INFORMED RECURSION: STRUCTURAL AWARENESS FOR RECURSIVE LANGUAGE AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recursive Language Models allow language models to undertake context lengths beyond their designed context windows by treating long prompts as external environments and recursively querying themselves. However, a closer look at this performance reveals a large drop in accuracy on tasks requiring exact numerical counts relative to comparison and existential tasks.

We trace this gap to a failure mode in which ordinary classification errors near ambiguous category boundaries are amplified by exact-count and combinatorial aggregation. We introduce Informed Recursion, a single-trajectory approach with two lightweight mechanisms: Query-Structure Routing, which selects recursion depth based on the structure of the query, and Targeted Verification, which rechecks intermediate decisions that can affect the final aggregation.

Across our evaluations, Informed Recursion improves RLM performance by adapting recursion depth to query structure and selectively verifying intermediate decisions. Doing so avoids the cost of applying deeper recursion or multiple trajectories to every query. Our analysis indicates that many RLM failures arise from local classification errors that become amplified when their outputs are aggregated, rather than from incomplete access to the context.

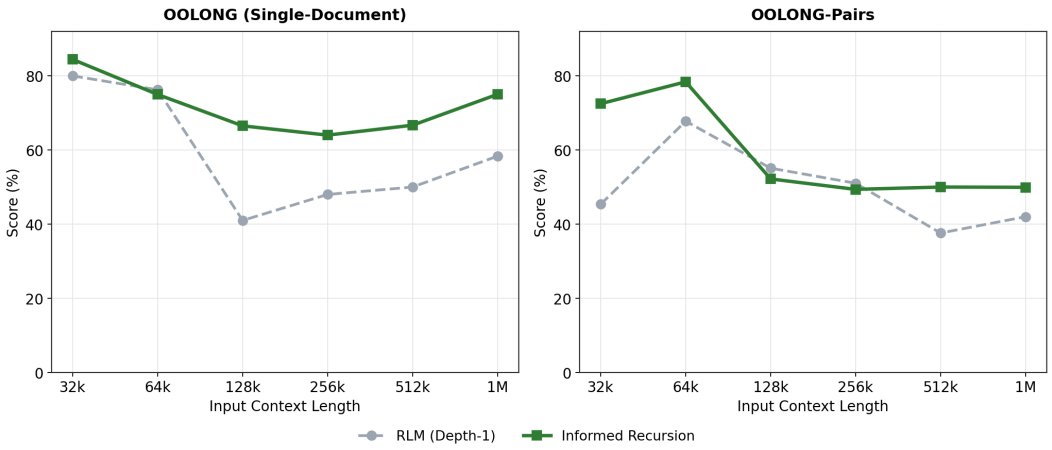

Figure 1: Accuracy as a function of input context length (32k–1M tokens) on OOLONG single-document (left) and OOLONG-Pairs (right). Informed Recursion outperforms RLM (Depth-1) at nearly every context length on both benchmarks.

1 INTRODUCTION

Large language models (LLMs) augmented with sandboxed execution environments and recursive sub-querying interfaces for long-context tasks, such as Recursive Language Models (RLMs) (Zhang et al., 2025), allow models to work with contexts beyond their designed context windows. Rather

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.

than placing millions of tokens directly in the context, an RLM treats the input as an external environment that can be inspected, filtered, sliced, and queried recursively.

While early reports demonstrated impressive aggregate improvements over long documents, a closer look reveals a large accuracy gap between task types. RLMs perform well on tasks characterized by existential or comparative properties (e.g., "Find any user who mentioned a location" or "Identify whether entity A appears more frequently than entity B"), but perform far worse on queries that demand an exact numerical count or combinatorial aggregation over exact constraints (e.g., "List all user pairs where User A has exactly one abbreviation"), consistent with the gap reported above.

We first rule out incomplete context coverage and redundant computation: audited trajectories process 100% of input lines, while duplicate sub-calls account for less than 1.4% of token usage. Instead, we find that small classification errors near semantic boundaries can accumulate during aggregation, making exact counts considerably more fragile than existential decisions. This motivates an approach that adapts recursion to query structure and selectively verifies intermediate decisions.

To address this fragility without re-processing an entire trajectory, we introduce Informed Recursion, which combines two lightweight single-trajectory mechanisms: Query-Structure Routing, which uses the query to choose an appropriate recursion strategy, and Targeted Verification, which rechecks only the intermediate classifications that can affect the final aggregation.

2 Recursive Language Models and the Failure Setting

2.1 Recursive Language Models

Recursive Language Models (RLMs). RLMs handle long contexts by offloading their context into an external environment that can be inspected and recursively queried. A root model can delegate subproblems to recursive calls and combine their outputs, allowing RLMs to operate beyond the base LLM's context window.

3 Understanding the Failure

3.1 Ruling Out Incomplete Coverage and Redundant Computation

We observed complete trajectory execution logs across failing tasks in OOLONG-Pairs and CodeQA. Table 1 summarizes the two most direct candidate explanations for the observed failures.

Table 1: Trajectory audit of candidate explanations for the observed failures.

Candidate Hypothesis	**Instrumented Trajectory Evidence**	**Verdict**
Incomplete Coverage	Model systematically reads 100% of input lines (1,585 records) in REPL variables	Not supported by the audit
Redundancy & Looping	Hash deduplication shows duplicate sub-calls represent $< 1.4\%$ of token spend	Not supported by the audit

3.2 Why Exact Counts Are Fragile

A simple way to see the difference is to consider how classification errors affect the final predicate. Let S be the set of items being classified, and let c(x) denote the predicted category for item x. We compare two common types of queries:

1. Existential queries ask whether at least one item satisfies a condition. In this case, an isolated classification error often does not change the final answer: if several items satisfy the condition, one

mistaken item can be absorbed by the remaining correct items. In notation, $P_{\text{exist}}(S)$ = 1 if there exists an x in S such that c(x) = y.

2. Exact-count queries ask whether exactly k items satisfy a condition. Here, a single false positive or false negative near the threshold can change the answer. In notation, $P_{\text{count}}(S)$ = 1 if exactly k items in S satisfy c(x) = y. Thus, exact-count queries are especially sensitive to classification errors near the decision boundary.

3.3 Error Amplification in Pairwise Aggregation

In OOLONG-Pairs, membership decisions are combined to construct pairs, so an error doesn't just mislabel one user, it propagates into every pair that user participates in. Let U_E denote users satisfying the existential condition and U_C those satisfying the exact-count condition. An error in U_C can affect on the order of $|U_E|$ pairs, and an error in U_E can affect on the order of $|U_C|$ pairs. The resulting pair-level error is approximately

$$\Delta_{\text{pairs}} = |U_E|\,\text{Err}_C + |U_C|\,\text{Err}_E - \delta_{\text{overlap}},$$

where δ_{overlap} corrects for excluded or double-counted pairs. Because $|U_E|$ and $|U_C|$ are often tens to hundreds of users in our tasks, even one or two misclassified users near the count threshold can shift the predicted pair count by hundreds of pairs relative to gold — which is why exact-count and pairwise aggregation are far more sensitive to boundary errors than existential queries.

3.4 When Deeper Recursion Hurts

Conversely, on CodeQA (repository comprehension across millions of characters), forcing multi-tier recursion (Depth-2) reduces accuracy: Vanilla Depth-1 achieves 64.00%, whereas Vanilla Depth-2 achieves 62.00%. In the audited trajectories, deeper recursion can add delegation planning even when local grep/cat searches have already provided grounded evidence. Choosing recursion depth from query structure, rather than fixing it globally, follows directly from this pattern.

4 Informed Recursion

Informed Recursion combines Query-Structure Routing and Targeted Verification in a single RLM trajectory. Query-Structure Routing adapts recursion strategy to the structure of the query, while Targeted Verification selectively re-checks intermediate decisions that can affect the final aggregation. Figure 2 summarizes the resulting workflow.

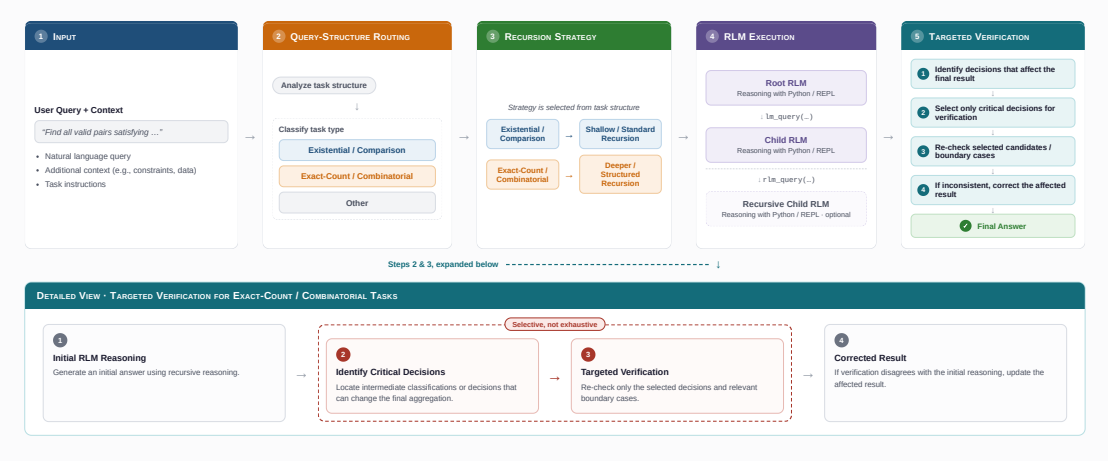

Figure 2: Overview of Informed Recursion. Query-Structure Routing selects a recursion strategy from the query structure before execution. Targeted Verification selectively re-checks intermediate decisions that can materially affect the final result.

4.1 Query-Structure Routing

Query-Structure Routing inspects the query before context processing and selects a recursion strategy based on its reasoning structure. Aggregation-heavy queries, such as exact-count or pairwise tasks, are routed toward deeper recursive decomposition, while queries that don't demand excessive recursion depths remain shallow. The router uses lightweight structural cues from the query rather than analyzing the full context. If the query does not provide a clear routing signal, the system falls back to the standard execution strategy.

4.2 Targeted Verification

Targeted Verification is applied after the initial reasoning trajectory and focuses on intermediate decisions that can affect the final answer. For exact-count tasks, these include candidates near the decision boundary or cases with conflicting evidence. Instead of re-processing the entire context, the verifier re-checks only the selected decisions and updates the result when it disagrees with the initial reasoning. Verification is limited to a single pass, keeping the additional inference bounded while targeting errors that can propagate into the final aggregation.

5 Experimental Setup

5.1 Evaluation Benchmarks

We evaluate Informed Recursion on four primary benchmarks and one held-out generalization benchmark. The primary benchmarks cover long-context aggregation, pairwise reasoning, repository understanding, and multi-domain reasoning. The held-out benchmark tests whether the approach generalizes beyond the task families used to develop the method.

OOLONG (Bertsch et al., 2025). OOLONG is a long-context reasoning benchmark that requires semantically labeling records and aggregating those labels to produce a final answer. We use the 131K single-document setting with 50 tasks. Each task requires reasoning over a large fraction of the input, making it a useful setting for studying how classification errors propagate through aggregation. We report answer accuracy.

OOLONG-Pairs (Zhang et al., 2025). OOLONG-Pairs is a modified variant of OOLONG in which the final answer requires aggregating pairs of entries. We use the 64K setting with 20 queries, consisting of 10 symmetric existential tasks and 10 asymmetric hybrid tasks. Because the answer

depends on pairwise combinations of classified entries, the number of pairwise decisions grows quadratically with the number of entries. We report F1 over the predicted answer pairs.

LongBench-v2 CodeQA (Bai et al., 2025). CodeQA is a multi-choice code repository understanding benchmark in which each instance requires reasoning over a fixed number of files in a codebase to determine the correct answer. We evaluate 50 instances and report answer accuracy. CodeQA therefore contrasts with OOLONG-style aggregation: local evidence from a small number of relevant files is often sufficient to answer the query.

LongCoT-mini (Motwani et al., 2026). LongCoT-mini evaluates multi-domain reasoning across Logic, Mathematics, Computer Science, Chemistry, and Chess. We use 50 tasks, with 10 tasks from each domain, and report answer accuracy. We present the results separately by reasoning domain in §6.4 rather than combining them with the main ablation table.

BrowseComp-Plus (1K documents) (Chen et al., 2025). We use BrowseComp-Plus as a held-out generalization benchmark because its multi-hop questions require combining evidence across an offline corpus rather than performing OOLONG-style aggregation.We use 150 randomly sampled instances and provide 1,000 randomly selected documents as input, with the gold and evidence documents guaranteed to be present. We report the percentage of correct answers. Importantly, BrowseComp-Plus is evaluated only after the method and routing rules have been fixed, it is not used to design or tune Informed Recursion.

5.2 Methods and Baselines

We compare three approaches: the original Recursive Language Model (RLM), Informed Recursion, and Self-Reflective Recursive Language Models (SRLM) (Alizadeh et al., 2026). For Informed Recursion, we separately evaluate the routing-only and verification-only variants to isolate the contribution of each component. Finally, we compare against SRLM to test whether the gains of Informed Recursion can instead be explained by multi-trajectory program search.

RLM. We implement RLM with a Python REPL environment that stores the long context and provides the root model with programmatic access to it. The root model can delegate portions of the context to language-model sub-calls. We evaluate three recursion settings: **Depth-0**, where sub-calls are disabled. **Depth-1**, where the root model can make flat sub-calls; and **Depth-2**, where sub-calls can themselves invoke recursive RLM instances. Unless otherwise specified, these settings use the same model, context, and execution protocol as Informed Recursion.

Informed Recursion. Informed Recursion uses the same underlying RLM execution framework but adds two task-conditional mechanisms. **Query-Structure Routing** selects the recursion strategy from the structure of the query, while **Targeted Verification** re-checks intermediate decisions that can materially affect the final answer. We evaluate three variants: **Routing Only**, **Verification Only**, and the **Full** system containing both components. These ablations isolate whether the gains come from adaptive recursion, targeted verification, or their combination.

Self-Reflective Recursive Language Models. We compare against SRLM (Alizadeh et al., 2026), which generate multiple independent trajectories and select among them using self-consistency, verbalized confidence, and reasoning-trace length. We evaluate with both, recursive sub-calls enabled (Depth-1) and sub-calls disabled (Depth-0). The original SRLM experiments use K=8; our K=5 setting is therefore a cost-controlled adaptation rather than an exact reproduction. We evaluate SRLM on OOLONG-Pairs and CodeQA, where the comparison is most directly relevant to our targeted single-trajectory intervention.

5.3 Models and Evaluation Protocol

Models. All evaluations are conducted using **GPT-5.6-Luna**, configured with Medium reasoning effort for the root agent and Low reasoning effort for recursive sub-agents.

Evaluation Protocol. We report task accuracy and F1 separately from measured trajectory costs. All agentic ablations use an identical execution environment and sandboxed Python REPL. We compute costs based on the compute provider Azure OpenAI.

6 Results and Analysis

6.1 Main Results

Table 2 shows that recursion depth is task-dependent: deeper recursion helps OOLONG-Pairs but slightly hurts CodeQA. Informed Recursion avoids this fixed-depth trade-off by adapting computation to the query structure. The gain is largest on OOLONG-Pairs, where errors in intermediate membership decisions can propagate through pair construction. Figure 3 visualizes these results across all evaluated methods.

Table 2: Performance comparison of methods across long-context evaluation benchmarks. Scores are F1 for OOLONG-Pairs and accuracy for CodeQA and OOLONG. Average API cost is shown in gray as mean ± std.

Method	OOLONG-Pairs 64K Overall F1	CodeQA Accuracy	OOLONG 131K Accuracy
Base LLM	0.01%† $0.0129 ± 0.0017	34.00%† $0.0167 ± 0.0254	50.00% $0.0199 ± 0.0009
RLM Depth-0	52.55% $0.1093 ± 0.0622	56.00% $0.0181 ± 0.0376	56.00% $0.0362 ± 0.0329
RLM Depth-1	67.76% $0.0854 ± 0.0759	64.00% $0.0344 ± 0.0523	42.00% $0.0418 ± 0.0312
RLM Depth-2	70.53% $0.0724 ± 0.0530	62.00% $0.0237 ± 0.0426	65.97% $0.0750 ± 0.0752
Query-Structure Routing	72.32% $0.1038 ± 0.1266	58.00% $0.0241 ± 0.0319	57.00% $0.0415 ± 0.0380
Targeted Verification	67.40% $0.0702 ± 0.0463	64.00% $0.0401 ± 0.0660	56.33% $0.0796 ± 0.0829
Informed Recursion	**78.31%** $0.0869 ± 0.0723	**66.00%** $0.0220 ± 0.0254	**66.50%** $0.0604 ± 0.0724

†Base LLM truncates inputs exceeding its context window. Costs reflect a single truncated call, not a per-task average.

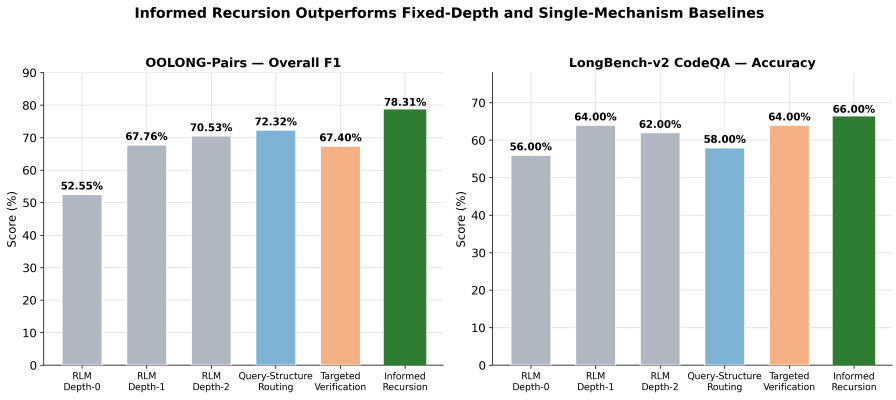

Figure 3: Visual comparison of the main results reported in Table 2. Informed Recursion improves over the fixed-depth RLM baselines and over the single-mechanism ablations (Query-Structure Routing only, Targeted Verification only) on both OOLONG-Pairs F1 and CodeQA accuracy.

6.2 Task-Dependent Recursion

The depth ablation suggests that recursion is useful when the query demands hierarchical aggregation, while unnecessary recursion can hurt more localized reasoning tasks. Query-Structure Routing

captures this distinction by assigning depth per query rather than globally. Figure 4 makes this divergence explicit: OOLONG-Pairs F1 rises monotonically with recursion depth, while CodeQA accuracy peaks at Depth-1 and declines at Depth-2.

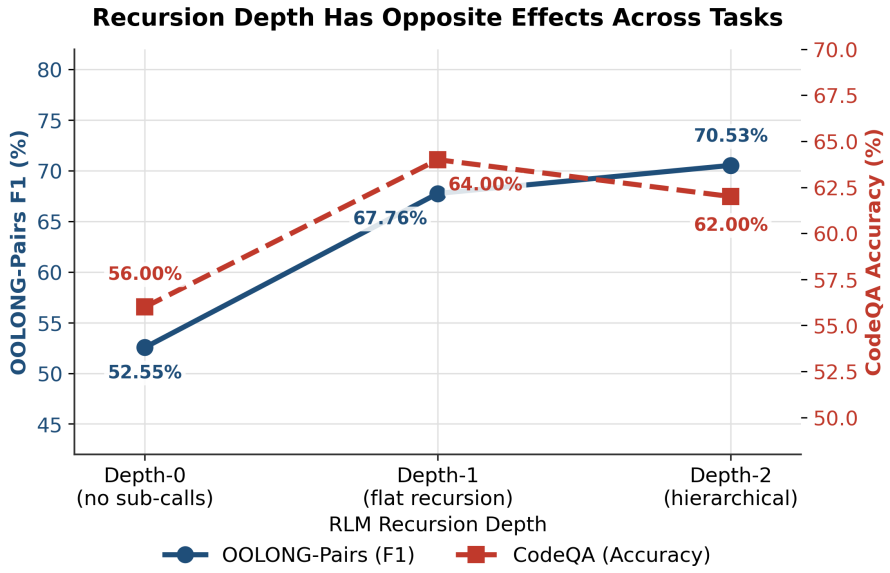

Figure 4: Effect of recursion depth on OOLONG-Pairs F1 and CodeQA accuracy. Deeper recursion monotonically improves pairwise aggregation but hurts localized code retrieval, illustrating why a single global recursion depth cannot be optimal for both task types.

6.3 Targeted Verification

Verification is most useful when an intermediate decision has many downstream consequences. In Task 15, correcting only two borderline membership decisions removes 1,558 spurious pairs, illustrating why selective verification can have a much larger effect than the number of corrected decisions suggests.

Table 3: OOLONG-Pairs predicate topology ablation across existential (monotonic) and hybrid (non-monotonic) task cohorts. Net Gain on Hybrid is each method's Hybrid F1 minus the RLM Depth-0 baseline (40.50%), applied consistently across all rows.

Method	Overall F1	Existential F1	Hybrid F1	Net Gain on Hybrid
RLM Depth-0	52.55%	64.60%	40.50%	Baseline
RLM Depth-1	67.76%	70.94%	64.57%	+24.07%
RLM Depth-2	70.53%	77.68%	63.38%	+22.88%
Query-Structure Routing	72.32%	75.90%	68.73%	+28.23%
Targeted Verification	67.40%	79.05%	55.76%	+15.26%
Informed Recursion	**78.31%**	**85.46%**	**71.15%**	**+30.65%**

The full system reaches 71.15% F1 on the hybrid cohort, compared with 64.57% for Depth-1 RLM. Query-Structure Routing alone reaches 68.73%, while the verification-only variant is less effective on this cohort, indicating that the two mechanisms address different parts of the failure mode.

6.4 GENERALIZATION

We evaluate two settings beyond the development tasks. LongCoT-mini tests whether the observed failure pattern extends beyond OOLONG-style aggregation, while BrowseComp-Plus tests generalization to open-ended multi-document search.

On LongCoT-mini, results vary widely by domain rather than showing a uniform improvement. Informed Recursion improves chemistry and chess, but underperforms the fixed-depth baseline on logic, and computer science. The routing and verification strategy is therefore not uniformly beneficial outside the aggregation setting that motivated its design. Table 4 breaks down this pattern by domain.

Table 4: LongCoT-mini results by reasoning domain. Values are answer accuracy.

Method	**Overall**	**Logic & Planning**	**Mathematics**	**Computer Science**	**Chemistry & Molecular**	**Chess & Strategic Search**
Vanilla Depth-1	**48.00%**	**60.0%**	**20.0%**	**30.0%**	40.0%	90.0%
Informed Recursion	44.90%	40.0%	**20.0%**	20.0%	**50.0%**	**100.0%**

For BrowseComp-Plus, our evaluation shows that Informed Recursion achieves 89.33% accuracy (134/150 correct), compared with 82.00% for RLM (123/150 correct), as reported in Table 5. The routing policy is frozen before evaluation: evidence-seeking and comparison queries use Depth-1, while open-ended or out-of-distribution queries use Depth-2. The held-out test therefore checks whether query-dependent recursion remains useful when relevant evidence must first be located across a large document set.

Table 5: Held-out generalization results on BrowseComp-Plus (150 instances, 1K documents each).

Method	**Accuracy**	**Correct / Total**
RLM	82.00%	123 / 150
Informed Recursion	**89.33%**	**134 / 150**

6.5 COMPARISON WITH MULTI-TRAJECTORY SEARCH

SRLM provides a direct multi-trajectory alternative to our single-trajectory design.

For each task, the five candidate trajectories receive the same query, context, model, and sampling configuration as the corresponding RLM experiments.

Table 6: Comparison with SRLM using a cost-controlled $K = 5$ multi-trajectory budget.

Method	Recursion	K	OOLONG-Pairs Overall F1	CodeQA Accuracy	Mean Cost per Task
SRLM	Enabled (Depth-1)	5	63.51%	60.00%	\$0.3201 ± 0.2202
SRLM (no sub-call)	Disabled (Depth-0)	5	58.02%	58.00%	\$0.2676 ± 0.2388
Informed Recursion	Selective	1	**78.31%**	**66.00%**	**\$0.0405 ± 0.0530**

Informed Recursion's Mean Cost per Task is the pooled micro-average (mean and standard deviation) of the audited per-task costs across all 70 tasks – 20 OOLONG-Pairs (\$0.0869 ± 0.0723) and 50 CodeQA (\$0.0220 ± 0.0254; see Table 2) – rather than a single re-used figure.

7 Related Work

Long-Context and Recursive Agents. Long-context systems have explored retrieval, programmatic context interaction, and recursive reasoning. RLM treats long prompts as external state that can be queried recursively (Zhang et al., 2025), while benchmarks such as LongBench-v2 and OOLONG evaluate reasoning over long contexts (Bai et al., 2025; Bertsch et al., 2025). We focus on how query structure should determine recursive computation.

Multi-Trajectory Sampling and Verification. Self-Consistency samples multiple reasoning paths and selects the answer based on their agreement (Wang et al., 2023). Similarly, SRLM (Alizadeh et al., 2026) uses multiple recursive trajectories with self-consistency, confidence, and trace length. Informed Recursion instead uses a single trajectory, adapting recursion depth and selectively checking intermediate decisions.

8 Limitations and Future Work

Query-Structure Routing currently relies on lightweight query cues, so unusual or ambiguous queries may be routed incorrectly. A learned router could improve this. Verification can add to the costs when many decisions require checking. Our evaluation covers a limited set of tasks and models, so the observed failure pattern should be tested more broadly.

9 Conclusion

Our audit points to a specific mechanism behind RLM failures on exact-count and combinatorial queries: ordinary classification noise near category boundaries, which existential queries can absorb, gets multiplied once those decisions feed into pairwise or exact aggregation. Query-Structure Routing and Targeted Verification address this at the two points where it matters – how much recursion a query gets, and which intermediate decisions get a second look – without re-running an entire trajectory. The uneven results on LongCoT-mini indicate that this is a fix for aggregation-sensitive failures specifically, not a general-purpose improvement to recursive reasoning; a natural next step is a learned router that can recognize when a query falls outside the aggregation regime the method was built for.

AI Usage Disclosure

Large language models were used as writing and coding aids during the preparation of this work. An LLM (Gemini 3.8 Flash) assisted with implementation and coding of the experimental pipeline, and an LLM (Claude Sonnet 5) assisted with polishing and writing portions of the manuscript text. The authors take full responsibility for the content, experimental design, and conclusions of this paper.

REFERENCES

Keivan Alizadeh, Parshin Shojaee, Minsik Cho, and Mehrdad Farajtabar. Recursive Language Models Meet Uncertainty: The Surprising Effectiveness of Self-Reflective Program Search for Long Context. arXiv preprint arXiv:2603.15653, 2026. https://arxiv.org/abs/2603.15653.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench v2: Towards Deeper Understanding and Reasoning on Realistic Long-context Multitasks. arXiv preprint arXiv:2412.15204, 2025. https://arxiv.org/abs/2412.15204.

Amanda Bertsch, Adithya Pratapa, Teruko Mitamura, Graham Neubig, and Matthew R. Gormley. Oolong: Evaluating Long Context Reasoning and Aggregation Capabilities. arXiv preprint arXiv:2511.02817, 2025. https://arxiv.org/abs/2511.02817.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Sahel Sharifymoghaddam, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and Jimmy Lin. BrowseComp-Plus: A More Fair and Transparent Evaluation Benchmark of Deep-Research Agent. arXiv preprint arXiv:2508.06600, 2025. https://arxiv.org/abs/2508.06600.

Sumeet Ramesh Motwani, Daniel Nichols, Charles London, Peggy Li, Fabio Pizzati, Acer Blake, Hasan Hammoud, Tavish McDonald, Akshat Naik, Alesia Ivanova, Vignesh Baskaran, Ivan Laptev, Ruben Glatt, Tal Ben-Nun, Philip Torr, Natasha Jaques, Ameya Prabhu, Brian Bartoldson, Bhavya Kailkhura, and Christian Schroeder de Witt. LongCoT: Benchmarking Long-Horizon Chain-of-Thought Reasoning. arXiv preprint arXiv:2604.14140, 2026. https://arxiv.org/abs/2604.14140.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *International Conference on Learning Representations (ICLR)*, 2023. https://openreview.net/forum?id=1PL1NIMMrw.

Alex L. Zhang, Tim Kraska, and Omar Khattab. Recursive Language Models. arXiv preprint arXiv:2512.24601, 2025. https://arxiv.org/abs/2512.24601.

A PROMPTS USED FOR INFORMED RECURSION

The following are the exact prompts used for the two proposed mechanisms. Query-Structure Routing is invoked before the main recursive trajectory, while Targeted Verification is invoked selectively when an intermediate decision or final answer warrants additional checking.

A.1 QUERY-STRUCTURE ROUTING PROMPT

```
You are an expert query complexity and structure analyzer for recursive reasoning
     architectures.
Analyze the user query and determine the minimal recursive execution depth required to answer
     it reliably.

Routing Rules:
- depth_0 (Flat Execution / No Sub-calls):
  * Simple, direct factual lookup or short-context single-step retrieval.
  * Queries that can be resolved in a single LLM generation without iterative inspection.

- depth_1 (Single-Level Recursion):
  * Multi-hop lookups over large contexts (e.g., standard document QA, localized code search).
  * Direct constraint matching or filtering where sub-problems are independent and do not
     require iterative sub-querying.

- depth_2 (Deep / Hierarchical Recursion):
  * Complex multi-predicate queries.
  * Open-ended, multi-document exploratory search .
  * Cross-file / multi-module dependency tracing or structural reasoning where each sub-call
     itself spawns further sub-examinations.

Query to Analyze:
{query}

Respond in JSON:
{
  "reasoning": "Brief explanation of query predicate topology and branching requirements",
  "assigned_depth": 0 | 1 | 2
}
```

A.2 TARGETED VERIFICATION PROMPT

```
Your task is to inspect a candidate decision and determine whether it satisfies the required
     predicate or if it is a false positive / false negative.

Task Context & Query:
{task_query}

Extracted Candidate Answer / Sub-Decision:
{candidate_decision}

Supporting Context / Evidence Snippet:
{evidence_snippet}

Verification Instructions:
1. Identify the exact logical constraints demanded by the query.
2. Verify whether the candidate decision is strictly supported by the evidence or if critical
     constraints were overlooked, hallucinated, or partially met.
3. Check for borderline edge cases:
   - For pair/relation tasks: Did both entities independently satisfy their respective
     predicates?
   - For code/retrieval tasks: Does the selected code path or passage actually implement the
     requested behavior without conflicting side effects?

Respond in JSON:
{
  "is_verified": true | false,
  "confidence": 0.0 to 1.0,
  "criticism": "Precise explanation if flawed, ambiguous, or ungrounded; otherwise rationale
     for confirmation",
  "corrected_decision": "Corrected output if is_verified is false, or null if is_verified is
     true"
}
```