MICRO-TURNS FOR EFFICIENT CONTEXT AND STATE MANAGEMENT IN MULTI-HOP RETRIEVAL AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-running retrieval agents must continually decide what context and state to preserve, compress, rank, or discard. In many harnesses, these decisions are handled by fixed programmatic rules that truncate old messages, apply hand-written compression, or evict content when a capacity limit is reached. We introduce *micro-turns*, short harness-injected model calls dedicated to context and state management between main agent turns. A micro-turn branches from the current trajectory, has the model perform a scoped management operation, and returns a compact state update. After applying the update, the harness discards the micro-turn prompt and response rather than recording them in the persistent trajectory. This makes context and state management model-mediated without lengthening the main session or exposing later main turns to auxiliary management exchanges. We instantiate micro-turns in a multi-hop retrieval agent built on the Harness-1 retrieval framework, where they summarize retrieved chunks, resolve curated-set overflow, and order curated evidence. We empirically show that across a diverse suite of retrieval benchmarks with Qwen3.5-4B, micro-turns improve recall while reducing the persistent-context token and attention cost of every episode, and that ordered curation further strengthens ranking at a bounded, dataset-dependent comparison cost, suggesting a practical harness primitive for efficient, context-heavy retrieval agents.

1 INTRODUCTION

Retrieval-augmented generation grounds language-model outputs on external documents retrieved from a corpus (Lewis et al., 2020). Many realistic information-seeking tasks, however, require more than a single retrieved passage because the answer may depend on composing evidence across multiple documents or using one piece of evidence to determine what should be searched next (Yang et al., 2018; Trivedi et al., 2022). This has motivated multi-hop and agentic retrieval methods in which a model interleaves reasoning, search actions, and observations over several steps (Trivedi et al., 2023; Press et al., 2023; Yao et al., 2023). In such systems, the agent not only chooses the next query, but also operates inside a harness that determines what retrieved content is shown, what state is carried forward, and what evidence remains available for later decisions.

These harness decisions are often implemented as fixed programmatic rules. Tool-using and search agents can call external APIs or retrieval backends (Yao et al., 2023; Schick et al., 2023; Jin et al., 2025), while the harness around the model renders observations (retrieved tool results), truncates long contexts, maintains working state, and enforces capacity limits. These rules make long-running agents predictable and keep the main policy focused on task-level actions. But not all context-and-state management decisions are purely mechanical. Deciding whether a retrieved passage is redundant, whether a weak lead should remain available for backtracking, or which of two pieces of evidence is more relevant can require the same semantic understanding that the agent uses for the main task.

We introduce *micro-turns*, a harness primitive for model-mediated context and state management between main agent turns (Figure 1). A micro-turn is a short, harness-injected model call dedicated to a management operation such as summarizing, removing, ranking, or otherwise updating state. It branches from the current trajectory, returns a compact state update, and is then discarded from the persistent main trajectory. Future main turns see only the effect of the micro-turn, not the auxiliary

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

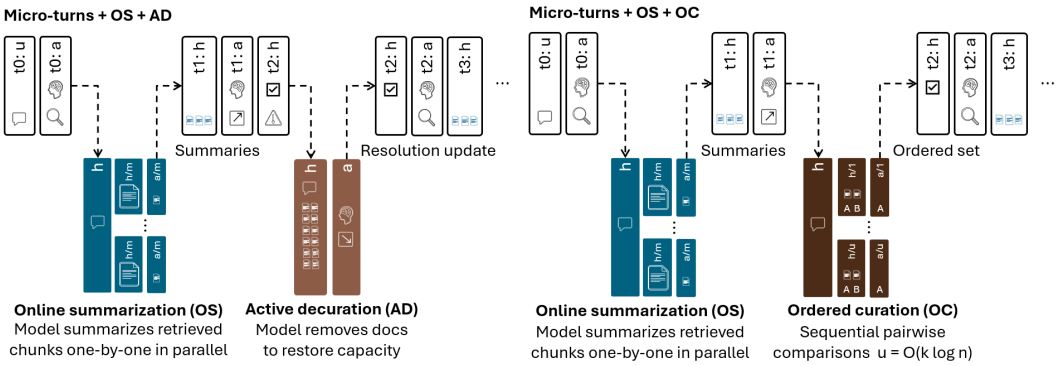

Figure 1: Micro-turns for context and state management in the retrieval harness. t_i is the turn index; u/a are the harness's/model's turns; h is a harness-injected prompt. Main turns search and curate evidence. Online summarization (OS), active decuration (AD), and ordered curation (OC) each branch off to perform a scoped update – chunk summarization, overflow removal, or pairwise ordering, respectively (Section 3.2) – and only the resulting compact state update persists to later main turns.

prompt and response. This lets the harness use model judgment for semantic management decisions without lengthening the main session with additional management exchanges.

We study this idea in multi-hop retrieval because retrieval agents make the context-and-state management problem both severe and measurable. A retrieval agent repeatedly introduces large external text into the session, but its final output is often a compact evidence set for a downstream answerer or evaluator (Bashir et al., 2026; Jiang et al., 2026). Thus, success depends not only on whether evidence was retrieved, but also on whether the harness exposed, compressed, retained, and ordered that evidence appropriately. We build on Harness-1, a strong stateful retrieval harness with a document pool, curated set, recent-turn rendering, and programmatic curation rules (Jiang et al., 2026).

Within this retrieval harness, we instantiate three micro-turn designs. Online summarization (OS) summarizes each returned retrieval chunk before the next main turn, so the agent can see compact summaries of chunks that would otherwise be truncated or omitted from the rendered observation. Active decuration (AD) triggers when the curated set overflows and asks the model to explicitly remove documents, rather than letting the harness resolve overflow by a fixed eviction rule. Alternatively, ordered curation (OC) replaces the curated list's coarse tag order with a total order built by binary insertion. Each new document is compared pairwise against those already retained, giving documents a finer order than the tag tiers could express. This order determines which documents are evicted from the set at capacity and which appear first when a downstream answerer can inspect only a prefix of the curated list.

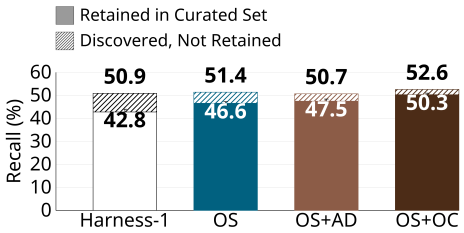

Figure 2: BrowseComp+ recall result (Table 1). Micro-turn variants retain more gold+evidence in the curated set while discovering comparable or greater total gold+evidence in the trajectory.

Across BrowseCompPlus and the Harness-1 retrieval benchmark families (Chen et al., 2026; Bashir et al., 2026; Jiang et al., 2026), these micro-turns improve final curated recall under the same small-model retrieval setup (Figure 2 previews the BrowseCompPlus result). They also improve Recall@n for truncated curated prefixes, especially with ordered curation, and reduce the persistent-context compute burden for OS and AD. OC gives the strongest ranking gains at a modest, dataset-dependent comparison cost.

Our contributions can be summarized as follows:

- We introduce micro-turns as a harness primitive for model-mediated context and state management between main agent turns.

- We instantiate micro-turns in a multi-hop retrieval harness through online summarization, active decuration, and ordered curation.
- We analyze how these micro-turns address concrete failure modes of programmatic retrieval harnesses, including partial exposure to retrieved chunks, repeated prefill over chunk-heavy context, and curation overflow or ordering mismatch.
- We evaluate the resulting harness on a suite of retrieval benchmarks, showing improved final curated recall, stronger curated-prefix quality, and reduced episode-level persistent-context compute for the summarization and decuration variants; ordered curation (OC) adds the strongest ranking gains at a modest, dataset-dependent comparison cost.

2 Multi-Hop Retrieval with a Programmatic Harness

This section gives the background on which our micro-turn design is built. We first place retrieval agents within the broader landscape of multi-hop question-answering methods (Section 2.1), then describe Harness-1, the stateful retrieval harness we adopt (Section 2.2), and finally identify three concrete failure modes of its programmatic context and state management (Section 2.3), which motivate our multi-turns-based approach presented in Section 3.

2.1 Background: From Agentic Search to Stateful Retrieval

In much prior work on multi-hop question answering, the same agent searches for evidence and generates the final answer. Search-R1, for example, trains such an agent to interleave reasoning with retrieval (Jin et al., 2025). As search results accumulate, the retrieved text can consume the working context. Masking stale observations reduces that burden by hiding older tool results, although its effect depends on whether those results are still needed (Zhang et al., 2026). These approaches keep retrieval and answer generation coupled, and when context management is used, it acts on the agent's accumulated search observations.

Retrieval agents isolate evidence collection from answer generation. They return documents intended to support an answer to the user's query, either for direct inspection by the user or for use by a downstream answering model. Over multiple turns, they query a corpus, inspect returned passages identified by stable documents or chunk IDs, and decide which documents to retain. Context-1 allows the agent to explicitly prune retrieved chunks from the model's view, freeing space for subsequent searches (Bashir et al., 2026). Thus, managing the context of the retrieval agent is considered as part of the search process. We next describe Harness-1, a more structured retrieval harness built for this setting.

2.2 Harness-1

Harness-1 (Jiang et al., 2026) provides the model with tools for search, inspection, curation, verification, and stopping, while maintaining the state around those decisions. Operationally, each episode starts with a user query and enters a tool-using search loop whose final output is a curated set of evidence documents required to answer the query. At each turn, the harness assembles the model's context from its maintained state together with a small window of recent tool interactions. The model then issues one or more tool calls, and the harness executes those calls, updates the state, and assembles the context for the next turn.

The harness state tracks the retrieval episode across turns. The *document pool* contains the document IDs encountered by retrieval tools, while the *curated set* contains the documents the model explicitly selects as relevant after seeing rendered evidence. The key retrieval tools for our setting are `search_corpus`, which issues a targeted search for the current query or subproblem, and `fan_out_search`, which runs several related searches in parallel to broaden coverage. Both tools return candidate chunks that are added to the document pool and rendered back to the model under the harness context budget. Subsequent turns therefore operate over a maintained retrieval state rather than an append-only transcript alone.

Curation is the central state update. The `curate` action edits a bounded curated set by adding and removing documents, and assigning an importance tag of `very_high`, `high`, `fair`, or `low` to each added document. In the Harness-1 recipe, the curated set is capped at 30 documents. The

first successful search also auto-seeds a tentative curated set with up to eight `fair`-importance candidates, so the model can refine an initial state rather than build the set from empty. When the cap is reached, the harness resolves the capacity constraint by comparing importance tags and either evicting a lower-importance document or rejecting the new addition. The harness nudges the agent toward a search-then-curate rhythm via injected messages (visible in the main trajectory of Figure 1).

Finally, Harness-1 uses a budget-aware renderer to expose this state to the model. It does not render the full episode transcript verbatim. Instead, the prompt contains a recent window of up to `RECENT_K` (= 5) turns, including the model's recent tool calls and the corresponding tool responses. For retrieval tools, these responses contain retrieved document chunks or snippets, which are compressed, truncated, and display-limited to fit the context budget. Older turns are folded into a compact harness-state summary rather than kept as raw interaction text. This summary includes the curated set, document-pool IDs with short one-line snippets from retrieved chunks, and the search queries issued in earlier turns. Through these interfaces, the model invokes `search_corpus`, `fan_out_search`, and `curate`, while fixed harness rules determine which returned chunks are displayed, how older evidence is represented, and how importance tags resolve curated-set overflow.

2.3 Limitations of Programmatic Context and State Management

Harness-1's fixed state-management rules are effective, but they also expose the boundary where programmatic harnessing becomes too rigid. For small models designed for efficiency (low memory/compute) operating in long deep-retrieval episodes, the design of context rendering and state maintenance can substantially affect inference cost. We trace this rigidity to three concrete failure modes we identify in Harness-1's fixed rules. Its rendering rules can hide retrieved evidence from the model, its capacity rule can override the model's own retention judgment, and its caching interaction forces repeated prefill over context the model has already seen.

Trajectory growth and context truncation. Retrieval tools such as `search_corpus` and `fan_out_search` can return many document chunks, each with substantial text. Harness-1 manages this with hard-coded context rules that control both how many returned chunks are shown in the tool response and how much content is shown per chunk. As a result, some retrieved chunks enter the document pool but are not visible to the model in the immediate prompt, whereas rendered chunks may only expose truncated content. This gap is especially pronounced for `fan_out_search`. Four or five parallel subqueries may return roughly 20–22 chunks in total, but the budgeted response may only show about 8–9 of them (Appendix D.4). Thus, retrieval can surface potentially relevant evidence that the model does not actually see when deciding what to keep next.

Suboptimal curation state management. Harness-1 maintains a bounded curated set with importance tags such as `very_high`, `high`, `fair`, and `low`, giving the model a coarse way to express preference over documents. These tags do not define a complete order within a tier, so the retained curated set also lacks a model-provided ranking when a downstream consumer can only inspect a prefix. Once the curated set reaches capacity, a fixed harness rule admits an incoming document only if its importance can evict a document currently ranked lower; otherwise, the add is rejected (Appendix E reports how often each variant operates at this cap). This can misalign the actual state update with the model's intent. For example, the model may add a `low`-importance document to preserve a weak but potentially useful lead for later comparison, backtracking, or answer synthesis. At capacity, that deliberate low-confidence addition can instead trigger a programmatic rejection or eviction that the model did not explicitly choose. In deep retrieval, where weak clues can become important after later searches, this overflow behavior can drop recall even when the model identified the document as worth retaining.

Context length and prefill cost. Even after truncation, retrieval tool responses contain substantial verbatim chunk text. Although these responses remain in the `RECENT_K` window, they are rendered again for each main turn, increasing the number of input tokens at each turn. Once the window starts sliding, older turns are replaced by an updated working-state summary. This changes the long prompt prefix and prevents reuse of its KV cache, so subsequent turns must prefill the retained chunks again, increasing attention operations despite the bounded context length (Appendix F.1).

Together, these limitations show where fixed state management of the harness falls short. It can hide or discard useful evidence while carrying costly verbatim chunks forward. Moving these decisions into additional main turns would pollute the trajectory and add overhead.

3 MICRO-TURNS FOR CONTEXT AND STATE MANAGEMENT

We first define micro-turns as a general harness primitive, then instantiate them as three concrete retrieval-specific designs (Section 3.2) that address the failure modes discussed in Section 2.3.

3.1 MICRO-TURNS AS A HARNESS PRIMITIVE

As illustrated in Figure 1, a micro-turn branches from the current state of the agent at a point where the harness would otherwise make a fixed context or state-management decision. The harness determines when to invoke the call, what information it receives, and which state update it may produce, while the model handles the content-dependent judgment. After applying the update, the harness resumes the main loop without recording the auxiliary prompt and response. Scheduling and bookkeeping remain programmatic, while decisions about what to compress, keep, or remove can use model judgment.

The call can use the main agent or a separate model, with the current trajectory prefix or a narrower view selected by the harness. It resembles a short-lived sub-agent, but sub-agents may pursue broad delegated subgoals, whereas each micro-turn is scoped to a specific context- or state-management operation. The harness, rather than the main agent, initiates the micro-turn and discards its exchange after applying the update. Recursive Language Models also use additional model calls to inspect selected context, but their model directs that recursive inspection rather than having it scheduled by the harness (Zhang et al., 2025). A narrower analogy comes from Codex's `/side` and Claude Code's `/btw`, which let users ask side questions without adding them to the main conversation.[1] Unlike these side questions, micro-turn outputs can change the context or state of the main agent.

Each call incurs inference cost, but its discarded exchange does not enlarge the context of subsequent main turns. Because a model makes the semantic management decision, that behavior could improve with stronger models or dedicated training rather than remaining a fixed harness heuristic. We study the inference-time harness design here, without training a separate micro-turn policy.

3.2 MICRO-TURNS IN THE RETRIEVAL HARNESS

In our retrieval agent, each micro-turn starts from the main agent's current trajectory prefix (Figure 1) and uses the same model, enabling KV-cache reuse for that prefix.

Online summarization (OS). Rendering retrieved chunks verbatim causes both context truncation and trajectory growth. Online summarization (OS) intervenes after a retrieval tool such as `search_corpus` or `fan_out_search` returns chunks. Before rendering the next main turn, the harness launches one OS micro-turn per chunk, with the calls running in parallel. Each call sees the trajectory so far with exactly one complete, untruncated chunk appended to its prompt, and summarizes that chunk in light of the agent's current context and state. Thus, each auxiliary call processes only one full chunk. We call the operation *online* because these summaries are generated as retrieval proceeds. The harness stores each summary with its chunk and renders the summaries in the next tool response, while discarding the auxiliary exchanges from the main trajectory (Figure 1).

This mechanism directly addresses the chunk omission in Harness-1's `fan_out_search` responses, where the rendering budget can exclude some returned chunks. In our experiments, summaries of all returned chunks fit within that budget (Appendix D.4), so the model can consider the complete retrieval batch. Only the summaries persist in the main trajectory, so later main turns render a compact chunk description rather than the raw text, which is why their prefill cost falls (Table 3).

Active decuration (AD). Active decuration (AD) replaces automatic overflow resolution with an explicit model-mediated removal step. When a curation update would exceed the capacity of the curated set (the warning trigger in Figure 1), the harness temporarily allows overflow rather than rejecting incoming documents or evicting existing ones by a fixed rule. It then triggers an AD micro-turn whose prompt displays summaries of the existing curated documents and newly added documents, ordered from lower to higher priority. The model is directed to remove enough documents to restore the capacity limit. This order places likely removal candidates first while keeping

[1] Codex side chats; Claude Code `/btw`.

the entire overflowed set visible. The model therefore chooses which documents to remove. Low-importance documents can remain as tentative leads when the model considers them useful, rather than being silently discarded by a harness-side capacity rule.

Ordered curation (OC). Ordered curation (OC) addresses the remaining coarseness of importance tags by replacing them with a total order. Two documents marked `fair`, for example, may have a very different value for the current query, yet Harness-1 has no model-provided ordering signal within that tier. Beyond this coarseness, tags are also rigid and imprecise in practice. Once assigned, they are rarely revised without new evidence, and even within a tier precision can be weak (Appendix B).

Ordered curation replaces absolute importance tags with a relative ordering interface. When the model curates documents, it adds document IDs without assigning importance labels. In a subsequent ordered-curation turn, each newly added document is inserted into the existing curated list by a sequence of binary comparisons. In each comparison, the model is asked which of two documents is more relevant to the query and the accumulated evidence. This uses the observation that language models are often more reliable at pairwise preference judgments than at assigning globally calibrated absolute scores. Pairwise prompting has also been shown to be effective in document ranking, including settings where direct relevance labels are less reliable (Qin et al., 2024; Yan et al., 2024).

We treat the curated set as having a perfect total order induced by these comparisons. This makes overflow management straightforward. If the ordered curated list exceeds the capacity limit, the harness drops the documents beyond the cap. In this sense, overflow handling comes for free once the ordered state has been constructed. Unlike tag-based eviction, this removal is not an implicit harness decision over coarse buckets. The model has explicitly placed those documents below the retained documents in relevance order. Ordered curation therefore turns capacity management into a direct consequence of the model's own pairwise preferences, while also providing a ranked prefix for downstream consumption.

The cost of ordered curation is modest because the curated set is capped. If k new documents are inserted into an existing ordered set of size n, the micro-turn performs $O(k \log n)$ pairwise comparisons. In our setting $n \leq n_{\max} = 30$, so the additional comparison budget is bounded and can be measured directly as part of the total compute per episode. OC replaces AD as the curated-set management variant; both are used with OS, but not with each other. Appendix G.3 details the binary-insertion procedure; Appendix G gives the exact prompt templates for all three micro-turns.

4 Experiments

We evaluate whether micro-turns improve final curated recall, episode-level efficiency, and the quality of the prefix consumed by a downstream answerer or evaluator, as motivated in Section 3. All variants use the same model, retrieval backend, turn budget, and curated-set cap; the only change is how the harness manages context and retrieval state between main agent turns.

4.1 Experimental Setup

Our primary model is Qwen3.5-4B-Instruct/Reasoning. In this setting, long prompt prefills and repeated retrieval observations contribute substantially to episode cost. Decoding (up to 2048 new tokens, temperature 0.6, top-p 0.95, top-k 20) is fixed across the harness variants, and all reported means use 5 seeds. The current evaluation focuses on BrowseCompPlus (BC+) (Chen et al., 2026), using the 631-query hold-out test split. BC+ is useful for this paper because it separates gold answer documents from broader supporting evidence, allowing us to measure both gold-only and gold+evidence recalls. We also evaluate Harness-1 benchmark families used outside BC+, including the Context-1/Harness-1 Web, WebSimple, Patents, and SECSimple benchmark families (Bashir et al., 2026; Jiang et al., 2026). We evaluate 150 queries each for Web and Patents and 75 each for WebSimple and SECSimple. Across all benchmarks, we match the Harness-1 retrieval setup where possible and use BM25 retrieval without reranking because it is straightforward to host.

The baseline is the Harness-1 recipe ported to the same model and benchmark setup. We compare it with **Micro-turns+OS**, which summarizes retrieval observations online; **Micro-**

Table 1: BrowseCompPlus retrieval results on 631 queries. Entries are five-seed mean$\pm$population-std; bold marks the best mean in each column. OS, OS+AD, and OS+OC are the micro-turn variants.

Harness	Recall	Traj. Recall	FAR	Traj. FAR
Harness-1	0.428$\pm$0.012	0.509$\pm$0.004	0.461$\pm$0.018	0.544$\pm$0.004
Micro-turns+OS	0.466$\pm$0.007	0.514$\pm$0.009	0.516$\pm$0.009	0.560$\pm$0.009
Micro-turns+OS+AD	0.475$\pm$0.009	0.507$\pm$0.007	0.519$\pm$0.006	0.546$\pm$0.006
Micro-turns+OS+OC	**0.503**$\pm$0.007	**0.526**$\pm$0.008	**0.554**$\pm$0.009	**0.573**$\pm$0.008

turns+OS+AD, which additionally lets the model resolve curated-set overflow explicitly; and **Micro-turns+OS+OC**, which replaces tag-based overflow with a model-ordered curated list. AD and OC are exclusive curation-management variants in this paper. To keep the OS comparison conservative, we summarize the documents that the retrieval tool already returned under the Harness-1 retrieval budget. We do not increase the number of retrieved documents simply because summaries would allow more evidence to fit in later turns (Appendix D.4 quantifies how many raw chunks versus summaries fit the same per-call render budget).

Metrics. We use Harness-1 terminology throughout. *Recall* and *FAR* measure gold+evidence and gold-only coverage, respectively, in the final curated set – what the agent actually returns; *Traj. Recall* and *Traj. FAR* measure the same over the trajectory pool (all documents returned by retrieval tools, including chunks never rendered to the model), a diagnostic upper bound before state-management and curation losses. On the non-BC+ benchmarks, the released relevant-document and final-answer-document sets coincide, so Recall $\equiv$ FAR and Traj. Recall $\equiv$ Traj. FAR; we report the full four-metric set only on BC+, where gold and evidence annotations differ, and omit the duplicate FAR columns elsewhere. Recall@n over the top-n curated documents measures prefix quality, the key metric for ordered curation since downstream consumers may not inspect all 30 curated documents. For efficiency, we report attention operations and processed tokens over the full episode (main turns plus micro-turns), a main-turn-only token measure isolating the persistent-trajectory reduction, and peak per-call context as the memory measure; episode totals charge each variant for its own micro-turn calls, which is where costs diverge.

Recall takes priority over trajectory coverage. Recall and FAR measure the curated set the agent actually returns and decide which harness is better; Traj. Recall and Traj. FAR only bound that coverage from above and cannot compensate for a lower final Recall.

4.2 Main Results

We present our results on how micro-turns improve recall performance, reduce compute and memory cost, and affect the quality of top-n truncated curated set recall and accuracy when consumed by a downstream answering agent.

Recall improvements. Micro-turns improve the final curated set under the same model and retrieval backend. In BC+ (Table 1; Figure 2 visualizes the Recall and Traj. Recall columns), OS raises Recall from 0.428 to 0.466 and FAR from 0.461 to 0.516. AD gives a further small improvement in Recall, consistent with the goal of letting the model resolve overflow rather than relying on automatic tag-based eviction. OC gives the strongest results, reaching 0.503 Recall and 0.554 FAR, and also the best trajectory-pool diagnostics.

Table 2 reports Recall and Traj. Recall on the remaining benchmarks, and the cross-benchmark pattern supports the same interpretation. Patents shows the clearest progression, with OS improving over Harness-1, AD improving further, and OC performing best. On the noisier Web benchmark, Harness-1 attains the highest Traj. Recall but the lowest final Recall of the four variants: it discovers more gold+evidence than it retains. OC's final Recall is the highest nonetheless, despite a lower Traj. Recall than Harness-1, which is only possible if OC keeps a larger share of a smaller discovered pool – better selection, not broader discovery, is the source of its advantage here. WebSimple and SECSimple are near ceiling, so the small differences there mainly indicate that the micro-turn variants do not disrupt easy retrieval settings. The Harness-1 trajectory-to-final Recall gap is only 0.003 on WebSimple and 0.008 on SECSimple (Table 2), leaving little headroom for a selection mechanism to improve on either benchmark; Appendix C isolates this same headroom effect in the BC+ progression.

Table 2: Retrieval results on the non-BC+ benchmarks. Entries are five-seed mean$\pm$population-std. Traj. denotes trajectory recall; bold marks the best mean in each column.

	Patents		Web		WebSimple		SECSimple	
Harness	Recall	Traj.	Recall	Traj.	Recall	Traj.	Recall	Traj.
Harness-1	0.544$\pm$0.037	0.731$\pm$0.035	0.470$\pm$0.010	**0.558**$\pm$0.014	**0.861**$\pm$0.007	0.864$\pm$0.005	0.928$\pm$0.015	0.936$\pm$0.011
OS	0.647$\pm$0.023	0.698$\pm$0.010	0.477$\pm$0.009	0.533$\pm$0.015	**0.861**$\pm$0.011	0.864$\pm$0.005	0.929$\pm$0.024	0.932$\pm$0.021
OS+AD	0.677$\pm$0.030	0.727$\pm$0.020	0.478$\pm$0.021	0.524$\pm$0.022	**0.861**$\pm$0.007	0.864$\pm$0.005	0.926$\pm$0.025	0.936$\pm$0.010
OS+OC	**0.724**$\pm$0.011	**0.765**$\pm$0.016	**0.509**$\pm$0.008	0.538$\pm$0.008	**0.861**$\pm$0.011	**0.867**$\pm$0.000	**0.930**$\pm$0.019	**0.948**$\pm$0.019

Compute and memory savings. Micro-turns add model calls, so their efficiency must be evaluated at the full episode level rather than by looking only at shorter main-turn prompts. Table 3 includes both main turns and micro-turns. Micro-turns reuse the triggering main turn's cached prefix, whereas after the `RECENT_K` window starts sliding, the updated state summary forces main turns to prefill their rendered context again. The token totals and attention operations measure episode-level inference work; the peak-context row below them measures memory separately. OS and AD reduce the total processed tokens on every benchmark, with OS ranging from $0.58\times$ to $0.77\times$ of Harness-1 and AD from $0.60\times$ to $0.82\times$. The attention-operation reductions are larger on most benchmarks, because replacing long verbatim chunks with summaries shortens the repeated prefill contexts. The second value in each token entry shows that all micro-turn variants substantially reduce the main-turn rendered context across an episode.

OC trades some efficiency for finer curation. Its main-turn token use falls on every benchmark, but comparison calls raise episode cost. BC+ and Web remain below Harness-1 on both total measures. Patents reaches $1.21\times$ on both, while WebSimple and SECSimple exceed the baseline only in total tokens. Patents combines large chunks with long summaries, making its comparison prompts particularly costly. OC thus spends additional inference on ordering while retaining the main-turn savings from compact retrieval context. Peak per-call context – the largest single-call prompt-plus-decode length in an episode, and the primary driver of KV-cache memory rather than cumulative inference work – falls between $0.55\times$ and $0.79\times$ Harness-1 for every variant on every dataset, including OC on Patents. There, comparator calls raise total tokens and attention operations to $1.21\times$, but do not raise the episode's largest single-call context: Harness-1's renderer, a working-state summary plus a verbatim recent-turn window, is what sets peak context, and every micro-turn variant shrinks exactly that regardless of how many auxiliary calls it issues. Appendix F.3 gives the underlying cached/fresh/output token decomposition, including a per-dataset breakdown usable for cost estimation under commercial inference pricing.

Table 3: Efficiency relative to Harness-1. Token pairs give normalized all-turn/main-turn totals; attention includes all turns; peak context is the largest single-call context (prompt+decode) in the episode, the KV-memory driver (Appendix F.3). † marks costs larger than the baseline.

Harness	BC+	Patents	Web	WebSimple	SECSimple
Processed tokens (all turns / main turns)					
OS	0.58/0.52	0.77/0.65	0.63/0.53	0.69/0.59	0.72/0.63
OS+AD	0.60/0.52	0.82/0.68	0.67/0.55	0.71/0.59	0.79/0.69
OS+OC	0.76/0.49	1.21†/0.73	0.91/0.52	1.02†/0.69	1.13†/0.88
Episode attention operations					
OS	0.37	0.65	0.45	0.46	0.53
OS+AD	0.40	0.72	0.50	0.53	0.59
OS+OC	0.54	1.21†	0.74	0.75	0.83
Peak per-call context (memory)					
OS	0.59	0.73	0.60	0.71	0.74
OS+AD	0.59	0.74	0.61	0.72	0.75
OS+OC	0.55	0.72	0.58	0.74	0.79

Top-n curated-set quality. If the final consumer can only inspect the first n curated documents, a fully ordered list should place useful evidence earlier than one sorted only by coarse importance tags. Figure 3 plots Gold+Evidence Recall@n for $n = 1, \ldots, 30$, using the stored curated order for OS+OC and the harness's importance-tier order for Harness-1, OS, and OS+AD. Where the curated set holds enough relevant and distractor evidence for ordering to matter, OC lifts the entire prefix curve, most sharply at small n: at $n = 10$ it leads Harness-1 by 13.6 points in BC+ (0.475 vs. 0.339), by 24.0 points in Patents (0.683 vs. 0.443), and by 10.3 points in Web (0.475 vs. 0.372), as shown in the figure. OS+AD sits between the two at the same depth (0.375, 0.558, 0.400), so most of OC's lead comes from ordering rather than from AD's overflow handling alone. These gains exceed the corresponding full-set gaps at $n = 30$, the pattern expected if OC reorders the curated evidence rather than merely finding more of it.

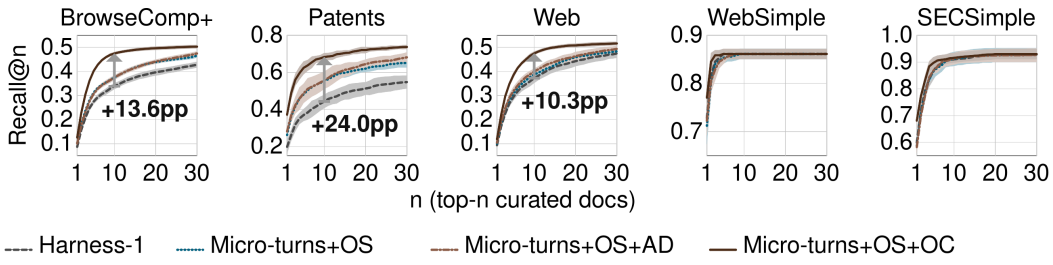

Figure 3: Recall@n over curated-set prefixes. OC uses its stored order; Harness-1, OS, and OS+AD use importance-tier order. Bands show one population standard deviation across five seeds.

OS and OS+AD are much closer to each other throughout the Recall@n curves. This follows their design because OS changes what evidence is exposed and preserved, while AD restores model agency during overflow, but still leaves the final set organized by coarse tags. OC adds the missing ordering operation, so overflow is handled by dropping the tail of a model-ordered list, and downstream consumers receive a stronger prefix. In Web, the OC advantage is visible in small n but is mostly close by $n = 30$, suggesting that the same evidence is often eventually present, but is ordered differently. WebSimple and SECSimple are near saturation for all variants, so the curves mainly serve as sanity checks. When the task already fits easily into the curated prefix, ordering has little room to improve recall.

Downstream answering performance. We evaluate the quality of the curated set in a separate stage where the answerer receives only the query and curated documents, not the search trajectory, isolating the quality of the returned state from any benefit of the full interaction history.

For each harness variant, we truncate the curated-set order used above (OC's stored order; importance-tier order for Harness-1, OS, and AD) to the first n documents and give Qwen3.5-9B the query and the stored chunk text of those documents, exactly as the retrieval agent saw it; a separate judge call, using the same model, scores the answer against the gold string. We evaluate on a 150-query fixed-seed subsample of the 631-query BC+ set, restricted to queries with a curated set available in every harness–seed cell so the comparison is paired, over the same 5 seeds as the main results.

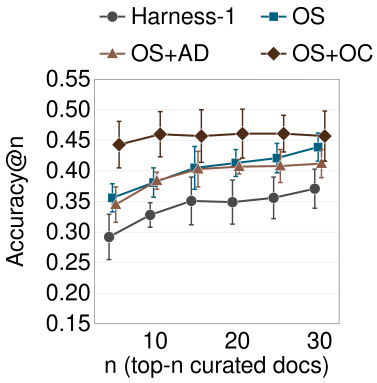

Figure 4: Answer accuracy@n on BC+ with Qwen3.5-9B as the answering agent.

Figure 4 shows the same three-way separation as the retrieval curves, now in answer accuracy. OC's margin is largest at small n: at $n = 10$ it leads Harness-1 by +13.2pp (0.460 vs. 0.328). OC is also nearly flat across the whole range, dropping only 1.4 points from $n = 30$ to $n = 5$ (0.457 to 0.443), so truncating its prefix costs the answerer almost nothing. OS, OS+AD, and Harness-1 all fall off substantially more as the prefix shrinks (-8.3pp, -6.7pp, and -7.9pp from $n = 30$ to $n = 5$, respectively, vs. only -1.4pp for OC): accuracy under a truncated prefix depends on the order of what little evidence survives, not just how much of it does.

5 Conclusion

We introduced micro-turns, short harness-injected calls that let the model operating in long sessions handle context and state management without lengthening the main trajectory. Instantiated in the setting of multi-hop retrieval as online summarization, active decuration, and ordered curation, micro-turns improve final curated recall and curated-prefix quality across BrowseCompPlus and other benchmarks, reduce persistent-context and peak-memory cost, and let ordered curation add the strongest ranking gains, including on downstream answer accuracy, at a modest, dataset-dependent comparison cost, establishing them as a practical harness primitive for retrieval agents managing context and state.

AI USE STATEMENT

In this work, we used generative AI tools to write and polish the paper's content. We did not use generative AI tools for research ideation or for developing the paper's structure. We also used generative AI tools to generate code for our experiments. We reviewed and tested all AI-assisted work, including LLM-generated code, citations, and numerical values, for correctness. We take full responsibility for the final content of this work, including any text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Hammad Bashir, Kelly Hong, Patrick Jiang, and Zhiyi Shi. Chroma Context-1: Training a self-editing search agent. Technical report, Chroma, 2026. URL https://trychroma.com/research/context-1.

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, 2026. URL https://aclanthology.org/2026.acl-long.1023/.

Pengcheng Jiang, Jiacheng Lin, Lang Cao, Runchu Tian, SeongKu Kang, Zifeng Wang, Jimeng Sun, and Jiawei Han. DeepRetrieval: Hacking real search engines and retrievers with large language models via reinforcement learning. In *Conference on Language Modeling*, 2025a. URL https://openreview.net/forum?id=u9JXu4L17I.

Pengcheng Jiang, Xueqiang Xu, Jiacheng Lin, Jinfeng Xiao, Zifeng Wang, Jimeng Sun, and Jiawei Han. s3: You don't need that much data to train a search agent via RL. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 21599–21617, 2025b. URL https://aclanthology.org/2025.emnlp-main.1095/.

Pengcheng Jiang, Zhiyi Shi, Kelly Hong, Xueqiang Xu, Jiashuo Sun, Jimeng Sun, Hammad Bashir, and Jiawei Han. Harness-1: Reinforcement learning for search agents with state-externalizing harnesses. *arXiv preprint arXiv:2606.02373*, 2026.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Rwhi91ideu.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-Harness: End-to-end optimization of model harnesses. In *Conference on Language Modeling*, 2026. URL https://arxiv.org/abs/2603.28052.

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 *Advances in Neural Information Processing Systems*, volume 33, 2020.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5420–5438, 2025. URL https://aclanthology.org/2025.emnlp-main.276/.

Xinghua Lou, Miguel Lázaro-Gredilla, Antoine Dedieu, Carter Wendelken, Wolfgang Lehrach, and Kevin P. Murphy. AutoHarness: Improving LLM agents by automatically synthesizing a code harness, 2026. URL https://arxiv.org/abs/2603.03329. ICLR Workshop on Recursive Self-Improvement.

Miao Lu, Weiwei Sun, Weihua Du, Zhan Ling, Xuesong Yao, Kang Liu, and Jiecao Chen. Beyond the context window: Scaling agentic RL via end-to-end optimized context compression. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pp. 21074–21125, 2026. URL https://aclanthology.org/2026.acl-long.966/.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems, 2024. URL https://arxiv.org/abs/2310.08560.

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. URL https://aclanthology.org/2023.findings-emnlp.378/.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. Large language models are effective text rankers with pairwise ranking prompting. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 1504–1518, 2024. URL https://aclanthology.org/2024.findings-naacl.97/.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

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 *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pp. 10014–10037, 2023.

Xixi Wu, Kuan Li, Yida Zhao, Liwen Zhang, Litu Ou, Huifeng Yin, Zhongwang Zhang, Xinmiao Yu, Dingchu Zhang, Yong Jiang, Pengjun Xie, Fei Huang, Minhao Cheng, Shuai Wang, Hong Cheng, and Jingren Zhou. ReSum: Unlocking long-horizon search intelligence via context summarization, 2026. URL https://arxiv.org/abs/2509.13313.

Le Yan, Zhen Qin, Honglei Zhuang, Rolf Jagerman, Xuanhui Wang, Michael Bendersky, and Harrie Oosterhuis. Consolidating ranking and relevance predictions of large language models through post-processing. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 410–423, 2024. URL https://aclanthology.org/2024.emnlp-main.25/.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. 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 Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Chanwoong Yoon, Taewhoo Lee, Hyeon Hwang, Minbyul Jeong, and Jaewoo Kang. CompAct: Compressing retrieved documents actively for question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 21424–21439, 2024. URL https://aclanthology.org/2024.emnlp-main.1194/.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pp. 21457–21483, 2026. URL https://aclanthology.org/2026.acl-long.981/.

Alex L. Zhang, Tim Kraska, and Omar Khattab. Recursive language models. *arXiv preprint arXiv:2512.24601*, 2025. URL https://arxiv.org/abs/2512.24601.

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. *arXiv preprint arXiv:2606.00408*, 2026. URL https://arxiv.org/abs/2606.00408.

A Related Work

Iterative retrieval and search agents. Methods such as Self-Ask, IRCoT, and ReAct interleave reasoning with retrieval or other tool use rather than relying on a single retrieval pass (Press et al., 2023; Trivedi et al., 2023; Yao et al., 2023). More recent work trains search behavior directly. Search-R1 and DeepRetrieval use reinforcement learning to improve the model's decisions about when and how to search (Jin et al., 2025; Jiang et al., 2025a). These methods primarily study the search policy within a given interface. A related line separates retrieval from answer generation. The s3 framework trains a searcher separately from its downstream generator (Jiang et al., 2025b), while Context-1 and Harness-1 construct evidence sets for downstream use (Bashir et al., 2026; Jiang et al., 2026). In our comparison with Harness-1, we keep the main model and retrieval backend fixed while changing how the harness presents evidence and manages curated state.

Harness design. The environment around an agent can substantially affect its behavior even without changing the underlying model. SWE-agent illustrates the importance of the agent-computer interface in software engineering, while AutoHarness and Meta-Harness optimize harness designs themselves (Yang et al., 2024; Lou et al., 2026; Lee et al., 2026). Harness-1 takes a stateful approach to retrieval, maintaining a document pool and curated set outside the model's transcript (Jiang et al., 2026). Our work keeps that stateful substrate and studies a different design choice within it. The harness still schedules and applies state transitions, but it asks a model to make selected content-dependent management decisions that were previously resolved by fixed rules.

Managing retrieved context and agent state. Search-o1 processes retrieved documents before passing refined information to an answering model, while CompAct actively compresses retrieved documents for multi-hop question answering (Li et al., 2025; Yoon et al., 2024). These works are closest to the document-level compression in our online summarization, but do not address management of a retrieval agent's curated evidence set. At the trajectory level, ReSum condenses accumulated search history, and SUPO periodically replaces the working tool-use history with a model-generated summary for subsequent turns (Wu et al., 2026; Lu et al., 2026). The latter also trains summarization jointly with tool use. Our micro-turns instead produce scoped updates without resetting the main trajectory. Observation masking offers another way to reduce search context, although removing old observations can also discard useful evidence (Zhang et al., 2026).

Other work manages information outside or alongside the working context. MemGPT moves information between a limited main context and external memory, and Recursive Language Models let a model inspect externally stored prompts through further model calls (Packer et al., 2024; Zhang et al., 2025). AgeMem exposes storage, retrieval, summarization, and deletion as actions in the agent's policy (Yu et al., 2026); Context-1 likewise allows a retrieval agent to prune its search context (Bashir et al., 2026). In contrast, our harness triggers management calls at specified points, applies their updates to retrieval context or curated state, and does not retain the auxiliary exchange in the main trajectory.

Ranking curated evidence. Pairwise Ranking Prompting uses relative comparisons to rank retrieved documents, and related work finds that pairwise preferences can improve on directly elicited relevance labels (Qin et al., 2024; Yan et al., 2024). We use pairwise judgments for a different role: maintaining a bounded curated set whose order both determines overflow eviction and exposes a

useful ranked prefix to downstream consumers. We assume the resulting comparisons define a total order, rather than studying inconsistencies or calibration in the comparisons themselves.

B Importance Tags and the Motivation for Ordered Curation

Harness-1 uses importance tags to prioritize a bounded curated set, and AD retains these tags while letting the model choose removals at overflow. We examine whether existing tags are reconsidered and how well each tier distinguishes annotated evidence. These analyses characterize a limitation that OC addresses with relative ordering; they do not directly measure how long any individual document occupies capacity.

B.1 Reconsidering Existing Tags

We replay `curate` calls in Harness-1 and OS+AD trajectories. A *re-tag submission* adds an already-curated document with an importance value, whether or not that value changes. An *evidence-free revision* is a submission that changes the tag without a new retrieved or inspected chunk of that document since its previous tag assignment. This distinction matters because a new chunk can legitimately change the model's assessment; the second event measures reconsideration without such new evidence. Table 4 reports the share of re-tag submissions that produce such revisions.

Table 4: Evidence-free tag revisions as a percentage of re-tag submissions. The denominator includes submissions that repeat the existing tag. Percentages are computed from the ratio of the corresponding five-seed mean event rates and rounded to one decimal place.

Benchmark	Harness-1	OS+AD
BrowseComp+	1.9%	1.9%
Web	1.9%	1.4%
Patents	3.8%	3.0%
WebSimple	2.6%	1.6%
SECSimple	3.3%	1.8%

On BrowseComp+, Harness-1 produces 837 evidence-free revisions among 43,530 re-tag submissions (1.9%). Most submissions restate the current tag: 27,504 are unchanged, while 10,518 downgrade and 5,508 upgrade the document. Thus, tags do change, often after new evidence, but revisions made solely to reconsider an existing document are uncommon. The percentages in Table 4 show the same qualitative pattern across benchmarks and under OS+AD. This analysis does not condition on curated-set overflow and does not count explicit removals, so it should not be read as a direct estimate of how often a high-tagged document blocks a new candidate. It does show that the tag interface seldom elicits a change in relative priority absent new evidence. OC does not revise tags. Its binary-insertion procedure compares each new document with the ordered set, allowing a candidate to change the retained ranking without an explicit reconsideration of an old importance label. The two interfaces therefore cannot be compared by counting tag revisions. Figure 3 instead evaluates the quality of their resulting ranked prefixes.

B.2 Precision Within Importance Tiers

For each importance tier, we measure the fraction of its final curated documents that belong to the annotated gold-plus-evidence set. On benchmarks other than BrowseComp+, the available annotations make this the gold-document set. Table 5 compares Harness-1 with OS+AD. OC has no tags and is therefore not included. Values are means across five seeds.

Table 5: Final curated-set precision by importance tag. Each entry is the five-seed mean fraction of documents in that tier matching the annotated gold-plus-evidence set. OS+AD is shown as a complete variant, not as an isolated estimate of AD's effect.

Benchmark	Harness	Very high	High	Fair	Low
BrowseComp+	Harness-1	0.942	0.476	0.045	0.011
	OS+AD	0.932	0.507	0.053	0.018
Web	Harness-1	0.795	0.387	0.044	0.010
	OS+AD	0.771	0.438	0.056	0.014
Patents	Harness-1	0.342	0.061	0.007	0.002
	OS+AD	0.314	0.067	0.008	0.002
WebSimple	Harness-1	0.747	0.368	0.015	0.001
	OS+AD	0.789	0.227	0.025	0.009
SECSimple	Harness-1	0.851	0.471	0.118	0.033
	OS+AD	0.828	0.436	0.113	0.050

The tags carry useful signal, with precision rising sharply from low to very high on every benchmark. Yet they do not provide a complete ordering. On BrowseComp+, for example, Harness-1's high tier has precision 0.476, so many documents in that tier do not match the annotated evidence set. The comparison with OS+AD shows no consistent improvement in final tag precision, although it cannot isolate AD from the accompanying OS change. AD restores model choice over removals, but retains the coarse tiers rather than recalibrating the tags. A high-tagged document that is not useful for the current query can therefore remain ahead of a more useful candidate unless the model explicitly removes or retags it. OC instead compares new documents with those already retained and assigns them a position in the same ordered set. This gives both overflow eviction and a truncated downstream prefix a finer ranking signal, consistent with the Recall@n gains in Figure 3. The precision and revision studies motivate this mechanism, but neither alone proves that persistent high-tagged documents cause those gains.

C Why the Final Curated Set Progression Separates: Fixed Exposure, Better Selection

The final curated-set progression on BrowseComp+ (Table 1) could in principle be produced by three different mechanisms: the micro-turn variants might simply discover more evidence, they might discard less of what they discover, or they might select better at the same discovery level. This appendix separates the three mechanisms using the paired trajectory and final-set metrics, and shows that the progression is driven by selection quality under an essentially fixed exposure budget.

We decompose the gold-only progression (FAR) using its two diagnostic counterparts. Traj. FAR measures gold coverage over the trajectory pool, i.e., everything the agent ever saw a hit for; the gap between Traj. FAR and FAR measures how much gold evidence was discovered but then discarded between pool and final curated set. Table 6 reports both quantities on BC+.

Table 6: Mechanism decomposition of the BC+ FAR progression. FAR is the gold-only final curated recall; Traj. FAR is the same quantity over the trajectory pool. The trajectory column is nearly flat across variants while the pool-to-final discard gap is constant, so the FAR movement is attributable to better selection rather than more discovery or less discard. Per-episode values over the 5×631 matrix.

Harness	FAR (final)	Traj. FAR (pool)	Discard (gap)
Harness-1	0.461	0.544	0.083
Micro-turns+OS	0.516	0.560	0.044
Micro-turns+OS+AD	0.519	0.546	0.027
Micro-turns+OS+OC	**0.554**	**0.573**	**0.019**

Three observations pin down the mechanism:

Discovery is not the driver. Trajectory-pool coverage moves by at most 2.9 points from Harness-1 to OC (0.544 to 0.573) while final FAR moves by 9.3 points (0.461 to 0.554). If the micro-turn

variants worked by surfacing more evidence, the trajectory diagnostic would move first; it barely moves. This is also visible in the action profile: micro-turn episodes issue slightly fewer main actions than Harness-1 (mean 32.6 vs. 36.2 generations per episode on BC+), so they explore marginally less, yet retain more.

Discard does not close the gap. The pool-to-final discard gap (Traj. FAR − FAR) is between 1.9 and 8.3 points and does not mirror the FAR progression across variants: OS discards 4.4, AD 2.7, and OC 1.9 points of trajectory-pool coverage. Discard accounting is directionally consistent with the top of the progression – OC retains the largest fraction of what it found – but discard differences across variants are far smaller than the 9.3-point FAR spread, and OS and AD invert this relationship (AD discards less than OS yet ties it on final FAR). Selection quality, not discard avoidance, is the load-bearing mechanism.

Same pool, better picks. Because trajectory coverage is nearly flat and discard is small and near-constant, the FAR differences reflect which documents the variants retain from a comparable discovery set. This matches the per-dataset pattern: the identity between relevant and final-answer annotations outside BC+ means pool and trajectory metrics coincide there, so the large non-BC+ gains are selection effects. On Patents (per-episode values), final recall rises from 0.548 (Harness-1) to 0.651 (OS), 0.681 (AD), and 0.737 (OC) with trajectory recall flat-to-higher (0.735, 0.701, 0.731, 0.779); the OC gain is produced almost entirely inside the curated-set update, not by retrieving anything the baseline could not see. On Web, OC posts the *lowest* trajectory recall of the four variants (0.538 vs. 0.558 for Harness-1, Table 2) while holding the highest final recall (0.509) – the cleanest version of the selection-not-discovery story.

Saturation boundaries. The selection mechanism also predicts where the gaps between variants should close. WebSimple and SECSimple sit at 0.86 and 0.93 final recall with Recall@1 already at 0.71–0.77 and Recall@5 $\approx$ 0.86–0.90; there is no headroom for a selection-quality difference to express itself, and the four variants tie accordingly. Where headroom exists (BC+, Web, Patents), the ordering gains appear at every prefix depth (Section 4.2).

D Exposure and Rendering Anatomy

This appendix quantifies what the agent actually observes under each harness variant, at three layers: the summarization transform, the retrieval layer, and the rendering layer. Together these pin down which mechanism produces the efficiency gains of Table 3 and clarify what ordered curation does and does not change.

D.1 The Summarization Transform

Summaries are drastic compressions, not paraphrases at similar length. Across the 1.64M summarize micro-turns in the final evaluation matrix, the mean retrieved chunk carries 2,455 characters of pre-truncation text while the mean generated summary is 483 characters, a 5.1× character-level compression at the observation layer (before accounting for the removal of chunk formatting and headers). Replacement coverage is essentially total: the observation slot is replaced by the summary in all but two of the 1,637,320 summarize turns; the exceptions are episodes in which a summary came back empty and the harness conservatively kept the raw chunk text rather than replacing the observation with nothing.

This transform is the compactness mechanism underlying the efficiency table. Under Harness-1, each retrieval observation carries verbatim chunk text, so a `search_corpus` response alone fills a large fraction of the render budget. Under the micro-turn harness, the same response renders as summaries whose total length is a small fraction of the raw text, which is why main-turn prompt lengths drop from a mean of 14,628 raw prompt tokens per generation on BC+ (Harness-1) to 8,168 (OS), and why the persistent-trajectory columns of Table 3 fall as far as they do.

D.2 The Retrieval Layer Is Harness-Invariant

Micro-turn harnesses do not change what the retrieval backend returns. On BC+, a `search_corpus` call returns a mean of 5.2 documents and ∼14.7K characters of raw chunk text for every variant (5.21–5.24 docs, 14,683–14,756 chars across Harness-1, OS, AD, and OC), and

a `fan_out_search` call returns 18.3 documents (18.26 Harness-1, 18.34 OS) with pre-truncation observations averaging ∼53K characters. Retrieval budgets, corpus indices, and query construction are identical; the harness variants differ only in what happens to the returned text.

D.3 Total Exposure Falls, Retention Rises

Although retrieval yield per call is invariant, the total number of documents entering an episode's trajectory falls under the micro-turn harness: on BC+ the mean number of documents returned by retrieval tool calls per episode drops from 117 (Harness-1) to 93 (OS), on Patents from 193 to 149, and on Web from 162 to 131. This drop is a policy effect, not a retrieval-budget effect: decomposing the BC+ gap, roughly half comes from fewer search actions per episode (consistent with shorter episodes) and the rest from slightly lower new-document yield per action, because the `ignore_ids` mechanism skips documents already in the pool and the pool saturates sooner when summaries keep more of it resident.

At the same time, curated-set retention of what was seen rises: on BC+, the final curated set holds 28.6% of returned documents under OS versus 24.1% under Harness-1. The micro-turn variants see less and keep proportionally more of it – the same quality-over-quantity signature as the selection mechanism of Appendix C.

D.4 Rendering: Summaries Fit, Raw Chunks Do Not

The per-call render budget (which is approximately 15K chars per tool response, split across sub-queries for `fan_out_search`) is the bottleneck that OS relaxes. Under Harness-1, a fan-out response renders raw chunk blocks of roughly 1,200–2,200 characters (depending on dataset), so the per-subquery budget of ∼4,100–5,000 chars shows only 1.7–3.8 raw chunks out of the ∼5–9 each subquery retrieved; the remainder enters the pool unrendered. Under OS, every retrieved chunk is rendered as a ∼420–560-character summary, and on four of five datasets the full retrieved set fits the same budget as summaries; only on Patents, where summaries are longest (562 chars, patent text being dense), does the rendered count fall below the retrieved count (∼24 of ∼33 blocks per call). The efficiency gains (Table 3) follow precisely due the change in rendering (summaries instead of raw chunks) supported quantitatively: mean raw prompt/input tokens per main turn fall from 14,628 to 8,168 (BC+), 15,034 to 8,933 (Web), and 13,811 to 9,994 (Patents).

D.5 What Ordered Curation Adds: Order, Not Exposure

Because OS+AD and OS+OC render identically to OS only (all three summarize), and OC's comparator changes no retrieval and no rendering capacity, the Recall@n gains of Section 4.2 cannot come from additional exposure. They come from the ordering OC imposes on the same underlying content: OC's stored order is strictly better than harness importance-tier order at every depth on BC+ (Recall@10 of 0.475 vs. 0.375 for both OS and AD), with the full-set Recall@30 values close (0.503 vs. 0.475/0.466). The fan-out usage profile is also nearly identical across variants (fan-out share of search actions on BC+: 12.6% Harness-1 vs. 10.5% OS), so OC's gain cannot be attributed to different search strategy either. Ordered curation is the re-ranking of fixed evidence, exactly as the design in Section 3.2 intends and the improvements are directly attributable to this model-driven ordering.

E Curation Overflow Accounting

This appendix quantifies how often each harness variant operates at the curated-set cap and how each resolves overflow, completing the picture of the curation failure mode described in Section 2.3.

Cap-hit rates. The fraction of episodes whose final curated set sits at the 30-document cap separates the harness variants by construction. Harness-1 hits the cap on 81–91% of episodes on the three long-horizon datasets (BC+ 86%, Web 81%, Patents 91%), reflecting its fixed tag-based eviction rule under sustained curation pressure. The micro-turn variants hit the cap less often (OS 67–75%, AD 68–80%, OC 64–79% on the same datasets) as their curation is more selective and they keep exploring. On the two ceiling datasets cap hits are rare for all variants (6–16%), consistent with small relevant-set tasks that never fill the curated set.

Overflow pressure under AD. When AD fires, the curated set is genuinely over capacity: across the final matrix, the 33,740 decuration micro-turns trigger at a mean curated count of 34.2–36.2 (by dataset; up to six documents above the cap on the busiest episodes), i.e., insertions typically arrive 4–6 documents above the cap before the model trims. Decuration covers 78% (BC+), 81% (Patents), and 69% (Web) of episodes, and is correspondingly rare on the saturated datasets (WebSimple, SECSimple).

OC never violates the cap. Under OC, overflow is resolved by construction. The ordered curated list is truncated to the cap after every insertion, so no OC episode ever exceeds 30 curated documents, 67% of OC episodes end exactly at the cap, and OC's automatic (non-comparator) curation path never fires. This directly verifies the design claim that overflow handling "comes for free" once the ordered state exists. The price is the comparator traffic quantified in Appendix F: OC issues a mean of roughly 484 pairwise comparisons per BC+ episode, which is where its episode-total cost above Harness-1 originates (for example, in Patents).

OS without nudges. Plain OS ends episodes with a smaller final curated set than Harness-1 (~ 26.7 vs. ~ 28.1 mean size on BC+) without any overflow-resolution mechanism; its cap-hit reduction reflects lower insertion pressure rather than active trimming.

F How the Efficiency Numbers Are Computed

This appendix specifies the token and attention-operation accounting behind Table 3, so that the reported ratios can be reproduced from per-episode trajectories. All quantities are per-episode sums over generation calls, averaged over the 5-seed matrix and normalized to Harness-1 on the same dataset.

F.1 Per-Turn Accounting Model

Every generation call is decomposed into three token counts: K (prompt tokens already resident in the KV cache and therefore not re-prefilled), F (fresh prompt tokens that must be prefilled), and O (decode/output tokens). Accounting then uses:

$$\text{prefill ops} = F \cdot K + \frac{F(F+1)}{2}, \tag{1}$$

$$\text{decode ops} = O\,(K + F) + \frac{O(O-1)}{2}, \tag{2}$$

i.e., standard causal-attention operation counts with the cache-hit prefix charged once. "Processed tokens" sums F and O; "attention ops" applies the formulas above.

Main turns. Harness-1's renderer repeats a working-state summary plus a recent window of up to $\texttt{RECENT_K} = 5$ turns, and rewrites the summary as state changes, so the prompt prefix is byte-stable only within a short horizon. We model this as: turn 0 has $K = 0$; turns 1–4 have K equal to the previous main turn's full prompt length (automatic prefix caching hit); turns ≥ 5 have $K = 0$, because the sliding window shifts token positions and the rewritten state summary breaks the prefix, forcing a full re-prefill. Fresh and decode token counts are the recorded per-turn values from the trajectories.

Micro-turns (summarize and decurate). Each micro-turn prompt is built from the triggering main turn's prompt IDs plus a short final user segment, so the prefix through that segment is already cached: K equals the micro-turn's full prompt length minus the tokenized fresh final segment (tokenized exactly with the Qwen3.5-4B tokenizer), and F is just the fresh segment – a mean of only $\sim$255 tokens for summarize calls against a $\sim$9,200-token full prompt. Summing raw prompt lengths instead would inflate reported summarize-turn token volume by an order of magnitude or more ($\sim 36\times$ per call on BC+).

Compare turns (OC). Comparator prompts reuse the same prefix structure: K is modeled as the triggering main turn's prompt-plus-decode length, validated to within 0.03% against exact tokenization on sampled turns. Within one insertion, the comparator holds the new document's summary fixed as Summary A across every comparison and places Summary A before Summary B, so after the first comparison of a given (curate step, Summary A) pair, the prefix through "Summary B:" is also cache-stable; we credit that prefix to K for all subsequent comparisons in the group. The number of comparisons itself follows the design bound $O(k \log n)$ with $n \leq 30$.

F.2 What the Columns Measure

Attention ops (column 2) is fully cache-adjusted. Both axes of the attentionOps column apply the (K, F, O) model to main and micro turns alike, so it measures actual quadratic-attention work including cache hits.

Processed tokens (column 1) is a hybrid measure. For main turns it counts raw prompt tokens (the full rendering the harness assembles each turn, whether or not cacheable); for micro-turns it counts only the fresh segment, and decode counts are added for all calls. The main-turn convention is deliberate: the quantity of interest is the size of the persistent per-turn context, which is what micro-turns shrink. Nonetheless the two turn classes are not charged on identical bases, and Table 3 should be read with that in mind. Fully cache-adjusting the main-turn token counts would tighten the OS/AD ratios slightly (e.g., BC+ OS $0.58\times \rightarrow 0.57\times$) but does not change any qualitative conclusion.

Main-turn tokens (column 3). This column isolates the persistent trajectory effect: it counts only main-turn processed tokens, normalized to Harness-1. Because every micro-turn variant shortens what later main turns must render, this column is below $1.00\times$ everywhere, and is the cleanest single-number summary of the "smaller persistent state" mechanism.

F.3 Peak Context and the Cached/Fresh/Output Decomposition

Every generation call's (K, F, O) triple (Appendix F.1) also determines two quantities not shown in Table 3's first two blocks: peak context and the cached/fresh/output split.

Peak context. For each episode we take $\max_t(K_t + F_t + O_t)$ over its main turns – the largest single-call context the serving stack must hold in memory at once. This differs from the processed-tokens columns, which sum F and O across all turns and so measure cumulative inference work rather than standing memory. Table 3's peak-context row reports this quantity, episode-mean, normalized to Harness-1.

Why OC still saves memory where it costs more compute. On Patents, OC's total tokens and attention operations both reach $1.21\times$ Harness-1 (Section 4.2), driven by comparator calls – OC issues a mean of ~ 484 pairwise comparisons per BC+ episode (Appendix E), and each adds fresh compute. But no single comparator call's context exceeds what the OS-based summarized renderer already produces for a main turn, so the episode's largest single-call context is unaffected: peak context falls to $0.72\times$ Harness-1 on Patents, and to 0.55–$0.79\times$ across the full suite for every variant on every dataset. Harness-1's renderer – a working-state summary plus a verbatim recent-turn window – is what sets peak context, and every micro-turn variant shrinks exactly that, independent of how many auxiliary calls it issues.

Cached/fresh/output decomposition. Splitting each call's tokens into cached (K, served from the KV cache), fresh input (F, actually prefilled), and output (O, decoded) separates the two quantities the processed-tokens column conflates. Table 7 reports the fresh-input and output ratios underlying this appendix. Output tokens are higher for every micro-turn variant on every dataset (1.7–$3.0\times$ Harness-1), the honest counterweight to the prefill savings: summarization and comparison calls are additional generations. This decomposition also supports cost estimation under commercial inference pricing, where input (prefill) and output (decode) tokens are typically billed at different per-token rates – combining the fresh-input and output ratios below with a provider's price-per-token gives a direct $-cost estimate, which the single hybrid processed-tokens figure cannot support.

F.4 Modeling Caveats

- **Prefix-cache model is idealized.** The turn-type rules above approximate server-side automatic prefix caching; real hit rates depend on block boundaries, eviction, and request ordering. We conservatively treat turns ≥ 5 as full re-prefills, so reported savings are, if anything, understated relative to a serving stack with more persistent cache reuse.
- **OC's quadratic-insert refinement is not separately modeled.** The comparator count reflects the observed binary-insertion trace; refinement steps that re-compare replaced documents are included in the observed per-episode comparison counts, so no additional charge is missing.

Table 7: Fresh-input and output token ratios normalized to Harness-1, episode means over the 5-seed matrix. Fresh input is prefilled (billed) tokens only, excluding cache-served tokens; output is decoded tokens. † marks costs above baseline.

Harness	BC+	Patents	Web	WebSimple	SECSimple
Fresh input tokens					
OS	0.55	0.73	0.59	0.64	0.69
OS+AD	0.57	0.77	0.62	0.66	0.76
OS+OC	0.74	1.19†	0.88	0.97	1.10†
Output (decode) tokens					
OS	1.68	2.10	1.95	2.47	2.06
OS+AD	2.19	2.52	2.46	2.86	2.25
OS+OC	1.77	2.48	2.08	2.99	2.58

- **Compare-group chunking.** The pod-safe chunked recomputation decides first-vs-later comparisons per (episode, curate step, Summary A) group within each chunk; a group whose rows straddle a chunk boundary is handled by a merge-time recomputation, and residual approximation affects a negligible fraction of groups.
- **Decode-side accounting.** Decode ops charge each new token against the full visible context ($K + F$ plus prior decode), which matches incremental-decoding cost; no speculative-decoding or batching effects are modeled.
- **Retry and error rows.** Format-retry generations (which occur but emit no tool call) are included; episode-level error rows (no generation) are excluded.

G MICRO-TURN PROMPT TEMPLATES

This appendix records the prompt templates used for the three retrieval-specific micro-turns. In all cases, the harness injects the prompt as a user message for a scoped management call. The resulting prompt and response are not appended to the persistent main trajectory; only the induced state update is retained.

G.1 ONLINE SUMMARIZATION

Online summarization is called after retrieval tools such as `search_corpus` or `fan_out_search`. Each returned chunk is summarized independently.

```
You are summarizing a retrieved document chunk as part of an active
multi-turn research session. The full conversation history --
    including all
prior searches, retrieved documents, and your curated set -- is
    visible above.

Research context:
Main query (overall goal): {main_query}
Search query (what was searched): {search_query}

Write a 2-4 sentence factual summary of the document chunk below. Keep
    all
specific names, dates, places, and numbers. Then, in 1 sentence, note
    how this
chunk relates to the research so far -- e.g., whether it confirms,
contradicts, extends, or is redundant with what you have already
    curated.
Do NOT issue any tool calls. /no_think

Wrap your full response in <summary>...</summary> tags.

<tool_response>
{document_chunk}
```

```
</tool_response>
```

G.2 Active Decuration

Active decuration is called only when the curated set overflows its capacity. The prompt renders the lowest-priority tiers that must be considered for trimming, using document summaries rather than full chunks.

```
The full conversation history -- including all prior searches,
    retrieved
documents, and your curated set -- is visible above. Based on the
    conversation
so far, decide which curated documents to remove.

[ACTION REQUIRED] Your curated set has {overflow_curated_list_size}/
{max_curated_list_cap} documents -- {num_docs_over_cap} over the cap.

[STATUS] Only the lowest-importance tiers that need trimming are shown
    below.

Call curate(remove_ids=[...]) to remove exactly {num_docs_over_cap}
document(s) -- no more than necessary -- starting from the lowest
    tier(s) first
(low -> fair -> high -> very_high). Removing more than
    {num_docs_over_cap}
discards evidence you may still need, so do not over-trim. Do NOT
    search again
until the set is within the {max_curated_list_cap}-doc cap.

--- LOW importance tier ---
{doc1_id}: {doc1_summary}
{doc2_id}: {doc2_summary}
...
```

G.3 Ordered Curation

Ordered curation inserts newly curated documents into an ordered curated set by binary insertion. Each comparison asks the model to choose the more relevant of two document summaries. Concretely, each new document is located by bisection over the current ordered list: the harness compares the new document against the list's median element and recurses into the half indicated by the comparison verdict, so a single insertion into a list of size n costs at most $\lceil \log_2(n+1) \rceil$ comparisons, matching the $O(k \log n)$ bound for inserting k new documents (Section 3.2). The comparison prompt below disallows ties, so every comparison strictly resolves the search to one half of the remaining range.

```
You are ranking two document summaries by relevance to an ongoing
    research
task, as part of an active multi-turn research session. The full
    conversation
history -- including all prior searches, retrieved documents, and your
    curated
set -- is visible above. Do NOT issue any tool calls. /no_think

Research context:
Main query (overall goal): {main_query}

Summary A:
{summary_doc_A}

Summary B:
{summary_doc_B}
```

```
Based on the conversation so far, which summary is MORE relevant and
    useful
for answering the main query? You must choose one -- ties are not
    allowed.
Answer with exactly one letter, A or B, wrapped in
    <winner>...</winner> tags.
```