluding fixed prompts, tool schemas, and the question. Evidence Coverage is the fraction of annotated evidence documents whose source text enters an agent input during the trajectory; candidate membership alone does not count as observation. Query-level metrics are averaged over questions, including failed and budget-exhausted runs. Answer-scoring procedures, the judge model and prompt, and metric definitions are provided in Appendix C.

5.2 MAIN RESULTS

Performance on BC+. Table 1 shows that INDEXACT achieves the highest answer accuracy and Evidence Coverage on BC+. Among the corpus interfaces that outperform conventional Retrieval Agents, RARG is the strongest baseline, while INDEXACT further improves both metrics. RARG combines retriever-based prioritization with fine-grained source-text exploration by traversing documents in the order induced by the retriever. However, this also means that which search matches reach the agent first depends on the retriever's ranking. Because only a bounded number of matches can be returned to the agent, matches from higher-ranked documents can occupy the available output while useful clues in lower-ranked documents remain unobserved (Li et al., 2026a). In contrast, INDEXACT lets the agent directly refine and combine retained candidate sets according to its current search conditions, assess their effects through statistical feedback, and separately decide what source text to inspect. The improvements in both accuracy and Evidence Coverage suggest that, beyond relevance-guided document traversal, explicit control over candidate sets and feedback on their refinement can further support effective evidence search.

Context use. Compared with the terminal-based alternatives, INDEXACT achieves higher answer accuracy and Evidence Coverage with a lower Average Live Context. Specifically, its Average Live Context is 13,960 tokens, approximately half that of RARG and also lower than those of DCI-Agent-Lite and DR-DCI. The selective-reading baselines provide a complementary comparison.

INDEXACT improves answer accuracy and Evidence Coverage over SIEVE while also reducing Average Live Context. PI-SERINI has an even lower Average Live Context than INDEXACT, but its answer accuracy and Evidence Coverage are lower. These comparisons highlight the importance of using context effectively to support evidence discovery, rather than simply minimizing Average Live Context. This pattern is consistent with separating candidate states retained for continued exploration from source text retained in the agent's context.

Multi-hop QA. The benefits also extend to multi-hop QA over Wikipedia-18. Here, Retrieval Agent (Dense) remains a strong alternative: it is the highest-scoring baseline on MuSiQue, 2Wiki, and Bamboogle, whereas DR-DCI performs best on HotpotQA. Strong BC+ performance therefore does not uniformly translate into an advantage on these QA benchmarks. Nevertheless, INDEXACT achieves the highest accuracy on all, outperforming the strongest baseline across benchmarks. Although the margins are smaller than on BC+, the improvements across these additional benchmarks show that the observed benefits are not confined to the primary evaluation. Index-native corpus interaction also provides an effective interface for multi-hop QA over a different underlying corpus.

6 ANALYSIS

6.1 CANDIDATE REFINEMENT AND EVIDENCE ACQUISITION

We examine how refinement feedback affects evidence acquisition and context use, keeping candidate-state operations and explicit reading unchanged. *Auto Preview* adds bounded source-text snippets to `FILTER` and set-composition outputs. *Binary Feedback* replaces exact counts in state-producing outputs and `COUNT_DOCS` responses with empty/non-empty indicators. Alongside accuracy, evidence coverage, and average live context, we report model calls per query and cumulative source tokens returned by tools per query.

Observation policy. Auto Preview uses fewer model calls but lowers both accuracy and evidence coverage (Table 2). Average live context remains close to that of INDEXACT, as additional tool-response content is offset by less agent-command history. The joint reduction in calls and coverage suggests that previews encourage earlier answer commitment before further evidence is sought or verified. An immediate clue can guide search, but does not establish that the remaining constraints have been satisfied. INDEXACT instead achieves broader evidence coverage with more model decisions but less source-text exposure. Thus, shorter trajectories do not necessarily yield more effective search.

Table 2: Observation-policy ablation on BC+.

Method	Acc.↑	EviCov.↑	AvgCtx↓	Calls	SrcTok.↓
Auto Preview	70.0	52.5	14,842	38.5	32.0K
Binary Feedback	62.0	55.9	**14,492**	39.8	35.2K
INDEXACT	**74.0**	**63.9**	15,107	42.9	**30.8K**

Statistical feedback. Binary Feedback has the smallest average live context, yet exposes the most source text and yields the lowest accuracy. Its smaller context reflects less accumulated non-source interaction history, not reduced source exposure. Exact counts reveal whether a condition substantially narrows the candidates or leaves the search largely unchanged, helping the agent decide whether to refine further or read. The lower accuracy and evidence coverage despite greater source exposure suggest that additional text does not compensate for the loss of this state-level guidance. Statistical feedback is therefore part of search control, not merely compact metadata. Together, the comparisons favor informative feedback and selective reading over minimizing working context or interaction count alone.

6.2 CANDIDATE STATES AND EXPLORATION

Candidate states. We examine the contributions of candidate states and index operations by removing each capability separately while retaining the observation policy of INDEXACT (Table 3). More details are provided in Appendix D.2. Removing state reuse

Table 3: Candidate-state and operation ablations on BC+.

Method	Acc.↑	EviCov.↑	AvgCtx↓	Calls
INDEXACT	**74.0**	**63.9**	15,107	42.9
w/o State Reuse	66.0	58.4	**14,365**	42.2
w/o Set Composition	62.0	53.1	16,946	54.9
w/o Positional Ops.	70.0	59.1	14,907	44.3

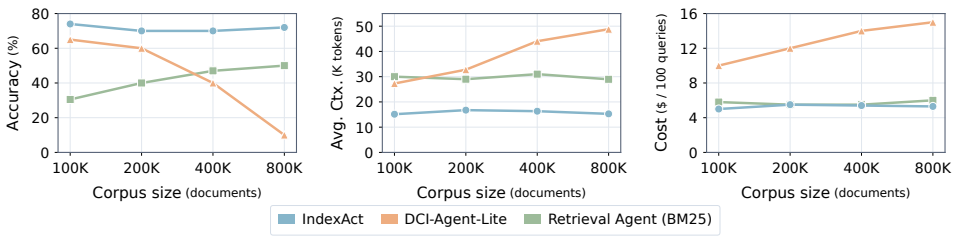

Figure 4: Effect of corpus expansion on accuracy, average live context, and API cost on BC+. The corpus grows from 100K to 800K documents, with the same 100 questions and evidence.

lowers both accuracy and evidence coverage despite a similar number of model calls, suggesting that access to intermediate results supports exploration beyond simply increasing the number of interaction steps. Removing set composition, while preserving Boolean filtering, causes the largest performance drop and increases both model calls and average live context. The additional interaction does not recover the loss in evidence coverage, indicating that direct composition provides value beyond expressing multiple conditions in a new query. Together, these support treating intermediate candidate sets as reusable and composable search states rather than one-off retrieval outputs.

Positional operations. Removing `PHRASE` and `NEAR` also lowers accuracy and evidence coverage, despite slightly more model calls and similar average live context. This suggests that document-level term co-occurrence alone does not provide the same control over evidence search as explicit phrase and proximity conditions. Positional operations therefore complement Boolean filtering by allowing the agent to specify more precise lexical relationships when selecting candidates.

6.3 Robustness to Corpus Scale

Corpus expansion. To examine whether fine-grained corpus exploration can retain the scaling advantages of indexed retrieval, we compare IndexAct with Retrieval Agent (BM25) and DCI-Agent-Lite as the corpus expands. We keep 100 BC+ questions and their original evidence fixed, expanding the corpus from approximately 100K to 200K, 400K, and 800K documents using nested samples of FineWeb pages (Penedo et al., 2024). Model settings and execution budgets remain fixed within each method across scales. Corpus construction is detailed in Appendix E.

Performance and resource use. Figure 4 shows that IndexAct maintains 70.0–74.0% accuracy as the corpus grows eightfold, with little change in average live context or API cost. DCI-Agent-Lite instead uses more context and incurs higher costs while its accuracy declines. Broad terminal searches may scan more raw text as the corpus grows, while returned matches and tool-error messages can accumulate in the interaction history without advancing the search.

IndexAct evaluates lexical and positional conditions over index postings and retained candidate sets rather than repeatedly scanning raw documents. Refinement returns references and statistics, with source text accessed through separate reads. A larger corpus can therefore increase index-side processing without a corresponding increase in the text returned by each refinement. Retrieval Agent (BM25) also exhibits stable resource use, consistent with indexed access, but remains less accurate than IndexAct across tested sizes. These support extending index-based execution to fine-grained candidate exploration while maintaining answer quality and stable working-context use.

7 Conclusion

This paper proposes IndexAct, an interface for Index-Native Corpus Interaction that separates candidate-set refinement from source-text inspection. It lets agents construct, refine, and reuse persistent candidate states through index-based operations, using statistical feedback to guide exploration while selectively reading source text when needed. Our experiments demonstrate strong performance across agentic search and multi-hop question answering, while further analyses show that informative refinement feedback and reusable candidate states support effective evidence discovery and remain robust as corpus size grows. We hope that IndexAct will contribute to future work on more effective and context-efficient interfaces for agentic search.

AI USE STATEMENT

LLMs were used to assist with manuscript writing and editing for clarity and readability, as well as with code implementation for the experiments. All AI-assisted text was reviewed by the authors, and all AI-generated code was manually verified before being used in the experiments. The authors remain fully responsible for the final content of the manuscript and the code used in this work.

REFERENCES

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, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and Jimmy Lin. Browsecomp-plus: A more fair and transparent evaluation benchmark of deep-research agent. In *First Workshop on Multi-Turn Interactions in Large Language Models*, 2025. URL https://openreview.net/forum?id=YJAA2PzfDi.

E. F. Codd. Relational completeness of data base sublanguages. *Research Report / RJ / IBM / San Jose, California*, RJ987, 1972. URL https://api.semanticscholar.org/CorpusID:41445196.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In Donia Scott, Nuria Bel, and Chengqing Zong (eds.), *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.580. URL https://aclanthology.org/2020.coling-main.580/.

Tz-Huan Hsu, Jheng-Hong Yang, and Jimmy Lin. Rethinking agentic search with pi-serini: Is lexical retrieval sufficient?, 2026. URL https://arxiv.org/abs/2605.10848.

Yulong Hui, Chao Chen, Zhihang Fu, Yihao Liu, Jieping Ye, and Huanchen Zhang. Interact-rag: Reason and interact with the corpus, beyond black-box retrieval. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 112154–112174, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/b61f288da3c106f65d57b0d45b470b6b-Paper-Conference.pdf.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 conference of the north american chapter of the association for computational linguistics: Human language technologies (volume 1: Long papers)*, pp. 7036–7050, 2024.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Rwhi91ideu.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550/.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

Jiangnan Li, Yuqing Li, Mo Yu, Jinchao Zhang, and Jie Zhou. A new role for relevance: Guiding corpus interaction in agentic search, 2026a. URL https://arxiv.org/abs/2607.24223.

Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yongkang Wu, Ji-Rong Wen, Yutao Zhu, and Zhicheng Dou. Webthinker: Empowering large reasoning models with deep research capability. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 120091–120131. Curran Associates, Inc., 2025. doi: 10.52202/085713-4011. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/ae03bdef276132fae089692445725635-Paper-Conference.pdf.

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

Yi Lu, Zhuofeng Li, Ping Nie, Haoxiang Zhang, Yuyu Zhang, Kai Zou, Wenhu Chen, Jimmy Lin, Dongfu Jiang, and Yu Zhang. Dr-dci: Scaling direct corpus interaction via dynamic workspace expansion, 2026a. URL https://arxiv.org/abs/2606.14885.

Yijun Lu, Rui Ye, Yuwen Du, Jiajun Wang, Songhua Liu, and Siheng Chen. Longseeker: Elastic context orchestration for long-horizon search agents, 2026b. URL https://arxiv.org/abs/2605.05191.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

OpenAI. Introducing gpt-5.4 mini and nano, 2026. URL https://openai.com/index/introducing-gpt-5-4-mini-and-nano/.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, 2023.

Stephen E. Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3:333–389, 2009. URL https://api.semanticscholar.org/CorpusID:207178704.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9248–9274, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.620. URL https://aclanthology.org/2023.findings-emnlp.620/.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2503.05592.

Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. Zerosearch: Incentivize the search capability of llms without searching, 2026. URL https://arxiv.org/abs/2505.04588.

Egemen Tanin, Amnon Lotem, Ihab Haddadin, Ben Shneiderman, Catherine Plaisant, and Laura Slaughter. Facilitating data exploration with query previews: A study of user performance and preference. *Behaviour & Information Technology*, 19(6):393–403, 2000. doi: 10.1080/014492900750052651. URL https://doi.org/10.1080/014492900750052651.

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

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10014–10037, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.557. URL https://aclanthology.org/2023.acl-long.557/.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

Shuai Wang, Haodong Chen, Yu Yin, Shengyao Zhuang, Bevan Koopman, and Guido Zuccon. Search, inspect, fetch: Exploiting structure-aware boolean retrieval for deep-search agents, 2026. URL https://arxiv.org/abs/2608.02751.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*, 2025.

Zeyu Yang, Xu Han, Qi Ma, Jason Chen, and Anshumali Shrivastava. Superintelligent retrieval agent: The next frontier of agentic retrieval, 2026. URL https://arxiv.org/abs/2605.06647.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Mario Zechner and Pi Contributors. Pi: A minimal terminal coding harness. https://shittycodingagent.ai, 2026.

Haoxiang Zhang, Qixin Xu, Zhuofeng Li, Lei Zhang, Pengcheng Jiang, Yu Zhang, and Julian McAuley. Masking stale observations helps search agents – until it doesn't: A regime map and its mechanism, 2026. URL https://arxiv.org/abs/2606.00408.

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

Shengyao Zhuang, Yuansheng Ni, Hengxin Fun, Jimmy Lin, and Xueguang Ma. Towards retrieving interaction spaces for agentic search, 2026. URL https://arxiv.org/abs/2606.06880.

CONTENTS OF APPENDIX

Page

A Index-Native Interaction **14**
A.1 Operations and matching semantics 14
A.2 Reading and execution 15
A.3 A worked interaction example 15

B Experimental Details **16**
B.1 Datasets and sampling 16
B.2 Baselines 16

C Context and Evidence Measurement **17**
C.1 Scoring and accounting 17
C.2 Preliminary analysis 18

D Controlled Variants and Additional Analyses **20**
D.1 Refinement feedback and selective reading 20
D.2 Candidate-state capabilities 20
D.3 Exploration behavior 21

E Corpus Scale and Resource Use **22**

F Agent Instructions **22**

A INDEX-NATIVE INTERACTION

The interface separates candidate membership, inspection order, and source-text observation. An unordered `SetRef` identifies candidate documents, and a `RankedRef` adds an inspection order. The root `CORPUS` denotes the entire corpus. State-producing operations create new states without overwriting their inputs and return a reference, count, and state metadata rather than text or membership lists. References remain available within the session until released.

A.1 OPERATIONS AND MATCHING SEMANTICS

The grammar uses S for an unordered state, R for a ranked state, and H for either. It summarizes operation syntax, not JSON serialization. `AS_SET` converts ranked states before unordered-set operations.

Candidate-state and expression grammar

```
State operations and counts
  FILTER(S, p)                          -> SetRef
  INTERSECT(S1, S2) | UNION(S1, S2)      -> SetRef
  DIFFERENCE(S1, S2) | AS_SET(R)         -> SetRef
  RANK(S, score) | TOPK(R, k)            -> RankedRef
  RESTRICT(R, S)                        -> RankedRef
  COUNT(H) | COUNT_DOCS(H, p)            -> integer
Lexical expression
  e ::= TERM(text) | ANY_OF(e1,...,en)
      | PHRASE(e1,...,en)
      | NEAR(e1,...,en, ordered, max_gaps)
Document condition
  p ::= e | AND(p1,...,pn) | OR(p1,...,pn) | NOT(p)
Ranking expression
  score ::= TERM(text) | COMBINE(TERM(t1),...,TERM(tn))
          | WEIGHT(w1,TERM(t1),...,wn,TERM(tn))
```

Filtering, composition, and counts. `FILTER` evaluates a document condition within its input set. In particular, `NOT` excludes matching documents from that set, not from an implicit corpus-wide search. `INTERSECT`, `UNION`, and `DIFFERENCE` combine set membership without duplicates. `COUNT` returns the size of an existing state; `COUNT_DOCS` returns the number of documents satisfying a condition within that state without registering another state. Neither count operation returns matching passages.

Lexical and positional conditions. The analyzer applies Unicode normalization and lowercase mapping without stemming or stopword removal. Each `TERM` argument must produce exactly one analyzed token, and matching is performed at the token level. `AND` and `OR` combine document-level conditions. In contrast, `ANY_OF` preserves matching locations from its alternatives, allowing it to occur within a positional expression or serve as a reading anchor. `PHRASE` requires ordered adjacency. `NEAR` bounds the number of uncovered token positions between its child matches and optionally requires their submitted order. The same matching semantics apply to candidate selection and `AROUND` anchors.

Ranking and membership. `RANK` uses a weighted sum of BM25 term contributions, with non-negative weights; `COMBINE` assigns unit weights. The index implementation uses $k_1 = 1.2$ and $b = 0.75$, with corpus-wide statistics rather than statistics recomputed within each candidate set. `RANK` changes only the inspection order and preserves all input members. Ties are broken by document-key order. `TOPK` retains the highest-ranked k members, whereas `RESTRICT` retains members also found in a supplied set without changing their order. `AS_SET` removes ordering but preserves membership.

A.2 READING AND EXECUTION

Source-reading grammar

```
Source reading
  READ(H, documents, region, budget)
  READ(DocKey, region, budget)
  documents ::= ALL_DOCUMENTS | PAGE(limit, after?)
  region    ::= DOCUMENT | RANGE(start,end)
              | AROUND(e, selector, before, after)
  selector  ::= FIRST | NTH(n) | ALL
  budget    ::= {max_output_codepoints, max_evidence_count}
```

Document and region selection. A state read selects all documents or a page without changing candidate membership. Unordered states follow document-key order; ranked states follow their established order. `DOCUMENT` reads the whole selected document, and `RANGE` reads a half-open interval in an individually identified document. `AROUND` locates an anchor match and returns the requested context before and after it. Its selector chooses the first, the nth, or all occurrences; overlapping windows are merged when all occurrences are requested. Reading offsets and context extents are measured in raw-text Unicode code points, whereas phrase and proximity conditions use analyzed-token positions.

Returned evidence and output limits. Passages include their source document, raw-text offsets, and exact text. To preserve passage integrity, over-budget requests return budget feedback so that the agent can issue a smaller read. A candidate-state reference by itself is not an evidence citation.

Batched operations. The model-facing `index_execute` tool batches non-text operations, and `index_read` returns text. An `EXECUTE` batch supports dependent state operations, while adaptive decisions are made across tool calls. State metadata and lineage can be inspected through `index_state` without reading candidate contents.

A.3 A WORKED INTERACTION EXAMPLE

The implementation includes a four-document fixture: `doc-a` contains `a b`, `doc-b` contains `a c`, `doc-c` contains `c`, and `doc-empty` is empty. Table 4 illustrates the information returned by successive operations. Handles are symbolic and only the relevant response fields are shown. This is an operation-semantics example, not a benchmark question or an LLM-generated trajectory.

Table 4: Candidate feedback and source observation on the four-document fixture. `READ` uses the complete-document region and a sufficient output budget.

Operation	Returned information
`FILTER(CORPUS, TERM("a"))`	h_a, count 2
`FILTER(CORPUS, TERM("c"))`	h_c, count 2
`INTERSECT`(h_a, h_c)	$h_\cap$, count 1
`DIFFERENCE`(h_a, h_c)	$h_\setminus$, count 1
`COUNT`(h_a)	2; parent unchanged
`READ`$(h_\cap, \texttt{DOCUMENT})$	`doc-b`, $[0, 3)$, `a c`

The first five responses contain no source text. Intersection and difference create new candidate states while preserving their input states. Only the final read exposes a passage. The agent could instead inspect the residual branch or apply another condition to an earlier state. A singleton result narrows eligibility but does not by itself establish that the document supports an answer.

B EXPERIMENTAL DETAILS

B.1 DATASETS AND SAMPLING

The primary evaluation uses all 830 BC+ questions and the released corpus (Chen et al., 2025). For multi-hop QA, we use fixed 500-question subsets of HotpotQA, the answerable version of MuSiQue, and 2WikiMultiHopQA, and all 125 Bamboogle questions (Yang et al., 2018; Trivedi et al., 2022; Ho et al., 2020; Press et al., 2023). The 500-question subsets are randomly sampled from the respective datasets. Each subset is held fixed across methods.

All four QA benchmarks use the full Wikipedia-18 corpus, containing approximately 20M passages, rather than task-specific evidence collections (Karpukhin et al., 2020). Each comparison uses a common corpus. The preliminary and scaling analyses each use 100 BC+ questions; Appendix D specifies the ablation subsets.

B.2 BASELINES

We compare INDEXACT with agents that access corpora through ranked retrieval results, direct corpus interaction, or structured search and selective reading.

Retrieval Agent (BM25/Dense). Retrieval Agent follows a ReAct-style search–read loop: the agent submits a query, inspects the retrieved evidence, and either continues searching or produces an answer. The BM25 variant retrieves documents using lexical relevance scores, whereas the dense variant uses similarity between query and document embeddings.

DCI-Agent-Lite. DCI-Agent-Lite gives the agent direct access to the raw corpus through general-purpose terminal tools, rather than a separate retrieval API (Li et al., 2026b). The agent composes lexical searches, locates relevant passages, and inspects local context to discover clues and verify candidate answers.

DR-DCI. DR-DCI combines retrieval-based candidate discovery with direct corpus interaction (Lu et al., 2026a). Through its `pull` action, the agent retrieves documents into a persistent local workspace and then searches and reads within that workspace. Additional pulls expand the workspace as new clues or unresolved evidence needs emerge, allowing retrieval and local investigation to alternate throughout the search.

RARG. RARG uses retrieval relevance to guide grep-style corpus exploration (Li et al., 2026a). It orders documents by relevance and directs `ripgrep` to traverse them in that order, so that promising documents are searched before less relevant ones.

PI-SERINI. PI-SERINI separates retrieval, result browsing, and document reading into distinct actions (Hsu et al., 2026). It retains a cached ranking that the agent can browse beyond the initially displayed results without issuing another query. The agent selectively opens promising documents and reads relevant portions, allowing deeper retrieval without immediately placing all retrieved document contents in its context.

Sieve. Sieve follows a Search–Inspect–Fetch workflow that preserves webpage structure during retrieval and evidence acquisition (Wang et al., 2026). The agent uses fielded Boolean queries to select eligible webpages, which are then ordered by a ranker. Structured result cards expose headings and other navigation cues, allowing the agent to inspect the available structure and fetch selected sections rather than entire webpages.

C Context and Evidence Measurement

We distinguish candidate membership, source-text observation, and dynamic context retained by the model. Table 5 summarizes the definitions and accounting scopes of these quantities.

C.1 Scoring and accounting

Answer quality. For BC+, an LLM judge determines whether the final answer extracted from the agent's response is semantically equivalent to the reference answer. We use the grading prompt from the official BC+ evaluation code (Chen et al., 2025), reproduced verbatim below. The judge is `gpt-5.4-nano` with reasoning effort set to `low`. Accuracy is the percentage of questions receiving a `correct: yes` verdict.

BC+ judge prompt

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 judgement 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 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.

For the four multi-hop QA benchmarks, reference answers are short entities, dates, or numbers, so string-matching metrics such as EM and F1 penalize correct answers that differ only in surface form. We therefore also use an LLM judge, with the grading prompt from the official BrowseComp evaluation code (Wei et al., 2025), reproduced verbatim below. The judge model and accuracy definition are the same as for BC+.

Multi-hop QA judge prompt

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

[question]: {question}

[response]: {response}

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

extracted_final_answer: The final exact answer extracted from the [response]. Put the extracted answer as 'None' if there is no exact, final answer to extract from the response.

[correct_answer]: {correct_answer}

reasoning: Explain why the extracted_final_answer is correct or incorrect based on [correct_answer], focusing only on if there are meaningful differences between [correct_answer] and the extracted_final_answer. 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 match.

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 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.

Average live context. Let $H_{q,t}^{\mathrm{dyn}}$ be the dynamic-history text supplied to decision-making model call t for question q, after truncation or compaction. We exclude fixed prompts, tool schemas, and the original question. With T_q such calls, the reported quantity is

$$\mathrm{AvgCtx} = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{T_q} \sum_{t=1}^{T_q} \mathrm{tok}(H_{q,t}^{\mathrm{dyn}}). \tag{2}$$

Token counts use the shared `o200k_base` tokenizer, applied to the recorded readable dynamic-history text. Opaque reasoning payloads and provider-specific framing are excluded. Dynamic summaries and ledgers are included whenever they are supplied to the model, including in system or developer messages. We average first within each question and then across questions, giving each question equal weight. AvgCtx therefore measures the mean readable dynamic-history size per decision.

Evidence coverage. Let E_q be the annotated evidence-document set and O_q the documents whose source text enters at least one agent input. Following the definition in the main text,

$$\mathrm{EviCov} = \frac{100}{|Q|} \sum_{q \in Q} \frac{|E_q \cap O_q|}{|E_q|}. \tag{3}$$

Evidence coverage measures document-level source observation, rather than passage-level evidence use. Candidate membership, workspace import, and a filename alone do not count as observation. We use the full BC+ evidence annotations (`evidence_doc_keys`, derived from `qrel_evidence.txt`), rather than only the smaller gold-document subset.

Model calls and source tokens. *Calls* is the mean number of decision-making model requests per question. *SrcTok* is the per-question mean of cumulative source-text tokens returned by tool executions. A new tool execution returning previously seen text is counted again, whereas replaying an existing tool result in subsequent model inputs does not add to SrcTok. The preliminary analysis separately measures deduplicated source exposure.

Table 5: Definitions and accounting scopes of context and evidence measures.

Quantity	What is counted	Accounting scope
Candidate membership	Documents in a retained state.	Membership in the exploration state.
Evidence coverage	Annotated evidence documents with observed source text.	Document-level source observation.
Avg. live context	Readable dynamic-history tokens supplied at each decision.	Averaged within questions, then across questions.
Source delivery	Source-text tokens returned by tool executions.	Cumulative delivery per question, including repeated reads.
Unique exposure	Observed source intervals merged by document and position.	Deduplicated exposure within each trajectory.

C.2 Preliminary analysis

We analyze Retrieval Agent (BM25), based on the official BC+ implementation, and DCI-Agent-Lite on the same fixed 100-question subset. Both use GPT-5.4 nano with high reasoning effort. The two analyses use the same runs.

Source provenance. We group source text into gold documents, other annotated evidence documents, and documents outside the annotated evidence set. *Other* includes agent-generated messages, non-source tool output, and text whose source attribution cannot be verified.

Returned passages and local search/read outputs are associated with documents using explicit document identifiers for structured tool results and resolved corpus paths for terminal outputs. We then verify the returned text against the corresponding canonical document body. For terminal outputs, verification requires whitespace-normalized body fragments of at least 24 Unicode codepoints. Filenames, metadata headers, and title-only matches do not establish source-text exposure. Token accounting follows Appendix C.1.

Analysis I: retained context. We examine the final logged model request after truncation or compaction, excluding the question, fixed instructions, and tool specifications. Category shares are calculated within each request and then averaged across trajectories, giving each trajectory equal weight. This analysis characterizes the composition of the dynamic context retained at the final model request.

Analysis II: first evidence exposure. Among trajectories that reach an annotated evidence source, we identify the first model request containing its text. Deduplicated non-evidence source text is divided into content first observed up to and including that request and content first observed afterward. We normalize both amounts by the trajectory's total deduplicated non-evidence text and average the shares for each agent. Non-evidence text in the first evidence-bearing request belongs to the first group.

Repeated content is deduplicated using within-trajectory unions of observed intervals in each canonical document body. Intervals are located using recorded offsets, verified line numbers, or unique whitespace-normalized substring matches. Overlapping intervals are counted only once and assigned to their earliest observed request; newly exposed regions of a previously read document still count. Identical text at different source positions or in different documents is not merged.

These analyses characterize context retention and exposure timing relative to the annotated evidence set.

D CONTROLLED VARIANTS AND ADDITIONAL ANALYSES

The ablations examine refinement feedback and candidate-state capabilities separately. Within each comparison, the question set, corpus, backbone, and remaining interface components are held fixed. Both ablation groups use the same fixed 100-question BC+ subset as the preliminary analysis (Appendix C.2). Results are reported in Tables 2 and 3.

D.1 REFINEMENT FEEDBACK AND SELECTIVE READING

Auto Preview. This variant adds bounded source-text snippets to successfully completed `FILTER`, `INTERSECT`, `UNION`, and `DIFFERENCE` responses, while preserving candidate-state operations and explicit reading. Each preview selects up to five documents in the resulting set's canonical document-key order. Each selected document contributes one snippet of at most 440 Unicode codepoints, yielding at most 2,200 source codepoints per qualifying operation.

Snippet anchors are the first 32 distinct positive lexical terms collected from the operation and its parent-state lineage. Negated terms are excluded, and `DIFFERENCE` inherits terms only from its left operand. When no lineage terms are available, the first 32 distinct analyzed question tokens are used. The snippet extractor selects a term-matched window, falling back to the document prefix when no terms match. Empty states return no source text.

Previews are attached to the existing tool response without an additional model call. The preview budget applies independently to each qualifying operation within a batch, and repeated or overlapping previews are retained. State references and statistical feedback remain available.

Binary Feedback. This variant replaces exact counts in state-producing outputs and `COUNT_DOCS` responses with empty/non-empty indicators. The same masking applies to `COUNT`, state metadata, state listings, and model-visible ledgers, including the corpus state. Read diagnostics omit document/evidence count totals and lower bounds. Free-form error messages and details are sanitized to remove numerical cardinality information. Exact internal values are retained in audit artifacts outside the model-visible interface.

All operations remain enabled. Source text, document identifiers, offsets, pagination cursors, and codepoint-budget information are preserved. The intervention targets explicit numerical cardinality feedback; record enumeration and repeated binary queries remain available through the retained interface.

D.2 CANDIDATE-STATE CAPABILITIES

The state ablations remove one capability at a time while retaining the observation policy of INDEX-ACT. State operations return references and statistical feedback, and source text is obtained through explicit reading.

Without state reuse. This variant disables cross-call reuse of intermediate candidate states for search and refinement. Sequential refinement remains available within a single `index_execute` call through batch-local bindings. Search or refinement in a subsequent call must reconstruct the candidate state from `CORPUS`.

Previously created states remain available for explicit reading, including repeated and paginated reads, plain `COUNT`, and state inspection. The ablation therefore isolates cross-call search reuse while preserving access to previously selected documents.

Without set composition. This variant removes `INTERSECT`, `UNION`, `DIFFERENCE`, and `RESTRICT`, while preserving Boolean filtering. Removing `RESTRICT` also closes the ranked-set route for combining retained memberships.

The agent can express multiple conditions within a new filter. For Boolean conditions over the same corpus,

$$F(\mathcal{C}, p) \cap F(\mathcal{C}, q) = F(\mathcal{C}, p \wedge q).$$

The intervention isolates direct composition of previously computed candidate states while retaining conjunctive filtering.

Without positional operations. This variant removes `PHRASE` and `NEAR`, while retaining term matching and Boolean conditions. The restriction applies recursively to candidate conditions in `FILTER`/`COUNT_DOCS` and to reading anchors in `AROUND`. Term-anchored reading and existing term-based snippet extraction remain available, and the underlying positional index is unchanged. The ablation evaluates phrase and proximity constraints across both candidate selection and targeted reading.

D.3 Exploration behavior

We define cross-call search reuse and set composition through successful operations and their state dependencies.

Cross-call search reuse. A reuse event is a successful search or refinement operation that consumes a state created by an earlier `index_execute` call. This definition includes continued refinement of the latest branch as well as reuse of an earlier state. Same-batch bindings, explicit reads, and plain metadata observations are outside this event definition.

Set composition. A composition event is a successful `INTERSECT`, `UNION`, `DIFFERENCE`, or `RESTRICT` operation. Both event types are counted per successfully committed operation, including operations committed before a later failure in the same batch.

E CORPUS SCALE AND RESOURCE USE

The scaling study keeps 100 BC+ questions and their original evidence fixed while expanding the released corpus of approximately 100K documents to approximately 200K, 400K, and 800K documents with nested samples of FineWeb pages. Each larger collection contains the original corpus and the distractors included at the smaller scale. At a given scale, INDEXACT, Retrieval Agent (BM25), and DCI-Agent-Lite use the same collection. Each method's model settings and execution budgets remain fixed across scales.

Backend resource use. Backend latency, timeout rates, and state storage describe execution outside the model's live context and are reported separately from API cost.

API and index costs. Live-context size is not a billing measure: earlier history can be processed at multiple model calls, and cached input can be priced differently from uncached input. API cost is calculated from recorded usage using the experiment's fixed `gpt-5.4-nano` rates of $0.20 per million uncached input tokens, $0.02 per million cached input tokens, and $1.25 per million output tokens. Cached input is charged at the discounted rate rather than treated as free; answer judging is reported separately.

Completion and interpretation. Latency and cost should be interpreted alongside task completion, as unfinished runs may consume fewer resources without solving the question. Candidate states retained outside the model's context require storage and computation in the index service, which are distinct from model-context usage.

F AGENT INSTRUCTIONS

The following block reproduces the instruction in the supplied `index_native.md`. Markdown structure and JSON whitespace are typeset for readability without changing instruction wording or JSON values. The question is supplied separately. Numeric values in the example tool calls illustrate usage rather than define run-wide limits. The instruction permits reading before the candidate set becomes small; it imposes no fixed count threshold for reading. The answer-judge instruction is specified separately in Appendix C.

Index-Native Corpus Agent Instructions v2

Objective

Answer the user's question using evidence from the current immutable corpus snapshot. Treat corpus evidence observed through `index_read` as the authoritative basis for factual claims. Do not claim to have seen corpus content that has not appeared in an `index_read` result, and never invent an evidence ID.

Finish by calling `submit_answer`.

Mental model

You work with persistent retrieval states, not transient search-result pages.

- `CORPUS` is the root `SetRef` containing every document in the snapshot.
- A `SetRef` is an unordered, persistent document membership set.
- A `RankedRef` has persistent membership plus a deterministic reading order.
- State-producing operations create a new immutable state and return its exact cardinality.
- Existing states remain available across turns. Reuse them, branch from them, and recombine them.
- State handles such as `s3` and `r2` persist across tool calls.
- A batch `binding` exists only inside one `index_execute` call. Use the persistent handle returned in the result on later turns.
- Search, counting, ranking, and state-control operations do not return corpus text.
- Only `index_read` can place corpus text in your context.

The runtime may show a deterministic state ledger and evidence ledger. They are memory aids containing only previously observed state metadata and evidence. They are not new corpus evidence.

Expressions and states

Lexical expressions

- `TERM("x")` matches one analyzed token. It is not substring, wildcard, or regex search.
- `PHRASE(...)` matches ordered, adjacent lexical expressions.
- `NEAR(...)` matches lexical expressions under the specified order and maximum-gap constraint.
- `ANY_OF(...)` is positional choice between lexical expressions and may be used where match locations matter.

Boolean conditions

- `AND`, Boolean `OR`, and `NOT` combine unexecuted document conditions inside `FILTER` or `COUNT_DOCS`.
- Boolean `OR` is document-level truth. It is different from positional `ANY_OF`.

Persistent-state algebra

- `INTERSECT`, `UNION`, and `DIFFERENCE` combine already computed `SetRef` states.
- Use condition algebra when composing one predicate before execution.
- Use state algebra when independently computed intermediate results should remain reusable or branchable.

Eligibility and ordering

Keep hard eligibility separate from soft reading order.

- `FILTER` changes membership by applying an exact lexical or Boolean condition to a `SetRef`.
- `COUNT_DOCS` tests a condition within an existing state without creating another state.
- Most state-producing results already include cardinality; use `COUNT` when you need to re-read an existing state's cardinality.
- `RANK` preserves every input member and adds an order from soft lexical evidence.
- `TOPK` keeps a prefix of a `RankedRef`.
- `RESTRICT` keeps only documents allowed by a `SetRef` while preserving an existing ranked order.
- `AS_SET` deliberately discards rank and preserves membership.

Do not use a soft clue as a hard filter merely because it may be relevant. Conversely, do not rank on a condition that must be true for the answer.

Native tools

`index_execute`

Use `index_execute` for ordered batches of non-text operations:

```
FILTER, INTERSECT, UNION, DIFFERENCE,
COUNT, COUNT_DOCS,
RANK, TOPK, RESTRICT, AS_SET
```

Use one batch when later operations depend on earlier results. Batch-local bindings avoid extra model turns, but they do not replace the persistent handles returned by state-producing results.

Example: narrow a state and inspect its size without reading text.

```
{
  "operations": [
    {"op": "FILTER", "target": "CORPUS", "condition": {"term": "mars"},
     "bind": "mars_docs"},
    {"op": "FILTER", "target": {"binding": "mars_docs"},
     "condition": {"term": "crater"}, "bind": "candidates"},
    {"op": "COUNT", "state": {"binding": "candidates"}}
  ]
}
```

Example: preserve alternative hypotheses and recombine them.

```
{
  "operations": [
    {"op": "FILTER", "target": {"handle": "s1"},
     "condition": {"term": "volcano"}, "bind": "volcanic"},
    {"op": "FILTER", "target": {"handle": "s1"},
```

```
      "condition": {"term": "impact"}, "bind": "impact"},
    {"op": "UNION", "left": {"binding": "volcanic"},
     "right": {"binding": "impact"}, "bind": "combined"}
  ]
}
```

index_read

index_read is the only tool that materializes corpus text.

Use it deliberately when text is likely to reveal:

- a fact needed for the answer;
- a bridge entity for another search hop;
- the corpus's actual vocabulary;
- evidence needed to distinguish or verify hypotheses.

You do not need to wait for a fixed cardinality before reading. Balance the expected information value of text against its context cost.

Reading modes:

- DOCUMENT: read a complete selected document.
- AROUND: read bounded windows around lexical occurrences.
- RANGE: read an exact raw-text range and is valid only for a direct doc_key target.
- For a state target, use a bounded PAGE unless you deliberately intend to process every document.

Example: read two ranked or set-state documents around the first occurrence of a lexical anchor.

```
{
  "target": {"state": {"handle": "s7"}},
  "documents": {"page": {"limit": 2, "after": null}},
  "region": {"around": {
    "anchor": {"term": "crater"}, "selector": {"first": {}},
    "before": 120, "after": 120
  }},
  "budget": {"max_output_codepoints": 4000, "max_evidence_count": 8}
}
```

A ReadBudgetExceeded result contains no partial evidence text. Use its structured diagnostics and page-selection metadata to choose among:

- retrying with a smaller page or region;
- reading a selected doc_key with a bounded RANGE;
- advancing past the selected page when appropriate.

index_state

Use index_state to:

- list active states when the runtime ledger is insufficient;
- inspect a lineage node when derivation provenance matters;
- release states that are no longer useful.

Do not release a state that may still be needed for a branch or later recombination.

submit_answer

Call submit_answer only when the answer is sufficiently supported by observed corpus evidence or when you have exhausted the allowed search and must report the best supported result.

- Include the IDs of Evidence that directly support the answer.
- Cite only Evidence IDs actually returned by prior index_read calls.
- Do not cite state handles as evidence.
- Do not invent document text, ranges, or evidence IDs.

Search policy

Use cardinality and observed evidence as feedback, not as rigid rules.

- A zero-result state can mean that a surface form is absent, the analyzer produced a different vocabulary item, or the condition is too specific. Revise the relevant hypothesis rather than blindly repeating it.
- A very broad state may benefit from another hard condition, an explicit branch, set recombination, or soft ranking.
- A manageable state may be read directly, but there is no universal size threshold.
- Read earlier when new vocabulary or a bridge entity is needed to formulate the next predicate.
- After reading, use newly observed vocabulary to refine the most relevant existing state or create a deliberate new branch.
- Preserve alternative hypotheses as separate states when later evidence may distinguish them.
- Reuse an existing state instead of rerunning an equivalent derivation.
- Inspect structured tool errors and correct the specific invalid reference, type, value, cursor, or budget. Do not repeat the same invalid call unchanged.

The objective is not to minimize READ calls at all costs. The objective is to control when evidence enters context while using persistent index state to test and revise search hypotheses efficiently.