CODE2EVIDENCE: EXPLICIT KNOWLEDGE ORGANIZATION FOR MULTI-AGENT SEARCH AND REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Retrieval agents that search a corpus over many steps keep everything they have done in one interaction history, where the few observations that support the answer sit among irrelevant hits, search errors, failed commands, and the agent's own reasoning traces. This mixture can mislead later search and the final reasoning step, and truncating or summarizing the history can discard supporting evidence along with the noise. We argue that retrieval agents should keep two separate states: a *transient interaction history* that may be compressed freely, and a *persistent evidence state* of source spans, with provenance, that the agent explicitly banks during search. The evidence state is never compressed, guides further search, and is the only input to the final answer. We implement this separation in Code2Evidence, a coding-based retrieval agent that searches raw corpora with shell commands. Holding the model, tools, turn budget, and compression policy fixed, answering from the evidence state matches or beats answering from the compressed history in all ten task–backbone pairs across six Wikipedia QA datasets (MHQA), FinQA, and BioProBench with GPT-5.4-mini and GPT-5.5. With GLM-5.3 on MHQA, adding the evidence state to a single agent raises meaning-match accuracy from 74.3% to 80.7%, and reconciling the evidence states of three role-specialized searchers reaches 85.2%. Code2Evidence is also more accurate than DCI-Agent-Lite on FinQA with all four backbones and on MHQA (85.2% vs. 79.4%), with the largest gains for GPT-5.4-mini on Legal RAG Bench and BioProBench.

1 INTRODUCTION

Retrieval-augmented generation (RAG) gives language models access to external evidence for knowledge-intensive questions (Lewis et al., 2020). Many such questions are answered over several searches rather than one. A financial comparison may depend on values from separate tables, each tied to a reporting period and unit, and all of them must still be available when the arithmetic is done. A multi-hop question may reveal an entity that guides the next search, while the passage that revealed it is still needed to support the final answer. In both cases, the agent must retain what it has found, together with the source context needed to interpret it, across a long and growing search.

Recent methods make retrieval increasingly adaptive. ReAct interleaves reasoning with tool use (Yao et al., 2023), IRCoT couples retrieval to intermediate reasoning (Trivedi et al., 2023), and Self-RAG learns when to retrieve and how to critique retrieved passages (Asai et al., 2024). Direct corpus interaction (DCI) further enables agents to search raw documents with shell commands and short scripts, composing lexical filters and following newly discovered entities (Li et al., 2026). In these agents, every command, raw output, failed search, and provisional conclusion enters one interaction history, and what the agent has found is carried forward only as part of that history. After a few dozen steps, the handful of lines that support the answer are buried among irrelevant matches.

We focus on three challenges that arise when findings live only in this history. First, the history mixes findings with search errors, failed commands, abandoned hypotheses, and the agent's own reasoning traces. Irrelevant context is known to reduce answer accuracy (Yoran et al., 2024; Liu et al., 2024), and the agent's judgment of whether it has enough support, and of what is still missing, becomes entangled with this noise. Second, long histories must eventually be truncated, summarized, or compacted (Xu et al., 2024; Packer et al., 2023), and compression cannot tell a supporting line from the noise around it, so evidence the agent has already found can be discarded. Third, parallel

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.

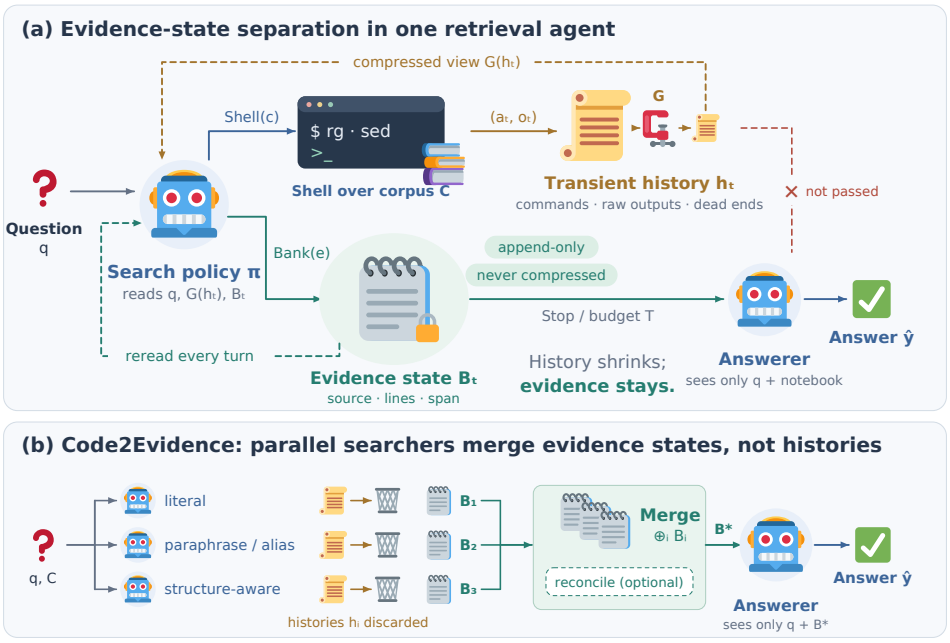


Figure 1: Evidence-state separation. **(a)** Actions and raw observations enter the transient history h_t, which the compressor shortens to $\widetilde{h}_t$. Spans the agent judges relevant are banked, with provenance, into the persistent evidence state $\mathcal{B}_t$, a clean record that is never compressed and is shown to the policy at every step (dashed arrows). The answer is generated from q and $\mathcal{B}_T$ alone. **(b)** Several searchers with complementary prompts run (a); their evidence states are merged, optionally reconciled, and passed to the answerer.

searchers produce complementary but overlapping findings, and their histories cannot be meaningfully concatenated into a basis for answering. All three motivate the same design: keep what has been found in a state of its own, separate from the record of how it was found, and let that state guide both the search and the answer.

Motivated by these challenges, we introduce *evidence-state separation* (Figure 1a), built on a simple premise: *evidence should outlive the search that produced it*. A retrieval agent therefore keeps two states with different lifetimes. The *transient interaction history* of actions and raw observations may be compressed freely. The *persistent evidence state* is an append-only record of source spans with document and line provenance, written only through an explicit BANK action, never compressed, and shown in full at every step, so that what has been established guides the search for what is still missing. When the agent stops, a tool-free answerer receives the question and the evidence state alone, without the search history. Because evidence states can be merged and histories cannot, the separation extends directly to parallel search (Figure 1b). Concretely, CODE2EVIDENCE runs several searchers that explore a raw corpus with shell commands, each following a complementary strategy: literal matching, paraphrase and alias search, or a structure-aware role that follows multi-hop links, financial operands, or protocol structure. Their histories are discarded; their evidence states are merged, optionally reconciled to remove duplicate entries and entries that violate the question's constraints, and passed to the answerer. Notably, the design *adds no new model, verifier, or learned representation*: all components run at inference time with fixed model parameters, the same LLM policy decides what to bank and when to stop, and what changes is only where the agent's findings live. Our contributions are summarized as follows:

- **Principle.** We identify the interaction history as the wrong place for a retrieval agent's findings and propose *evidence-state separation*, a two-state design in which a persistent, provenance-carrying evidence state guides the search, survives compression, and is the only input to the answer. Because evidence states compose and histories do not, the design extends directly to parallel search with merged and reconciled evidence states.

- **System and protocol.** We develop CODE2EVIDENCE, a coding-based retrieval agent with three role-specialized searchers, and design controlled comparisons that fix the model, tools, turn budget, and compression policy and vary only the evidence state or the answerer's input; comparisons with DCI-Agent-Lite and search-agent baselines, including on Legal RAG Bench (Butler & Butler, 2026), are reported as system-level results.
- **Evidence.** Across six Wikipedia QA datasets, FinQA (Chen et al., 2021), and BioProBench (Liu et al., 2026a), answering from the evidence state matches or beats answering from the compressed history in all ten task–backbone pairs with GPT-5.4-mini and GPT-5.5, by up to 4.0 points (Table 1). With GLM-5.3 on Wikipedia QA, the evidence state raises an otherwise identical single agent from 74.3% to 80.7%, and three role-specialized searchers with reconciled evidence states reach 85.2%, 5.8 points above DCI-Agent-Lite; the largest system-level gains come with the smallest backbone, GPT-5.4-mini, on Legal RAG Bench and BioProBench.

2 RELATED WORK

Iterative retrieval and reasoning. RAG conditions generation on retrieved passages (Lewis et al., 2020). Iterative methods interleave retrieval with reasoning: IRCoT with chain-of-thought steps (Trivedi et al., 2023), self-ask with follow-up questions (Press et al., 2023), ReAct with tool actions (Yao et al., 2023), and Self-RAG with learned retrieval and critique (Asai et al., 2024). RL-trained search agents such as Search-R1 (Jin et al., 2025), R1-Searcher (Song et al., 2025), and ASearcher (Gao et al., 2025) learn when and what to search from outcome rewards. Search-o1 condenses retrieved documents with a Reason-in-Documents step (Li et al., 2025), the closest single-agent analogue to our evidence state, but its refined knowledge is model-written and stays inside the reasoning chain. In all of these, findings are carried implicitly in one growing trace of reasoning and observations. Unlike these methods, we carry them explicitly in a separate, evidence-only state.

Context compression and irrelevant context. Models use long contexts unevenly (Liu et al., 2024) and can be misled by irrelevant passages (Yoran et al., 2024). RECOMP summarizes retrieved documents (Xu et al., 2024), and MemGPT pages information between working context and external storage (Packer et al., 2023). These methods decide how much context to keep and how to shorten it. We instead distinguish what kind of context it is: interaction history, compressed freely, versus selected evidence, never compressed and the sole input to the answer. The evidence state is neither a learned compressor nor a general-purpose memory, but an append-only record of source spans chosen by the policy.

Coding-based retrieval. Toolformer trains models to call APIs (Schick et al., 2023), CodeAct uses code as an action space (Wang et al., 2024), and SWE-agent shows that the agent–computer interface shapes repository navigation (Yang et al., 2024). Direct corpus interaction (DCI) lets an agent search raw corpora with shell commands (Li et al., 2026); we adopt it as both our retrieval substrate and main baseline. Its long, noisy observations make the history/evidence distinction especially sharp.

Multi-agent RAG and deep research. MA-RAG coordinates planner, step-definer, extractor, and QA agents (Nguyen et al., 2025); MASS-RAG has summarizer, extractor, and reasoner agents whose outputs a synthesis agent combines (Xiao et al., 2026); and deep-research agents run long, planned web searches (Zheng et al., 2025). Our parallel extension (Section 3.4) shares this broad pattern; the difference is what agents exchange: MA-RAG and MASS-RAG pass model-written extractions or summaries of retrieved passages, whereas each CODE2EVIDENCE searcher passes only verbatim, line-addressed spans banked while searching a raw corpus, and the reconciler can delete but not rewrite them. This makes independent findings mergeable while histories are not. Because these systems operate on retriever outputs rather than raw corpora, we isolate specialization and reconciliation with ablations instead of running them as baselines (Section 4.4).

3 EVIDENCE-STATE SEPARATION

Sections 3.1–3.3 define the two states and the loop that connects them, Section 3.4 the parallel-search extension, and Section 3.5 our coding-based instantiation; the separation itself does not depend on the retrieval interface.

3.1 TWO STATES WITH DIFFERENT LIFETIMES

A question q is answered against a corpus $\mathcal{C} = \{d_j\}$, where each document $d_j = (x_{j1}, \ldots, x_{jL_j})$ is a sequence of lines. At step t the agent takes an action a_t and receives an observation o_t.

Transient interaction history. The raw history $h_t = (a_1, o_1, \ldots, a_{t-1}, o_{t-1})$ records everything the agent has done: commands, their outputs, failed searches, and intermediate reasoning. It grows quickly, since a single shell observation can run to thousands of characters, and most of it is irrelevant to the answer. A compressor G presents the policy with a shortened view

$$\widetilde{h}_t = G(h_t) = \big(h^{\mathrm{sys}},\, q,\, \mathrm{Summ}(h_t^{\mathrm{old}}),\, h_t^{\mathrm{recent}}\big),$$

which truncates each observation and replaces older tool outputs with short markers or a progress note. We call h_t *transient* because nothing in it is guaranteed to survive G.

Persistent evidence state. The evidence state $\mathcal{B}_t = (e_1, \ldots, e_{K_t})$ is an append-only list of entries

$$e_k = \big(s_k,\, [\ell_k^-, \ell_k^+],\, z_k,\, \rho_k\big),$$

where s_k identifies a source document, $[\ell_k^-, \ell_k^+]$ is a line range, z_k is a verbatim span, and ρ_k is an optional note on why the span is relevant. $\mathcal{B}$ has three defining properties:

(P1) **Explicit writes.** $\mathcal{B}$ changes only through a $\textsc{Bank}(e)$ action chosen by the policy: $\mathcal{B}_{t+1} = \mathcal{B}_t \oplus e$ if $a_t = \textsc{Bank}(e)$, and $\mathcal{B}_{t+1} = \mathcal{B}_t$ otherwise, with $\mathcal{B}_1 = \varnothing$.

(P2) **Clean, evidence-only trace.** $\mathcal{B}$ contains only the evidence the agent has collected, each span with its provenance. Commands, raw outputs, failed searches, and reasoning traces never enter it. It is also exempt from compression: G never modifies $\mathcal{B}$, and messages that contain the evidence state are skipped by every compressor.

(P3) **Always visible.** $\mathcal{B}_t$ is returned in full after every append and conditions every later action.

Provenance fields are written by the agent and are not verified cryptographically. Adapters without a line field keep document-level provenance only.

What a separate state provides. By (P2), $\mathcal{B}_t$ is a clean trace of what has been found. The history, even when uncompressed, mixes findings with search errors, failed commands, abandoned hypotheses, and reasoning traces. $\mathcal{B}_t$ instead gives the policy and the answerer a compact, explicit record of the collected evidence alone, which guides the search toward what is still missing and grounds the answer. Because $\mathcal{B}_t$ is exempt from compression, it also preserves findings whose original observations, and even the turn in which they were banked, G may have removed from $\widetilde{h}_t$. We do not claim that any single one of these roles accounts for the empirical gains.

3.2 SEARCH GUIDED BY THE EVIDENCE STATE

The policy chooses actions conditioned on both states,

$$a_t \sim \pi_\theta\big(\cdot \mid q,\, \widetilde{h}_t,\, \mathcal{B}_t\big), \qquad a_t \in \{\textsc{Shell}(c),\, \textsc{Bank}(e),\, \textsc{Stop}\},$$

where θ is a fixed inference backend; no parameters are learned. The two states play different roles. $\widetilde{h}_t$ tells the agent what it has recently tried, so it can avoid repeating searches. $\mathcal{B}_t$ tells it what it has established, so it can direct search toward what is still missing: an unresolved hop, a missing operand, or an unconfirmed date.

Sufficiency is a policy decision, not a verification. The agent stops by emitting $\textsc{Stop}$ (an `EVIDENCE_STATUS: enough` line), or when the turn budget T runs out, at which point tools are disabled. The evidence state affects this decision only because the policy is conditioned on it, and nothing checks that the decision is correct: $\mathcal{B}$ does not certify sufficiency, it only makes the basis for the decision explicit and inspectable.

3.3 ANSWERING FROM THE EVIDENCE STATE

After stopping, a tool-free answerer produces

$$\widehat{y} = \mathrm{Parse}\big[A_\theta(q,\ \mathcal{B}_T)\big].$$

The history h_T is not passed to it. This is an *architectural* information bottleneck, not a learned one: whatever the agent did not bank is unavailable at answer time. For numerical tasks, the answerer may call a whitelisted arithmetic calculator. The *evidence-plus-trajectory* variant used in our experiments instead gives the answerer $(q, \mathcal{B}_T, \widetilde{h}_T)$. The *history-only* baseline has no $\mathcal{B}$ and answers from $(q, \widetilde{h}_T)$.

3.4 EXTENSION: MERGING EVIDENCE STATES ACROSS SEARCHERS

Evidence states compose; histories do not. A history is a sequence of commands and raw outputs whose meaning depends on its own earlier steps, so concatenating several histories for one answerer mostly multiplies the noise. An evidence state is a list of self-contained, source-attributed spans, so the states of different searchers can be pooled. We use this to run search in parallel (Figure 1b).

Parallel searchers. N searchers each run the loop of Section 3.2 with their own history h_i, evidence state $\mathcal{B}_i$, turn budget, and compressor. Each works in a separate workspace that contains the same corpus, so searchers share source content but not notes, and they do not communicate during search. Each stops at its own STOP or when its budget T_i runs out. Because only the merged evidence is used, searchers need not be redundant draws of one policy; they can be given complementary prompts. By default there are three roles, cycled when $N > 3$: a *literal* role, which searches for exact phrases and entity or metric names from the question; a *paraphrase/alias* role, which searches for alternate wordings, aliases, units, and table headers; and a *structure-aware* role, which follows hop chains for Wikipedia QA, decomposes the question into operands and locates each one for financial QA, and follows procedure structure for protocols. Roles differ only in the searcher prompt. Tools, budget, and compressor are shared.

Merging and reconciliation. When every searcher has stopped, the histories are discarded and the evidence states are merged:

$$\mathcal{B}_{\mathrm{mech}} = \bigoplus_{i=1}^{N} \mathrm{Renumber}(\mathcal{B}_{i,T_i}), \qquad \mathcal{B}^\star = \begin{cases} R_\theta(q, \mathcal{B}_{\mathrm{mech}}) & \text{with reconciliation,} \\ \mathcal{B}_{\mathrm{mech}} & \text{otherwise.} \end{cases}$$

$\mathrm{Renumber}$ assigns fresh entry indices and leaves each entry's source, line range, and span unchanged. The mechanical merge $\mathcal{B}_{\mathrm{mech}}$ keeps everything, including duplicates and entries banked under a mistaken reading of the question. The optional reconciler R_θ is a single tool-free call that sees only q and $\mathcal{B}_{\mathrm{mech}}$. It removes duplicate entries and entries that violate the question's entity, time, or hop constraints. It has no tools, so it cannot add evidence that no searcher banked. If it fails, the mechanical merge is used.

Answering. The answerer is the same tool-free A_θ as in Section 3.3, applied to $(q, \mathcal{B}^\star)$; no searcher's history reaches it. If its answer is missing or it reports insufficient evidence, it is retried on the per-searcher evidence states $\mathcal{B}_1, \ldots, \mathcal{B}_N$ kept separate, in case reconciliation removed a needed entry. If that also fails, a searcher's proposed answer is returned and marked as a fallback. Specialization, reconciliation, and searcher count all act on evidence states rather than histories. We evaluate each with ablations after the central comparison (Section 4.4).

3.5 INSTANTIATION: CODING-BASED RETRIEVAL

We call our implementation CODE2EVIDENCE. Following direct corpus interaction (Li et al., 2026), SHELL(c) runs a bash command (`rg`, `grep`, `sed`, etc.) in a workspace containing the corpus. Each command has a time limit τ and an output cap L. BANK is implemented by an `append_evidence` tool that returns the updated evidence state. Our default compressor G has three parts: each tool result is truncated to 20,000 characters; after each banking step, and after five consecutive steps without banking, a *collapse* step replaces all but the two most recent tool outputs with short markers plus a model-written progress note; and older tool bodies are compacted once accumulated tool text exceeds

240,000 characters. The progress note is itself part of the transient history. Evidence messages are exempt from all three. Command filtering and working-directory conventions are interaction controls, not operating-system sandboxing. Appendix B gives the per-benchmark tools, budgets, fallbacks, and the financial-QA source-acquisition step, which is outside the evidence-state mechanism.

4 RESULTS

The experiments answer three questions from mechanism to system. (i) With everything else fixed, does answering from an explicit evidence state improve accuracy, and what does the answerer gain from also seeing the search trajectory (Section 4.2)? (ii) How does the full Code2Evidence system compare with other agents (Section 4.3)? (iii) How much does each component contribute (Section 4.4)? Only (i) isolates the evidence state; the system-level comparisons in (ii) and (iii) change several factors at once.

4.1 EXPERIMENTAL SETUPS

Datasets and Tasks. We evaluate on Natural Questions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), Bamboogle (Press et al., 2023), HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), and MuSiQue (Trivedi et al., 2022) over the 2018 Wikipedia dump (Karpukhin et al., 2020), grouped as MHQA (three 300-question samples); FinQA (Chen et al., 2021) (three 150-question subsets); Legal RAG Bench (Butler & Butler, 2026) (its fixed 100-question test split, one run); and BioProBench (Liu et al., 2026a) (PQA, ORD, ERR, and GEN; 100 items per task for each of three seeds).

Baselines. Our main baseline is DCI-Agent-Lite (Li et al., 2026), the state-of-the-art coding-based retrieval agent, run with the same backbone, tools, context window, retry policy, and 50-turn budget as Code2Evidence. FinQA adds the RL-trained search agents Search-R1, R1-Searcher, and ASearcher (Jin et al., 2025; Song et al., 2025; Gao et al., 2025), and BM25 and Qwen3-4B-Embedding retrieval with GLM-5.3; BioProBench adds BioProAgent (Liu et al., 2026b).

Implementation Details. Code2Evidence runs three role-specialized searchers with 15 turns each and a five-turn answerer (50 turns) on all four benchmarks, with GLM-5.3, GLM-5.2, GPT-5.5, and GPT-5.4-mini as fixed-parameter backbones and dense retrieval disabled (Appendix B).

Evaluation. Metrics are numeric accuracy for FinQA, meaning match (MHQA) and correctness (Legal RAG Bench) judged by Grok-4.6, and the official BioProBench metrics. Mean $\pm$ sample SD is always over three runs, one per subset, sample, or seed; values without an SD are single runs or pool the three runs. Appendix B.8 gives the full protocol.

4.2 WHAT THE ANSWERER SEES: HISTORY, EVIDENCE STATE, OR BOTH

Settings. Table 1 compares three single-agent settings that differ only in what is retained during search and what the answerer receives: (i) *history only*, where the search run has no BANK action and the answerer receives the question and the compressed history $\widetilde{h}$; (ii) *evidence state* (i.e., notebook), where the answerer receives the question and the evidence state $\mathcal{B}$; and (iii) *evidence plus trajectory*, where the answerer receives both. Within a row, the three settings share the backbone, tools, turn budget, compressor G, policy prompt, stopping behavior, number of answer retries, and questions. The comparison tests two predictions of the design in Section 3: the evidence state improves on history alone, and once evidence is banked, adding the trajectory contributes little.

The evidence state helps across tasks and backbones. Setting (ii) matches or exceeds setting (i) in all ten rows of Table 1, and setting (iii) exceeds it in all ten. Gains reach 4.0 points (GPT-5.5, ORD exact match) and 2.9 points (GPT-5.4-mini, FinQA); four of the ten differences are at most one point, so the strength of the evidence lies in its consistency across three benchmarks, five metrics, and two backbones. With GLM-5.3, adding the evidence state to a single agent raises MHQA accuracy from $74.3 \pm 0.82\%$ to $80.7 \pm 1.63\%$ (Table 3, rows 1–2). These results show that the evidence state helps, not *why*: beyond protecting evidence from compression, it keeps findings apart from search errors and reasoning traces and guides further search.

Table 1: What the agent keeps and what the answerer sees. Single-agent settings: (i) history only, (ii) evidence state only, (iii) evidence state plus trajectory. Δ columns are in percentage points. Settings (i) and (iii) differ only in whether the evidence state exists. Best setting per row in bold. No SD is reported. MHQA: Grok-4.6 meaning match on one 300-question run; FinQA: numeric accuracy pooled over the three subsets; BioProBench: PQA accuracy, ORD exact match, and ERR accuracy pooled over the three seeds.

Task	Backbone	(i)	(ii)	(iii)	(ii)−(i)	(iii)−(i)
MHQA	GPT-5.4-mini	58.3	58.3	**61.0**	+0.0	+2.7
	GPT-5.5	83.7	**84.7**	**84.7**	+1.0	+1.0
FinQA	GPT-5.4-mini	46.0	48.9	**49.6**	+2.9	+3.6
	GPT-5.5	72.7	**73.6**	73.3	+0.9	+0.6
PQA	GPT-5.4-mini	59.7	62.0	**63.0**	+2.3	+3.3
	GPT-5.5	71.0	**73.3**	**73.3**	+2.3	+2.3
ORD	GPT-5.4-mini	33.7	35.4	**36.7**	+1.7	+3.0
	GPT-5.5	51.2	**55.2**	54.9	+4.0	+3.7
ERR	GPT-5.4-mini	58.3	**59.3**	**59.3**	+1.0	+1.0
	GPT-5.5	68.0	69.3	**69.7**	+1.3	+1.7

The value of the trajectory depends on the backbone. Comparing settings (iii) and (ii) isolates the trajectory's contribution once evidence is banked. With GPT-5.5, adding the trajectory changes accuracy by at most 0.4 points on every task, so the evidence state carries essentially all of the useful information. With GPT-5.4-mini, the trajectory adds a small amount on four of five tasks, most on MHQA (58.3 to 61.0), and nothing on ERR. The multi-agent GLM-5.3 comparison in Table 3 points the same way: answering from the reconciled evidence state alone ($85.2 \pm 1.03\%$, row 4) is 1.9 points more accurate than also exposing the searcher trajectories ($83.3 \pm 1.52\%$, row 5), a difference comparable to the across-run SDs. The second prediction therefore holds for the two stronger backbones, while the smaller model can still extract some additional signal from its own search history.

4.3 System-level comparisons

FinQA. GLM-5.3 Code2Evidence reaches $76.67 \pm 3.06\%$ accuracy over the three FinQA subsets, compared with $36.22 \pm 1.68\%$ for Search-R1, $42.44 \pm 0.77\%$ for R1-Searcher, and $41.22 \pm 0.77\%$ for ASearcher (Figure 2(a)). These search agents differ in backbone, retriever, and source access: they use GPT-5.5 to fetch a full Form 10-K and search it with BM25, whereas Code2Evidence acquires filings live from EDGAR and searches them with shell commands. With the same GLM-5.3 backbone, retrieval baselines reach $51.89 \pm 3.06\%$ with BM25 and $60.44 \pm 1.68\%$ with Qwen3-4B-Embedding, 24.78 and 16.23 points below Code2Evidence. Against DCI with the same backbone, Code2Evidence is more accurate on all four backbones, by 5.78 points with GLM-5.3, 5.33 with GLM-5.2, 3.33 with GPT-5.4-mini, and 0.22 with GPT-5.5; the GPT-5.5 difference is well within the across-subset spread. With GPT-5.4-mini, Code2Evidence also has a smaller across-subset SD than DCI (1.39 versus 4.07).

MHQA. Averaged over three runs, GLM-5.3 Code2Evidence achieves $85.2 \pm 1.03\%$ meaning-match accuracy, compared with $79.4 \pm 1.09\%$ for DCI-Agent-Lite (Figure 2(b)). The 5.8-point gain, more than five times either system's across-run SD, corresponds to about 17 more correct answers per 300-question run and removes approximately 28% of DCI's errors. Code2Evidence exceeds DCI on all six datasets, by 6.4 points on Natural Questions, 6.6 on TriviaQA and Bamboogle, 5.0 on HotpotQA, 6.0 on 2WikiMultiHopQA, and 4.0 on MuSiQue; every per-dataset gain is larger than either system's across-run SD on that dataset. Gains persist where DCI is already strong: the TriviaQA error rate falls from 11.3% to 4.7% and the HotpotQA error rate from 11.3% to 6.3%.

Legal RAG Bench. In this single-run comparison (Figure 2(c)), Code2Evidence reaches 93% correctness with GLM-5.2 and 91% with GPT-5.5, one to two points above DCI (a difference within plausible run-to-run variation), and improves annotated-passage retrieval by four and five points with these backbones. The largest gain comes with GPT-5.4-mini: correctness rises from 53% to 72% and answer availability from 66% to 98%. Code2Evidence reaches this higher correctness despite a

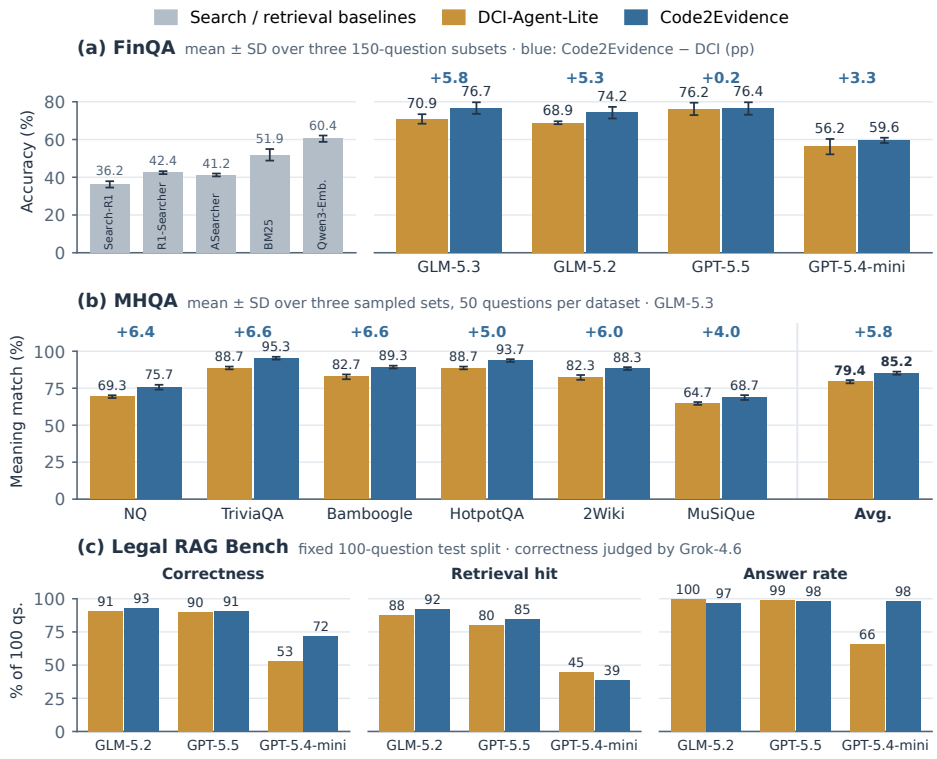


Figure 2: FinQA, MHQA, and Legal RAG Bench results (%). Blue numbers: Code2Evidence minus DCI-Agent-Lite (points). **(a) FinQA:** mean ± sample SD over three 150-question subsets; Search-R1, R1-Searcher, and ASearcher use their released Qwen2.5-7B checkpoints, and BM25 and Qwen3-4B-Embedding use GLM-5.3. **(b) MHQA:** Grok-4.6-judged meaning match, mean ± sample SD over three 300-question runs, GLM-5.3. **(c) Legal RAG Bench:** single run on the 100-question test split; correctness judged by Grok-4.6, retrieval hit is the annotated passage among touched passages, and answer rate counts nonempty outputs.

Table 2: BioProBench primary and secondary scores. Mean ± sample SD over three seeds; arrows indicate the preferred direction.

		PQA		ORD		ERR
Backbone	Agent	Acc.↑	Brier↓	EM↑	τ↑	Acc.↑
GLM-5.3	C2E	0.683±0.031	0.205±0.018	0.508±0.056	0.798±0.035	0.677±0.057
	DCI	0.675±0.041	0.210±0.033	0.505±0.051	0.773±0.042	0.665±0.073
	BioProAgent	0.650±0.038	0.253±0.035	0.475±0.061	0.750±0.030	0.657±0.043
GPT-5.5	C2E	0.743±0.050	0.189±0.034	0.549±0.076	0.819±0.052	0.690±0.036
	DCI	0.743±0.015	0.178±0.012	0.525±0.051	0.833±0.036	0.673±0.050
GPT-5.4-mini	C2E	0.610±0.050	0.294±0.026	0.404±0.046	0.686±0.023	0.557±0.057
	DCI	0.477±0.055	0.307±0.053	0.279±0.046	0.675±0.040	0.607±0.050

C2E denotes Code2Evidence; BioProAgent is the protocol agent of Liu et al. (2026b). Brier uses valid confidence-bearing outputs; Kendall's τ uses valid orderings, so these secondary metrics have different coverage from accuracy/EM. Coverage and generation scores appear in Appendix F.

lower annotated-passage hit rate (39% versus 45%), so most of the gain comes from turning retrieved material into completed, correct answers rather than from locating the designated passage more often.

BioProBench. On the PQA, ORD, and ERR tasks (Table 2; generation in Appendix F), with GLM-5.3, Code2Evidence is ahead of DCI and BioProAgent (Liu et al., 2026b) on all five metrics,

Table 3: Building up Code2Evidence component by component (MHQA, GLM-5.3, meaning match judged by Grok-4.6, %, mean ± sample SD over three runs). Each row adds one component to the row above. Δ_{step} is the change from the previous row and $\Delta_{(1)}$ the change from the single-agent baseline, in percentage points.

	Configuration	States	Reconcile	Traj.	Accuracy	Δ_{step}	$\Delta_{(1)}$
(1)	Single agent	–	–	–	74.3 ± 0.82	–	–
(2)	+ Evidence state	1	–	–	80.7 ± 1.63	$+6.4$	$+6.4$
(3)	+ Multi-agent	3 concat.	–	–	79.7 ± 1.19	-1.0	$+5.4$
(4)	+ Reconciliation	1 merged	✓	–	$\mathbf{85.2} \pm 1.03$	$+5.5$	$+10.9$
(5)	+ Trajectories	1 merged	✓	✓	83.3 ± 1.52	-1.9	$+9.0$
(4′)	(4) w/o roles	1 merged	✓	–	79.2 ± 1.81	-6.0 vs. (4)	

States: what the answerer reads. Rows (3)–(5) run three searchers with the literal, alias/paraphrase, and hop-chain roles; in (3) the answerer reads the mechanical merge of the three searcher evidence states, which keeps every entry, while from (4) on the reconciler turns them into a single reconciled evidence state (Section 3.4). Traj.: the answerer also sees the searcher trajectories. (4′) removes role specialization from the default (4).

although its margins over DCI (0.3 to 2.5 points) are within one seed SD, reaching 0.683 ± 0.031 PQA accuracy and 0.508 ± 0.056 ORD exact match. With GPT-5.5, it matches DCI's PQA accuracy (0.743) and improves ORD exact match from 0.525 to 0.549 and ERR accuracy from 0.673 to 0.690, while DCI has the lower Brier score and higher Kendall's τ. The largest multi-seed gains occur with GPT-5.4-mini: PQA accuracy improves by 13.3 points and ORD exact match by 12.5 points, relative gains of approximately 27.9% and 44.8%, while DCI is ahead on ERR. A single-seed GPT-5.6-sol comparison (Appendix F) favors Code2Evidence by one point on PQA, four on ORD exact match, and 14 on ERR accuracy.

4.4 Ablation Study

Table 3 builds the GLM-5.3 system one component at a time on MHQA. Giving the single agent an evidence state raises accuracy from $74.3 \pm 0.82\%$ to $80.7 \pm 1.63\%$ (+6.4 points). Running three role-specialized searchers and passing the answerer the mechanical merge of their evidence states does not help by itself ($79.7 \pm 1.19\%$, -1.0). Reconciling those states into a single one before answering produces the largest step, +5.5 points to $85.2 \pm 1.03\%$, the Code2Evidence default and 10.9 points above the single agent. Also showing the answerer the searcher trajectories lowers accuracy to $83.3 \pm 1.52\%$ (-1.9). Role specialization is also necessary: removing it from the default drops accuracy to $79.2 \pm 1.81\%$ (-6.0). Relative to the preceding configuration's error rate, the evidence state removes approximately 25% of errors and reconciliation 27%, and removing role specialization raises the default's error rate from 14.8% to 20.8%. The evidence-state, reconciliation, and role effects (6.4, 5.5, and 6.0 points) are each at least three times the larger across-run SD of the two rows compared, whereas the -1.0 and -1.9 steps are within about one to two SDs. They are also several times the one point gained by going from three to five searchers (Appendix H.1), so *how workers search and combine their findings matters more than how many run*. Search-budget and searcher-count studies and observational evidence-state diagnostics are in Appendix H.

5 Conclusion

In this paper, we propose *evidence-state separation*: a retrieval agent keeps what it has found in a persistent, provenance-carrying evidence state that is never compressed, guides further search, and is the only input to the answer, while its history may be compressed freely. We instantiate the principle in Code2Evidence, a coding-based retrieval agent with role-specialized searchers that merge and reconcile their evidence states. Answering from the evidence state matches or beats the compressed history in all ten controlled task–backbone pairs, and Code2Evidence outperforms DCI-Agent-Lite by 5.8 points on MHQA and by 19 points with the smallest backbone on Legal RAG Bench. Future work will isolate why the evidence state helps, verify banked spans and sufficiency instead of trusting the policy's judgment, and extend evaluation to full test splits and untrusted corpora.

Reproducibility Statement

The appendices specify the datasets, sample sizes, seeds, metrics, turn budgets, compression settings, and fallback rules for every reported run, and identify single-run numbers wherever they appear; Appendix G gives per-question paired statistics for the controlled comparisons.

AI Use Statement

Generative AI tools were used to assist with manuscript organization, editing, consistency checks, and as LLM-Judges for evaluating answers. Generative AI tools were not used to produce experimental runs or numerical results except as part of the disclosed evaluation procedures; all reported numbers come from the runs described in the appendices. The authors remain responsible for the final manuscript.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.11511.

Abdur-Rahman Butler and Umar Butler. Legal RAG Bench: An end-to-end benchmark for legal RAG. *arXiv preprint arXiv:2603.01710*, 2026. URL https://arxiv.org/abs/2603.01710.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. FinQA: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3697–3711. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.300. URL https://aclanthology.org/2021.emnlp-main.300/.

Jiaxuan Gao, Wei Fu, Minyang Xie, Shusheng Xu, Chuyi He, Zhiyu Mei, Banghua Zhu, and Yi Wu. Beyond ten turns: Unlocking long-horizon agentic search with large-scale asynchronous RL. *arXiv preprint arXiv:2508.07976*, 2025. URL https://arxiv.org/abs/2508.07976.

Neel Guha, Julian Nyarko, Daniel E. Ho, Christopher Ré, Adam Chilton, Aditya K, Alex Chohlas-Wood, Austin Peters, Brandon Waldon, Daniel N. Rockmore, Diego Zambrano, Dmitry Talisman, Enam Hoque, Faiz Surani, Frank Fagan, Galit Sarfaty, Gregory M. Dickinson, Haggai Porat, Jason Hegland, Jessica Wu, Joe Nudell, Joel Niklaus, John Nay, Jonathan H. Choi, Kevin Tobia, Margaret Hagan, Megan Ma, Michael Livermore, Nikon Rasumov-Rahe, Nils Holzenberger, Noam Kolt, Peter Henderson, Sean Rehaag, Sharad Goel, Shang Gao, Spencer Williams, Sunny Gandhi, Tom Zur, Varun Iyer, and Zehua Li. LegalBench: A collaboratively built benchmark for measuring legal reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/89e44582fd28ddfea1ea4dcb0ebbf4b0-Abstract-Datasets_and_Benchmarks.html.

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

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. *arXiv preprint arXiv:2503.09516*, 2025. URL https://arxiv.org/abs/2503.09516.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611. Association for Computational Linguistics, 2017. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

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 *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 6769–6781, 2020.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

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, pp. 9459–9474, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

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. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.276. URL https://aclanthology.org/2025.emnlp-main.276/.

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. *arXiv preprint arXiv:2605.05242*, 2026. URL https://arxiv.org/abs/2605.05242.

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

Yuyang Liu, Liuzhenghao Lv, Xiancheng Zhang, Jingya Wang, Li Yuan, and Yonghong Tian. BioProBench: A corpus and benchmark for biological protocol reasoning in autonomous science. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026a. URL https://icml.cc/virtual/2026/poster/65981.

Yuyang Liu, Jingya Wang, Liuzhenghao Lv, and Yonghong Tian. BioProAgent: Neuro-symbolic grounding for constrained scientific planning. *arXiv preprint arXiv:2603.00876*, 2026b. URL https://arxiv.org/abs/2603.00876.

Thang Nguyen, Peter Chin, and Yu-Wing Tai. MA-RAG: Multi-agent retrieval-augmented generation via collaborative chain-of-thought reasoning. *arXiv preprint arXiv:2505.20096*, 2025. URL https://arxiv.org/abs/2505.20096.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah 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. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-emnlp.378. URL https://aclanthology.org/2023.findings-emnlp.378/.

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, pp. 68539–68551, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract.html.

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

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475. URL https://aclanthology.org/2022.tacl-1.31/.

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 (Volume 1: Long Papers)*, pp. 10014–10037. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.557. URL https://aclanthology.org/2023.acl-long.557/.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better LLM agents. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 50208–50232. PMLR, 2024. URL https://proceedings.mlr.press/v235/wang24h.html.

Xingchen Xiao, Heyan Huang, Runheng Liu, and Jincheng Xie. MASS-RAG: Multi-agent synthesis retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 9865–9883. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.480. URL https://aclanthology.org/2026.findings-acl.480/.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. RECOMP: Improving retrieval-augmented LMs with compression and selective augmentation. In *International Conference on Learning Representations*, 2024.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R. Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract.html.

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. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

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. URL https://arxiv.org/abs/2210.03629.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. Making retrieval-augmented language models robust to irrelevant context. In *International Conference on Learning Representations*, 2024.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. DeepResearcher: Scaling deep research via reinforcement learning in real-world environments. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 414–431. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.22. URL https://aclanthology.org/2025.emnlp-main.22/.

A LIMITATIONS

Scope of the causal evidence. The matched comparisons of Section 4.2 isolate the evidence state and the answerer's input, but not every alternative explanation. The evidence-state setting's prompt asks the agent to bank evidence and re-read it, which may change how carefully it searches, independently of whether evidence survives compression; we have not run a compression stress test or a token-matched control that would separate these explanations, and the observation-volume breakdown in Appendix G.2 does not by itself support the survival mechanism.

Statistical coverage. MHQA uses three 50-question samples per dataset, FinQA three 150-question subsets rather than a full split (with a scorer that forgives sign and scale errors), and BioProBench 100 items per task and seed rather than the full test pools. Legal RAG Bench has one fixed split without repeated runs, and its correctness metric uses one LLM judge.

System-level comparisons. Code2Evidence, DCI-Agent-Lite, and the search-agent wrappers differ in prompts and context management, so the comparisons of Section 4.3 characterize systems rather than the evidence state alone.

Cost and deployment. An evidence state adds tokens, and parallel search multiplies them; dollar costs are available only for selected GPT runs, and comparable latency logs are absent. The command filter and workspace conventions are not a security boundary, so deployment on untrusted corpora requires operating-system isolation and prompt-injection controls.

B METHOD: CODE2EVIDENCE

B.1 SCOPE AND REFERENCE IMPLEMENTATION

Code2Evidence is a tool-calling framework for retrieving evidence from long documents with ordinary Linux commands, recording selected spans in a persistent notebook, and separating retrieval from answer production. Throughout the appendix, *notebook* refers to the implemented evidence state of Section 3.

B.2 ROLES AND INFORMATION BOUNDARIES

For FinQA-style questions, the reference pipeline has four logical roles. A *filing brain* selects and validates the source filing; N independent *shell searchers* retrieve cited spans; an optional *dense searcher* provides an independent semantic channel; and a *main answerer* resolves the evidence and computes the answer. Retrieval workers do not coordinate directly. By default, the main answerer receives a derived evidence notebook rather than worker trajectories, which creates an explicit information bottleneck between exploration and reasoning.

Filing brain. The brain is restricted to document acquisition tools: `edgar_search`, `parse_html_page`, `run_shell`, `list_workspace_files`, `get_stored_data`, and `web_search_ddgs`. Its prompt forbids answering the question or banking numeric evidence. It searches for the requested issuer's 10-K, saves the selected HTML body as a local text document, verifies that it contains the filing body for the requested registrant, and emits `FILING_READY: yes` only after verification. The implementation additionally checks the downloaded text outside the model by rejecting short pages, EDGAR index pages, submissions JSON, non-10-K text, and mismatched ticker, company, or CIK evidence.

Shell searchers. The default role set cycles through three prompt specializations as N grows: (i) a literal-keyword worker that searches exact metric phrases, (ii) a paraphrase/HyDE-style worker that searches alternate wording, units, and table headers, and (iii) an operand/table worker that decomposes a question and locates each required operand. An ablation mode replaces these specializations with homogeneous independent searchers. Each worker may call only shell, notebook, file-listing, stored-text, and calculator tools; EDGAR access remains brain-only.

Dense searcher. An optional dense searcher, disabled in all reported runs, provides an embedding-based channel over the same filing; it returns line-addressed spans in the same notebook schema as the shell workers.

Main answerer. The main answerer has no retrieval tools. It sees either the final notebook or, in an explicit ablation, clipped worker transcripts; its only enabled tool is the calculator. The prompt requires evidence, a computation expression (or `none`), and a bare numeric or yes/no `FINAL ANSWER`. Ratio, growth, sum, and average calculations are delegated to the calculator rather than performed implicitly by the model.

B.3 Shell and notebook interface

Every worker operates in a filesystem workspace. `parse_html_page` removes `script` and `style` elements, converts the remaining HTML to newline-delimited text, stores it in memory, and writes a text file with a sanitized name. The `run_shell` tool launches `bash -lc` with the workspace as its current directory, captures standard output and error, records the exit code, applies a configurable timeout (30 seconds by default for financial QA), and truncates captured output at a configurable limit (40,000 characters by default). Prompts recommend line-numbered `grep`/`rg` followed by narrow `sed`/`awk` windows; short Python commands are permitted.

The shell is constrained by a soft denylist covering destructive, privilege-escalating, external network, process-control, mount, and absolute-`cd` patterns. This is not an operating-system container: the implementation explicitly relies on working-directory confinement, command filtering, output caps, and timeouts. Consequently, we describe it as a bounded workspace interface rather than a formally isolated sandbox.

Each retrieval worker owns an append-only evidence notebook. An entry contains an ordinal, the source filename, a line range, an optional relevance note, and the selected snippet. `append_evidence` returns the complete notebook after every write, allowing the worker to test whether all required facts have been accumulated. Context-local workspace, storage, and file registries are implemented with `ContextVar`; before parallel search begins, the validated filing is copied into each worker's directory. Thus workers share source content but not mutable notes.

The notebook is also the durable context boundary. After a worker banks evidence, older large tool messages are compressed while notebook-bearing messages are retained. A short trajectory synthesizer records what was tried, what remains missing, and the next search action after each banking event and after five idle rounds. Saved worker trajectories and messages are enabled by default, but observations are clipped for storage; they are diagnostic artifacts and are not shown to the default main answerer.

B.4 End-to-end inference flow

The implemented control flow is the following:

1. Normalize the FinQA identifier by removing its page suffix, infer the fiscal year from the remaining identifier, and form an EDGAR query from company, ticker, years, and question.
2. Run the filing brain for at most eight turns by default. Accept only a validated, materialized filing; if validation fails, attempt deterministic CIK-based recovery from SEC submissions metadata. If no filing is recovered, return `brain_no_filing` without launching searchers.
3. Instantiate $N = 3$ shell workers by default and, unless disabled, one dense worker. Give every worker an isolated workspace containing the same filing. Execute the workers in parallel with a thread pool by default.
4. Let each worker alternate model calls and tools for at most 15 turns by default. A worker stops early after emitting `EVIDENCE_STATUS: enough` or a proposed final answer; the last worker turn disables tools and exposes its notebook.
5. Renumber and concatenate shell notebook entries. If dense retrieval is active, reconcile the shell and dense notebooks with a tool-free model call; if reconciliation raises an exception, use a tagged mechanical merge.

6. Ask a tool-free synthesizer to extract canonical operands and a formula, prepend this summary to the notebook, and invoke the main answerer for at most five turns. The answerer can call only the calculator, and tools are disabled on its last turn.
7. If the answer is absent, a refusal, or not a bare number/yes/no, retry with the separate shell and dense notebooks (when present), then retry with an explicit numeric-answer nudge. If these attempts fail but a searcher proposed an answer, return the first proposal as a marked fallback.

The turn accounting deliberately excludes retrieval work from the reported main budget: the returned `turns` field sums brain and answerer turns, while `searcher_turns_total` is reported separately. Searcher failures are localized to the corresponding result when parallel futures are collected, but an unrecoverable brain failure terminates the pipeline before retrieval.

B.5 Stopping and error handling

All agent loops have finite turn limits and disable tools on their final turn. Unknown or disallowed tool names and malformed JSON arguments are returned to the model as textual errors; common tool aliases and prefix variants are normalized before dispatch. Individual tool output is clipped before re-entering model context. API exceptions terminate the current role and are recorded in its trajectory. Reconciliation failure has a deterministic merge fallback, shell timeouts return partial output, and calculator errors (including division by zero) are returned as strings rather than raised.

Document acquisition receives additional deterministic safeguards. Full-text EDGAR results are ranked to prefer primary 10-K/10-Q HTML and penalize exhibits and index pages. Searches can be pinned to a CIK, and recovery enumerates recent and archived SEC submissions, preferring a 10-K whose report year equals the fiscal year or whose filing year is the following year. The brain also extracts CIKs from candidate archive URLs when the live ticker map is missing.

B.6 Backends and implementation parameters

The role logic is backend-agnostic through a `chat_fn` interface with OpenAI-style messages and native function calls. The client defaults to GLM-5.2, caps non-streaming completions at 8,000 tokens, allows at most ten concurrent requests, retries transient service errors with exponential backoff, and can fail over across configured model identifiers. The FinQA evaluation entry point also routes GPT-family names through an OpenAI-compatible client, including optional reasoning-effort configuration.

The reported runs use $N = 3$ specialized shell searchers with 15 turns each and five answerer turns (a 50-turn budget), parallel workers, dense retrieval disabled, and hidden trajectories. Filing acquisition (at most eight filing-brain turns) happens before search. All settings are controlled by environment variables in the implementation.

B.7 MHQA adaptation

The MHQA adaptation preserves the notebook bottleneck and parallel shell-search pattern but changes the corpus and roles. Every workspace contains a symbolic link to a locally stored Wikipedia passage corpus; the link avoids copying the corpus. Searchers are literal, paraphrase/alias, and sequential hop-chain workers, and web retrieval is disallowed. There is no filing brain because the corpus is preloaded. The default final notebook is LLM-reconciled, and the main answer format is a concise exact answer plus explanation and confidence.

MHQA adds explicit long-context policies. The default L3 policy truncates each tool result to 20,000 characters and compacts older tool bodies after accumulated tool text reaches 240,000 characters, retaining the twelve most recent tool turns. L4 additionally summarizes older history with a model call. Notebook messages are exempt from compaction. If the notebook-only answerer declares insufficient evidence, a five-turn rescue phase may search the corpus directly for the missing hop, after which the system retries answer production and can fall back to per-searcher notebooks or a worker proposal.

The released MHQA evaluator covers NQ, TriviaQA, Bamboogle, HotpotQA, 2WikiMultiHopQA, and MuSiQue using the released DCI-bench samples and a local 2018 Wikipedia passage dump. It can select GLM-5.3/GLM-5.2 or GPT-family backends.

B.8 Evaluation scope

Benchmarks and runs. FinQA uses three randomly selected subsets ($n = 150$ each, 450 evaluations covering 438 unique questions). The six-dataset QA suite (MHQA: Natural Questions, TriviaQA, Bamboogle, HotpotQA, 2WikiMultiHopQA, and MuSiQue) uses three 50-question samples per dataset: the released sample-50 files and two additional random samples, so each run covers 300 questions (900 over the three samples). BioProBench uses 100 items per task for each of three seeds (35/42/50) on protocol QA (PQA), ordering (ORD), error detection (ERR), and generation (GEN). Legal RAG Bench (Butler & Butler, 2026) uses its only test split, a fixed set of 100 questions, and is run once. Code2Evidence runs with three searchers on all four benchmarks. Whenever we report mean $\pm$ sample standard deviation (SD), the configuration was run three times, once per FinQA subset, MHQA sample, or BioProBench seed, and the SD summarizes variation across these three runs. Values without an SD come from a single run, or, where stated, pool the three runs into one score.

Metrics and comparability. We report each benchmark separately: numeric answer accuracy under the FinQA scorer's sign and scale normalization; meaning match for MHQA, judged by Grok-4.6; Grok-4.6-as-a-judge correctness for Legal RAG Bench; and official task metrics for BioProBench. In the system-level comparisons we keep the budget (50 turns for each framework), backbone model, tool pool, context window, and retry policy uniform within each dataset.

B.9 Protocol for the controlled comparisons

The controlled runs in Table 1 use GPT-5.4-mini and GPT-5.5 on MHQA (one 300-question run), FinQA (the three 150-question subsets, pooled), and BioProBench PQA, ORD, and ERR (the three seeds, pooled). The GLM-5.3 single-agent and three-searcher runs in Table 3 follow the same protocol on each of the three MHQA samples.

- **Single agent.** One searcher with a generic prompt runs the unchanged search loop of each benchmark.
  - MHQA: 30 turns.
  - FinQA: 15 turns, after the filing brain acquires the 10-K. The filing is acquired once per question and both search runs use it.
  - BioProBench: 15 turns, with one seed grep; in the notebook setting, relevant shell outputs are also auto-banked.
- **Shared answerer.** All single-agent settings end with the same answerer, so the settings differ only in its input. It is tool-free except for the calculator on FinQA. Each benchmark keeps its usual fallbacks, applied identically to every setting:
  - FinQA: a forced recovery pass on the same input after an insufficient answer, then the searcher's proposed answer.
  - BioProBench: one format retry, then format coercion.
  - MHQA: one format retry.
- **History given to the answerer.** The answerer receives the search context as it stood when the agent stopped, including compressed tool outputs and progress notes, with controller instructions and the agent's final stop message removed. It is capped at 300,000 characters, keeping the most recent part.
- **Re-answering.** Settings (ii) and (iii) answer from the same notebook search run. In the three-searcher setting, the searcher trajectories are clipped with the default 12,000-character budget and answered with and without $\mathcal{B}^\star$, in the same run.
- **History-only setting.** The BANK and notebook-reading tools are removed, and notebook mentions are removed from the prompts and compression placeholders. The collapse step fires after five consecutive steps without banking in both settings; in the notebook setting it also fires after each banking step.

Table 4: Recorded Code2Evidence token usage and estimated API cost.

Model	Seeds	Examples	Total tokens	Est. USD / USD per example
GPT-5.4-mini	35,42,50	450	66,193,065	43.07 / 0.096
GPT-5.5	35,42,50	450	50,191,351	237.05 / 0.527

- **Scoring.** FinQA and BioProBench settings are scored with the same scorers as the main results. MHQA settings are scored with meaning match judged by Grok-4.6.

C FinQA Experiments

C.1 Dataset and evaluation protocol

FinQA (Chen et al., 2021) contains questions grounded in financial-report text and tables, together with executable reasoning programs and their execution results. We sample without replacement using Python's seeded pseudorandom shuffle. Unless stated otherwise, each configuration is run on three 150-question subsets drawn from the FinQA training split with seeds 35, 42, and 50 (450 evaluations; 438 unique IDs, with 3–5 pairwise overlaps). Full-report access means access to the entire filing, not evaluation on a full FinQA split, and the reported numbers are not official FinQA test-set scores.

Code2Evidence first identifies and downloads the relevant filing, then launches three specialized shell searchers (literal-keyword, paraphrase/HyDE, and operand/table search), merges their evidence notebooks, and gives the answerer at most five turns, for a total budget of 50 turns (45 searcher turns and 5 answerer turns). Dense retrieval is disabled in the completed three-seed runs. The reference budget is 15 searcher turns. The answerer emits a numeric value or yes/no answer rather than a FinQA program.

Consequently, our metric is *not* FinQA's official program accuracy. We report answer accuracy over all sampled questions. The repository scorer extracts the first non-year numeric token from the `FINAL ANSWER` line, uses relative tolerance 10^{-2} and absolute tolerance 10^{-3}, compares absolute values, and additionally accepts scale differences of 100, $1{,}000$, or 10^6 (including equality after rounding a scaled value to one decimal). Unanswered and errored examples remain in the denominator. Means and sample standard deviations, where reported, are computed across the three runs, one per subset; they are not confidence intervals.

C.2 DCI-Agent-Lite configurations

The main comparison gives DCI-Agent-Lite a uniform 50-turn budget, matching Code2Evidence's 45 searcher turns and 5 answerer turns. Accuracies for GLM-5.3, GLM-5.2, GPT-5.5, and GPT-5.4-mini under those configurations are in Figure 2. Scores were recomputed from saved extracted predictions with the same permissive numeric scorer used for Code2Evidence. Zero-valued gold answers are preserved during rescoring.

C.3 Searcher-turn ablation

The GLM-5.3 comparison of five, ten, and fifteen turns per searcher, including answer rates, is reported in Figure 3(a).

C.4 Efficiency and cost

API cost was recorded for the OpenAI-backed Code2Evidence runs. Table 4 reports estimates at the list prices in effect at the time of the runs; these are not invoices and exclude EDGAR, storage, local compute, and engineering costs. GLM-5.2, GLM-5.3, and all search-agent baselines lack comparable cost records. Wall-clock durations are not logged consistently enough for a fair latency comparison.

D LEGAL RETRIEVAL QA: LEGAL RAG BENCH

Scope and naming. We evaluate on the test split of Legal RAG Bench, a retrieval-augmented question-answering benchmark about Victorian criminal law and procedure. Accordingly, we call the evaluation *Legal RAG Bench* throughout, not to be confused with LegalBench (Guha et al., 2023), and make no aggregate or task-level claim about LegalBench. The split has 100 questions and a corpus of 4,876 passages. The questions are used in their dataset order (IDs 1–100), without subsampling, shuffling, or a random seed. Thus, "100 questions" denotes one fixed full-split run, not a seeded sample.

D.1 ADAPTATION AND EXPERIMENTAL SETUP

Corpus materialization. Both agent families use the same downloaded test corpus. Each source passage is written as a separate Markdown document beginning with a `DOC:` passage identifier and including its title, body, and any footnotes. A manifest maps passage identifiers to documents. Code2Evidence and DCI-Agent-Lite use separate materialized workspaces containing the same 4,876 passages. Neither main agent uses a precomputed vector index in this evaluation: retrieval is performed by issuing shell commands over the files.

Code2Evidence adaptation. The Code2Evidence system prompt identifies the Judicial College of Victoria Criminal Charge Book as the source and requires local-corpus grounding. It instructs the model to inspect files, search using `rg` or `grep`, read local windows with standard Linux utilities, append useful snippets to the evidence notebook, re-read that notebook, and finish with `FINAL ANSWER:`. Answers should be direct and concise (one to three sentences) and cite `DOC` IDs when possible. The exposed function tools are `list_workspace_files`, `run_shell`, `append_evidence`, `read_evidence`, `get_stored_data`, and `calculator`, together with three finance/web tools that the legal prompt explicitly prohibits: `edgar_search`, `parse_html_page`, and `web_search_ddgs`. The evidence notebook is reset for each question. The last turn disables tools and asks for a final answer from banked evidence.

DCI-Agent-Lite adaptation. DCI-Agent-Lite receives the same question-level instruction: use only the local corpus, do not use web search, prefer `rg`, cite `DOC` IDs, write one to three direct sentences, and include `FINAL ANSWER:`. It is launched in the corpus root with only generic `read` and `bash` tools. GPT runs use context-management level 3, high thinking, and 50 turns, matching the turn budget of Code2Evidence. DCI does not use Code2Evidence's evidence-notebook functions in this evaluation.

Models. We report results for GLM-5.2, GPT-5.4-mini, and GPT-5.5 in both agent families.

D.2 METRICS

Answer and retrieval scoring. Answer rate is the fraction with a nonempty extracted final answer. Retrieval accuracy is the fraction for which the single annotated relevant passage ID appears in a heuristically extracted list of touched passage IDs. This is neither recall over all valid evidence nor a semantic retrieval measure. Correctness and groundedness are binary decisions from Grok-4.6 as a judge, the same judge used for correctness in the main text. The judge compares the candidate with the reference answer for correctness and with a constructed context for groundedness. Empty answers receive zero for both. Correctness, retrieval hit, and answer rate for GLM-5.2, GPT-5.5, and GPT-5.4-mini are in Figure 2(c). All rates below use the full 100-question denominator. The error taxonomy gives hallucination priority, then labels a grounded answer as correct, retrieval error (wrong and annotated passage absent), or reasoning error (wrong despite annotated passage present). Consequently, "correct" in the error column can be lower than the standalone correctness count when an answer is judged correct but ungrounded.

D.3 RESULTS AND FAILURE ANALYSIS

Descriptive findings. Headline correctness, retrieval, and answer rates are in Figure 2(c). Table 5 adds groundedness and the error taxonomy. These remain paired descriptive outcomes, not con-

Table 5: Legal RAG Bench groundedness and error taxonomy on the fixed 100-question test split. Correctness, retrieval hit, and answer rate for GLM-5.2, GPT-5.5, and GPT-5.4-mini are in Figure 2(c). Percentages use $n = 100$. Error counts are hallucination (H), retrieval error (Ret), reasoning error (Reas), taxonomy-correct (C), and no answer (NA). These are legal retrieval-QA results, not LegalBench results.

Agent	Model	Grounded	n	Error counts
Code2Evidence	GLM-5.2	100	100	H0 / Ret0 / Reas4 / C93 / NA3
Code2Evidence	GPT-5.4-mini	95	100	H3 / Ret21 / Reas3 / C71 / NA2
Code2Evidence	GPT-5.5	98	100	H0 / Ret4 / Reas3 / C91 / NA2
DCI-Agent-Lite	GLM-5.2	97	100	H0 / Ret2 / Reas7 / C91 / NA0
DCI-Agent-Lite	GPT-5.4-mini	66	100	H0 / Ret8 / Reas5 / C53 / NA34
DCI-Agent-Lite	GPT-5.5	99	100	H0 / Ret4 / Reas5 / C90 / NA1

Table 6: Recorded generation usage and estimated cost for complete 100-question GPT runs. Costs exclude Grok-4.6 judging.

Agent	Model	Requests	Total tokens	Cost (USD)
Code2Evidence	GPT-5.4-mini	981	46,614,764	9.73
Code2Evidence	GPT-5.5	873	41,285,368	73.04
DCI-Agent-Lite	GPT-5.4-mini	411	2,929,093	2.00
DCI-Agent-Lite	GPT-5.5	674	9,349,763	5.89

trolled architecture effects: with the same 50-turn budget, the systems still differ in token usage, endpoint/runtime, prompt, tool interface, context management, and evidence handling, and there are no repeated runs or uncertainty estimates. DCI's empty GPT-5.4-mini answers (Section 4.3) still dominate that run's error column (NA34 in Table 5).

Operational failures. Code2Evidence has two empty answers in each of its GPT-5.4-mini and GPT-5.5 runs; all four are Azure content-filter errors. Its GLM-5.2 run has three empty outputs but no recorded execution error. DCI records one Azure content-filter failure for GPT-5.5, and its GLM-5.2 run has no empty outputs. The 34 empty GPT-5.4-mini DCI outputs are nevertheless marked "completed" with no run error, so they indicate an extraction or termination failure not captured by the launcher status. Table 5 uses the complete Grok-4.6 judgments (no judge failures).

Usage interpretation. The cost files apply recorded list-price schedules to generation calls and do not include judge cost, corpus preparation, or hardware. Moreover, the two runtimes account for cached tokens differently: Code2Evidence's total includes prompt and completion tokens while reporting cached tokens as a subset, whereas DCI's displayed total is the sum of prompt, cache-read, and completion fields. The dollar figures are therefore useful as run records, but token and cost differences should not be interpreted as a clean systems-efficiency comparison. Usage was not recorded for the GLM-5.2 runs.

E Open-Domain Wikipedia QA

The six-dataset protocol follows the knowledge-intensive QA setting of DCI-Agent (Li et al., 2026): Natural Questions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), Bamboogle (Press et al., 2023), HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), and MuSiQue (Trivedi et al., 2022). Evidence is the 2018 Wikipedia dump of Karpukhin et al. (Karpukhin et al., 2020), stored locally as a passage corpus (14.394 GB on disk). The corpus has 21,015,324 passages of about 100 words.

E.1 Data, splits, and sample

Three 50-question samples per dataset. Each MHQA run covers 300 questions, 50 per dataset. The first sample is the official released DCI-bench sample-50 file for each dataset; the other two are

Table 7: Released DCI-bench sample-50 files, the first of the three MHQA samples. "Orig. split" denotes the source dataset partition, identified by the question-ID prefix. n_{list} is the number of golds stored as lists; n_{multi} is the number of lists with more than one alias.

Dataset	Orig. split	n	ID prefix	n_{list}	n_{multi}
NQ	test	50	`test_`	8	5
TriviaQA	test	50	`test_`	4	4
Bamboogle	test	50	`test_`	0	0
HotpotQA	dev	50	`dev_`	7	0
2WikiMultiHopQA	dev	50	`dev_`	35	22
MuSiQue	dev	50	`dev_`	20	4
Total	—	300	—	74	35

additional randomly selected 50-question samples per dataset. A configuration reported as mean ± sample SD was run once on each of the three samples (900 questions in total); single-run results, including the searcher-count study and the span-metric tables below, use the released sample-50 files. The MHQA runner loads each file in order and does not shuffle, resample, or take a `-limit`. Bamboogle's original set has 125 questions; the config notes that the released dci-bench file is the sample-50 split used in DCI QA scripts, not the full 125.

Each row has only `query_id`, `query`, and `answer`. Gold may be a string or a list of aliases; scoring takes the maximum over aliases. Query IDs keep the original-split prefix: NQ, TriviaQA, and Bamboogle use `test_*`; HotpotQA, 2WikiMultiHopQA, and MuSiQue use `dev_*`. Table 7 records the released samples.

E.2 Agents, budgets, and scoring

Code2Evidence default. Unless an ablation says otherwise, Code2Evidence uses pipeline `multi_search`, three specialized searchers (`literal_keyword`, `paraphrase_hyde`, `hop_chain`), 15 searcher turns, 5 main-answerer turns, rescue disabled, context level L3. GLM-5.3 uses `-max-tokens 2048`; GPT-5.6-sol uses 16384. GLM-5.2 uses the same settings.

DCI-Agent-Lite. The DCI baseline is the official `dci-agent-lite` launcher on the same question samples and the same Wikipedia corpus. Tools are `read,bash`; max turns are 50; runtime context is `level3`. The prompt forbids web search and asks the agent to use `rg`. GPT DCI runs set thinking to `high`. This matches the 50-turn budget of Code2Evidence (three 15-turn searchers and 5 answerer turns, versus a single 50-turn loop).

Exact match and token F1. The span scorer is shared by Code2Evidence and DCI. Normalization lowercases, strips punctuation, deletes articles `a`/`an`/`the`, and collapses whitespace. Exact match is 1 iff the normalized prediction equals any gold alias. Token F1 is the maximum bag-of-tokens F1 over aliases. Empty predictions score 0. Code2Evidence extracts the first line after `Exact Answer:` or `FINAL ANSWER:`. DCI uses the same extractors and, if they fail, the first nonempty line of the final message. That fallback is why DCI exact-match is near zero: many answers are long prose whose first line is a preamble, not a span. **EM/F1 must not be used to rank Code2Evidence against DCI.** Meaning match, which reads the raw response, is the comparable answer metric.

Meaning match. Meaning match is judged by Grok-4.6 for every MHQA result. The judge sees up to 12,000 characters of the raw response, not the extracted span. Results reported as single runs (the searcher-count study and the diagnostics in Appendix G) are the Grok-4.6-judged run on the released sample-50 files, which is one of the three runs averaged in the mean ± SD results.

E.3 Main six-dataset results

Table 9 records DCI span metrics for completeness. They are *not* comparable to Table 8: GLM-5.2 DCI EM is 0.00 on all six datasets because the extracted span is usually a preamble line. Empty extracted spans: GLM-5.3 DCI 22/300, GPT-5.6-sol DCI 2/300, GLM-5.2 DCI 9/300.

Table 8: Code2Evidence exact match / token F1 (%), default architecture, single run on the released sample-50 files. Dense retrieval is off. Overall is the micro-average over 300 items, equal here to the macro-average of the six datasets.

System	NQ	TriviaQA	Bamb.	Hotpot	2Wiki	MuSiQue	Overall
GLM-5.2 Code2Evidence	38.00/51.28	28.00/60.99	58.00/73.65	48.00/64.93	60.00/74.62	22.00/33.98	42.33/59.91
GLM-5.3 Code2Evidence	26.00/39.42	24.00/58.44	54.00/72.15	42.00/66.63	50.00/70.05	22.00/43.65	36.33/58.39
GPT-5.6-sol Code2Evidence	36.00/48.58	32.00/64.35	58.00/75.16	50.00/66.69	52.00/67.65	32.00/52.82	43.33/62.54

Table 9: DCI-Agent-Lite exact match / token F1 (%), single run on the released sample-50 files. These numbers score the extracted span, usually the first line of a long final message. Do not compare with Table 8.

System	NQ	TriviaQA	Bamb.	Hotpot	2Wiki	MuSiQue	Overall
GLM-5.2 DCI	0.00/10.06	0.00/15.44	0.00/5.23	0.00/4.90	0.00/8.44	0.00/3.28	0.00/7.89
GLM-5.3 DCI	2.00/12.46	4.00/28.15	6.00/26.62	4.00/28.70	2.00/21.57	2.00/11.79	3.33/21.55
GPT-5.6-sol DCI	0.00/14.30	2.00/34.81	0.00/23.96	2.00/25.75	0.00/26.69	2.00/18.77	1.00/24.05

Meaning match for GLM-5.3 Code2Evidence and GLM-5.3 DCI is in Figure 2(b), as mean $\pm$ sample SD over the three samples.

E.4 Architecture ablations (GLM-5.3)

All ablations share GLM-5.3, 15 searcher turns, 5 answerer turns, rescue 0, L3, dense off, and `-max-tokens 2048`. The span metrics below are from the single runs on the released sample-50 files; the Grok-4.6 meaning match in Table 3 averages three runs, one per sample. The $n = 3$ specialized cell is the run in Table 8, not a second draw.

Mechanical merging has the highest EM (38.00) but lower meaning-match accuracy than the default (Table 3): span match rewards short extracted answers that the meaning-match rubric rejects or that the default expresses in a longer form. Per-dataset ablation EM appears in Table 11.

E.5 Execution accounting and limitations

Recorded execution flags (not empty answers). Code2Evidence items often set `error=fallback_searcher_proposal` while still emitting an answer; this is a searcher-side fallback, not a dropped example. Counts on the $n = 300$ default runs: GLM-5.3 default: 51 flags (39 fallback, 12 `main_max_turns_exceeded`) and 5 empty predictions; GPT-5.6-sol Code2Evidence 60 flags (24 fallback, 36 max-turns) and 19 empty; GLM-5.2 Code2Evidence 17 empty (15 of them on MuSiQue) plus 33 fallback and 5 max-turns. DCI launcher status is `completed` for 300/300 GLM-5.3 and GLM-5.2 items; GPT-5.6-sol has one `failed` item with an RPC/node crash in `error`. Empty DCI spans are extraction failures, not missing rows.

What this section does not claim. Each run uses 50 questions per dataset; across-run SDs over three samples are reported where available, but we report no binomial intervals. With the same 50-turn budget, Code2Evidence and DCI still differ in tool API, notebook, prompt, and answer extraction. Meaning match relies on one LLM judge, Grok-4.6. No MHQA run here enables the optional dense retriever, so these numbers are shell-only.

F BioProBench

BioProBench (Liu et al., 2026a) tests whether a model can *use* a large library of wet-lab protocols rather than recite biomedical trivia. We treat it as a cross-domain stress test of Code2Evidence: the same search-then-synthesize agent operates over a file-system corpus of protocols instead of Wikipedia dumps or SEC filings, and is scored with the official BioProBench task parsers.

Table 10: Completed GLM-5.3 architecture ablations: exact match and token F1 (%) on the released sample-50 files (300 questions, single runs). Meaning match for these variants is in Table 3 and Figure 3(b).

Variant	EM	F1	Notes
Single agent, no notebook	31.67	50.80	single loop, no notebook
Single agent, with notebook	31.33	52.16	single loop, notebook on
$n = 3$ identical roles	34.33	54.87	three copies of one prompt
$n = 3$ specialized, no reconcile	38.00	58.56	mechanical notebook merge
$n = 2$ specialized	35.67	57.12	truncated role set
$n = 3$ specialized	36.33	58.39	default
$n = 4$ specialized	36.67	56.42	roles cycle

Table 11: GLM-5.3 ablation exact match (%) by dataset, $n = 50$ each. Default is $n = 3$ specialized.

Variant	NQ	TriviaQA	Bamb.	Hotpot	2Wiki	MuSiQue
Single, no notebook	18.00	22.00	54.00	40.00	38.00	18.00
Single, notebook	14.00	24.00	48.00	38.00	38.00	26.00
$n = 3$ identical	26.00	22.00	56.00	38.00	42.00	22.00
$n = 3$ no reconcile	30.00	28.00	58.00	44.00	46.00	22.00
$n = 2$ specialized	30.00	24.00	54.00	42.00	44.00	20.00
$n = 3$ specialized	26.00	24.00	54.00	42.00	50.00	22.00
$n = 4$ specialized	28.00	28.00	52.00	42.00	46.00	24.00

F.1 SETUP

Tasks and metrics. We evaluate the four BioProBench test categories that the official evaluation script implements (profile `bioprobench`):

- **PQA** (protocol question answering): multiple-choice fill-in of a protocol parameter. Metrics: accuracy and Brier score (lower is better). Answers must copy a choice verbatim and attach a confidence in $[0, 100]$, wrapped as `[ANSWER_START]...[ANSWER_END]`.
- **ORD** (step ordering): recover the canonical order of shuffled procedure steps as a Python list of original indices. Metrics: exact match and Kendall's τ.
- **ERR** (error detection): decide whether a target step is valid (`True`) or corrupted (`False`). Metrics: accuracy, precision, recall, and F1. The evaluation script's positive class is `has_error=true` (gold `is_correct=false`).
- **GEN** (protocol generation): emit a numbered procedure. We report the official *core* lexical metrics only (exact match and token precision/recall/F1). Semantic metrics (BERTScore, embedding similarity) were not run; GEN evaluation uses `-offline`. Exact match is 0 in every complete run we scored.

The public test pools contain 1200 PQA, 1161 ORD, 1200 ERR, and 772 GEN items. All Code2Evidence and DCI numbers in this section are on **fixed** n=100 **subsamples per task**, drawn independently with `random.Random(seed).sample` at seeds 35, 42, and 50. Each subsample has 100 unique IDs. Example heads: `TEST-PQA-001123` (seed 35), `TEST-PQA-000228` (seed 42), `TEST-PQA-001018` (seed 50). PQA/ORD/ERR IDs share numeric suffixes within a seed; GEN IDs are sampled separately.

Every subsample contains exactly one ORD item whose gold `correct_steps` are not unique, so Kendall's τ and exact match are undefined for that row (`TEST-ORD-001088` on seed 35, `TEST-ORD-000600` on seed 42, `TEST-ORD-000459` on seed 50). ERR class balance is mildly error-heavy: $57/43$, $54/46$, and $52/48$ corrupted/valid steps on seeds $35/42/50$.

Corpus. The retrieval corpus is the concatenation of BioProBench's protocols.io and Protocol Exchange collections, materialized as 10,307 **unique text documents**, one per protocol, with `ID`, `TITLE`, and `SOURCE` headers. The corpus manifest lists 10,518 entries, of which 211 are duplicates. Code2Evidence searches this corpus without an embedding index.

Table 12: BioProBench secondary metrics omitted from Table 2: Code2Evidence ERR F1 and GEN token F1 (mean±s.d. over seeds $\{35, 42, 50\}$) and DCI GEN token F1 (mean over the same seeds).

Backbone	C2E ERR F1↑	C2E GEN tok-F1↑	DCI GEN tok-F1↑
GLM-5.3	0.729±0.050	0.249±0.005	–
GPT-5.5	0.702±0.037	0.295±0.006	0.275
GPT-5.4-mini	0.687±0.050	0.304±0.002	0.312

Table 13: Single-seed (35) GPT-5.6-sol (high) comparison on the BioProBench primary metrics.

		PQA		ORD		ERR
Backbone	Agent	Acc.↑	Brier↓	EM↑	τ↑	Acc.↑
GPT-5.6-sol (high)	C2E	0.770	0.189	0.515	0.811	0.790
	DCI	0.760	0.113	0.475	0.792	0.650

How Code2Evidence is adapted. The agent is the same three-searcher Code2Evidence used for FinQA and MHQA, not a BioProBench-specific planner. Domain adaptation is almost entirely in the *environment and output contract*:

1. **Files, not indices.** Searchers issue sandboxed `grep`/`rg`, `find`, `cat`, `head`/`tail`, `awk`/`sed`, `ls`, `wc`, `sort`, `uniq` over the protocol directory. Pipes, redirects, and destructive commands are blocked; grep alternation with `\|` is allowed. Commands citing another corpus location are redirected to the configured corpus.
2. **Task playbooks, not a new controller.** Each searcher is told what evidence to bank: PQA looks for the sentence that states a dose/volume/time; ORD cats one contiguous numbered procedure; ERR banks the target step plus neighbors; GEN prefers one full procedure over scattered snippets.
3. **Default architecture (complete runs).** Three specialized searchers (literal keyword, paraphrase/HyDE, procedure structure) run in parallel for up to 15 shell turns each. A mechanical first grep is executed per role before the LLM loop. Hits are written into a per-searcher `[BANK]` notebook (capped at 14,000 characters after merge). An answerer that *cannot* issue shell commands answers from the merged notebook only, with at most 5 turns, for a total budget of 50 turns. Idle searchers collapse every 5 non-banking turns. Decoding is greedy (T=0).
4. **Answer tags.** The answerer must wrap its answer in BioProBench's `[ANSWER_START]`/`[ANSWER_END]` tags, with a one-shot format retry and a last-ditch regex coerce for PQA/ORD/ERR.

DCI baselines use the same on-disk protocol files through a coding agent with `read` and `bash` tools and the same 50-turn budget. All complete runs use the three seeds 35, 42, and 50, and mean ± sample SD is taken over these three runs.

F.2 Generation and secondary ERR metrics

PQA accuracy, Brier, ORD exact match, Kendall's τ, and ERR accuracy for GLM-5.3, GPT-5.5, and GPT-5.4-mini are in Table 2; the seed-35 GPT-5.6-sol pair is in Table 13. Table 12 adds ERR F1 and GEN token F1 for the complete Code2Evidence backbones in that table. Headline PQA/ERR/GEN scores count unparseable outputs as incorrect over all 100 items; ORD exact match uses 99 items after dropping one duplicate-gold row per seed. ORD has 93–99 scored items per 100 for the backbones in Table 12. GEN exact match is 0 throughout. GPT-5.5 GEN seed 35 includes one scored safety-blocked empty row (`TEST-GEN-000169`).

GLM-5.3 PQA is fully parseable (100/100 on each seed). ERR F1 is high relative to accuracy because recall on the error class is high and the subsamples are error-majority. GEN token F1 is stable near 0.25 for GLM-5.3, with high recall and low precision: the answerer dumps long retrieved procedures that overlap the gold tokens but do not match the gold document.

GPT-5.4-mini has the lowest three-seed PQA among the Code2Evidence backbones in the main table but has the best GEN token F1 (0.304); its generations are shorter and more precise (token precision ≈ 0.37 vs. GLM-5.3 ≈ 0.19).

F.3 BASELINES

The comparison holds the backbone and the 50-turn budget fixed and reports completed three-seed runs. It is not compute-matched in tokens: Code2Evidence splits its budget across three 15-turn shell workers and an answerer, while DCI uses one sequential coding-agent loop.

Same backbones, DCI vs. Code2Evidence. When the backbone can already follow the answer schema, Code2Evidence is not uniformly better. The GPT-5.5 and GPT-5.4-mini PQA/ORD/ERR comparison is in Table 2. The secondary metrics there use valid-output subsets: Brier averages confidence-bearing PQA outputs, and Kendall's τ pools pairwise ordering agreement across valid permutations. Code2Evidence has 100 Brier samples per seed for every displayed backbone; DCI has 99/99/96 for GPT-5.5 and 78/79/83 for GPT-5.4-mini. Thus DCI's lower GPT-5.5 Brier and higher τ on their valid-output subsets do not imply higher full-sample PQA accuracy or ORD exact match.

Code2Evidence and DCI generation token F1 are in Table 12.

Because the systems differ in prompts, context management, and parallel search, these comparisons do not support a causal architecture interpretation.

F.4 QUALITATIVE OBSERVATIONS

We restrict examples to seed 42 GLM-5.3 PQA, where predictions, notebooks, and gold IDs are complete. Mean shell commands per question: 35.7 (min 8, max 45), i.e. the three searchers usually exhaust the 15-turn budget. $66/100$ notebooks contain the gold choice string; $43/100$ contain that string in a window that also has at least two distinctive question tokens *and* the answerer copies the gold choice.

`TEST-PQA-000457` is a clean hit. Gold is `TRI Reagent` for "Add 800 μl pre-cooled _____ solution into each tube and resuspend the tissue powder." The literal searcher banks a matching corpus document with the rationale that tissue powder is transferred into chilled tubes with Trizol (TRI Reagent). The answerer emits `[ANSWER_START]TRI Reagent & ...[ANSWER_END]`.

`TEST-PQA-000864` is a near miss that still scores as correct. The question asks for whole-blood centrifugation at ____$\times g$ *at room temperature*. Gold is `500`. The notebook quotes a protocol that centrifuges whole blood at $500 \times g$ *at* $4°C$ for 5 min. The answerer copies 500 and ignores the temperature clash. String-level PQA accuracy does not measure this.

`TEST-PQA-000051` (zebrafish embryos per Petri dish; gold `50`) also scores correctly, but a naive "gold substring in notebook" check is misleading: the notebook's first `50` is "needle 50 times" from an unrelated mitochondrial isolation protocol. We do not count substring overlap as evidence of successful retrieval unless the surrounding tokens match the question.

F.5 COSTS

The combined Code2Evidence GPT-5.4-mini ledger covers all three seeds and four tasks: **$19.72** for 9,803 requests and 30.5M tokens, with three searchers and 15 turns per searcher. This is a recorded API estimate, not a compute-matched efficiency comparison.

GPT-5.5 GEN seed 35 includes a provider safety block on `TEST-GEN-000169` ("limited access to this content for safety reasons"), which is the unique empty GEN row in that complete 3×4 grid.

G ADDITIONAL SEARCH AND EVIDENCE DIAGNOSTICS

This appendix gives per-row paired statistics for the controlled comparisons (Appendix G.1), the found-then-lost breakdowns (Appendix G.2), and descriptive partitions of completed runs (Appendices G.3–G.4). The descriptive partitions are observational, not interventions. FinQA recounts use

the GLM-5.2 Code2Evidence runs on the three 150-question subsets (seeds 35/42/50), with 438 unique questions. The MHQA analyses use the Grok-4.6-judged GLM-5.3 runs on the released sample-50 files, the first of the three MHQA samples (Appendix E.2); for configurations in Table 3, this run is one of the three averaged there.

G.1 Paired statistics for the controlled comparisons

For each pair of settings evaluated on the same questions, we resample questions with replacement 10,000 times to obtain a 95% interval on the accuracy difference. We also report wins and losses (questions only one setting answers correctly) and an exact two-sided McNemar test on those discordant counts. For FinQA and BioProBench, the 450 and 3×100 evaluations are pooled across seeds; items repeated across seeds are treated as separate evaluations.

Table 14: Paired statistics for the GLM-5.3 MHQA runs on the released sample-50 files (one of the three Grok-4.6-judged sample runs), covering the comparisons of Table 3, and for the three settings of Table 1.

Task	Backbone	Comparison	Δ (pts)	95% CI	W/L	McNemar p
MHQA	GLM-5.3	notebook vs. history only	$+6.67$	$[+3.3, +10.0]$	24/4	0.0002
MHQA	GLM-5.3	evidence vs. trajectory (3 searchers)	$+4.12$	$[+1.1, +7.1]$	14/3	0.013
MHQA	GLM-5.3	specialized vs. identical roles	$+6.00$	$[+3.0, +9.0]$	20/2	0.0001
MHQA	GLM-5.3	reconciled vs. mechanical merge	$+5.33$	$[+2.0, +8.7]$	22/6	0.004
MHQA	GLM-5.3	3 searchers vs. 1 (notebook)	$+4.00$	$[+0.7, +7.3]$	20/8	0.036

G.2 Found, then lost

We use the saved GLM-5.3 MHQA single-agent runs (history only; with notebook), excluding the three yes/no questions ($n = 297$). A question is *seen* if a normalized gold alias occurs in the normalized 2,000-character preview of any shell observation. Previews are shorter than the observations the agent received, so "seen" is a lower bound. A question is *banked* if a gold alias occurs in the arguments of any `append_evidence` call. Correctness uses the Grok-4.6 meaning-match judgments of that run.

Table 15: Found-then-lost breakdown (GLM-5.3, MHQA, $n = 297$). Wrong: answered incorrectly despite the condition in that row.

Group	History only: wrong / n	Notebook: wrong / n
Answer seen in an observation	31 / 204 (15.2%)	25 / 200 (12.5%)
seen and banked	–	12 / 179 (6.7%)
seen, not banked	31 / 204	13 / 21 (61.9%)
Answer never seen	46 / 93 (49.5%)	32 / 97 (33.0%)

By observation volume. We split questions into thirds by the total pre-clip shell-observation length of the history-only run. The thresholds are 23.7K and 47.1K characters, with a maximum of 249K. Accuracy (history only $\rightarrow$ notebook) is:

- lowest third: 81.0% $\rightarrow$ 95.0% ($+14.0$);
- middle third: 85.0% $\rightarrow$ 86.0% ($+1.0$);
- highest third: 57.0% $\rightarrow$ 62.0% ($+5.0$).

Only one history-only run exceeded the 240K-character compaction trigger, so under the default compressor, loss of evidence happens mainly through per-result truncation and collapse steps, not compaction. A compression stress test that varies the collapse and compaction settings directly would test this mechanism; we have not run it. Two further differences between the settings should be kept in mind. First, the collapse step fired in 175 of 300 history-only runs and in all 300 notebook runs, because banking also triggers it. Second, the compression placeholders mention a notebook file even in the history-only setting.

Table 16: GLM-5.3 MHQA quality, evidence coverage, and cost (per-question means on the released sample-50 files, 300 questions; one of the three Grok-4.6-judged sample runs). MM is Grok-4.6 meaning-match accuracy (%); Tokens is thousands of logged tokens; Shell is shell calls summed over workers; Obs. is thousands of pre-clip observation characters; Entries is the number of notebook entries given to the answerer; E^{ans} is answer presence (%) in the merged notebook $\mathcal{B}_{\text{mech}}$ and in the answerer's notebook $\mathcal{B}^{\star}$.

Configuration	MM	EM	Tokens	Shell	Obs.	Entries	$E^{\text{ans}}_{\text{mech}}$	$E^{\text{ans}}_{\star}$
Single agent, no notebook	74.33	31.67	78	8.4	44	–	–	–
Single agent, notebook	81.00	31.33	103	8.7	50	2.3	69.4	69.4
2 specialized	84.33	35.67	143	11.6	103	3.2	72.1	68.7
3 specialized (default)	85.00	36.33	208	17.6	149	5.4	73.4	71.7
4 specialized	85.67	36.67	282	23.7	205	6.7	74.7	71.7
5 specialized	86.00	37.33	350	29.4	253	9.0	76.8	73.4
3 identical roles	79.00	34.33	206	17.5	152	4.6	72.1	70.0
3 specialized, mechanical merge	79.67	38.00	215	17.7	154	6.3	73.4	73.4
3 specialized, trajectories shown	–	34.33	214	17.9	151	–	73.4	–

Provenance. All groupings below were recounted from the saved per-question predictions.

G.3 Searcher participation and notebook state

We count a contributing searcher when its saved `banked_any` flag is true. Appendix H.2 reports accuracy by contributor count, per-role banking rates, notebook size, and the coordinator recovery flag. The zero-contributor group includes 18 evaluations that never launch searchers. This definition counts participation, not independently verified corroboration: different workers can bank overlapping or incorrect material. The groups are mutually exclusive and cover all 450 evaluations.

Similar marginal role participation does not establish the presence or absence of specialization by question type; that would require conditional analyses of evidence content or unique contributions. The MHQA role-configuration comparison in Table 3 addresses a different question.

Final-notebook size counts the numbered evidence entries in the saved `final_evidence_notebook`; Appendix H.2 reports all five groups. Entries are not necessarily distinct facts or correct operands. The coordinator analysis uses the saved `brain.recovered` flag. An unset flag does not certify that the first retrieved filing was correct. Both notebook size and the recovery flag expose potentially useful failure-analysis signals, but no predictive threshold or intervention is validated by these post hoc groups.

G.4 Measured quality, evidence, and cost

We measure three quantities separately rather than combining them into one objective. *Quality* is the benchmark answer metric. *Cost* is logged prompt-plus-completion tokens summed over all logged calls, together with the number of shell calls and the total pre-clip observation length. *Evidence coverage* would ideally be the fraction of required facts supported by some evidence entry. The MHQA samples do not annotate required facts, so we use an answer-presence proxy,

$$E^{\text{ans}}(\mathcal{B}, q) = \mathbf{1}\big\{\exists\, y \in \text{Aliases}(y^{\star}) :\ \text{norm}(y) \sqsubseteq \text{norm}(\mathcal{B})\big\},$$

where norm is the exact-match normalizer and $\sqsubseteq$ is substring containment; yes/no questions are excluded. E^{ans} is an upper-bound proxy: it ignores intermediate hops and can fire on unrelated mentions (Appendix F.4). Evaluating it on both $\mathcal{B}_{\text{mech}}$ and $\mathcal{B}^{\star}$ separates what the searchers found from what reaches the answerer. Table 16 reports all three quantities for the GLM-5.3 MHQA configurations of Section 4. All values are per-question means over the same 300 questions and are recomputed from the saved per-question predictions. Tokens sum every logged call (searchers and answerer; the single-agent loop answers in its own history). The reconciler call's usage is not logged and is excluded; its input is capped at 8,000 characters per worker notebook and its output at 1,800 tokens. E^{ans} excludes the three yes/no questions ($n = 297$).

Cost. Cost scales nearly linearly with searcher count at a fixed 15-turn budget: from two to five searchers, tokens rise from 143K to 350K and shell calls from 11.6 to 29.4, while meaning-match accuracy rises by 1.67 points. Moving from the default three searchers to five adds 142K tokens (+68%) for one point. Enabling the notebook in the single-agent loop adds 25K tokens (+32%) and 6.67 points. Configurations with three searchers have nearly identical cost, so the role-specialization and merge-mode comparisons are cost-matched to within 5%.

Evidence coverage. Answer presence in the merged notebook varies little across multi-searcher configurations (72.1–76.8%) and increases with searcher count. Identical and specialized roles reach similar coverage (72.1% vs. 73.4%) at the same cost but differ by 6.00 points in meaning-match accuracy, so that difference is not explained by whether the answer string was found. Reconciliation reduces the entries passed to the answerer and lowers answer presence from $\mathcal{B}_{\text{mech}}$ to $\mathcal{B}^\star$ by 1.7 points in the default configuration (5 questions); the mechanical merge preserves it by construction. The proxy is informative but incomplete. In the default run, meaning-match accuracy is 93.0% when $E_\star^{\text{ans}} = 1$ (213 questions) and 64.3% otherwise (84 questions), but 54 of the 252 correct answers lack the literal answer string in $\mathcal{B}^\star$, through paraphrase, derivation, or parametric knowledge. These are single runs without uncertainty estimates, and differences of one or two points are within plausible run-to-run variation.

Correct answers do not by themselves establish successful evidence retrieval. The artifact-grounded cases in Appendix F.4 illustrate this distinction: a correct answer can coexist with missing or mismatched evidence in the notebook.

H Additional System-Level Results

This appendix adds the search-depth and searcher-count studies and observational diagnostics to the system-level results of Sections 4.3–4.4. Like those comparisons, these results characterize the full Code2Evidence system and do not isolate the evidence state.

H.1 Search depth and searcher count

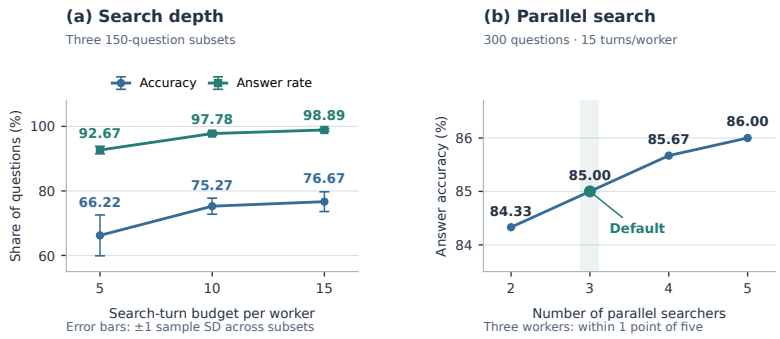


Figure 3: Search depth and parallelism. **(a) Searcher turns (FinQA):** accuracy gains 9.05 points from five to ten turns and reaches 76.67% at fifteen turns, with a 98.89% answer rate. Error bars show sample SD across three 150-question subsets. **(b) Parallel search (MHQA):** single Grok-4.6-judged run on the released sample-50 files (300 questions), the first of the three MHQA samples; accuracy increases from 84.33% with two searchers to 86.00% with five. The highlighted three-searcher default reaches 85.00% in this run using 40% fewer allocated searcher turns than five workers.

FinQA searcher-turn budget. Increasing the per-searcher budget from 5 to 10 turns raises mean accuracy from $66.22 \pm 6.34\%$ to $75.27 \pm 2.49\%$, and 15 turns reach $76.67 \pm 3.06\%$ (Figure 3(a)). The first five additional turns deliver 9.05 of the 10.45-point total gain (about 87%); the next five add 1.40 points. Answer rate rises from 92.67% through 97.78% to 98.89%, and the across-subset SD is smaller at ten and fifteen turns (2.49 and 3.06) than at five (6.34). Since accuracy rises by more than ten points while the unanswered share falls from 7.33% to 1.11%, the extra depth improves answers and not only availability.

Table 17: FinQA Code2Evidence (GLM-5.2) accuracy by final-notebook size, number of contributing searchers, and filing-recovery flag (observational; 450 evaluations over the three subsets). Groups contain different questions, so differences between groups reflect question difficulty as well as notebook content.

Grouping	Group	Evaluations	Correct	Accuracy (%)
Final-notebook entries	0	22	9	40.9
	1–2	36	25	69.4
	3–4	337	267	**79.2**
	5–6	48	31	64.6
	7 or more	7	2	28.6
Contributing searchers	1	15	9	60.0
	2	37	23	62.2
	3	376	293	**77.9**
Filing recovery	flagged	106	72	67.9
	not flagged	344	262	**76.2**

Number of searchers. In a single Grok-4.6-judged run on the released sample-50 files, the first of the three MHQA samples, accuracy rises monotonically from 84.33% with two searchers to 85.00%, 85.67%, and 86.00% with three, four, and five (Figure 3(b)). Three searchers cover the complete literal, alias/paraphrase, and hop-chain role set; larger ensembles repeat roles and extend the total search budget. The three-searcher default comes within one point of the best score with 40% fewer searchers and allocated searcher turns at a fixed per-worker budget, while five searchers remain an option when accuracy matters more than cost.

Cost. Total search budget grows with the number of searchers: tokens rise from 143K to 350K per question from two to five searchers (Table 16), while the fourth and fifth searchers add 0.67 and 0.33 points. Because the searcher-count study is a single run, one-point differences are within plausible run-to-run variation; for comparison, the three-run SD of the three-searcher default is 1.03 points (Table 3).

H.2 Notebook size and searcher participation (observational)

Notebook size. The saved GLM-5.2 FinQA runs (450 evaluations over the three subsets, 74.22% accuracy overall) show successful answers concentrated around compact evidence records (Table 17). The three-to-four-entry group contains 74.9% of the evaluations and 79.9% of the correct answers. We report this only as an association. Notebook size is not assigned; it results from the search. Harder questions may lead to both larger notebooks and lower accuracy, for example through many partial or conflicting matches. Questions with no entries include failed filing acquisition. The pattern therefore does not show that compact notebooks *cause* higher accuracy. Testing that would require changing notebook size while holding the question and search fixed, for example by capping entries or injecting distractor entries into saved notebooks before re-answering. We have not run that experiment.

Searcher participation. When all three searchers bank evidence, accuracy is 77.9% $(293/376)$. With exactly one or two contributors it is 60.0% $(9/15)$ and 62.2% $(23/37)$. Among the 432 evaluations that launch searchers, each role banks evidence in 93.1–94.7% of cases. Questions flagged for filing recovery reach 67.9% $(72/106)$, compared with 76.2% $(262/344)$ for unflagged ones. All of these groupings are observational, for the same reasons as above, but they match the intended workflow: explore with several strategies, then answer from a small retained evidence set.

Tokens. Code2Evidence uses more tokens than DCI in aggregate: $150{,}705$ versus $113{,}060$ tokens per FinQA question, about $1.33\times$. Appendix G.4 gives token accounting for the MHQA configurations.