WHEN DOES RECURSIVE ACCESS HELP? CONTEXT-NATIVE RECURSIVE LANGUAGE MODELS FOR MEDICAL EVIDENCE TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Medical evidence tasks often require locating a small amount of decision-relevant information within a collection too large for a single prompt, where the needed evidence rarely shares surface form with the query. Recursive Language Models (RLMs) address this by treating the evidence collection as a persistent, programmatically addressable environment that a controller can search and revisit across many steps, rather than compressing it into one prompt or relying on one-shot retrieval. We evaluate this access pattern on MIRAGE medical question answering (6,574 questions, four task types) and R2MED reasoning-driven retrieval (876 queries, 357,080 corpus documents, three task families), where RLMs are competitive with prior published methods and current leaderboard results on both medical benchmarks, and outperform standard retrieval-augmented generation and agentic search baselines under matched backbones and compute for R2MED. To understand the source of remaining errors in retrieval tasks, we introduce a trace-level diagnostic protocol that separates candidate misses, where relevant evidence is never encountered during search, from ordering misses, where evidence is found but subsequently lost or ranked too low to count, finding that ordering misses are the more common of the two. Motivated by this diagnosis, we show that a lightweight, weight-frozen self-improvement mechanism, which distills recurring trajectory failures into reusable retrieval policies, improves retrieval on the subsets it targets without any change to model parameters. Together, these results characterize not only how well recursive access performs on medical evidence tasks, but specifically where it succeeds and fails, and why, thus identifying evidence retention, rather than discovery, as the central bottleneck for future systems.

1 INTRODUCTION

Large language models answer fluently from what they memorized during training, but a growing share of useful tasks require answers grounded in evidence the model was never trained on, such as a document collection, knowledge base, or corpus that changes after training ends. In these cases, the limiting factor is not whether the model can reason, but how much of that evidence it can access. Long-context prompting places the full collection in the prompt, but performance degrades with input length and depends on the position of the relevant span (Liu et al., 2024; Hong et al., 2025; Du et al., 2025). Retrieval-augmented generation (RAG) instead selects a small evidence set before generation (Lewis et al., 2020), but this selection is fixed: evidence missing or mis-ranked at retrieval time cannot be recovered downstream (Barnett et al., 2024). Agentic RAG allows the model to issue further queries after the initial retrieval (Singh et al., 2025; Mishra et al., 2026), but querying remains its only mechanism for revising the evidence under consideration.

Recursive language models (RLMs) instead maintain the evidence collection as a persistent external environment, over which a controller searches, filters, and recursively delegates through executable code, revising a working set of candidates across many steps rather than fixing it at a single retrieval step (Zhang et al., 2025a). RLMs have so far been evaluated primarily on general long-context and coding tasks. Medical question answering and evidence retrieval present a more demanding setting: the terms used in a clinical query are often distant from those used in the relevant evidence, and corpora are large enough that reading them in full is infeasible under standard context or com-

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.

pute budgets. A retrieval error in this setting can mean that supporting evidence for a diagnosis or treatment claim was never found, or was found and subsequently lost.

Prior work on medical retrieval has improved upon static retrieval through agent specialization (Huang et al., 2026) and reasoning-intensive rerankers and embedding retrievers (Liu et al., 2026; Chen et al., 2026), with each approach reporting gains under its own backbone, training data, or retrieval stage. Despite these differences, they share a common assumption: evidence is a passive input, retrieved once and consumed as-is. However, the prior literature does not isolate whether persistent, programmatic access to evidence, i.e., the ability to revisit, filter, and reformulate, provides an independent benefit under a matched backbone, corpus, and compute budget. We address this question directly, treating access pattern as the variable of interest while controlling for architecture, training data, and retrieval stage.

In this work, we study the RLM access pattern on medical benchmarks of question answering (MIRAGE) and evidence retrieval (R2MED) through complementary analyses. We first position RLM against public MIRAGE and R2MED leaderboards and against the recent medical-domain systems (§5.1). We then isolate the access pattern itself in a controlled comparison against RAG and an agentic full-corpus baseline, matched in corpus and compute across two backbones and three recursion depths for medical retrieval tasks (§5.2). Since output-level metrics cannot distinguish evidence that was never found from evidence that was found and then lost or under-ranked, we use the RLM's recorded trajectory to classify each query's outcome as a top-10 success, an ordering miss, or a candidate miss; to test, for every judged-relevant document, whether it was exposed to the controller before submission; and to categorize recurring failure mechanisms from this combined signal (§3.2 and §5.3). To investigate whether this diagnosis can inform an intervention without retraining, we distill recurring failure patterns from a small development trajectories into a frozen, subset-specific retrieval recipe and evaluate it on held-out test queries from the development sample (§3.2 and §5.3). Our main contributions[1] are:

Leaderboard positioning. We evaluate RLM on medical benchmarks of question answering (MIRAGE) and evidence retrieval (R2MED) tasks against public leaderboards and recent medical-domain systems. RLM exceeds published results on MIRAGE (87.10% average accuracy) and ranks second overall on R2MED (43.08, within 0.10 points of the top entry).

Controlled comparison of access patterns. Under matched backbone, retrieval corpus, and compute budget, RLM outperforms both standard retrieval-augmented generation and an agentic full-corpus baseline on R2MED, with the largest gain at recursion depth 1 (+20.75 points over RAG on Qwen3.6-27B, +6.74 on Gemma 4 A4B), indicating that the advantage stems from the recursive access pattern itself rather than backbone scale or training data.

Trace-level diagnostic protocol. Applying our diagnostic protocol to RLM trajectories of retrieval tasks, we find that ordering misses (relevant evidence retrieved but not surfaced in the top 10) outnumber candidate misses (17.6% vs. 14.0% of queries), and exposure analysis shows many of these documents were inspected before submission, indicating evidence found and then discarded or misranked with failure categorization traces this to lexical over-matching as the single largest recurring mechanism; thus moving beyond aggregate metrics to explain *why* retrieval succeeds or fails.

Trace-guided self-improvement. We show that retrieval lessons distilled from failure traces can be frozen into a reusable, subset-specific recipe without any weight updates, improving retrieval on held-out queries across all three targeted R2MED subsets; evidence that trace-level diagnosis is actionable, not merely descriptive.

2 Related Work

Long-context and retrieval-augmented generation. Long-context prompting places evidence directly in the prompt, but performance degrades with input length and is sensitive to where relevant information sits within the context (Liu et al., 2024; Hong et al., 2025; Du et al., 2025). RAG instead retrieves a small evidence set before generation (Lewis et al., 2020), but bounds correctness entirely on a single, one-shot retrieval step: evidence missing or ranked below the cutoff cannot be recovered downstream (Barnett et al., 2024). Agentic RAG relaxes this by letting the model issue

[1] Our code repository is available at https://anonymous.4open.science/r/context-native-rlm.

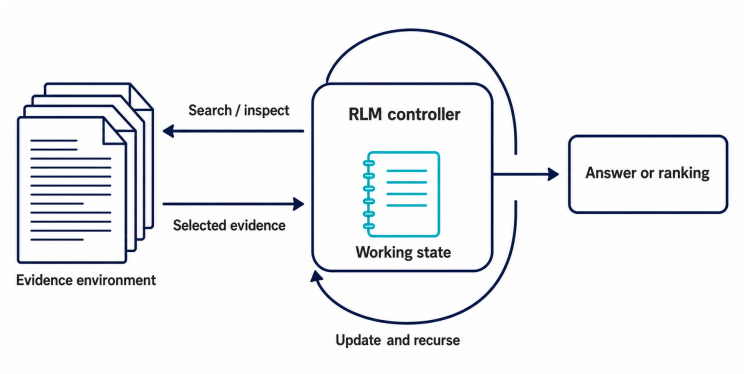


Figure 1: The RLM access pattern. A query drives repeated search and inspection of an external evidence environment; selected evidence updates the controller's working state until it emits a final answer label or ranked document list. Answer keys and relevance judgments are withheld from the controller throughout and are introduced only by the evaluator after the trajectory concludes.

further queries (Singh et al., 2025), but the model's only lever over evidence remains querying, not direct inspection, and iterative reformulation can drift from the original need (Mishra et al., 2026).

Recursive language models. RLMs instead give the model programmatic, persistent access to the evidence collection as an external environment, letting it inspect, filter, and recursively delegate over the context rather than only querying it (Zhang et al., 2025a). This capacity to hold and revise candidate evidence directly, rather than only through repeated querying, is what we isolate in our controlled comparison (§5.2).

Medical QA and retrieval. Recent medical-domain systems improve on static retrieval via agent specialization (Huang et al., 2026) or reasoning-intensive rerankers and embedding retrievers (Liu et al., 2026; Chen et al., 2026) (§4.2.2), while RLMs have so far been evaluated primarily on general long-context and coding tasks (Zhang et al., 2025a). We instead study the RLM access pattern directly in the medical domain, where the gap between a query's surface form and the evidence needed to answer it is often largest across both question answering and reasoning-driven retrieval. We additionally use the RLM's own recorded trajectory to distinguish evidence that was never found from evidence that was found but lost or under-ranked (§3.2), which output-only evaluation cannot separate.

3 Methodology

3.1 Recursive Language Model Access Pattern

Standard approaches to evidence-grounded QA either compress the evidence collection into the model's context window (long-context prompting) or select a fixed subset of it before generation (RAG). Both commit to an evidence set before reasoning begins. Recursive Language Models (RLMs) instead separate the evidence collection from the model's active text window, keeping it available as a persistent, external working environment that a controller can search, decompose, and revisit across many steps (Zhang et al., 2025a). Formally, let q denote the query, $\mathcal{D}$ the target evidence collection, and $S_0 = \mathrm{Init}(q, \mathcal{D})$ the initial execution state. At each step t, the controller produces an action $a_t = (r_t, c_t)$, a rationale paired with executable code, conditioned on the task specification I and its controller-visible history H_t:

$$\begin{aligned} a_t &\sim \pi_\theta(\cdot \mid I, H_t, t), \\ (S_{t+1}, o_t) &= \mathrm{Exec}(S_t, c_t), \\ H_{t+1} &= H_t \,\|\, (a_t, \rho(o_t)), \end{aligned} \tag{1}$$

where o_t is the execution observation, ρ bounds its controller-visible representation, $\|$ denotes history concatenation, and θ remains fixed throughout. Intermediate variables persist in S_t independently of H_t: a value can be computed, stored, and reused by later code even if it is never printed into

the history the controller reasons over. A final answer or ranking is produced only once this process concludes, yielding a trajectory $\tau = \{(a_1, o_1), \ldots, (a_T, o_T)\}$ for every run. Generated code can itself invoke the language model on selected evidence, delegating a subproblem to a further model call. We define *recursion depth* as the number of such delegation edges from the root controller to a descendant invocation (full delegation semantics in Appendix B). Figure 1 illustrates the resulting control-flow abstraction. Crucially, recursion here refers to repeated controller actions over a persistent evidence environment, not to self-referential function calls in the conventional programming sense.

3.2 Trace-Level Diagnostic Protocol

Output-level retrieval metrics cannot distinguish a query for which relevant evidence was never located from one for which it was located and subsequently discarded or mis-ranked. Because an RLM controller emits a trajectory $\tau = \{a_1, \ldots, a_T\}$, the ordered sequence of search, inspection, and candidate-formation actions preceding its final output, this distinction can be recovered directly from τ rather than inferred indirectly from scores alone. We therefore apply this protocol to retrieval traces: the ranked output makes intermediate decisions directly observable (each candidate either enters the top-100 pool or does not), whereas question answering tasks leave no analogous checkable artifact. The protocol comprises below components, each defined over tasks whose output is a ranked list of document identifiers scored against relevance judgments.

Outcome classification. Let R_q denote the top-100 ranking submitted for query q and G_q the set of judged-relevant document identifiers. We assign q to exactly one of three outcomes: *top-10 success* if $R_q[1{:}10] \cap G_q \neq \emptyset$; *ordering miss* if $R_q[1{:}100] \cap G_q \neq \emptyset$ but $R_q[1{:}10] \cap G_q = \emptyset$; *candidate miss* if $R_q[1{:}100] \cap G_q = \emptyset$.

Exposure analysis. For every $g \in G_q$, we test whether it occurs in τ prior to submission. *Exact-identifier exposure* holds if g's identifier appears verbatim in any action $a_i \in \tau$, detected via string matching over the serialized log. *Semantic-judgment exposure* is the strict subset of these cases in which the document is additionally passed as input to an LLM sub-call issued for relevance or ranking assessment, rather than merely appearing in a search result or candidate listing. Both are lower bounds: state held only in unprinted variables is not recoverable from τ, and exposure does not establish that the controller correctly interpreted the document. To relate exposure to outcome quality, we further subdivide top-10 successes by how completely and how well-ordered the relevant set was recovered: *partial top-ten retrieval* (some but not all judged-relevant documents reach the top 10), *full recall, imperfect order* (all reach the top 100 but not all the top 10), and *perfect nDCG@10* (the full relevant set is optimally ordered).

Failure categorization. Exposure analysis establishes that evidence is found and lost, but not the mechanism producing that loss. We characterize these mechanisms from execution traces via a codebook-then-apply protocol: an initial rubric is developed through open coding of a small trace subset, frozen, and then applied across a larger sample by model-based coders, each code anchored to a quoted, verifiable log span and checked by a human analyst. The sample is not conditioned on outcome, so correctly-resolved runs surface as counterexamples alongside failure modes (detailed in §4.3).

Trace-guided self-improvement. To test whether the access/retention diagnosis above can directly inform an intervention without retraining, we adapt Prime Agent's Continual Harness (Karten et al., 2026): completed trajectories on a small development sample are reviewed, and transferable retrieval lessons, not query-specific answers, are distilled into a persistent, subset-specific recipe. The recipe is then frozen and applied to held-out test queries from the same subset that do not overlap with the development sample. Full protocol details are given in Appendix C.

4 Experimental Set-up

4.1 Evaluation Benchmarks

To systematically evaluate RLM's performance and generalization capabilities across diverse medical scenarios, we select widely used datasets from the MIRAGE and R2MED benchmarks.

Table 1: Statistics of the medical QA datasets from MIRAGE used in our experiments.

Dataset	#Samples	Task
MedQA-US	1273	4-choice MCQ
MedMCQA	4183	4-choice MCQ
PubMedQA*	500	3-choice Y/N/M
BioASQ-Y/N	618	2-choice Y/N

Table 2: R2MED composition. Each query searches one dataset-specific corpus. "Mean rel." is the mean number of judged relevant documents per query.

Family	Subset	Queries	Corpus rows	Mean rel.
Q&A reference	Biology	103	57,359	3.63
	Bioinformatics	77	47,473	2.95
	Medical Sciences	88	34,810	2.77
Clinical evidence	MedXpertQA-Exam	97	61,379	3.01
	MedQA-Diag	118	56,250	4.42
	PMC-Treatment	150	28,954	2.10
Clinical case	PMC-Clinical	114	60,406	2.18
	IIYi-Clinical	129	10,449	3.54
Total		**876**	**357,080**	

MIRAGE Xiong et al. (2024) was introduced to evaluate retrieval-supported medical question answering (QA) across medical examination datasets with 4-choice multiple-choice questions (MedQA-US and MedMCQA) and two biomedical research QA datasets (PubMedQA, yes/no/-maybe, and BioASQ, yes/no). Together, these four datasets cover general medical knowledge, clinical examinations, and biomedical literature inference. We report accuracy as the evaluation metric and the metric does not assess whether the underlying rationale is grounded in the retrieved evidence. Please note that we exclude the fifth MIRAGE component, MMLU-Med, since it follows the same 4-choice MCQ format as MedQA-US and MedMCQA and draws from general MMLU rather than medical-licensing or entrance-exam sources. Table 1 summarizes the datasets.

R2MED Zhang et al. (2025b) was designed specifically for reasoning-driven medical retrieval. It contains 876 queries and 357,080 corpus documents across eight datasets, grouped into three task families: Q&A reference retrieval (Biology, Bioinformatics, Medical Sciences), clinical evidence retrieval (MedXpertQA-Exam, MedQA-Diag, PMC-Treatment), and clinical case retrieval (PMC-Clinical, IIYi-Clinical). Unlike MIRAGE, R2MED evaluates the ranking of retrieved evidence rather than the final answer, using nDCG@10 as the primary metric and Recall@100 as a diagnostic of candidate coverage. Each query searches only its assigned corpus, and relevant document identifiers, rather than a generated diagnosis or treatment recommendation, are scored. Table 2 summarizes the eight corpus-specific settings.

4.2 Models and Baselines

4.2.1 Backbone Models

We instantiate our RLM controller on two publicly accessible backbones. Our primary backbone is **Qwen3.6-27B**, a dense 27B-parameter model with a hybrid gated-attention architecture and a 262K native context window. To test generalization across model families and parameter scales, we additionally evaluate a **Gemma**-family backbone (Gemma 4 A4B), under otherwise identical experimental conditions.

4.2.2 Baselines

Reference baselines. are drawn from public leaderboards and recently published work, evaluated under their own original backbones, corpora, and retrieval configurations; we report these to situate our results within the broader field rather than as a substitute for the controlled comparison.

MIRAGE: We report the top entries from the public MIRAGE leaderboard (Xiong et al., 2024), alongside results from SEMA-RAG (Huang et al., 2026), a recent multi-agent RAG framework. For

SEMA-RAG, we report two configurations: its best-performing backbone and its *qwen3-coder-plus* configuration, the closest available match to our own Qwen backbone in model family.

R2MED: We report the top entries from the public R2MED leaderboard (R2MED, 2026), alongside results from ReasonRank (Liu et al., 2026), a reasoning-intensive passage reranker applied on top of E5-mistral retrieval, and ReasonEmbed (Chen et al., 2026), a first-stage reasoning intensive embedding retriever whose strongest variant (Qwen3-8B, trained with Redapter) corresponds to the current top public leaderboard entry. Because these reference systems differ from our setup in backbone, retrieval stage, training data, and in some cases retrieval corpus, we report them as heterogeneous reference points (Table 3, Table 4) rather than treating any single score difference as evidence of a controlled improvement.

Controlled baselines. For R2MED, we compare against standard retrieval-augmented generation using BM25 retrieval; Vanilla Pi (earendil-works, 2025), an open-source token-efficient coding agent used here as a full-corpus-access agentic baseline without RLM's recursive delegation structure; and RLM at recursion depths $\{0, 1, 2\}$. All conditions share the same backbone, retrieval corpus, and compute budget, enabling a direct comparison of access patterns (Table 5).

4.3 Implementation Details

Task I/O. For retrieval (R2MED), the controller receives the query and complete target corpus as typed inputs (document identifier and text), and must return a ranked list of up to 100 document identifiers; relevance judgments are withheld. For classification (MIRAGE), the controller receives the question, the answer choices when applicable, and a fixed, question-independent slice of the MedRAG Textbooks corpus as its evidence collection (Xiong et al., 2024), and must return one label in the dataset's native answer space. In both cases, the controller sees only variable descriptions and bounded previews at initialization and obtains further evidence by executing code against the corpus, which remains accessible throughout the trajectory (full I/O schema and output-validation rules in Appendix D and RLM prompts are present in Appendix E).

Recursion depth ablation. We enforce depth caps of zero, one, or two by extending the dispatcher's delegation limit (depth defined in §3.1). At depth zero the controller retains its execution environment but cannot delegate; at depth one, delegated children are direct model calls; at depth two, a child receives a fresh execution environment and may itself inspect and delegate before returning to the parent. The depth-zero condition retains persistent state and full-corpus access and is therefore distinct from controlled baselines (§4.2.2); enforcement mechanics are given in Appendix D.

Failure categorization. Sample selection, codebook development, and verification are detailed in Appendix F.

Decoding and compute. Controller and delegated calls use the same backbone with deterministic (greedy) decoding. Each condition is evaluated in a single run per query; trajectories that failed to return a valid output were re-executed, and only the final accepted run is scored. All models are served locally on NVIDIA A100 (40GB/80GB) GPUs; execution bounds are listed in Appendix D.

Logging. We log all controller actions, execution observations, delegated model histories, and final predictions, forming the trajectory τ used by the diagnostic protocol in §3.2.

5 Results

5.1 Comparison against Reference Baselines

MIRAGE Question Answering. Table 3 situates RLM (Qwen3.6-27B, recursion depth 1) against the public MIRAGE leaderboard and recent published methods. Our RLM attains the best reported score on three of the four datasets: MedMCQA (80.30), PubMedQA (84.00), and BioASQ-Y/N (91.10). The PubMedQA result is notable given the task requires preserving an explicit uncertainty label (yes/no/maybe) rather than defaulting to a binary judgment, suggesting the RLM's iterative inspection helps it weigh partial or conflicting evidence rather than commit early. LinCogRAG is marginally stronger on MedQA-US (93.40 vs. 92.99); however, our RLM attains the highest overall accuracy (87.10) averaged across all four datasets, ahead of the current leaderboard #1 entry (83.99) and both SEMA-RAG configurations (77.35, 74.85).

Table 3: Comparison (Accuracy) of our RLM (Qwen3.6-27B, depth 1) against leaderboard and recent baselines on MIRAGE.

Source	System	LLM	Corpus	MedQA-US	MedMCQA	PubMedQA*	BioASQ-Y/N	Avg.
Leaderboard #1	LinCogRAG	GPT-5-Mini	PubMed	**93.40**	79.51	72.60	90.45	83.99
Leaderboard #2	GPT-4 + MedRAG	GPT-4-32k	MedCorp	82.80	66.65	70.60	92.56	78.15
Leaderboard #3	Llama-3 + MedRAG	Llama-3-70B	MedCorp	76.90	68.87	71.60	89.97	76.84
ACL, 2026	SEMA-RAG	Gemini-2.0-Flash	Textbooks+StatPearls	90.42	71.60	59.20	88.19	77.35
ACL, 2026	SEMA-RAG	Qwen3-Coder-Plus	Textbooks+StatPearls	86.17	74.23	56.00	83.01	74.85
Ours	**RLM**	**Qwen3.6-27B**	MedRAG Textbooks	92.99	**80.30**	**84.00**	**91.10**	**87.10**

Table 4: Comparison (nDCG@10 (%)) of RLM (Qwen3.6-27B, depth 1) against leaderboard and recent baselines on R2MED. Overall is an equal-weight macro-average across all 8 subsets.

	Q&A Reference				Clinical Evidence				Clinical Case			
System	Biology	Bioin.	MedSci	Avg.	MedExam	MedDiag	PMC-Tx	Avg.	PMC	IIYi	Avg.	Overall
E5-mistral (7B), no rerank (ACL,2026)	18.28	41.47	41.01	33.59	6.44	11.38	19.81	12.54	30.97	21.37	26.17	23.84
ReasonRank (32B) reranker (ACL+Ldbd. #2)	45.56	**67.73**	63.45	58.91	18.90	30.60	41.08	30.19	46.11	**29.35**	**37.73**	42.85
o3-mini + NV-Embed-v2 (Ldbd. #3)	34.01	55.90	51.28	47.06	**28.99**	40.30	48.97	39.42	**50.86**	20.47	35.67	41.35
ReasonEmbed-Llama-3.1-8B (ACL,2026)	53.52	64.37	63.82	60.57	20.40	29.67	51.86	33.98	34.55	23.88	29.22	42.76
ReasonEmbed-Qwen3-8B (Ldbd. #1)	**54.01**	66.33	**67.64**	**62.66**	20.93	27.96	51.38	33.42	33.76	23.43	28.60	**43.18**
Ours (RLM)	50.00	47.63	52.07	49.90	27.47	**47.38**	**60.19**	**45.01**	42.59	17.27	29.93	*43.08*

R2MED Retrieval. As shown in Table 4, RLM (Qwen3.6-27B, depth 1) leads the Clinical Evidence family average (45.01), where inferring a latent diagnosis or clinical relation from an underspecified query rewards iterative search reformulation. It trails on Q&A Reference (49.90 vs. 62.66, ReasonEmbed-Qwen), which favors a strong pretrained embedding space, and on Clinical Case (29.93 vs. 37.73, ReasonRank), driven by IIYi-Clinical, where superficially similar same-diagnosis narratives appear to induce early hypothesis commitment. Nonetheless, averaged across all eight subsets, our RLM attains the second-highest overall score (43.08), within 0.10 points of the leaderboard #1 entry (43.18) and ahead of the strongest reranking baseline ReasonRank (42.85), indicating that its advantage concentrates in clinically inferential retrieval rather than holding uniformly across all settings. Per-subset nDCG@10, Recall@100, and oracle headroom are reported in Table 8.

5.2 Comparison against Controlled Baselines

Table 5 isolates the effect of the RLM access pattern under controlled baselines across two backbones (Qwen and Gemma families) and three recursion depths on R2MED datasets.

RLM outperforms both baselines on every backbone. On Qwen3.6-27B, the best RLM configuration (depth 1, 43.08) exceeds RAG and Vanilla Pi by 20.75 and 20.04 points respectively. On Gemma 4 A4B, RLM depth 1 (29.08) again leads both baselines, though by a narrower margin (6.74 points over RAG, 10.76 over Vanilla Pi). Notably, Vanilla Pi performs comparable to RAG or even lesser on backbones (23.04 vs. 22.33 on Qwen; 18.32 vs. 22.34 on Gemma) despite Vanilla Pi having the same full-corpus access as our RLM (Table 5). This indicates that unstructured agentic access alone, without the RLM's recursive delegation, persistent candidate state, and bounded execution protocol, provides little advantage over simple static retrieval; the gain is attributable specifically to RLM's architecture, not merely to having some form of agentic or full-corpus access.

Depth: one level of delegation is sufficient. Within RLM, performance is non-monotonic in recursion depth on both backbones: depth 1 is the strongest configuration, and depth 2 underperforms it substantially (Qwen: 43.08 $\rightarrow$ 33.84; Gemma: 29.08 $\rightarrow$ 19.54), in the latter case falling below even RAG (Table 5). Depth 0, which retains persistent state but cannot delegate, already improves over both baselines on Qwen (30.70 vs. 23.04/22.33) and Gemma (25.61 vs. 18.32/22.34). These results, on our studied backbones, suggest that the benefit of recursion is not monotonic in delegation depth: a single level is sufficient to exploit the corpus, while a second adds coordination overhead that a weaker backbone in particular is unable to absorb.

5.3 Error Analysis

Outcome Classification Table 6 classifies every R2MED query into one of three outcomes (§3.2). RLM succeeds on 68.4% of queries, split between ordering misses (17.6%) and candidate misses (14.0%). Both failure types concentrate in Clinical Evidence and Clinical Case. MedXpertQA-

Table 5: Controlled comparison (nDCG@10 (%)) of access patterns across backbones, recursion depths, and controlled baselines on R2MED.

		Q&A Reference				Clinical Evidence				Clinical Case			
LLM	**Method**	Biology	Bioin.	MedSci	**Avg.**	MedExam	MedDiag	PMC-Tx	**Avg.**	PMC	IIYi	**Avg.**	**Overall**
Qwen3.6-27B	Vanilla Pi	26.94	20.35	47.77	31.69	7.39	27.91	32.77	22.69	8.08	13.10	10.59	23.04
	RAG	15.34	36.10	29.13	26.86	2.78	4.85	32.69	13.44	36.41	21.34	28.88	22.33
	RLM (depth = 0)	51.61	32.48	34.43	39.51	21.34	39.80	30.37	30.50	27.48	8.07	17.78	30.70
	RLM (depth = 1)	50.00	47.63	52.07	49.90	27.47	47.38	60.19	45.01	42.59	17.27	29.93	**43.08**
	RLM (depth = 2)	45.07	37.70	36.27	39.68	32.95	35.14	41.61	36.57	27.79	14.20	21.00	33.84
Gemma 4 A4B	Vanilla Pi	23.77	25.00	25.38	24.72	6.25	25.20	24.89	18.78	10.61	5.45	8.03	18.32
	RAG	16.01	36.48	30.03	27.51	2.78	4.58	31.98	13.11	36.15	20.73	28.44	22.34
	RLM (depth = 0)	51.62	9.37	28.83	29.94	12.30	45.05	26.44	27.93	19.97	11.32	15.65	25.61
	RLM (depth = 1)	32.29	32.78	49.01	38.03	18.72	29.32	43.49	30.51	12.70	14.33	13.52	**29.08**
	RLM (depth = 2)	18.01	23.04	24.92	21.99	11.75	25.19	27.45	21.46	14.71	11.25	12.98	19.54

Table 6: RLM vs. RAG outcome breakdown on R2MED.

		RLM			RAG		
Subset	**n**	Success	Ord. Miss	Cand. Miss	Success	Ord. Miss	Cand. Miss
Q&A Reference							
Biology	103	79 (76.7%)	16 (15.5%)	8 (7.8%)	33 (32.0%)	18 (17.5%)	52 (50.5%)
Bioinformatics	77	57 (74.0%)	14 (18.2%)	6 (7.8%)	43 (55.8%)	20 (26.0%)	14 (18.2%)
Medical Sciences	88	72 (81.8%)	10 (11.4%)	6 (6.8%)	40 (45.5%)	18 (20.5%)	30 (34.1%)
Clinical Evidence							
MedXpertQA-Exam	97	50 (51.5%)	22 (22.7%)	25 (25.8%)	5 (5.2%)	11 (11.3%)	81 (83.5%)
MedQA-Diag	118	88 (74.6%)	18 (15.3%)	12 (10.2%)	13 (11.0%)	16 (13.6%)	89 (75.4%)
PMC-Treatment	150	116 (77.3%)	15 (10.0%)	19 (12.7%)	69 (46.0%)	22 (14.7%)	59 (39.3%)
Clinical Case							
PMC-Clinical	114	76 (66.7%)	22 (19.3%)	16 (14.0%)	62 (54.4%)	30 (26.3%)	22 (19.3%)
IIYi-Clinical	129	61 (47.3%)	37 (28.7%)	31 (24.0%)	62 (48.1%)	38 (29.5%)	29 (22.5%)
Total	**876**	**599 (68.4%)**	**154 (17.6%)**	**123 (14.0%)**	**327 (37.3%)**	**173 (19.7%)**	**376 (42.9%)**

Exam and PMC-Treatment are the only subsets where candidate misses exceed ordering misses (25.8% vs. 22.7%; 12.7% vs. 10.0%): both request specific diagnostic tests or interventions, so a search anchored on the wrong latent diagnosis can miss relevant evidence entirely rather than rank it poorly. IIYi-Clinical has the highest ordering-miss rate (28.7%) and second-highest candidate-miss rate (24.0%), as its long case narratives share symptoms across diagnoses, excluding relevant cases outright or burying them among similar distractors. Q&A Reference subsets have the lowest candidate-miss rates (6.8–7.8%), since their queries name concepts closer to the reference material's own terminology. RAG shows a substantially higher candidate-miss rate under the same protocol (Table 6), indicating RLM's search contributes primarily to whether evidence is found at all.

Exposure Analysis We examine failure types via trace-level exposure analysis to determine whether the underlying evidence was ever visible to the controller before the final ranking. Table 9 shows that *Exact-identifier exposure* rises monotonically with outcome quality (29.3% for candidate misses, 72.7% for ordering misses, 91.4–98.8% across the three tiers of top-10 success), while *semantic-judgment* exposure follows the same trend but at consistently lower rates (8.9% to 54.3%). This gradient indicates that the exposure measures track genuine differences in how thoroughly the controller engaged with relevant evidence, rather than being a binary artifact of whether a query happened to succeed. Notably, exact-identifier exposure is already high for ordering misses (72.7%): the relevant document was visible to the controller in nearly three-quarters of these cases, yet failed to reach the top 10; evidence that the 17.6% ordering-miss rate reported in Table 6 reflects a retention or ranking failure more often than a genuine absence of access.

Trace failure mechanisms. Table 7 surfaces breakdowns at several points across the retrieval pipeline (Figure 2). The most frequent mechanism is candidate-stage over-matching (C1, 30/50): the controller retains a candidate on lexical overlap despite a visibly off-target hit. Root hypothesis injection into a delegate prompt (Q1, 24/50) and delegate judgments lost or altered later in execution (R2, 19/50) are also common, followed by unexecuted observations cited in reasoning (E1, 15/50) and final orderings that ignore available delegate judgments (K1, 13/50). These counts overlap and describe this 50-trace sample, not the prevalence of errors across the full benchmark. Two trajectories make the distinction between an observation and its use concrete. In Run 10, the root fabricated 50 delegate assessments before any sub-call was logged and used them in its submitted ranking. In Run 07, 34 documents the delegate scored 1/10 were reassigned an imputed score of 3 during ranking reconstruction and ranked above documents scored 2–3/10. The reverse also occurs: 13



Figure 2: **Observed mechanisms span the retrieval process.** Run-level counts from 50 Qwen3.6-27B depth-one traces; categories can co-occur. Arrows denote execution order, not causal links between mechanisms. The delegate judgments refer to logged assessments, not benchmark relevance labels.

runs retain a final ordering consistent with the delegate's own judgments (P4), the counterexample against which the mechanisms above are read. These comparisons test consistency with the model's own intermediate decisions, not agreement with relevance judgments.

Trace Guided Self Improvement. As a preliminary test of whether the diagnosis above can inform an intervention, we apply the self-improvement protocol (§3.2) to three low-scoring R2MED subsets. For each subset, a recipe is derived from a small development sample (12–17 queries), frozen, and evaluated on disjoint held-out test queries; baselines are recomputed on the same held-out query identifiers (Table 10 and Figure 3). The adapted controller improves nDCG@10 on all three subsets: IIYi-Clinical ($0.1626 \rightarrow 0.2883$, $+0.1257$), MedQA-Diag ($0.4855 \rightarrow 0.5684$, $+0.0829$), and PMC-Clinical ($0.4401 \rightarrow 0.4875$, $+0.0474$). Recall@100 shows where these gains come from. On IIYi-Clinical, Recall@100 rises sharply ($+0.2020$), indicating that the recipe brings substantially more relevant evidence into the candidate pool, i.e., improved access. PMC-Clinical shows a moderate Recall@100 gain ($+0.0286$), so its improvement reflects both better coverage and better ranking. On MedQA-Diag, Recall@100 is slightly lower (-0.0130), yet nDCG@10 still rises substantially, indicating that the gain comes from better ordering of evidence the baseline could already reach. Overall, adaptation benefits retrieval consistently: mean nDCG@10 increases from 0.3627 to 0.4480 ($+0.0853$) and mean Recall@100 from 0.7015 to 0.7740 ($+0.0725$), without any change to model parameters.

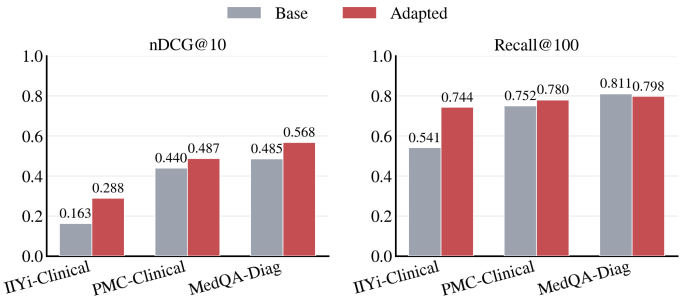


Figure 3: Trace-guided self-improvement on R2MED subsets.

6 Conclusion

We studied the recursive language model access pattern on medical question answering and evidence retrieval. RLM attains the best reported aggregate accuracy on MIRAGE, and ranks second overall on R2MED with outperforming controlled baselines across backbones on retrieval tasks. Using the RLM's own recorded trajectories for evidence retrieval, our trace-level diagnostic protocol separates candidate misses from ordering misses and finds that ordering misses evidence found and then lost or under-ranked are more common, with a catalog of recurring execution mechanisms explaining many of them. Motivated by this diagnosis, a preliminary, weight-frozen self-improvement mechanism that distills these mechanisms into a reusable retrieval recipe improved performance on the subsets it targeted. For medical evidence tasks, the primary bottleneck for recursive access is thus retention and ordering, not discovery, a distinction we expect to matter for designing future retrieval-oriented controllers.

REPRODUCIBILITY STATEMENT

We take the following steps to support reproducibility. All backbone models used in our experiments are publicly available: Qwen3.6-27B and Gemma 4 A4B are released on HuggingFace under their respective licenses. Hyperparameters, recursion depth configurations, compute budget constraints, and corpus details are reported in §4.3 and Appendix D. The failure categorization codebook, sampling procedure, and human adjudication protocol are described in Appendix F. The self-improvement recipe construction follows the Prime Agent Continual Harness (Karten et al., 2026), with subset-level query counts reported in § 3.2 and Appendix C. The anonymous code repository is available at [1].

AI USE STATEMENT

Large language models (LLMs) are central to the experimental pipeline reported in this work: we evaluate Recursive Language Models built on Qwen3.6-27B and Gemma 4 A4B as backbone models, and use Qwen3.6-27B as the model-based coder in our failure categorization protocol. in addition, Generative AI tools were used to assist with language editing tasks, including grammar correction, phrasing refinement, and clarity improvements during manuscript preparation. All technical aspects of the work, including experimental design, analysis, and conclusions, were developed and verified by the authors.

ETHICS STATEMENT

This work evaluates Recursive Language Models on publicly available medical benchmarks (MIRAGE and R2MED) and does not involve any collection, use, or processing of personal health information or patient data. All datasets used are released for research purposes under their respective licenses. The failure categorization analysis is conducted on model-generated retrieval traces, not on real clinical records or patient interactions.

The systems studied are research prototypes evaluated in controlled settings. We do not advocate for deployment of these models in clinical decision-making without appropriate validation, regulatory review, and human oversight. Improvements in retrieval performance on benchmark tasks do not imply clinical safety or efficacy. Results should be interpreted as contributions to the research literature on information retrieval and language model evaluation, not as evidence of readiness for medical practice.

REFERENCES

Scott Barnett, Stefanus Kurniawan, Srikanth Thudumu, Zach Brannelly, and Mohamed Abdelrazek. Seven failure points when engineering a retrieval augmented generation system. In *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering – Software Engineering for AI (CAIN)*, pp. 194–199, 2024.

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

Yufeng Du, Minyang Tian, Srikanth Ronanki, Subendhu Rongali, Sravan Bodapati, Aram Galstyan, Azton Wells, Roy Schwartz, Eliu A Huerta, and Hao Peng. Context length alone hurts LLM performance despite perfect retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025.

earendil-works. Pi Agent. `https://github.com/earendil-works/pi`, 2025. Accessed: September 2026.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models, 2022. URL `https://arxiv.org/abs/2211.10435`.

Kelly Hong, Anton Troynikov, and Jeff Huber. Context rot: How increasing input tokens impacts LLM performance. Technical report, Chroma, July 2025.

Yongfeng Huang, Ruiying Chen, and James Cheng. Sema-rag: A self-evolving multi-agent retrieval-augmented generation framework for medical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 18423–18442, 2026.

Seth Karten, Alex L. Zhang, Kevin Thomas, Sebastian Müller, Elie Bakouch, Daniel Auras, Mika Senghaas, Fares Obeid, Konstantin Dunas, Johannes Hagemann, and Sami Jaghouar. Prime agent: A self-improving RLM harness. *arXiv preprint arXiv:2608.23552*, 2026. doi: 10.48550/arXiv.2608.23552. URL https://arxiv.org/abs/2608.23552.

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.

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.

Wenhan Liu, Xinyu Ma, Weiwei Sun, Yutao Zhu, Yuchen Li, Dawei Yin, and Zhicheng Dou. ReasonRank: Empowering passage ranking with strong reasoning ability. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22007–22031, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1005. URL https://aclanthology.org/2026.acl-long.1005/.

Saroj Mishra, Suman Niroula, Umesh Yadav, Dilip Thakur, Srijan Gyawali, and Shiva Gaire. SoK: Agentic retrieval-augmented generation (RAG): Taxonomy, architectures, evaluation, and research directions. *arXiv preprint arXiv:2603.07379*, 2026.

R2MED. R2MED leaderboard. https://r2med.github.io/#leaderboard, 2026. Accessed: September 2026.

Aditi Singh, Abul Ehtesham, Saket Kumar, Tala Talaei Khoei, and Athanasios V. Vasilakos. Agentic retrieval-augmented generation: A survey on agentic RAG. *arXiv preprint arXiv:2501.09136*, 2025.

Guangzhi Xiong, Qiao Jin, Zhiyong Lu, and Aidong Zhang. Benchmarking retrieval-augmented generation for medicine. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 6233–6251, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models, 2022. URL https://arxiv.org/abs/2210.03629.

Alex L. Zhang, Tim Kraska, and Omar Khattab. Recursive language models. *arXiv preprint arXiv:2512.24601*, 2025a.

Xiang Zhang, Lei Li, Xiao Zhou, and Zheng Liu. R2MED: A benchmark for reasoning-driven medical retrieval. *arXiv preprint arXiv:2505.14558*, 2025b.

A Limitations

We acknowledge the limitation of our work. While RLM outperforms all controlled baselines under both backbones tested (Qwen3.6-27B and Gemma 4 A4B), generalization to a broader range of model families and scales remains unverified; the access-pattern advantages we observe may vary with backbone instruction-following ability or context capacity. Also, our failure categorization is grounded in 50 sampled traces, a scale chosen to permit careful human analysis; while this

produces a reliable codebook, the category frequencies may shift under automated analysis at full corpus scale. In addition, the trace-guided self-improvement recipe is validated on three R2MED subsets; whether the derived recipe transfers to held-out subsets or to other retrieval benchmarks requires further evaluation. Finally, consistent with standard practice in LLM-based retrieval evaluation, including SEMA-RAG (Huang et al., 2026) and the original RLM protocol (Zhang et al., 2025a), all results reflect single runs; the inference cost of recursive delegation makes repeated-run averaging impractical at this scale, and variance estimates are therefore unavailable. Nonetheless, the convergent pattern of results across task families, backbones, and diagnostic modalities supports the robustness of our core findings.

B RLM Delegation Semantics

Generated code can invoke the language model on selected evidence: the controller constructs a prompt over a subset of $\mathcal{D}$ and receives the response as a value in its running program, to be stored, parsed, or combined with other intermediate results. A delegated call receives only the prompt supplied by its parent, not the parent's execution state or interaction history, so recursion depth (§3.1) reflects nested delegation rather than iteration count within a single execution environment. This loop follows the interleaving of reasoning and action used in ReAct (Yao et al., 2022) and the use of an interpreter for executable computation in program-aided language models (Gao et al., 2022); here, these computations operate over the evidence collection itself, rather than over general-purpose subroutines.

C Trace-Guided Self-Improvement: Full Protocol

We implement policy-level adaptation with Prime Agent, a self-improving RLM harness rather than a new model architecture (Karten et al., 2026). Its persistent IPython environment realizes the RLM abstraction (§3.1) by allowing the controller to treat the corpus and intermediate state as programmatically accessible context. Prime Agent's Continual Harness supplies the cross-trajectory layer used here: completed execution histories can be reviewed, distilled into supplemental memories, skills, and prompt instructions, and reloaded in later sessions.

This mechanism forms a two-timescale architecture. Within a query, the RLM iteratively searches, inspects, and reranks documents. Across development queries, the harness diagnoses recurring failures from completed trajectories and consolidates only general retrieval procedures into a subset-specific recipe. For each subset, the controller processed a deterministic development sample using the same full-corpus interface as the main R2MED experiment. After committing a ranking, it received retrieval feedback and compared successful and failed actions in its trajectory. A lesson was retained only when it could be expressed as a transferable operation rather than a query-specific answer; the resulting recipe was frozen for adapted evaluation.

The learned recipes target the failure mechanisms identified in §5.3. Query-interpretation rules preserve the requested clinical relation e.g., separating a diagnosis from its cause, confirmatory test, or treatment. Candidate-generation rules expand beyond surface overlap when relevant cases use different terminology. Ranking rules require inspection and comparison of promising documents before submission. The harness also retains negative procedural evidence: when reweighting an unchanged candidate pool did not improve retrieval, subsequent queries were directed toward broader candidate generation instead of repeating the same operation.

Each subset produced an independent, frozen recipe prior to evaluation on held-out test queries. Evaluation sessions received the query, corpus, standard retrieval tools, and the corresponding read-only recipe; relevance judgments were used only for scoring outside the inference environment and were not available during ranking. Persistent memory was restricted to transferable retrieval instructions; query identifiers, document identifiers, patient names, and answer mappings were excluded; so the intervention changes how the controller searches and ranks evidence while leaving the underlying language model and benchmark task unchanged.

Please note that development and test queries were kept disjoint: the development samples contained 17 (IIYi-Clinical), 12 (PMC-Clinical), and 12 (MedQA-Diag) queries. Only these queries received retrieval feedback, and they were excluded from all reported adapted and baseline scores (§5.3). In

addition, due to timeouts and compute errors (e.g., exhausted execution budgets or memory limits), adapted runs did not produce a valid ranking for 9 of the 320 held-out queries (4 in IIYi-Clinical and 5 in MedQA-Diag). To keep the comparison matched, baselines are recomputed on exactly the query identifiers completed by each adapted run.

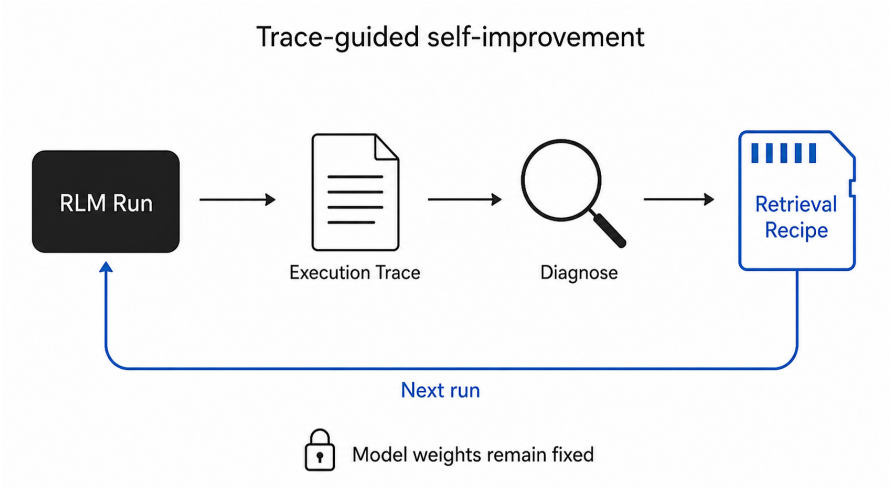


Figure 4: Trace-guided RLM self-improvement implemented through Prime Agent's Continual Harness (Karten et al., 2026). A completed execution is diagnosed and distilled into a persistent retrieval recipe for the next run, while model weights remain fixed.

D Implementation Details: Full Specification

D.1 Output Validation

The controller submits its output through an explicit operation checked against the task schema; if the iteration limit is reached without submission, a final model call attempts to extract an output from the recorded trajectory. For retrieval, we deduplicate repeated identifiers while preserving first-occurrence order, flag identifiers absent from the corpus, and accept a ranking only when the normalized output contains 100 identifiers with no unknown identifier proposed. These checks establish structural validity; relevance is scored separately by the evaluator.

D.2 Depth Enforcement

Depth caps are enforced via a shared counter over delegated invocations, with each prompt in a batch counted separately; a cap specifies an allowance, not a requirement to reach it. Independent delegated prompts can be dispatched concurrently as a batch, with results returned in submission order, preserving the association between each evidence subset and its response while leaving aggregation and final ordering to the parent. Controller and delegated invocations use the same model within each matched-backbone configuration.

D.3 Execution Bounds

Execution is bounded by limits on controller iterations, delegated calls, observation length, per-call generation length, and elapsed time. Each trajectory is capped at 30 controller iterations and 100 delegated calls per query; a call exceeding this budget is rejected by the environment, and the controller must continue with programmatic aggregation. In practice, R2MED trajectories of RLM (Qwen3.6-27B, depth 1) use a median of 10 controller steps (mean 11.8; 95th percentile: 24).

E RECURSIVE LANGUAGE MODEL PROMPTS

The following listings show representative retrieval and controller instructions used in our recursive language model runs. The query and document collection were supplied separately at runtime.

E.1 RETRIEVAL TASK INSTRUCTION

```
Rank the documents that best support the medical query.

Explore the complete corpus programmatically through the external
`corpus` variable. Use Python to inspect, filter, score, partition, and
validate documents. Use llm_query or llm_query_batched when semantic
medical analysis of selected subsets is useful. Do not answer the
medical question. Return exactly 100 unique document identifiers from
the supplied corpus in descending relevance order. Never invent IDs.
Recheck the highest-ranked evidence before submission.
```

E.2 CONTROLLER SYSTEM MESSAGE

```
Your input fields are:
1. `variables_info` (str): Metadata about the variables available in the REPL
2. `repl_history` (REPLHistory): Previous REPL code executions and their outputs
3. `iteration` (str): Current iteration number (1-indexed) out of max_iters
Your output fields are:
1. `reasoning` (str): Think step-by-step: what do you know? What remains? Plan your
    next action.
2. `code` (str): Python code to execute. Use markdown code block format:
    ```python\n<code>\n```
All interactions will be structured in the following way, with the appropriate values
    filled in.

[[ ## variables_info ## ]]
{variables_info}

[[ ## repl_history ## ]]
{repl_history}

[[ ## iteration ## ]]
{iteration}

[[ ## reasoning ## ]]
{reasoning}

[[ ## code ## ]]
{code}

[[ ## completed ## ]]
In adhering to this structure, your objective is:
        Rank the documents that best support the medical query.

        Explore the complete corpus programmatically through the external
        `corpus` variable. Use Python to inspect, filter, score, partition, and
        validate documents. Use llm_query or llm_query_batched when semantic
        medical analysis of selected subsets is useful. Do not answer the
        medical question. Return exactly 100 unique document identifiers from
        the supplied corpus in descending relevance order. Never invent IDs.
        Recheck the highest-ranked evidence before submission.

        You are tasked with producing the following outputs given the inputs `corpus`,
    `query`:
        - {ranked_doc_ids}        # note: the value you produce must adhere to the JSON
    schema: {"type": "array", "items": {"type": "string"}}

        You have access to a Python REPL environment. Write Python code and it will be
    executed. You will see the output, then write more code based on what you learned.
    This is an iterative process.
```
```

```
    Available:
    - Variables: `corpus`, `query` (your input data)
    - `llm_query(prompt)` - query a sub-LLM (~500K char capacity) for semantic
analysis
    - `llm_query_batched(prompts)` - query multiple prompts concurrently (much
faster for multiple queries)
    - `print()` - ALWAYS print to see results
    - `SUBMIT(ranked_doc_ids)` - submit final output when done
    - Standard libraries: re, json, collections, math, etc.

    IMPORTANT: This is ITERATIVE. Each code block you write will execute, you'll
see the output, then you decide what to do next. Do NOT try to solve everything in
one step.

    1. EXPLORE FIRST - Look at your data before processing it. Print samples, check
types/lengths, understand the structure.
    2. ITERATE - Write small code snippets, observe outputs, then decide next
steps. State persists between iterations.
    3. VERIFY BEFORE SUBMITTING - If results seem wrong (zeros, empty, unexpected),
reconsider your approach.
    4. USE llm_query FOR SEMANTICS - String matching finds WHERE things are;
llm_query understands WHAT things mean.
    5. MINIMIZE RETYPING (INPUTS & OUTPUTS) - When values are long, precise, or
error-prone (IDs, numbers, code, quotes), re-access them via variables and
parse/compute in code instead of retyping. Use small, targeted prints to
sanity-check, but avoid manual copying when variables can carry the exact value.
    6. SUBMIT ONLY AFTER SEEING OUTPUTS - SUBMIT ends the current run immediately.
If you need to inspect printed output, run it in one step, review the result, then
call SUBMIT in a later step.
```

F Trace Failure Categorization: Full Protocol

F.1 Sampling and Codebook Development

The sampling frame comprised 87 locally available Qwen3.6-27B depth-one runs with both trace and result files. We ordered their identifiers by a seeded SHA-256 key and selected the first 50, drawn without conditioning on outcome; the sample includes 49 completed runs and one invalid run. The first 15 selected traces were read for open coding; the resulting codebook was frozen before application to all 50 traces, including those first 15. The unit of counting is a run, and a run may receive multiple codes. Table 7 reports the codebook and run-level frequencies for the 50-trace sample.

F.2 Application and Verification

Each annotation names a root step or delegate call, quotes a log field, separates observation from interpretation, and records confidence. Model-based coders applied identical instructions to all 50 traces under the frozen rubric. All 670 quoted spans were exact matches to their referenced fields; human analyst adjudicated the resulting labels, rejecting one annotation and downgrading three.

F.3 Codebook

The codebook also distinguishes query paraphrasing, metadata-based candidate selection, lossy delegate-output contracts, score-scale mixing, batch-local ranking merges, and execution errors. A code was assigned only where the corresponding event was visible in the log; uncertain cases were marked insufficient evidence.

F.4 Internal Consistency Check

As a secondary check, we compared the submitted order with parsed delegate judgments rather than qrels. Forty-six runs had ID-mappable delegate outputs. Among the 38 runs with comparable judged pairs, 14 had a rank-discordance rate above 0.10. This measure can understate a departure when most delegate scores are tied.

Table 7: Selected trace codes and run-level frequencies in the 50-trace sample. Codes are non-exclusive.

Observed mechanism	Runs
Root hypothesis added to delegate prompt (Q1)	24
Lexical over-matching with visible off-target hit (C1)	30
Unexecuted REPL observation in reasoning (E1)	15
Delegate judgment lost or altered in later state (R2)	19
Final order ignores available delegate judgments (K1)	13
Delegate-consistent finalization (counterexample, P4)	13

F.5 Illustrative Traces

Run 03 illustrates a simulated observation: a variable appears in a purported output embedded in the root's reasoning, but never in an executed output; the subsequent code raises a `NameError` when it refers to that variable. Run 10 illustrates simulated delegation: 50 assessments appear before any logged sub-call, and the root uses them to form its ranking. Run 07 illustrates retention loss: 34 logged delegate scores of 1/10 are replaced by an imputed score of 3 during manual reconstruction of the ranking state. These examples identify process events; they do not assign benchmark relevance to the affected documents.

Table 8: Full-corpus RLM results by R2MED subset, including candidate coverage (Recall@100) and ordering headroom. Oracle re-sorts the same submitted top-100 candidates using the relevance judgments, without adding any new documents; Headroom is Oracle − nDCG@10. The oracle is a diagnostic upper bound conditional on the submitted candidate set, not a deployable reranker.

Subset	**n**	**nDCG@10**	**R@100**	**Oracle**	**Headroom**
Biology	103	0.5000	0.8330	0.8564	0.3564
Bioinformatics	77	0.4763	0.8544	0.8728	0.3965
Medical Sciences	88	0.5207	0.8360	0.8591	0.3384
MedXpertQA-Exam	97	0.2747	0.5668	0.6092	0.3345
MedQA-Diag	118	0.4738	0.7871	0.8166	0.3428
PMC-Treatment	150	0.6019	0.8012	0.8191	0.2172
PMC-Clinical	114	0.4259	0.7485	0.7744	0.3485
IIYi-Clinical	129	0.1727	0.5583	0.6045	0.4318
Equal-subset macro	**876**	**0.4308**	**0.7482**	**0.7765**	**0.3457**

Table 9: Trace-observable evidence state by final outcome.

Final outcome	n	**Mean nDCG**	**Exact ID exposed**	**Semantic judgment**
Candidate miss	123	0.0000	29.3%	8.9%
Ordering miss	154	0.0000	72.7%	29.9%
Partial top-ten retrieval	302	0.4443	91.4%	48.7%
Full recall, imperfect order	170	0.6854	98.8%	44.7%
Perfect nDCG@10	127	1.0000	97.6%	54.3%

Table 10: Trace-guided self-improvement on held-out R2MED test queries. Development queries used to derive each recipe are excluded, and baselines are recomputed on the same held-out query identifiers completed by each adapted run. n gives completed/held-out queries; the remainder failed due to timeouts or compute errors. Δ is adapted minus baseline.

	nDCG@10			**Recall@100**			
Subset	**Base**	**Adapted**	**Δ**	**Base**	**Adapted**	**Δ**	n
IIYi-Clinical	0.1626	**0.2883**	+0.1257	0.5415	**0.7435**	+0.2020	108/112
PMC-Clinical	0.4401	**0.4875**	+0.0474	0.7516	**0.7802**	+0.0286	102/102
MedQA-Diag	0.4855	**0.5684**	+0.0829	**0.8113**	0.7983	−0.0130	101/106
Unweighted mean	0.3627	**0.4480**	+0.0853	0.7015	**0.7740**	+0.0725	–