PARSER: READ IN PARALLEL, REASON IN DEPTH FOR LONG-CONTEXT LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reasoning over documents far beyond an LLM's context window remains challenging, as evidence may be sparsely distributed across hundreds of thousands of tokens. Sequential memory agents stream documents chunk by chunk into a compact recurrent memory, enabling bounded-context processing of arbitrarily long inputs. However, because every reading step immediately updates the memory on which subsequent processing depends, these agents couple document traversal with sequential reasoning. This coupling makes the reasoning sensitive to evidence position and forces the sequential inference path to grow with document length. We introduce PARSER (**Pa**rallel **R**eading, **Se**quential **R**easoning), a framework that separates document reading from question reasoning. To realize this separation, PARSER assigns local reading to chunk-bound subagents and global reasoning to a lead agent. The subagents read in parallel under the lead agent's queries; the lead agent integrates their findings and iteratively refines its queries as evidence accumulates. This decoupled design concentrates all learnable behavior in the lead agent and optimizes it with reinforcement learning, while the lightweight chunk readers remain frozen off-the-shelf models. On multi-hop QA with contexts ranging from 7K to 896K tokens, PARSER with a 4B backbone beats the strongest sequential memory baseline by 5.7 points on average and by 12.0 points at 896K tokens. With a 9B backbone, PARSER surpasses DeepSeek-V4-Pro by 6.3 points. Experiments show that PARSER remains robust to changes in evidence position, order, and distance that cause large accuracy swings in sequential methods. By parallelizing document coverage, PARSER reduces inference latency by up to $11\times$.

1 INTRODUCTION

Reasoning over long documents is a core capability for large language models, yet remains challenging: in tasks such as multi-document QA or legal analysis, the evidence for a single question can be scattered across hundreds of thousands of tokens. Despite context windows now reaching a million tokens or more (DeepSeek-AI, 2026), model accuracy degrades as context grows, a phenomenon known as *context rot* (Hong et al., 2025). One well-documented manifestation is positional bias: evidence placed away from the boundaries of the input is systematically ignored (Liu et al., 2024). We observe the same on multi-hop QA, where direct full-context answering drops by tens of points as documents lengthen from 7K to 896K tokens, even for million-token models. Since accuracy degrades even when the window is far from full, the bottleneck is no longer capacity; the open question is how to select what to read and in what order to reason about it.

One response to the selection problem is the sequential memory paradigm: the document is read chunk by chunk, and at each step the agent compresses the current chunk together with its previous memory into an updated memory. The final answer is generated from this memory alone. MemAgent (Yu et al., 2026) and its follow-ups (Shi et al., 2026; Sheng et al., 2026) train this recurrent workflow end-to-end with reinforcement learning, handling documents of millions of tokens within a small context window. These methods address a genuinely hard problem: bounded working memory, arbitrary input length, and end-to-end trainability.

However, these memory-based methods share a structural commitment: *the document is traversed exactly once, in order*. This single-pass recurrent structure imposes two constraints: evidence must be evaluated through a repeatedly compressed prefix state, and every chunk update depends on

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.

the preceding one. The first leads to a **sensitivity to evidence placement**: because each chunk is compressed before the rest of the document has been seen, the agent judges relevance under a strict information deficit. In particular, its accuracy could be affected by the absolute position of evidence, the logical order among evidence pieces, and the distance between them. This sensitivity is most damaging in multi-hop reasoning, where the answer depends on scattered evidence whose relevance emerges only incrementally. The second leads to **inference latency**: since each chunk update depends on the output of the previous one, the T steps form an irreducibly sequential chain whose wall-clock cost grows linearly with document length, regardless of available parallelism. Subsequent work within this paradigm has largely been a sequence of patches to these two symptoms, often trading one for the other, but leaves the sequential dependency chain intact (Shi et al., 2026; Sheng et al., 2026).

The order in which a long document is read is imposed by the document; the order in which a question is reasoned about is imposed by the question. Sequential memory agents let the first drive the second. We introduce PARSER (**Par**allel **R**eading, **Se**quential **R**easoning), which decouples the two orders. Rather than tying sequential depth to document length, PARSER reads all chunks in parallel and reserves sequential computation only for the reasoning the question demands. To achieve this decoupling, PARSER assigns reading and reasoning to two separate roles. A **lead agent**, blind to the document, *reasons* about the question in a ReAct-style (Yao et al., 2023) loop of interleaved thinking and action. A bank of **subagents** (one per chunk) *read* only their own chunk and extract evidence in parallel for a given query. At each round, the lead agent *scatters* queries to all subagents, then *gathers* their findings and decides what to ask next, iterating until committing to an answer.

This scatter–gather architecture changes the dependency structure of long-document processing. Sequential memory requires T dependent updates in sequence; PARSER replaces this with K scatter–gather rounds where all chunk-level calls execute in parallel, so latency scales with reasoning depth rather than document length. Furthermore, every chunk is re-read under a freshly formulated query at each round, eliminating the chunk-level position bias inherent in sequential traversal. Because no chunk is permanently discarded, the lead agent can condition each new query on previously discovered evidence, which is essential for multi-hop reasoning where a single static query cannot identify downstream hops (Zhou et al., 2024; Zhao et al., 2024; Xu et al., 2026). Although this requires reading every chunk over multiple rounds, cross-round KV-cache reuse avoids repeated prefill computation and sparse subagent outputs limit decoding computation. Finally, this decoupled design simplifies training: we train *only* the lead agent with Reinforcement Learning with Verifiable Reward (Shao et al., 2024; DeepSeek-AI, 2025a). Since each subagent's task is simple (locate evidence for a focused query in a short span), an off-the-shelf model suffices.

We evaluate PARSER on multi-hop long-context question answering, including the in-distribution HotpotQA (Yang et al., 2018) and the out-of-distribution 2WikiMultiHopQA (Ho et al., 2020), with context ranging from 7K to 896K tokens. On HotpotQA, PARSER with 4B and 9B backbones achieves average accuracies of 84.6% and 86.8% respectively, outperforming the strongest sequential memory baseline by 5.7 and 6.7 percentage points; at the longest setting (896K tokens) the gaps widen to 12.0 and 9.9 percentage points, as sequential methods degrade sharply with length while PARSER remains stable. Scaling to a 9B backbone, PARSER achieves an average of 86.8%, surpassing DeepSeek-V4-Pro (DeepSeek-AI, 2026), which natively supports a one-million-token context, by 6.3 percentage points. Controlled experiments that independently perturb the absolute position, the logical order, and the relative distance of evidence within context confirm the source of this stability: sequential memory agents exhibit large accuracy swings as any of these factors changes, whereas PARSER remains nearly flat across all three conditions. On inference latency, parallel reading yields an $11\times$ reduction at 896K tokens under single concurrency (876s vs. 78s per sample relative to MemAgent) and maintains a $1.7\times$ advantage under a concurrency of 16 (102s vs. 59s).

2 METHOD

2.1 PROBLEM FORMULATION

For the task of long-context question answering (QA), an agent is required to give an answer $\boldsymbol{a}$ to the question $\boldsymbol{q}$, conditioned on a corresponding long document $\boldsymbol{D}$. The document $\boldsymbol{D}$ can be extremely long, such as hundreds of thousands of tokens or even more. Typically, due to the limited LLM context window, $\boldsymbol{D}$ is split into a set of fixed-size chunks $\{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_T\}$ for processing. To answer

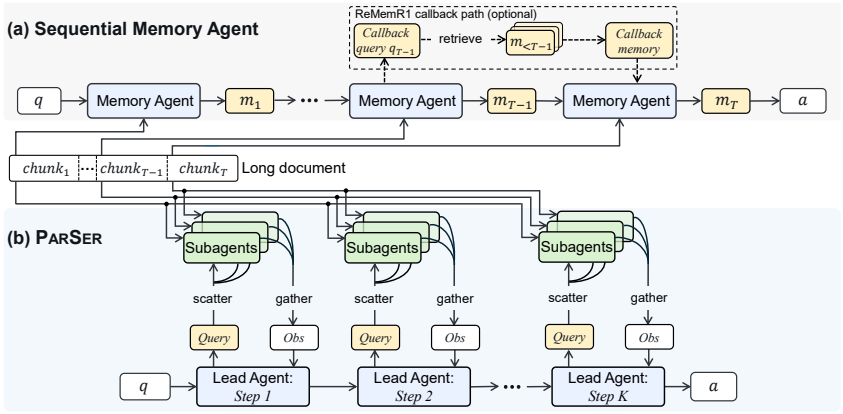

Figure 1: (a) Sequential memory traverses T chunks through a chain of dependent reading steps. (b) ParSer turns the chunks into parallel width: at each round, all chunks are read concurrently under the same query, while reasoning proceeds across rounds.

the question $\boldsymbol{q}$, the agent needs to accurately locate and then reason over a few pieces of key evidence, which are sparsely distributed within $\boldsymbol{D}$.

Sequential memory methods (the upper panel of Figure 1, e.g., MemAgent, ReMemR1) cast long-context reasoning as a sequential, recurrent, and chunk-by-chunk process: throughout the entire reasoning process, the agent maintains a textual memory, which stores key summaries of the chunks. Each update uses the question, the previous memory, and the current chunk to write the latest memory. Consequently, the update operations for chunk $\boldsymbol{d}_t$ with $0 < t \leq T$ must execute sequentially.

2.2 Workflow: Parallel Reading, Sequential Reasoning

Sequential memory turns document length into dependency depth: processing $\boldsymbol{d}_t$ cannot start until the memory from $\boldsymbol{d}_{<t}$ has been written. ParSer instead turns document length into *parallel width*. It assigns one subagent per chunk and organizes their interaction with a lead agent via repeated **scatter-gather** rounds (the lower panel of Figure 1; Algorithm 1). At round k, the lead agent *scatters* one or more focused queries to all T subagents; the subagents read their respective chunks concurrently, and their local findings are *gathered* as the observation for the lead agent. Each round therefore covers the entire document in parallel: increasing T adds parallel readers rather than dependent reading steps.

Parallel chunk readers. Each subagent is bound to one chunk and, given a lead-agent query, returns a finding grounded only in that chunk. Chunks can be irrelevant to a given focused query, so subagents may abstain from answering; abstentions are dropped during gathering. Appendix F.2 shows the subagent prompt. All T subagents run in parallel at every round, each receiving the same query while retaining its fixed chunk assignment. Binding each subagent to a short chunk also keeps its effective context compact, mitigating the context rot issue that arises as input length grows. Together with the decomposed query, this makes the subagent's reading task simpler; we thus let the subagents run in non-thinking mode. We deploy the subagents with SGLang (Zheng et al., 2024) and achieve parallel execution across all subagents through concurrent request dispatching. To reduce repeated chunk prefill across rounds, prompts with fixed instruction-chunk prefixes and cache-aware request routing maximize cross-round KV-cache reuse for subagent inference (Appendix F.4).

Question-driven reasoner. The lead agent controls this parallel reading. It takes as input $\boldsymbol{q}$, but never $\boldsymbol{D}$ or any chunk $\boldsymbol{d}_t$, so it reasons about the question rather than the document. Appendix F.1 gives the lead-agent prompt. The lead agent conducts this reasoning in a multi-step ReAct (Yao et al., 2023) loop of interleaved thinking and action—after thinking, it either performs a scatter-gather operation or commits to a final answer. Queries in the latest round are conditioned on the reasoning history including previously gathered findings. Only this reasoning process is sequential; document-wide reading remains parallel in every round. The loop terminates when the lead agent answers or reaches the maximum number of rounds. The number of rounds actually executed, K, is determined by the reasoning hops required by $\boldsymbol{q}$ rather than the number of chunks T.

Parallel reading yields two immediate consequences. First, every chunk is read under the same query in each round and can be revisited under a new query. Access to evidence is therefore symmetric with respect to chunk position. Second, parallel reading removes document coverage from the sequential critical path: the dependent reasoning depth is K rounds rather than T chunks, and long documents satisfy $T \gg K$, yielding lower wall-clock latency. On the other hand, parallelizing the readers raises two concerns: whether independent chunk reading undermines cross-chunk dependencies needed for answering complex questions, and whether rereading all chunks incurs excessive computation.

Adaptive parallel reading across rounds. Although independent chunk reading prevents subagents from seeing cross-chunk dependencies, PARSER does not ask subagents to solve the original multi-hop question. Instead, dependencies that span chunks are carried across reasoning rounds by the lead agent: the lead agent breaks the question into focused queries that can usually be answerable within a single chunk; findings gathered at round k then condition the query at round $k+1$. Cross-chunk dependencies are thus resolved through successive parallel reading rounds.

Efficiency through sparsity. The same decomposition also sparsifies subagent communication. For each query, only a small number of subagents return findings, while most emit only a short abstention that is dropped. Sequential memory methods, in contrast, generate a memory update (with hundreds or thousands of tokens) after every chunk. While multi-round reading may preserve or increase prefill computation (depending on whether KV-cache reuse is available), PARSER cuts decoding computation through far fewer generated tokens. Appendix B gives detailed analysis.

Taken together, PARSER separates query-conditioned local reading from evidence-conditioned global reasoning, composing cross-chunk evidence through successive parallel reading rounds.

2.3 OPTIMIZATION: AGENTIC REINFORCEMENT LEARNING

The decoupling of reading from reasoning also determines what we train. For the comparatively simple reading task, a frozen off-the-shelf model suffices as the subagent. What still has to be learned is the lead agent's policy: how to determine the next action based on reasoning history. We therefore train *only* the lead agent π_θ and keep the subagents frozen.

We optimize the lead agent with Reinforcement Learning with Verifiable Reward (Shao et al., 2024; DeepSeek-AI, 2025a). The reward is a binary exact-match score $r(\boldsymbol{q}, \boldsymbol{y}) = \mathrm{EM}(\boldsymbol{a}_{pred}, \boldsymbol{a}_{gold})$, where $\boldsymbol{a}_{pred}$ is the answer extracted from the reasoning trajectory $\boldsymbol{y}$ and $\boldsymbol{a}_{gold}$ is the ground truth. We do not use format rewards, as the lead agent uses the backbone's native multi-turn tool-calling format. With this reward, we use Group Relative Policy Optimization (GRPO; Shao et al. 2024) to maximize

$$\mathcal{J}_{\mathrm{GRPO}}(\theta) = \mathbb{E}_{\boldsymbol{q},\{\boldsymbol{y}_i\}_{i=1}^{G}\sim\pi_{\mathrm{old}}(\cdot|\boldsymbol{q};\mathcal{S}(\boldsymbol{D}))}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\boldsymbol{y}_i|}\sum_{t=1}^{|\boldsymbol{y}_i|}\min\big(\rho_{i,t}(\theta)\hat{A}_{i,t},\ \mathrm{clip}(\rho_{i,t}(\theta),1-\epsilon,1+\epsilon)\,\hat{A}_{i,t}\big) - \beta\mathbb{D}_{\mathrm{KL}}[\pi_\theta\|\pi_{\mathrm{ref}}]\right] \tag{1}$$

where $\rho_{i,t}(\theta) = \frac{\pi_\theta(\boldsymbol{y}_{i,t}|\boldsymbol{q},\boldsymbol{y}_{i,<t};\mathcal{S}(\boldsymbol{D}))}{\pi_{\mathrm{old}}(\boldsymbol{y}_{i,t}|\boldsymbol{q},\boldsymbol{y}_{i,<t};\mathcal{S}(\boldsymbol{D}))}$ is the importance ratio, $\mathcal{S}(\boldsymbol{D})$ denotes the frozen subagents bound to the chunks of $\boldsymbol{D}$, ϵ is the PPO clipping hyperparameter, β is the KL regularization coefficient, and $\hat{A}_{i,t}$ denotes the advantage computed from the relative rewards of outputs in each group. We mask observation tokens, the findings gathered from $\mathcal{S}(\boldsymbol{D})$, so the policy gradient is applied only to tokens generated by the lead agent.

3 EXPERIMENTS

3.1 IMPLEMENTATION

Following prior work (Yu et al., 2026; Shi et al., 2026), we utilize multi-hop long-context question answering tasks for our training. We synthesized 32,768 training samples from the HotpotQA (Yang et al., 2018) dataset by following the recipe of Yu et al. (2026), and each synthetic sample has a context composed of 200 paragraphs, with a total token length of $\sim$ 28K tokens. More details of sample synthesis are in Appendix D.

We choose Qwen3.5-4B and Qwen3.5-9B (Qwen, 2026) as backbone models. During training, we impose an upper bound of 9 total turns, i.e., $K \leq 9$; each turn is capped at 2048 tokens. The document is chunked into at most 512 tokens per chunk. The subagents use a temperature of 0.7 and

a 512-token generation budget. To streamline the aggregation of findings from subagents, we instruct the subagents to structure their output in JSON format. At inference, we increase the cap of K to 12 and the chunk size to $4{,}096$ tokens.[1] For training, to avoid leaving the subagents idle during policy updates, we adopt fully asynchronous RL: all training runs on 6 H100 GPUs; 4 GPUs serve rollouts and 2 GPUs train the actor. We use 10 additional H100 GPUs to deploy subagents with SGLang. We let Qwen3.5-4B serve as the subagent for both 4B and 9B lead agents. More details of training and inference are in Appendix F.4 and F.6, respectively.

3.2 Baselines & Evaluation

We compare our method against the following baselines: (1) **Long-context LLMs**, including Qwen3.5 (Qwen, 2026) and DeepSeek-V4-Pro (DeepSeek-AI, 2026), which take the question together with the entire associated document as input to generate direct answers. DeepSeek-V4-Pro (preview, 2026-04-24) supports 1M-token context, and we set its reasoning effort mode as `Max`, the highest level of reasoning effort of this model. For Qwen3.5, we use the models' default configuration without context extension for evaluations within 262K tokens; furthermore, to support evaluations beyond the models' native 262K-token context window, we apply YaRN (Peng et al., 2024)-based RoPE scaling with a factor of 4.0, extending its context window to approximately 1M tokens. The temperature for them is set to 0. (2) **Agentic RAG** (Jin et al., 2025), in which an agent iteratively searches relevant chunks from the long document through Qwen3-Embedding-4B (Zhang et al., 2025) and performs reasoning. (3) **Sequential memory agents**, such as MemAgent (Yu et al., 2026) and ReMemR1 (Shi et al., 2026). The baselines in the latter two categories are trained with RL, and we reimplement them using the same training data and backbone models as PARSER. Appendix E shows the details of the re-implementation.

For evaluation, we use the in-distribution HotpotQA (Yang et al., 2018) and the out-of-distribution 2WikiMultiHopQA (Ho et al., 2020). We use the HotpotQA test samples released by Yu et al. (2026), and regenerate the 2WikiMultiHopQA ones with Shi et al.'s (2026) public script. To enable comprehensive evaluations across diverse document lengths, the documents of the test samples have lengths ranging from 7K to 896K tokens. Following Yu et al. (2026) and Shi et al. (2026), we report `Sub_EM` as the evaluation metric. For each training method, we select the checkpoint that achieves the best in-distribution overall performance and report the average score over 3 runs. We also evaluate PARSER on ten non-QA tasks from RULER (Hsieh et al., 2024) in Appendix C.2.

3.3 Main Results

Table 1 presents all methods' results on two benchmarks. PARSER consistently demonstrates higher accuracy than all baselines across all subsets on HotpotQA and long-document subsets (with ≥ 1600 paragraphs) on 2WikiMultiHopQA. For the three long-context LLMs, the baselines that incorporate entire documents as input suffer severe performance degradation as the document length increases. While the sequential memory agent methods can alleviate this issue to some extent by storing salient information in memory, PARSER exhibits the smallest or even zero performance drop.

Among the trained methods, MemAgent and ReMemR1 perform far worse under out-of-distribution (OOD) conditions than under in-distribution scenarios; whereas, PARSER maintains favorable performance on OOD cases. We attribute this robustness to our decoupling of reading from reasoning. The lead agent never sees the document, so its training develops general reasoning capabilities rather than learning to generate document-specific memorization, as memory-based baselines do. Meanwhile, the subagents, with access to the document, remain frozen as off-the-shelf models. This decoupled reading-and-reasoning strategy thereby helps mitigate overfitting to the training documents.

4 Analysis

We study robustness, inference latency, and the roles of subagents and chunking; unless stated otherwise, all experiments use Qwen3.5-4B. An RL ablation is in Appendix C.

[1] To pursue faster training, we intentionally select a smaller chunk size for the training phase, despite the resulting mismatch with inference-time chunk sizes. As given in Equation (2), the prefill cost $\mathcal{O}(n^2/T)$ decreases as the number of chunks T grows.

Table 1: Long-context QA results on HotpotQA (Yang et al., 2018) and 2WikiMultiHopQA (Ho et al., 2020). Values are accuracy (Sub_EM, %).

(a) Accuracy on HotpotQA (In-Distribution)

Backbone	Method	# Paragraphs (Total Length)								Avg.
		50 (7K)	100 (14K)	200 (28K)	400 (56K)	800 (112K)	1600 (224K)	3200 (448K)	6400 (896K)	
DeepSeek-V4-Pro	Full-context (non-think)	78.12	77.34	76.56	79.69	77.34	75.78	73.44	62.50	75.10
	Full-context (think-max)	82.03	82.81	80.47	80.47	80.47	81.25	77.34	78.91	80.47
Qwen3.5-4B	Full-context (non-think)	75.78	75.78	71.88	74.22	67.97	61.72	53.13	34.38	64.36
	Full-context (think)	80.47	78.91	78.12	76.56	74.22	60.94	46.09	31.25	65.82
	Agentic RAG	78.91	71.88	60.94	61.72	57.03	45.31	47.66	46.09	58.69
	MemAgent	81.25	81.51	83.59	77.86	77.86	79.69	73.96	72.92	78.58
	ReMemR1	82.03	79.95	82.03	78.91	77.86	79.95	77.08	73.44	78.91
	PARSER	**85.68**	**84.64**	**83.60**	**85.68**	**85.42**	**83.07**	**83.07**	**85.42**	**84.57**
Qwen3.5-9B	Full-context (non-think)	75.00	72.66	72.66	71.88	70.31	65.62	58.59	47.66	66.80
	Full-context (think)	77.34	74.22	78.91	76.56	77.34	65.62	53.91	46.88	68.85
	Agentic RAG	77.34	68.75	60.16	62.50	58.59	47.66	46.88	46.88	58.59
	MemAgent	81.77	80.73	82.03	79.95	79.95	81.25	79.69	75.00	80.05
	ReMemR1	81.25	79.69	77.60	78.39	78.13	78.13	77.86	76.04	78.39
	PARSER	**86.72**	**88.80**	**87.24**	**85.68**	**86.46**	**86.72**	**86.72**	**85.94**	**86.79**

(b) Accuracy on 2WikiMultiHopQA (Out-of-Distribution)

Backbone	Method	# Paragraphs (Total Length)								Avg.
		50 (7K)	100 (14K)	200 (28K)	400 (56K)	800 (112K)	1600 (224K)	3200 (448K)	6400 (896K)	
DeepSeek-V4-Pro	Full-context (non-think)	89.84	86.72	87.50	87.50	87.50	77.34	78.91	68.75	83.01
	Full-context (think-max)	90.62	89.06	92.19	90.62	92.19	87.50	82.81	80.47	88.18
Qwen3.5-4B	Full-context (non-think)	85.94	85.94	82.81	78.12	74.22	69.53	66.41	46.88	73.73
	Full-context (think)	88.28	**88.28**	**88.28**	83.59	82.81	70.31	63.28	39.06	75.49
	Agentic RAG	85.94	76.56	77.34	67.19	62.50	58.59	57.81	53.13	67.38
	MemAgent	67.45	73.18	69.01	61.46	54.17	54.43	60.16	45.05	60.61
	ReMemR1	**88.54**	86.85	83.20	83.07	76.83	67.71	72.40	60.68	77.41
	PARSER	86.98	84.64	**88.28**	**87.76**	**87.50**	**85.94**	**88.28**	**86.98**	**87.04**
Qwen3.5-9B	Full-context (non-think)	80.47	82.81	81.25	80.47	74.22	78.91	61.72	50.00	73.73
	Full-context (think)	88.28	**89.06**	**92.19**	85.16	**89.06**	82.81	57.81	52.34	79.59
	Agentic RAG	**89.06**	82.81	78.12	79.69	62.50	64.84	60.94	54.69	71.58
	MemAgent	80.73	82.03	78.91	75.26	70.57	65.63	75.78	60.94	73.73
	ReMemR1	84.12	87.50	77.87	83.59	76.30	74.74	79.43	70.57	79.27
	PARSER	87.76	88.80	88.80	**89.06**	87.24	**89.58**	**88.54**	**88.02**	**88.48**

4.1 WHY PARSER OUTPERFORMS SEQUENTIAL MEMORY AGENTS

To understand why PARSER outperforms sequential memory agents, we construct three controlled evaluations that vary complementary aspects of evidence distribution: absolute position, logical order, and relative distance. Rather than comparing performance across methods, we focus on each method's sensitivity to perturbations applied to these three dimensions.

Evidence Position Control We vary the absolute position of supporting evidence within long documents. For each of 512 test questions sampled from HotpotQA, we place all evidence-bearing paragraphs at random positions in the $[st, st + 10]$ percentile range of an 896K-token document, where st ranges from 0 to 90 in increments of 10. The distractor paragraphs and their positions stay fixed across variants. Figure 2(a) shows performance across the position-controlled test sets. MemAgent drops sharply when the supporting evidence lies between the 50th and 70th percentiles of the document: MemAgent sequentially compresses document chunks into a fixed-capacity memory, making evidence availability dependent on where the evidence enters the memory-update sequence. Specifically, when all pieces of evidence appear near the beginning, MemAgent can aggregate them and derive an answer early; evidence near the end undergoes few subsequent memory updates. Evidence in the middle faces both prior memory saturation and subsequent overwriting, making it less likely to be retained. ReMemR1 partially mitigates this positional sensitivity by retrieving information from earlier memory states through its callback module. In contrast, PARSER gives the queries symmetric access to every chunk through parallel reading, making evidence retrieval independent of position and achieving stable performance.

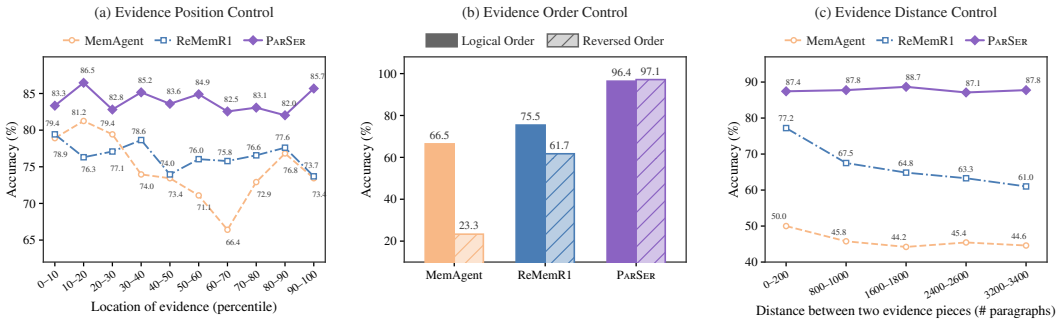

Figure 2: Performance under controlled evidence distributions. (a) Evidence position: supporting evidence is placed within different percentile ranges of documents. (b) Evidence order: evidence-bearing paragraphs in documents either follow or reverse their logical dependency order. (c) Evidence distance: the number of intervening paragraphs between two evidence pieces is varied.

Evidence Order Control We sample 512 two-hop bridge-comparison questions from 2WikiMultiHopQA[2]. In this question type, the supporting evidence forms a reasoning chain in which later hops depend on entities identified in earlier hops. For example, answering the question in Figure 5 requires first identifying the director of each film and then comparing the directors' dates of death. For each question, we build two 896K-token documents containing the same set of paragraphs, with all distractor paragraphs kept in the same positions. The two documents differ only in the evidence-bearing paragraphs' physical order in the document: one follows the logical dependency order, whereas the other reverses it (as in Figure 5). Figure 2(b) compares the two orders. Both MemAgent and ReMemR1 suffer a sharp performance drop when the order is reversed, whereas ParSer remains stable. In sequential memory agents, when an evidence paragraph precedes the prerequisite evidence needed to recognize its relevance to the question, it may either be ignored or, if stored initially, evicted from memory before the arrival of its prerequisite (see a MemAgent example in Appendix G.3). By revisiting all chunks with queries informed by previously gathered findings, ParSer follows the question's logical dependencies regardless of the evidence's physical order.

Evidence Distance Control We study how the distance between evidence pieces affects performance. We sample 512 questions requiring two pieces of supporting evidence from 2WikiMultiHopQA. For each question, we vary their separation by inserting different numbers of distractor paragraphs between the two evidence-bearing paragraphs. To avoid the confounds identified in the two preceding experiments, we place the two evidence paragraphs following their logical order and pad a fixed set of 1,600 paragraphs both before the first evidence paragraph and after the final evidence paragraph. Figure 2(c) shows that MemAgent and ReMemR1 degrade as the number of middle paragraphs increases, whereas ParSer remains stable. For sequential memory agents, the first evidence piece must survive an increasing number of memory updates before the second is encountered, making it more likely to be evicted from the fixed-capacity memory. In ParSer, the two evidence-bearing chunks are directly examined through parallel reading, and their findings are composed by the lead agent through a reasoning path independent of their physical distance.

Together, these experiments expose a common bottleneck in sequential memory agents: capacity-limited, document-ordered recurrent compression. ParSer decouples reading from reasoning into two complementary mechanisms: stateless, parallel reading gives symmetric access over the entire document regardless of evidence position or distance, while question-driven iterative reasoning aligns the inference path with the question's logical dependencies rather than the document's physical order.

4.2 Inference Latency

Table 2 reports amortized wall-clock inference time across all test subsets of HotpotQA with concurrency of 1, 16, and 32.[3] Each entry is the total subset wall-clock time divided by the size of the

[2]We use 2WikiMultiHopQA because, unlike HotpotQA, it provides ground-truth annotations of the logical order among pieces of supporting evidence.

[3]We exclude ReMemR1 from this analysis because it equips MemAgent with an extra retrieval module, which theoretically introduces higher inference latency than MemAgent.

Table 2: Amortized inference time per sample (sec), computed as the total subset wall-clock time divided by the subset size (128), across document lengths under different concurrency. Full-context methods fail to finish 896K-token evaluation at 16 or 32 concurrency due to GPU memory limits.

Concurrency	Method	# Paragraphs (Total Length)							
		50 (7K)	100 (14K)	200 (28K)	400 (56K)	800 (112K)	1600 (224K)	3200 (448K)	6400 (896K)
1	Full-context (non-think)	1.16	1.86	2.11	3.52	6.25	18.36	51.86	185.42
	Full-context (think)	12.33	17.78	18.93	31.89	42.73	68.19	162.97	375.32
	MemAgent	10.56	15.99	30.40	58.49	112.59	222.86	438.61	876.20
	ParSer	5.33	5.84	6.83	8.71	13.30	22.43	42.19	78.22
16	Full-context (non-think)	0.63	0.91	1.11	2.55	9.46	28.00	94.22	–
	Full-context (think)	2.00	2.84	4.05	8.70	21.37	53.95	177.01	–
	MemAgent	1.49	2.28	3.76	7.13	13.61	26.24	51.80	101.94
	ParSer	0.87	1.09	1.83	3.30	7.38	15.24	31.31	58.86
32	Full-context (non-think)	0.35	0.65	1.07	3.85	9.23	27.98	93.45	–
	Full-context (think)	1.70	2.25	3.39	9.32	20.64	54.16	191.91	–
	MemAgent	1.21	1.63	2.76	5.14	9.76	19.61	37.50	74.96
	ParSer	0.64	0.99	1.89	3.66	7.36	15.40	30.77	58.76

Table 3: Effect of subagent size on HotpotQA (Sub_EM, %). The lead agent is fixed as Qwen3.5-4B.

Subagent	# Paragraphs (Total Length)								Avg.
	50 (7K)	100 (14K)	200 (28K)	400 (56K)	800 (112K)	1600 (224K)	3200 (448K)	6400 (896K)	
Qwen3.5-2B	82.55	84.38	82.29	84.11	77.87	76.56	69.01	69.27	78.26
Qwen3.5-4B (default)	**85.68**	84.64	83.60	**85.68**	**85.42**	83.07	**83.07**	85.42	84.57
Qwen3.5-9B	82.81	**85.16**	**87.50**	84.38	84.38	**85.16**	82.81	**85.94**	**84.77**

subset (128), not per-request latency (which would typically increase under higher concurrency due to contention). Models are deployed with SGLang: we allocate one H100 GPU for Full-context and MemAgent, while ParSer employs two GPUs, one H100 dedicated to all subagents and one RTX3090 for the lead agent. For ParSer, the lead agent and subagents within a single inference instance run in an alternating fashion, as the lead agent has to await outputs from all subagents. Comparisons between our two-GPU ParSer and single-GPU baselines are thus valid.

Across all concurrency settings, Full-context (non-thinking) achieves the lowest latency on short-document subsets because it processes the input in one pass and generates only a short output. As document length increases, however, its single-pass processing becomes increasingly costly due to the quadratic complexity of attention module. Under high concurrency, GPU memory limits even prevent it from processing 896K-token inputs. In contrast, MemAgent and ParSer process documents in fixed-length chunks, supporting longer documents within limited GPU memory.

Figure 3 links inference step counts to latency. At a concurrency of 1, MemAgent's latency grows linearly with document length because its number of inference steps is proportional to the number of document chunks. In contrast, ParSer processes document chunks in parallel, reducing the number of sequential inference steps to the number of reasoning hops required to solve the question itself. Because the same set of questions is used across all subsets, the number of reasoning hops—and hence the inference step count of ParSer—remains nearly constant even as document length increases significantly. Consequently, ParSer achieves lower latency than MemAgent, with an order-of-magnitude advantage on long-document subsets. At higher concurrency, a more practical condition, MemAgent benefits from batching and closes the latency gap. Still, ParSer maintains lower latency across the evaluated concurrency levels. Appendix B explains why this advantage shrinks at higher concurrency.

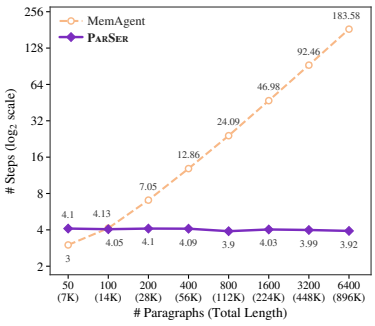

Figure 3: Step counts of MemAgent and ParSer across lengths.

4.3 Effect of Subagent

Subagent Size Besides the default Qwen3.5-4B, we also try 2B and 9B models as alternative subagents without retraining the lead agents. In Table 3, accuracy rises from the 2B subagent to the

Table 4: Effect of chunk size on HotpotQA (Sub_EM, %). All variants are based on Qwen3.5-4B. For the Full-document setting, YaRN-based RoPE scaling is enabled on the 3200 (448K) and 6400 (896K) subsets to accommodate contexts beyond the model's native window.

Chunk size	# Paragraphs (Total Length)								Avg.
	50 (7K)	100 (14K)	200 (28K)	400 (56K)	800 (112K)	1600 (224K)	3200 (448K)	6400 (896K)	
4,096	85.68	84.64	83.60	85.68	85.42	83.07	83.07	85.42	84.57
Entire	83.85	84.64	82.29	83.86	74.48	71.10	56.77	53.13	73.76
16,384	82.03	82.29	85.94	82.81	82.81	83.86	85.42	83.59	83.59
65,536	84.64	82.03	80.73	80.73	78.91	80.99	83.59	83.08	81.84
131,072	83.07	83.59	80.99	80.21	75.78	80.21	79.69	79.17	80.34

Table 5: Comparison of alternative subagent implementations with their full-context counterparts on HotpotQA (Sub_EM, %). All agents are built on Qwen3.5-4B.

Agent	# Paragraphs (Total Length)								Avg.
	50 (7K)	100 (14K)	200 (28K)	400 (56K)	800 (112K)	1600 (224K)	3200 (448K)	6400 (896K)	
DCI agent	73.96	75.78	74.74	76.56	74.48	75.52	77.60	76.30	75.61
Lead agent + DCI subagents	**86.72**	**88.28**	**89.06**	**86.72**	**85.94**	**80.47**	**81.25**	**78.91**	**84.67**
Full-context (think)	80.47	78.91	78.12	76.56	74.22	60.94	46.09	31.25	65.82
Lead agent + thinking subagents	**85.16**	**86.72**	**87.50**	**82.03**	**85.94**	**85.94**	**87.50**	**83.59**	**85.55**

4B version, and then plateaus at the 9B subagent. We attribute this saturation to the simplicity of the subagent task. After question decomposition and document chunking, each subagent only needs to answer a focused query over a short context, for which the 4B model already suffices. This shows the deployment efficiency of PARSER via the adoption of lightweight subagents.

Chunk Size The subagents of PARSER use the same working mode as the Full-context (non-thinking) baseline, but receive inputs at a different granularity (document chunk vs. entire document). Thus, the gain over Full-context (non-thinking) in Table 1 stems from both the lead agent's guidance and the chunk-level document decomposition. To separate these effects, we ablate chunking in PARSER, where a single subagent processes the entire document. Table 4 shows a clear drop without chunking, particularly on long-document subsets. We also evaluate intermediate chunk sizes and observe that performance drops as chunk size increases. This suggests that LLMs struggle more to capture key information from longer inputs than from shorter ones, consistent with *context rot* (Hong et al., 2025; Liu et al., 2024). PARSER addresses this with multiple subagents, and each subagent processes only a short chunk of the document.

4.4 COMPATIBILITY WITH ALTERNATIVE SUBAGENT IMPLEMENTATIONS

PARSER defaults to direct-answer subagents. We test whether the lead agent is compatible with other subagent implementations without retraining. We add two subagent variants: the thinking subagents and the DCI subagents. Direct Corpus Interaction (DCI; Li et al. 2026; Salemi et al. 2026; Sen et al. 2026) refers to the agentic search paradigm that lets LLMs directly query raw text corpora via composable Unix shell tools like `rg` and `grep` for fine-grained lexical matching and multi-step evidence collection. We follow the implementation of Li et al. (2026) to build our DCI subagents (Appendix E.2). Table 5 shows both variants of PARSER beat their corresponding standalone baselines. Thus the lead agent can effectively coordinate diverse subagent implementations.

5 CONCLUSION

We presented PARSER, a long-context reasoning framework that decouples reading from reasoning: frozen subagents read chunks in parallel, while an RL-trained lead agent iteratively refines queries and aggregates evidence. Across contexts from 7K to 896K tokens, PARSER outperforms the strongest sequential memory baselines by 5.7–6.7 points on average (up to 12.0 at 896K), while its 9B variant surpasses the million-token full-context DeepSeek-V4-Pro by 6.3 points. It also reduces latency while remaining robust to context length and evidence placement.

AI USE STATEMENT

In this work, we used generative AI tools to polish some passages of the manuscript, to search for and identify relevant literature, and to debug software code. We have reviewed all AI-assisted work. We checked and revised every AI-edited passage, read the original papers located by search, and verified debugging suggestions for correctness. We have not used generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate mathematical claims, propose or refine hypotheses, design or provide feedback on the methodology or experiments, implement the method, translate the manuscript, clean or reformat datasets, conduct qualitative or thematic analysis, or interpret the results. Assisting in the writing of proofs and providing ingredients for proving mathematical claims are not applicable to this work. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have provided the implementation of PARSER in Appendix F, including the prompts (Appendix F.1 and Appendix F.2), the RL training framework setup (Appendix F.4), the training hyperparameters (Table 9), and the inference configuration (Appendix F.6). Construction of the training and evaluation datasets, including the public sources and scripts used to build them, is described in Appendix D. The training and evaluation code, as well as the datasets and model weights, will be available on open-source platforms.

ETHICS STATEMENT

This work studies long-context question answering on publicly released benchmarks. We do not collect new data, involve human subjects, or release personal information. Training and evaluation follow existing public data-construction recipes and use only documents already distributed with those benchmarks.

The method is a general document-reading agent. On high-stakes text, such as legal or medical records, it can still omit evidence or produce incorrect answers, and it is not a substitute for professional review.

REFERENCES

Anthropic. How we built our multi-agent research system. Anthropic Engineering Blog, June 2025. URL https://www.anthropic.com/engineering/built-multi-agent-research-system.

Howard Chen, Ramakanth Pasunuru, Jason Weston, and Asli Celikyilmaz. Walking down the memory maze: Beyond context limit through interactive reading, 2023a. URL https://arxiv.org/abs/2310.05029.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation, 2023b. URL https://arxiv.org/abs/2306.15595.

Yufan Dang, Chen Qian, Xueheng Luo, Jingru Fan, Zihao Xie, Ruijie Shi, Weize Chen, Cheng Yang, Xiaoyin Che, Ye Tian, Xuantang Xiong, Lei Han, Zhiyuan Liu, and Maosong Sun. Multi-agent collaboration via evolving orchestration. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2505.19591.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025a. URL https://arxiv.org/abs/2501.12948.

DeepSeek-AI. Native sparse attention: Hardware-aligned and natively trainable sparse attention, 2025b. URL https://arxiv.org/abs/2502.11089.

DeepSeek-AI. Deepseek-v3.2: Pushing the frontier of open large language models, 2025c. URL https://arxiv.org/abs/2512.02556.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. LongRoPE: Extending LLM context window beyond 2 million tokens. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Jiani Guo, Zuchao Li, Jie Wu, Qianren Wang, Yun Li, Lefei Zhang, Hai Zhao, and Yujiu Yang. Tom: Leveraging tree-oriented mapreduce for long-context reasoning in large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025. arXiv:2511.00489.

Naman Gupta, Vaibhav Singh, Arun Iyer, Kirankumar Shiragur, Pratham Grover, Ramakrishna B. Bairi, Ritabrata Maiti, Sankarshan Damle, Shachee Mishra Gupta, Rishikesh Maurya, and Vageesh D. C. Chow-liu ordering for long-context reasoning in chain-of-agents, 2026. URL https://arxiv.org/abs/2603.09835.

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, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. URL https://www.aclweb.org/anthology/2020.coling-main.580.

Kelly Hong, Anton Troynikov, and Jeff Huber. Context rot: How increasing input tokens impacts llm performance. Technical report, Chroma, July 2025. URL https://trychroma.com/research/context-rot.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling (COLM)*, 2024. arXiv:2404.06654.

Mengkang Hu, Yuhang Zhou, Wendong Fan, Yuzhou Nie, Bowei Xia, Tao Sun, Ziyu Ye, Zhaoxuan Jin, Yingru Li, Qiguang Chen, Zeyu Zhang, Yifeng Wang, Qianshuo Ye, Bernard Ghanem, Ping Luo, and Guohao Li. Owl: Optimized workforce learning for general multi-agent assistance in real-world task automation, 2025. URL https://arxiv.org/abs/2505.23885.

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

Jiajie Jin, Yutao Zhu, Xinyu Yang, Chenghao Zhang, and Zhicheng Dou. Flashrag: A modular toolkit for efficient retrieval-augmented generation research. *CoRR*, abs/2405.13576, 2024. URL https://arxiv.org/abs/2405.13576.

Kimi Team. Kimi k3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

Kuang-Huei Lee, Xinyun Chen, Hiroki Furuta, John Canny, and Ian Fischer. A human-inspired reading agent with gist memory of very long contexts. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2402.09727.

Xiaomin Li, Zhou Yu, Zhiwei Zhang, Xupeng Chen, Ziji Zhang, Yingying Zhuang, Narayanan Sadagopan, and Anurag Beniwal. When thinking fails: The pitfalls of reasoning for instruction-following in LLMs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=w5uUvxp81b.

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.

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. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Dongyang Ma, Yan Wang, and Lan Tian. Block-attention for efficient prefilling. In *The Thirteenth International Conference on Learning Representations*, 2025. arXiv:2409.15355.

Thomas Merth, Qichen Fu, Mohammad Rastegari, and Mahyar Najibi. Superposition prompting: Improving and accelerating retrieval-augmented generation. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2404.06910.

MiniMax. MiniMax sparse attention, 2026. URL https://arxiv.org/abs/2606.13392.

Moonshot AI. Kimi k2.5: Visual agentic intelligence. Technical Blog, 2025. URL https://www.kimi.com/en/blog/kimi-k2-5.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=wHBfxhZu1u.

Qwen. Qwen3.5: Accelerating productivity with native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Nir Ratner, Yoav Levine, Yonatan Belinkov, Ori Ram, Inbal Magar, Omri Abend, Ehud Karpas, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. Parallel context windows for large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023. arXiv:2212.10947.

Alireza Salemi, Chang Zeng, Atharva Nijasure, Jui-Hui Chung, Razieh Rahimi, Fernando Diaz, and Hamed Zamani. Grepseek: Training search agents for direct corpus interaction, 2026. URL https://arxiv.org/abs/2605.29307.

Sahil Sen, Akhil Kasturi, Elias Lumer, Anmol Gulati, and Vamse Kumar Subbiah. Is grep all you need? how agent harnesses reshape agentic search, 2026. URL https://arxiv.org/abs/2605.15184.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, pp. 1279–1297, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400711961. doi: 10.1145/3689031.3696075. URL https://doi.org/10.1145/3689031.3696075.

Leheng Sheng, Yongtao Zhang, Wenchang Ma, Yaorui Shi, Ting Huang, Xiang Wang, An Zhang, Ke Shen, and Tat-Seng Chua. When to memorize and when to stop: Gated recurrent memory for long-context reasoning, 2026. URL https://arxiv.org/abs/2602.10560.

Yaorui Shi, Yuxin Chen, Siyuan Wang, Sihang Li, Hengxing Cai, Qi GU, Xiang Wang, and An Zhang. Look back to reason forward: Revisitable memory for long-context LLM agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=1cymflI2Lh.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Sibo Xiao, Zixin Lin, Wenyang Gao, Hui Chen, and Yue Zhang. Long context scaling: Divide and conquer via multi-agent question-driven collaboration, 2025. URL https://arxiv.org/abs/2505.20625.

Zhen Xu, Shang Zhu, Jue Wang, Junlin Wang, Ben Athiwaratkun, Chi Wang, James Zou, and Ce Zhang. When does divide and conquer work for long context llm? a noise decomposition framework. In *The Fourteenth International Conference on Learning Representations*, 2026. arXiv:2506.16411.

Xinyu Yang, Tianqi Chen, and Beidi Chen. Ape: Faster and longer context-augmented generation via adaptive parallel encoding. In *The Thirteenth International Conference on Learning Representations*, 2025. arXiv:2502.05431.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

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

Howard Yen, Tianyu Gao, and Danqi Chen. Long-context language modeling with parallel context encoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024. arXiv:2402.16617.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and Hao Zhou. Memagent: Reshaping long-context LLM with multi-conv RL-based memory agent. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=k5nIOvYGCL.

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

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Ö. Arik. Chain of agents: Large language models collaborating on long-context tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2406.02818.

Jun Zhao, Can Zu, Hao Xu, Yi Lu, Wei He, Yiwen Ding, Tao Gui, Qi Zhang, and Xuanjing Huang. Longagent: Scaling language models to 128k context through multi-agent collaboration, 2024. URL https://arxiv.org/abs/2402.11550.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=VqkAKQibpq.

Zihan Zhou, Chong Li, Xinyi Chen, Shuo Wang, Yu Chao, Zhili Li, Haoyu Wang, Rongqiao An, Qi Shi, Zhixing Tan, Xu Han, Xiaodong Shi, Zhiyuan Liu, and Maosong Sun. Llm×mapreduce: Simplified long-sequence processing using large language models, 2024. URL https://arxiv.org/abs/2410.09342.

A Related Work

A.1 Long-Context LLMs

Supporting million-token contexts *efficiently* has driven two complementary lines of architectural work. Positional interpolation rescales rotary embeddings so a model trained on short sequences extrapolates to far longer ones (Chen et al., 2023b; Peng et al., 2024; Ding et al., 2024). A separate family of attention mechanisms attacks the quadratic cost that dominates at long context: sparse attention attends only to a learned subset of relevant tokens per query (DeepSeek-AI, 2025b;c; MiniMax, 2026), and linear-attention variants reduce the complexity to linear in sequence length (Kimi Team, 2026). Yet a longer window does not by itself yield better *use* of that window. Models systematically underuse evidence placed in the middle of their input (Liu et al., 2024), and accuracy degrades as the input grows even when the nominal window is far from full, a phenomenon documented as *context rot* (Hong et al., 2025). A prominent response sidesteps the window limit altogether by reading the document in chunks while maintaining a compact textual memory that is repeatedly rewritten. Training-free methods precompute and then navigate such a memory. ReadAgent (Lee et al., 2024) uses gist lookup and MemWalker (Chen et al., 2023a) uses a summary tree, while Chain-of-Agents (Zhang et al., 2024) assigns one chunk per worker but passes a single message *sequentially* down the chain. MemAgent (Yu et al., 2026) instead trains this recurrent read-and-compress workflow end-to-end with reinforcement learning. ReMemR1 (Shi et al., 2026) adds a callback that revisits earlier memory states, and GRU-Mem (Sheng et al., 2026) introduces gated updates with an early-exit mechanism. These methods share one commitment: processing T chunks requires T *dependent* steps. Three consequences follow. Latency grows linearly with document length, a fixed-capacity memory must irreversibly decide what to retain before downstream relevance can be known, and the outcome depends on the order in which evidence is encountered (Gupta et al., 2026). §4.1 confirms that these are measurable biases with respect to evidence position, order, and separation.

A.2 Parallel Reading via Orchestrator–Worker Architectures

Reading chunks independently and aggregating their results is the natural parallel alternative to a sequential memory. Map-reduce pipelines such as LLM×MapReduce (Zhou et al., 2024) and ToM (Guo et al., 2025) explore this direction, but most are *single-shot*: the query sent to each chunk is fixed before any chunk is read. This cannot handle multi-hop questions, where later hops are not recognizable until earlier ones are found (Xu et al., 2026). LongAgent (Zhao et al., 2024) and XpandA (Xiao et al., 2025) pair a leader with per-chunk agents over multiple rounds, but coordinate through hand-specified protocols rather than a learned policy. A separate group achieves parallelism *inside* the model by encoding chunks independently and fusing them at the attention level (Ratner et al., 2023; Merth et al., 2024; Ma et al., 2025; Yang et al., 2025; Yen et al., 2024), but these are query-agnostic, single-round, and require architecture surgery. Structurally, a leader dispatching subtasks to workers that each run in an isolated context window is by now a common pattern in agentic systems (Anthropic, 2025), since a worker's intermediate tokens never occupy the leader's context. A growing line of work trains *only* this orchestrator while keeping the workers frozen (Hu et al., 2025; Dang et al., 2025), a design echoed by commercial agent swarms that optimize the scheduler alone (Moonshot AI, 2025). ParSer is the *long-context instantiation* of this design. Its subagents are bound to a disjoint partition of the input, so coverage is guaranteed by construction and the lead agent's task reduces to query formulation and aggregation. This structure makes freezing the subagents viable (§4.4) and keeps training cost independent of document length.

B Time Complexity Analysis

We analyze the time complexity of Full-context (direct answering given the entire document), MemAgent, and ParSer. ReMemR1 operates in a pipeline highly similar to MemAgent, so they have the same level of time complexity. Given a question and a document with n tokens, Full-context and MemAgent generate responses with r_F and r_M tokens, respectively. For ParSer, the lead agent and all subagents generate r_L and r_S tokens in total, respectively. For MemAgent and ParSer, each document is split into T chunks. As documents have significantly more tokens than the concatenation of task instructions and questions, the effective input sequence length for all three approaches can be approximated as n. To simplify the complexity derivation, we assume *the GPU memory capacity*

Table 6: Average output token counts per sample under different document lengths. Both methods are implemented based on Qwen3.5-4B. MemAgent entries report the total output tokens. For PARSER, entries report the combined output-token counts of the lead agent and all subagents. The final row reports the ratio of PARSER to MemAgent theoretical decoding computation, calculated from Equations (4) and (7) with $n/T = 5000$ and $K = 4$, under the assumption of sufficient GPU memory to retain all chunk KV caches.

	# Paragraphs (Total Length)							
	50 (7K)	100 (14K)	200 (28K)	400 (56K)	800 (112K)	1600 (224K)	3200 (448K)	6400 (896K)
MemAgent (# tokens)	2211	3379	6353	12334	23805	47174	93806	187058
PARSER (# tokens by Lead + Subagents)	534+312	533+348	525+379	485+392	523+474	522+515	546+586	541+804
PARSER/MemAgent (Decode Computation)	13.9%	10.6%	6.1%	3.2%	2.0%	1.1%	0.6%	0.4%

is sufficient to hold the entire contextual KV cache, thereby eliminating recomputation of context's key-value representations.

Prefill Owing to the inherent properties of the attention mechanism in Transformer architectures (Vaswani et al., 2017), the Full-context baseline exhibits the highest prefill-phase time complexity $\mathcal{O}(n^2)$, which scales quadratically with the input sequence length. Through the adoption of chunk-level input, MemAgent and PARSER reduce the prefill-phase time complexity as

$$T \cdot \mathcal{O}\left(\left(\frac{n}{T}\right)^2\right) = \mathcal{O}\left(\frac{n^2}{T}\right), \tag{2}$$

where each of the T chunks contains n/T tokens and is encoded independently.

Decode Under cached autoregressive decoding, each newly generated token attends to all preceding input and output tokens. Thus, decoding r_F tokens from the n-token Full-context input has time complexity

$$\mathcal{O}\left(\sum_{t=1}^{r_F}(n+t)\right) = \mathcal{O}\left(nr_F + r_F^2\right). \tag{3}$$

Based on the operation pipeline of MemAgent, for each chunk, the model generates an average of r_M/T tokens for memory update. Its decoding complexity is therefore

$$T \cdot \mathcal{O}\left(\frac{n}{T}\frac{r_M}{T} + \left(\frac{r_M}{T}\right)^2\right) = \mathcal{O}\left(\frac{nr_M + r_M^2}{T}\right). \tag{4}$$

For PARSER, let K denote the number of lead-agent rounds and r'_S the average number of tokens generated by each of the T subagents per round. Thus, the total number of tokens generated by all subagents is $r_S = KTr'_S$. In each round, the T subagents independently decode over their respective n/T-token chunks. Their aggregate decoding computation is

$$KT \cdot \mathcal{O}\left(\frac{n}{T}r'_S + (r'_S)^2\right) = \mathcal{O}\left(Knr'_S + KT\,(r'_S)^2\right). \tag{5}$$

The lead agent receives an observation of r_S tokens generated by subagents and generates r_L tokens. Its decoding complexity is $\mathcal{O}(r_S r_L + r_L^2)$. Consequently, the total decoding computation of PARSER is

$$\mathcal{O}\left(Knr'_S + KT\,(r'_S)^2 + r_S r_L + r_L^2\right). \tag{6}$$

Using the total subagent output $r_S = KTr'_S$, this complexity can equivalently be written as

$$\mathcal{O}\left(\frac{nr_S}{T} + \frac{r_S^2}{KT} + r_S r_L + r_L^2\right). \tag{7}$$

When the T subagents are executed in parallel, their decoding contribution to wall-clock latency is reduced by a factor of T. The corresponding latency is

$$\mathcal{O}\left(\frac{nr_S}{T^2} + \frac{r_S^2}{KT^2} + r_S r_L + r_L^2\right). \tag{8}$$

Based on the decoding time-complexity formulas in (4) and (7), we compute the ratio of PARSER to MemAgent decoding computation under each document length; the resulting ratios are reported in the last row of Table 6. The large gap arises mainly from the disparity in total generated tokens: MemAgent produces a memory of about 1K tokens for every chunk, whereas each PARSER subagent emits only a short finding or an "Unknown" response for its chunk—and the latter dominates in most cases. As a result, MemAgent's output length grows roughly with the number of chunks, while PARSER's remains comparatively small and stable, yielding substantially lower decoding complexity. Moreover, PARSER exposes parallelism across the T independent subagents. With sufficient hardware resources, this reduces the subagent contribution to wall-clock decoding latency from $\mathcal{O}(nr_S/T + r_S^2/(KT))$ to $\mathcal{O}(nr_S/T^2 + r_S^2/(KT^2))$, while the lead-agent terms remain sequential.

Note that this analysis assumes that GPU memory is sufficient to retain the KV caches for all encoded chunks. When this assumption does not hold, especially at high request concurrency, subagents' chunk KV caches may be evicted. The subagents must then re-prefill their chunks in each query round, increasing the aggregate prefill complexity from $\mathcal{O}(n^2/T)$ to $\mathcal{O}(Kn^2/T)$. With $K = 4$, as measured in §4.2, this is four times the prefill computation of MemAgent. In contrast, MemAgent updates memory by sequentially scanning chunks and does not revisit previously processed chunks; thus, it is unaffected by this KV-cache retention constraint. This explains why PARSER's latency advantage is less pronounced at higher concurrency, as shown in §4.2.

C Additional Experiments

C.1 Efficacy of Reinforcement Learning

Table 7 compares MemAgent, ReMemR1, and PARSER with and without RL training. Within the PARSER group, RL brings a clear gain: average HotpotQA accuracy rises from 74.42 to 84.57 on Qwen3.5-4B and from 76.24 to 86.79 on Qwen3.5-9B, with a smaller but consistent lift on out-of-distribution 2WikiMultiHopQA ($82.85 \rightarrow 87.04$ and $84.86 \rightarrow 88.48$). These gaps confirm that RL is an effective complement to PARSER.

Even without RL, PARSER already outperforms MemAgent and ReMemR1 by a wide margin and remains essentially flat as document length grows, whereas the sequential memory agents degrade sharply. We attribute this to a closer match between PARSER and the pretrained model: the lead agent follows the model's native multi-turn tool-calling template, a format the backbone is already trained to follow, whereas sequential memory agents impose a custom recurrent memory-update interface that the pretrained checkpoint has never seen.

C.2 Out-of-Distribution Evaluation on RULER

To examine whether PARSER generalizes beyond the multi-hop QA formats used for training, we evaluate it on ten synthetic tasks from the RULER benchmark (Hsieh et al., 2024). We directly use the test instances constructed by MemAgent (Yu et al., 2026), which cover context lengths from 8K to 512K tokens. The tasks span three long-context capabilities. The needle-in-a-haystack (NIAH) suite evaluates retrieval: the three single-key variants retrieve one target key–value pair under different distractor densities; the three multi-key variants retrieve a target among multiple distractor needles; multi-value requires extracting all values associated with the same key; and multi-query requires answering multiple distinct keys. Variable Tracking evaluates multi-hop tracing over chains of entity assignments, while Frequent Words Extraction evaluates aggregation by asking for the most frequent words in a power-law word distribution.

We evaluate both the 4B and 9B PARSER lead agents. For both variants, every chunk-bound subagent uses Qwen3.5-4B with a chunk size of $8{,}192$ tokens. All other inference settings follow the main experiments.

As shown in Figure 4, PARSER achieves consistently high Sub_EM across context lengths on most tasks. Interestingly, for VT and FWE, the performance of the 4B lead agent improves rather than degrades as the context grows. Inspection of these cases suggests that longer context distributes the fixed number of key information pieces across more individual chunks, and the independently operating subagents can then surface more pieces from their respective chunks.

Table 7: Effect of reinforcement learning on HotpotQA (Yang et al., 2018) and 2WikiMultiHopQA (Ho et al., 2020). Values are accuracy (Sub_EM, %).

(a) Accuracy on HotpotQA (In-Distribution)

Backbone	Method	# Paragraphs (Total Length)								Avg.
		50 (7K)	100 (14K)	200 (28K)	400 (56K)	800 (112K)	1600 (224K)	3200 (448K)	6400 (896K)	
Qwen3.5-4B	MemAgent w/o RL	69.53	62.76	58.86	50.78	40.62	32.29	25.78	19.79	45.05
	MemAgent w/ RL	81.25	81.51	83.59	77.86	77.86	79.69	73.96	72.92	78.58
	ReMemR1 w/o RL	66.41	63.28	59.90	48.44	44.53	34.38	27.87	20.58	45.67
	ReMemR1 w/ RL	82.03	79.95	82.03	78.91	77.86	79.95	77.08	73.44	78.91
	ParSer w/o RL	72.92	76.04	73.96	74.48	76.04	73.96	75.00	72.92	74.42
	ParSer w/ RL	85.68	84.64	83.60	85.68	85.42	83.07	83.07	85.42	84.57
Qwen3.5-9B	MemAgent w/o RL	68.49	61.20	53.64	52.08	47.14	38.80	26.82	26.30	46.81
	MemAgent w/ RL	81.77	80.73	82.03	79.95	79.95	81.25	79.69	75.00	80.05
	ReMemR1 w/o RL	61.20	53.64	46.10	46.35	42.71	39.58	40.62	32.03	45.28
	ReMemR1 w/ RL	81.25	79.69	77.60	78.39	78.13	78.13	77.86	76.04	78.39
	ParSer w/o RL	77.60	74.22	74.48	74.74	79.43	77.34	76.56	75.52	76.24
	ParSer w/ RL	86.72	88.80	87.24	85.68	86.46	86.72	86.72	85.94	86.79

(b) Accuracy on 2WikiMultiHopQA (Out-of-Distribution)

Backbone	Method	# Paragraphs (Total Length)								Avg.
		50 (7K)	100 (14K)	200 (28K)	400 (56K)	800 (112K)	1600 (224K)	3200 (448K)	6400 (896K)	
Qwen3.5-4B	MemAgent w/o RL	72.92	72.66	62.76	59.63	46.61	44.01	42.71	34.90	54.53
	MemAgent w/ RL	67.45	73.18	69.01	61.46	54.17	54.43	60.16	45.05	60.61
	ReMemR1 w/o RL	77.86	71.88	60.68	56.25	52.08	46.35	45.31	36.98	55.93
	ReMemR1 w/ RL	88.54	86.85	83.20	83.07	76.83	67.71	72.40	60.68	77.41
	ParSer w/o RL	82.29	84.38	85.68	84.90	80.99	81.51	83.33	79.69	82.85
	ParSer w/ RL	86.98	84.64	88.28	87.76	87.50	85.94	88.28	86.98	87.04
Qwen3.5-9B	MemAgent w/o RL	74.48	69.79	60.94	60.42	50.26	44.53	44.79	31.51	54.59
	MemAgent w/ RL	80.73	82.03	78.91	75.26	70.57	65.63	75.78	60.94	73.73
	ReMemR1 w/o RL	77.34	71.09	61.20	63.28	55.21	50.52	48.96	40.89	58.56
	ReMemR1 w/ RL	84.12	87.50	77.87	83.59	76.30	74.74	79.43	70.57	79.27
	ParSer w/o RL	84.37	86.46	86.98	83.07	83.59	83.86	85.94	84.64	84.86
	ParSer w/ RL	87.76	88.80	88.80	89.06	87.24	89.58	88.54	88.02	88.48



Figure 4: Out-of-distribution performance of the 4B and 9B ParSer on ten RULER tasks across context lengths from 8K to 512K tokens. Each cell reports Sub_EM (%).

D Datasets

D.1 Training Data Construction

We construct the training set by following Stage I of MemAgent (Yu et al., 2026); ReMemR1 (Shi et al., 2026) adopts the same Stage I recipe. We do *not* use MemAgent's Stage II data.

Concretely, we start from HotpotQA (Yang et al., 2018) training questions and retain each question's supporting Wikipedia articles as gold evidence. Following the RULER (Hsieh et al., 2024)-style packing used by MemAgent (Yu et al., 2026), we pad each sample with distractor articles sampled from the same HotpotQA corpus until the context contains 200 paragraphs ($\sim$ 28K tokens), then

Table 8: Statistics of training and evaluation data. Training contexts use 200 paragraphs ($\sim$ 28K tokens). Evaluation contexts range from 50 to 6400 paragraphs ($\sim$ 7K–896K tokens), with the same 128 questions reused across the eight length settings.

Split	# Samples	# Paragraphs per sample
Training (HotpotQA)	32,768	200
Eval (HotpotQA)	128×8	$\{50, 100, 200, 400, 800, 1600, 3200, 6400\}$
Eval (2WikiMultiHopQA)	128×8	$\{50, 100, 200, 400, 800, 1600, 3200, 6400\}$

shuffle the paragraph order with a fixed random seed. To remove questions that are already solvable from parametric knowledge alone, we query Qwen3.5-9B (Qwen, 2026) in the non-thinking mode *without* providing any document context, sample Best-of-3 responses, and discard any question for which the model achieves a 100% score under the rule-based verifier (substring / boxed-answer matching as in MemAgent). $41{,}027$ HotpotQA training examples are processed through this pipeline; we take the first $32{,}768$ remaining samples as our RL training set.

D.2 Evaluation Data Construction

HotpotQA (in-distribution). We directly reuse the long-context HotpotQA evaluation sets released by MemAgent (Yu et al., 2026)[4]. MemAgent synthesizes 128 questions from the HotpotQA validation split with the same packing recipe as training, then provides each question with contexts of $L \in \{50, 100, 200, 400, 800, 1600, 3200, 6400\}$ paragraphs (approximately 7K–896K tokens), reusing the same question indices across all length settings.

2WikiMultiHopQA (out-of-distribution). For out-of-distribution evaluation we use 2WikiMultiHopQA (Ho et al., 2020). Because ReMemR1 (Shi et al., 2026) does not release the constructed test files, we regenerate them with the authors' public data-processing script[5]. The script loads 2WikiMultiHopQA from FlashRAG (Jin et al., 2024), retains supporting-fact evidence for each question, pads the context with random distractor paragraphs to the same L grid as above, shuffles paragraphs and keeps 128 samples per length setting.

Table 8 summarizes the training and evaluation data. For each evaluation benchmark, the notation 128×8 indicates 128 shared questions evaluated under the eight paragraph-count settings listed in the last column.

E Baseline Implementation Details

E.1 Full-Context Answering

Full-context answering feeds the question together with the entire associated document into a single LLM call and asks the model to answer directly. We evaluate Qwen3.5 (Qwen, 2026) under this protocol (both thinking and non-thinking modes). For inputs within the model's native 262K-token window we use the default configuration; for longer documents we apply YaRN-based RoPE scaling (Peng et al., 2024) with a factor of 4.0, extending the effective context to about 1M tokens. Generation uses temperature 0. DeepSeek-V4-Pro (preview, 2026-04-24) (DeepSeek-AI, 2026) natively supports a 1M-token context window; we evaluate it under the same full-context input format with its reasoning effort set to `Max`.

Each sample is answered in two turns: the user prompt in the first turn provides the entire document and the question, and the models perform inference optionally with thinking; the second turn is a short follow-up that asks for a concise final answer only. We adopt this protocol because thinking models fail to strictly follow format instructions after finishing their reasoning more frequently than non-thinking models (e.g., wrapping the final answer in designated tags) (Li et al., 2025); the issue becomes more pronounced after lengthy deliberation over long documents. In a preliminary experiment that required placing the final answer inside `<answer>` and `</answer>`, DeepSeek-

[4] https://github.com/BytedTsinghua-SIA/MemAgent

[5] https://github.com/syr-cn/ReMemR1

V4-Pro under `think-max` violated the format instruction on 9.47% of in-distribution evaluation cases. We therefore separate reasoning from answer presentation in a two-turn format.

System prompt:

```
You are a helpful assistant that answers complex multi-hop questions
based on the provided reference documents.

Read the documents carefully and give an answer based on the documents.
```

User prompt for the first turn:

```
## Reference Document

{context}

## Question

{question}
```

User prompt for the second turn:

```
What is your final answer? Give only a concise answer. Do not include
reasoning or any other text.
```

E.2 Direct Corpus Interaction

We reimplement Direct Corpus Interaction (DCI; Li et al. 2026; Salemi et al. 2026; Sen et al. 2026) as a single-agent long-document QA baseline. Unlike other approaches that ingest document chunks into the context, DCI keeps the corpus outside the conversation and lets the model inspect it only through local shell tools. We mainly follow the implementation of DCI in Li et al. (2026)[6]. The agent is equipped with two native tools: (i) "`read`", which returns a line-numbered slice of a corpus file with a default window of 200 lines; and (ii) "`bash`", which executes a read-only shell command in the corpus directory (primarily `rg`, together with ordinary inspection utilities such as `ls`, `find`, `head`, and `wc`). Parallel tool calls within a single model turn are allowed. Bash execution is confined to a sandbox. Tool observations are truncated to at most $20{,}000$ characters (`read` keeps the head; `bash` keeps the tail). When the cumulative size of tool results exceeds $240{,}000$ characters, we apply the zero-LLM L3 history compaction of Li et al. (2026), replacing older tool results with a short placeholder while retaining the most recent 12 results.

Inference proceeds in a multi-turn ReAct-style loop with a maximum of 48 turns. We enable the model's thinking mode, set temperature to 0, and cap each generation at $8{,}192$ tokens; length-truncated turns receive a continuation observation and continue. When the model stops without tool calls, its response is taken as the final answer. If the turn budget is exhausted before a usable answer appears, we append a tool-free finalize prompt that asks the model to emit a concise answer from evidence already present in the trajectory. All DCI baselines are served with SGLang under the same Qwen3.5 backbone family as ParSer.

E.3 Agentic RAG

We implement Agentic RAG as a multi-turn retrieval baseline that uses the same Qwen3.5 backbone family (Qwen, 2026) and tool-calling interface as ParSer, but replaces the document-reading subagents with a dense retriever. The agent is given only the question at initialization; the corpus remains outside its context until it invokes `retrieve_documents` with a natural-language query. The tool returns the three highest-scoring chunks, after which the agent may issue another query or produce a final answer. This setup isolates the effect of agentic dense retrieval from both full-context prompting and LLM-based parallel document reading. Retrieval operates over fixed-size chunks rather than over the entire paragraph or individual sentences, consistent with other chunk-based approaches. It uses the same chunking strategy and training hyperparameters as for ParSer as described in §F.4, but uses a chunk size of $1{,}024$ tokens for inference.

[6] https://github.com/DCI-Agent/DCI-Agent-Lite

Qwen3-Embedding-4B (Zhang et al., 2025)[7] is used as the retriever. Let $\boldsymbol{e}_u$ denote the normalized query embedding and $\boldsymbol{e}_i$ the normalized embedding of chunk $\boldsymbol{d}_i$. We rank chunks by $s(u, \boldsymbol{d}_i) = \boldsymbol{e}_u^\top \boldsymbol{e}_i$, and return the top three. Each tool observation is a JSON object containing the original query and, for every retrieved chunk, its rank, similarity score, corpus index, and full text. The agent runs in ReAct style. At each turn it generates at temperature 0 with a maximum of $2{,}048$ new tokens. We allow at most 16 turns. The system prompt explicitly encourages query reformulation and decomposition when the first retrieved set is incomplete, which permits multi-hop evidence to be collected over multiple retrieval rounds.

System prompt:

```
You are an intelligent helper responsible for answering complex
multi-hop questions over a long document collection.  You can use the
'retrieve_documents' tool to retrieve the three chunks most relevant to
a search query.

# How to work

1.  Think about what evidence is needed, then call 'retrieve_documents'
with a clear, focused natural-language query.
2.  Read the retrieved chunks and decide whether they are sufficient.
Evidence for a multi-hop answer may require multiple retrieval calls
with different queries.
3.  If results are partial or irrelevant, simplify, rephrase, or
decompose the query instead of giving up.
4.  When you can answer the question, put only the concise final answer
inside <answer>your answer</answer>.

Do not claim that information is unavailable until you have tried
multiple focused retrieval strategies.
```

User prompt:

```
## Question

{question}
```

E.4 MemAgent and ReMemR1

We reproduce MemAgent (Yu et al., 2026) and ReMemR1 (Shi et al., 2026) from the authors' open-source repositories, adapting them only as needed to support Qwen3.5 (Qwen, 2026) by upgrading the underlying VERL (Sheng et al., 2025) dependency. We largely reuse the original training hyperparameters; the only intentional change is reducing the rollout group size from 16 to 8 (versus 5 for ParSer) to control training cost. Models are trained on 8 NVIDIA H100 GPUs for 250 steps, and a checkpoint is saved every 10 steps. Evaluation likewise follows the authors' released evaluation code. We refer readers to the original papers and code repositories for further implementation details.

F ParSer's Implementation Details

F.1 Lead Agent Prompt

The system prompt contains tool call function usage and QA task instructions. We use the default tool call function template of Qwen3.5, which the models readily follow.

System prompt:

```
# Tools

You have access to the following functions:

<tools>
{"type":  "function", "function":  {
  "name":  "query_agents",
```

[7] https://huggingface.co/Qwen/Qwen3-Embedding-4B

```
  "description": "Broadcast a query to all document agents in parallel.
Each agent reads only its chunk and may return evidence and a partial
answer. The tool response is a JSON string of aggregated findings.",
  "parameters": {
    "type": "object", "properties": {
      "query": {"type": "string", "description": "Question or search
query to send to every agent."}
    }, "required": ["query"]
  }
}
}
</tools>

If you choose to call a function ONLY reply in the following format
with NO suffix:

<tool_call>
<function=example_function_name>
<parameter=example_parameter_1>
value_1
</parameter>
<parameter=example_parameter_2>
This is the value for the second parameter that can span
multiple lines
</parameter>
</function>
</tool_call>

<IMPORTANT> Reminder:
- Function calls MUST follow the specified format: an
inner <function=...></function> block must be nested within
<tool_call></tool_call> XML tags.
- Required parameters MUST be specified.
- You may provide optional reasoning for your function call in natural
language BEFORE the function call, but NOT after.
- If there is no function call available, answer the question like
normal with your current knowledge and do not tell the user about
function calls.
</IMPORTANT>

You are an intelligent helper responsible for answering complex
multi-hop questions. Multiple agents exist, each with access to a
different chunk of a long document. The evidence you need may be
spread across chunks. You must use the query_agents tool to retrieve
information, then synthesize the final answer.

How to work (multi-turn)

1. After thinking, when you need evidence from the document, call the
tool query_agents with a clear natural-language query. The same query
is broadcast to every agent; each searches only its chunk.
2. Read the tool response (JSON summarizing agent findings) and decide
whether you need another query.
  - CRITICAL: DO NOT GIVE UP EASILY. If a query returns no results, or
only partial results, you MUST NOT immediately conclude "Information
not available."
  - Instead, you MUST simplify, rephrase, or break down your query into
smaller parts and call query_agents again.
  - You MUST make multiple different attempts to query the agents
before giving up. Only output the final answer when you are absolutely
certain no more information can be found after exhausting multiple
search strategies.
3. After thinking, when you can answer the question, write the final
answer only inside <answer>your answer</answer>.
```

User prompt:

Question
{question}

F.2 SUBAGENT PROMPT

User prompt:

```
You are a precise document analysis assistant. Your task is to answer
questions based ONLY on the information in your assigned document
chunk.

Core Principles
1. Strict Evidence-Based: Answer ONLY if the chunk contains explicit
information.
2. Verbatim Extraction: Evidence must be exact quotes, not
paraphrases.
3. Binary Output: Either provide JSON with evidence, or output
exactly Unknown.

Response Protocol

If Evidence Exists
```json
{"evidence": "Exact verbatim quote from the chunk that supports the
answer. Must be copy-paste accurate.", "answer": "Concise answer
based on the evidence"}
```

If No Evidence
Output exactly Unknown without any other text.

Examples

Example 1: Document Chunk: <A chunk of text that contains the
information about the invention of the telephone>
Question: Who invented the telephone?
Response:
```json
{"evidence": "In 1876, Alexander Graham Bell was awarded the first
US patent for the invention of the telephone.", "answer": "Alexander
Graham Bell"}
```

Example 2: Document Chunk: <A chunk of text that contains no
information about the invention of the telephone>
Question: Who invented the telephone?
Response: Unknown

Your Task

Document Chunk: {chunk_content}

Question: {question}

Analyze the chunk against the question. Output ONLY the required
format above, JSON or Unknown. Do NOT provide any explanations or
additional text.
```

F.3 OVERALL WORKFLOW

Algorithm 1 shows the overall workflow of PARSER. If a lead-agent generation contains neither a parseable `query_agents` tool call nor an `<answer>` block, the environment does not terminate the trajectory. Instead, it appends INVALIDACTIONHINT as the observation and continues the ReAct loop, prompting the lead agent to retry with a well-formed action. The hint text is:

INVALIDACTIONHINT:

```
Invalid step: start with a short Thinking section (what you know,
what you need, what you will do), then either call query_agents via
<tool_call>...</tool_call> with {"query": "..."}, or give the final
answer as <answer>your answer</answer>.
```

Algorithm 1 PARSER.

Require: Question $\boldsymbol{q}$, document chunks $\{\boldsymbol{d}_i\}_{i=1}^{T}$, maximum number of steps K
Ensure: Lead agent's reasoning trajectory, final answer

$\mathcal{H} \leftarrow [\textsc{SystemPrompt}, \textsc{UserPrompt}(\boldsymbol{q})]$ ▷ initialize the lead agent's context
for $t = 1$ to K **do**
  $(\boldsymbol{z}_t, \mathcal{A}_t) \leftarrow \textsc{LeadAgent}(\mathcal{H})$ ▷ $\boldsymbol{z}_t$: thinking content; $\mathcal{A}_t$: action (subagent calling/answering)
  if $\textsc{ExtractAnswer}(\mathcal{A}_t) \neq \varnothing$ **then**
    $\mathcal{H} \leftarrow \mathcal{H} \cup \{(\boldsymbol{z}_t, \mathcal{A}_t)\}$ ▷ append current turn's generation to $\mathcal{H}$
    return $(\mathcal{H}, \textsc{ExtractAnswer}(\mathcal{A}_t))$
  else if $\textsc{ExtractQueries}(\mathcal{A}_t) \neq \varnothing$ **then**
    parallel for $u \in \textsc{ExtractQueries}(\mathcal{A}_t)$ **do** ▷ queries in the same turn run concurrently
      parallel for $i = 1$ **to** T **do** ▷ Scatter: process all chunks concurrently
        $o_i \leftarrow \textsc{Subagent}(u, \boldsymbol{d}_i)$
      end parallel for
      $\mathcal{R}_u \leftarrow \textsc{Gather}\big(\{o_i\}_{i=1}^{T}\big)$ ▷ Gather: aggregate local findings
    end parallel for
    $\mathcal{H} \leftarrow \mathcal{H} \cup \{(\boldsymbol{z}_t, \mathcal{A}_t, \{\mathcal{R}_u\}_{u \in \textsc{ExtractQueries}(\mathcal{A}_t)})\}$ ▷ append current turn's generation and observations to $\mathcal{H}$
  else
    $\mathcal{H} \leftarrow \mathcal{H} \cup \{(\boldsymbol{z}_t, \mathcal{A}_t, \textsc{InvalidActionHint})\}$ ▷ observation flags an invalid generation
  end if
end for
return $(\mathcal{H}, \textsc{ExtractAnswer}(\mathcal{A}_t))$

F.4 TRAINING SETUP

Environment and deployment. We train PARSER with VERL's Megatron backend. We adopt the fully asynchronous RL training architecture,[8] which decouples policy updating and rollout onto separate GPUs and allows both stages to run continuously. Beyond higher training throughput, this setup yields an additional practical benefit in our training: subagents continue serving rollout requests while the lead agent policy is being updated, rather than remaining idle during the update. We use six NVIDIA H100 GPUs for the trainer: two GPUs for policy updates and four GPUs for SGLang-based lead agent rollouts. We deploy the subagents with a separate SGLang Model Gateway[9] cluster on 10 additional NVIDIA H100 GPUs.

We optimize KV cache reuse for faster subagent inference. According to §2.2, across all lead-agent turns, despite having different queries, each subagent is persistently assigned a fixed document chunk. Note that the subagent prompt is ordered as *fixed instructions* → *assigned document chunk* → *current query*. Consequently, when the lead agent issues a new query in a later turn, requests sent to the same chunk differ only in the query prompt and subsequent text; the instructions and the chunk, which constitute most of the prompt, remain an identical prefix. During the first such request, SGLang computes the prefix's KV states and stores them in its Radix Cache. For subsequent requests with that prefix, the server can retrieve the cached KV states and prefill only the new query suffix, avoiding repeated computation over the long chunk.

This reuse requires sending a request to an instance that already holds the relevant cached prefix. We therefore configure the SGLang Router with the `cache_aware` policy. For each incoming subagent request, the router compares its prompt prefix with the prefixes cached by its serving instances and preferentially routes the request to the instance with the longest match. The router thus preserves

[8] https://verl.readthedocs.io/en/latest/advance/fully_async.html
[9] https://docs.sglang.io/docs/advanced_features/sgl_model_gateway

cache locality across lead agent turns, while the Radix Cache performs the KV reuse within the selected instance. This combination reduces redundant chunk-prefill computation and accelerates subagent inference.

Training hyperparameters. Table 9 lists the training configuration. We train the lead agent with GRPO, use five rollouts per prompt, and set the learning rate to $1e-6$. The lead agent operates for at most 9 turns; each subagent reads a 512-token chunk and generates at most 512 tokens per query. Training for 180 steps takes about 312 hours for the 4B model and 400 hours for the 9B model.

Table 9: Training hyperparameters for PARSER.

Category	Hyperparameter	Value
Actor optimization	Optimizer	Adam
	Learning rate; warm-up steps; schedule	1×10^{-6}; 15; constant
	Adam (β_1, β_2); weight decay	$(0.9, 0.999)$; 0.01
	Gradient clipping	1.0
	PPO mini-batch size	128
	KL regularization	1×10^{-3}
	PPO clip range; entropy coefficient	$[0.2, 0.2]$; 0
Megatron backend	Precision	bfloat16
	Tensor / pipeline / context / expert parallelism	$1/1/1/1$
	Parameter / gradient / optimizer offload	enabled / enabled / enabled
Asynchronous training	Rollout engine; mode	SGLang; asynchronous
	Rollout GPUs; policy-update GPUs	4; 2
	Rollout GPU memory utilization	0.8
	Staleness threshold; parameter-sync interval	2; 4 steps
	Dynamic sampling	enabled
	Checkpoint interval	10 steps
Lead agent	Rollouts per prompt	5
	Sampling temperature; top-p; top-k	1.0; 1.0; -1
	Maximum turns	9
	Maximum generation per turn	2,048 tokens
Subagents	Temperature; output length	0.7; 512 tokens

Chunking. For HotpotQA and 2WikiMultiHopQA, we first recover paragraph boundaries by splitting each packed context at its original "`Document N:`" delimiters and prefix each resulting unit with its paragraph index. Adjacent paragraphs are then greedily packed, in their original order, into chunks of at most $chunk_size$ tokens according to the corresponding Qwen3.5 tokenizer. A paragraph that individually exceeds this limit is divided into contiguous token windows.

F.5 TRAINING DYNAMICS

Figure 6 summarizes the training dynamics of the 4B and 9B lead agents. Comparing the averages over the first and last ten steps, the training reward increases from 44.9% to 83.2% for the 4B model and from 49.8% to 83.3% for the 9B model. The two models reach similar rewards through different interaction dynamics. The mean number of turns increases from 4.65 to 5.04 for the 4B model, whereas it decreases from 4.85 to 4.12 for the 9B model. Meanwhile, the mean length of lead-agent responses grows from 1.36K to 2.40K tokens for 4B and from 1.47K to 2.48K tokens for 9B. For the 9B run, where the corresponding communication statistics were logged, subagent queries per turn increase steadily from 1.06 to 1.62. Notably, our reward neither penalizes the number of interaction turns nor explicitly encourages issuing multiple subagent queries in parallel. The increase therefore indicates an emergent strategy: the lead agent learns to identify queries without direct dependencies and place them in the same turn for parallel execution, rather than executing them sequentially across turns (see the example in Figures 7 and 9).

The lead agents also maintain reliable action formatting from the beginning of training: over the first ten steps, the tool-call format error ratio is only 0.22% for 4B and 0.66% for 9B, and it subsequently

approaches zero. This reliability is achieved without a format reward because we use the backbone's native multi-turn tool-calling format, which the backbone has already been trained to follow. Finally, the absolute log-perplexity difference between the rollout and actor policies remains on the order of 10^{-3} throughout training (at most 1.41×10^{-3} across both runs). Because rollout generation and policy optimization proceed concurrently, a trajectory may be generated by a rollout worker whose policy parameters lag behind the current actor by several updates. The consistently small discrepancy shows that this staleness causes only a minor shift in the token probabilities assigned to collected trajectories, thereby limiting the off-policy mismatch introduced by fully asynchronous training.

F.6 Inference

Inference follows the same process as training. We serve the lead agent with a local SGLang offline engine on the trained checkpoint, and deploy subagents following the training setup described above.

The lead agent runs in thinking mode at temperature 0 for up to 12 turns and generates at most $2{,}048$ tokens per turn. Subagents use temperature 0.7 and a 512-token generation budget. Relative to training, evaluation uses larger chunks ($4{,}096$ vs. 512 tokens) and a higher turn budget (12 vs. 9). We intentionally adopt a smaller chunk size during training because finer chunking reduces prefill cost: as given in Equation (2), the prefill complexity $\mathcal{O}(n^2/T)$ decreases as the number of chunks T grows, which speeds up training. Following MemAgent (Yu et al., 2026) and ReMemR1 (Shi et al., 2026), we report `Sub_EM`: after standard answer normalization (lowercasing, removing articles and punctuation), a prediction is counted as correct if either the prediction or a ground-truth answer is a substring of the other.[10]

G Case Study

G.1 Success Case

In the example in Figure 7, the lead agent decomposes the comparative question into two parallel `query_agents` broadcasts that execute simultaneously, gathers birth-date evidence from different subagents, and synthesizes the final answer in a second reasoning step. Figure 8 shows a two-hop director lookup. The lead agent first recovers that *I Want Someone to Eat Cheese With* was directed by Jeff Garlin, then queries his birth date. Figure 9 shows a comparative multi-hop success trajectory. The lead agent broadcasts parallel director queries for *Everything's Ducky* and *Karthika*, then parallel death-date queries, and selects the film whose director died earlier.

Figure 10 illustrates how the lead agent resolves conflicting subagent responses. In the first round, different subagents return François Coli alone, Charles Nungesser and François Coli together, or Jean Metzinger. The last answer is triggered by the similarly named painting *L'Oiseau bleu*, (as mentioned in the subagent's evidence), rather than the target aircraft *L'Oiseau Blanc*, and is therefore not adopted by the lead agent. Instead of committing to either of the remaining answers, the lead agent retains Nungesser and Coli as candidates and issues targeted parallel queries about both. The resulting evidence identifies Nungesser as the French ace pilot and adventurer and Coli as a pilot and navigator, allowing the lead agent to disambiguate the candidates and select Nungesser.

G.2 Failure Case

Figure 11 shows a failure on the question *"Who is the husband of Princess Elene of Georgia?"* Relevant findings identify Elene as the daughter of Heraclius II and the mother of Solomon II, whose parents were Prince Archil of Imereti and Helene (an alternative name for Elene), supporting the correct answer *Prince Archil of Imereti*. However, a subagent reading an unrelated chunk about Grand Duchess Elena Vladimirovna returns *"Her husband was Prince Nicholas of Greece and Denmark."* It conflates the similarly named women, and the lead agent accepts this apparently direct answer despite the identity-grounded contrary evidence.

[10] Strictly speaking, `Sub_EM` should be a unidirectional match in which the ground-truth answer is a substring of the model prediction; MemAgent's paper also describes the metric in this sense. However, the open-source evaluation code of MemAgent and ReMemR1 implements a bidirectional variant. We follow the latter definition.

This example exposes a failure caused by *context isolation between the lead agent and subagents*. A subagent sees only the query and its local chunk, not the reasoning trajectory, so an ambiguous query can cause an entity mismatch. Meanwhile, the lead agent lacks the source context needed to verify the finding and may over-trust it. Such failures occur when the lead agent does not issue a more specific follow-up query or cross-validation fails to override the misleading finding. This behavior is occasional: in other cases, the lead agent issues additional queries that are more specific and reconciles evidence from different subagents, like in Figure 10.

G.3 Comparison against MemAgent on a Reverse-evidence Case

Figure 5 shows a question–document sample from the reverse-evidence setting. The question is a comparative two-hop query: identify the directors of *Everything's Ducky* and *Karthika*, then select the film whose director died earlier. The four supporting paragraphs are embedded in a 6,400-paragraph document. Director death dates appear first (paragraphs 483 and 3911), while the film–director mappings appear later (paragraphs 5527 and 5631), i.e., in reverse logical order.

Figure 12 shows MemAgent's sequential memory updates on this sample. When the agent encounters paragraph 483 (M. Krishnan Nair, died 2001) and later paragraph 3911 (Don Taylor, died 1998), it does not write either person's identity or death date into memory. At those steps the two names have not yet been linked to the films in the question and are not written into the memory. Instead, the memory keeps recording directors of other films mentioned in the incoming chunks. Only after paragraphs 5527 and 5631 does the memory record that *Karthika* was directed by M. Krishnan Nair and *Everything's Ducky* by Don Taylor. By then the death dates have already been dropped. The final memory therefore contains the two director names without their death dates, and MemAgent cannot complete the comparison. This trajectory shows that MemAgent's reasoning is heavily driven by the document: the order and surface content of incoming chunks determine what is written into memory. By contrast, Figure 9 shows ParSer on the same question. ParSer reasons from the question, issuing multi-round queries that execute in parallel over the entire document, and is therefore completely insensitive to the order of evidence.

Question

Which film has the director who died earlier, Everything's Ducky or Karthika (Film)?

Document

Only evidence paragraphs are shown; the full document contains 6,400 paragraphs.

...

Paragraph 483 *M. Krishnan Nair (director)*
M. Krishnan Nair (2 November 1926 – 10 May 2001 in Thiruvananthapuram) was an Indian film director of Malayalam films. He directed over 100 films. He also directed 18 Tamil movies including four films starring M. G. Ramachandran and two Telugu movies...

...

Paragraph 3911 *Don Taylor (American actor and director)*
Donald Richie Taylor (December 13, 1920 – December 29, 1998) was an American actor and film director. He co-starred in 1940s and 1950s classics, including the 1948 film noir "The Naked City", "Battleground", "Father of the Bride", "Father's Little Dividend" and "Stalag 17". He later turned to directing films such as "Escape from the Planet of the Apes" (1971), "Tom Sawyer" (1973).

...

Paragraph 5527 *Karthika (film)*
Karthika is a 1968 Indian Malayalam film, directed by M. Krishnan Nair and produced by V. M. Sreenivasan and A. R. Divakar. The film stars Sathyan, Sharada, Adoor Bhasi, Ummar and Manavalan Joseph in the lead roles. The film had musical score by M. S. Baburaj.

...

Paragraph 5631 *Everything's Ducky*
Everything's Ducky is a 1961 comedy film directed by Don Taylor and written by Benedict Freedman and John Fenton Murray. The film stars Mickey Rooney, Buddy Hackett, Jackie Cooper, Joanie Sommers, Roland Winters and Elizabeth MacRae. The film was released on December 20, 1961, by Columbia Pictures.

... (a total of 6,400 paragraphs)

Figure 5: Reverse-evidence question–document sample constructed from 2WikiMultiHopQA: death-date paragraphs (483, 3911) precede the film–director mappings (5527, 5631) in a 6,400-paragraph document. The correct answer is *Everything's Ducky*.

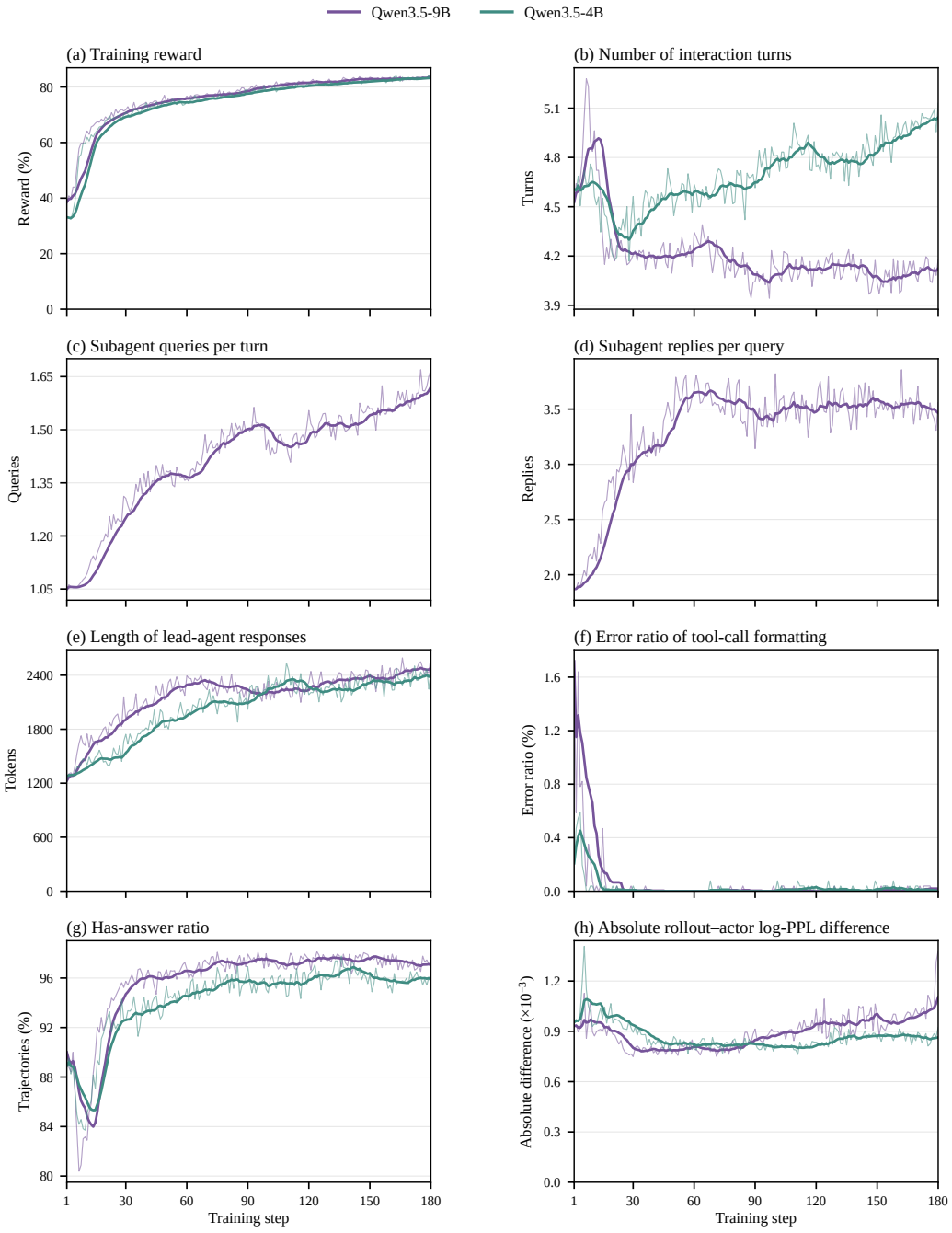

Figure 6: Training dynamics of the Qwen3.5-4B and Qwen3.5-9B lead agents over 180 reinforcement-learning steps. Thin curves show per-step measurements, while thick curves show 10-step moving averages. Logging for tool calls per turn and subagent replies per query was introduced after the 4B run had completed; consequently, these metrics are available only for the 9B run.

Question

User:
Who was born earlier, Emma Bull or Virginia Woolf?

Step 1

Assistant:
<think> I need to find information about the birth dates of Emma Bull and Virginia Woolf to determine who was born earlier. Let me start by querying the agents to find information about both individuals. </think>

```
{"name": "query_agents", "arguments": {"query": "When was Emma Bull born?"}}
```

```
{"name": "query_agents", "arguments": {"query": "When was Virginia Woolf born?"}}
```

Observations:

```
{"agent_129": {"evidence": "Emma Bull (born December 13, 1954) is an American science fiction and fantasy author.",
 "answer": "December 13, 1954"}}
```

```
{"agent_92": {"evidence": "Adeline Virginia Woolf (née Stephen; 25 January 1882 – 28 March 1941) was an English writer...",
 "answer": "25 January 1882"}}
```

Step 2

Assistant:
<think> Great! I found the birth dates:
– Emma Bull: December 13, 1954
– Virginia Woolf: 25 January 1882
Comparing these dates, Virginia Woolf was born in 1882 and Emma Bull was born in 1954. Therefore, Virginia Woolf was born earlier than Emma Bull.
The answer is Virginia Woolf. </think>
Based on the information retrieved:
– Virginia Woolf was born on 25 January 1882
– Emma Bull was born on December 13, 1954
Virginia Woolf was born earlier than Emma Bull by approximately 72 years.
<answer>Adeline Virginia Woolf</answer>

Figure 7: Success case 1 (PARSER-9B; HotpotQA, 6400-paragraph): the lead agent broadcasts two birth-date queries that execute simultaneously, gathers subagent findings, and synthesizes a comparative answer.

Question

User:
When was the director of film I Want Someone To Eat Cheese With born?

Step 1

Assistant:
<think> I need to find information about the film "I Want Someone To Eat Cheese With" and then find out when its director was born. Let me start by querying the agents to find information about this film. </think>

```
{"name": "query_agents", "arguments": {"query": "film I Want Someone To Eat Cheese With director"}}
```

Observations:

```
{"agent_99": {"evidence": "I Want Someone to Eat Cheese With is a 2006 American independent romantic comedy film
 written, produced, directed by, and...", "answer": "Jeff Garlin"}}
```

Step 2

Assistant:
<think> Great! I found that the film "I Want Someone To Eat Cheese With" was directed by Jeff Garlin. Now I need to find out when Jeff Garlin was born. Let me query the agents for this information. </think>

```
{"name": "query_agents", "arguments": {"query": "Jeff Garlin birth date"}}
```

Observations:

```
{"agent_130": {"evidence": "Jeffrey Todd Garlin (born June 5, 1962) is an American comedian and actor.", "answer": "June 5,
 1962"}}
```

Step 3

Assistant:

<think> Perfect! I now have all the information I need:
– The film "I Want Someone To Eat Cheese With" was directed by Jeff Garlin
– Jeff Garlin was born on June 5, 1962
So the answer to the question "When was the director of film I Want Someone To Eat Cheese With born?" is June 5, 1962. </think>
Based on my research, I found that:
1. The film "I Want Someone To Eat Cheese With" (2006) was written, produced, directed by, and starring Jeff Garlin 2. Jeff Garlin (full name: Jeffrey Todd Garlin) was born on June 5, 1962
<answer>June 5, 1962</answer>

Figure 8: Success case 2 (PARSER-9B; 2WikiMultiHopQA, 6400-paragraph): the lead agent first retrieves the film's director, then queries the birth date and synthesizes the answer.

Question

User:

Which film has the director who died earlier, Everything's Ducky or Karthika (Film)?

Step 1

Assistant:

<think> This is a multi-hop question that requires me to find information about two films: "Everything's Ducky" and "Karthika". I need to find out who directed each film, and then determine which director died earlier.
Let me start by querying the agents to find information about these films and their directors. </think>

```
{"name": "query_agents", "arguments": {"query": "Everything's Ducky film director"}}
```

```
{"name": "query_agents", "arguments": {"query": "Karthika film director"}}
```

Observations:

```
{"agent_15": {"evidence": "Everything's Ducky is a 1961 comedy film directed by Don Taylor", "answer": "Don Taylor"}}
```

```
{"agent_121": {"evidence": "Karthika is a 1968 Indian Malayalam film, directed by M. Krishnan Nair", "answer": "M. Krishnan Nair"}}
```

Step 2

Assistant:

<think> Great! Now I have the information that:
– "Everything's Ducky" is directed by Don Taylor
– "Karthika (Film)" is directed by M. Krishnan Nair
Now I need to find out which of these two directors died earlier. Let me query the agents for information about their death dates.
</think>

```
{"name": "query_agents", "arguments": {"query": "Don Taylor director death date"}}
```

```
{"name": "query_agents", "arguments": {"query": "M. Krishnan Nair death date"}}
```

Observations:

```
{"agent_172": {"evidence": "Donald Richie Taylor (December 13, 1920 – December 29, 1998) was an American actor and film director.", "answer": "December 29, 1998"}}
```

```
{"agent_175": {"evidence": "M. Krishnan Nair (2 November 1926 – 10 May 2001 in Thiruvananthapuram) was an Indian film director of Malayalam films.", "answer": "10 May 2001"}}
```

Step 3

Assistant:

<think> Perfect! Now I have the death dates for both directors:
– Don Taylor: December 29, 1998
– M. Krishnan Nair: 10 May 2001
Don Taylor died earlier (December 29, 1998) than M. Krishnan Nair (10 May 2001). Therefore, the film "Everything's Ducky" has the director who died earlier. </think>
<answer>Everything's Ducky</answer>

Figure 9: Success case 3 (PARSER-4B; 2WikiMultiHopQA, 6400-paragraph): the lead agent broadcasts parallel director queries, then parallel death-date queries, and selects the film whose director died earlier.

Question

User:
Which French ace pilot and adventurer fly L'Oiseau Blanc?

Step 1

Assistant:
<think> I need to find information about a French ace pilot and adventurer who flew "L'Oiseau Blanc". This is a multi-hop question that requires me to:
1. First, find information about L'Oiseau Blanc and who flew it 2. Then identify if that person is a French ace pilot and adventurer
Let me start by querying the agents to find information about L'Oiseau Blanc. </think>

{"name": "query_agents", "arguments": {"query": "L'Oiseau Blanc who flew it"}}

Observations:

{"agent_20": {"evidence": "François Coli (June 5, 1881 – presumably on or after May 8, 1927) was a French pilot and navigator best known as the one-eyed...", "answer": "François Coli"}}
{"agent_111": {"evidence": "L'Oiseau bleu (also known as The Blue Bird and Der Blaue Vogel) is a large oil painting created in 1912–1913 by the French...", "answer": "Jean Metzinger"}}
{"agent_146": {"evidence": "Only two examples of the type were built, with the first PL.8-01 named "L'Oiseau Blanc" (The White Bird), that gained fame as...", "answer": "Charles Nungesser and François Coli"}}
{"agent_167": {"evidence": "The aircraft was flown by French World War I aviation heroes, Charles Nungesser and François Coli.", "answer": "Charles Nungesser and François Coli"}}

Step 2

Assistant:
<think> Good, I found that L'Oiseau Blanc was flown by Charles Nungesser and François Coli. Now I need to find out which of these two is described as a "French ace pilot and adventurer". Let me search for information about both of them to see which one fits this description.
Let me search for information about Charles Nungesser first. </think>

{"name": "query_agents", "arguments": {"query": "Charles Nungesser French ace pilot adventurer"}}

{"name": "query_agents", "arguments": {"query": "François Coli French ace pilot adventurer"}}

Observations:

{"agent_20": {"evidence": "François Coli (June 5, 1881 – presumably on or after May 8, 1927) was a French pilot and navigator best known as the one-eyed...", "answer": "François Coli"}}
{"agent_80": {"evidence": "On the Paris side of the Atlantic their competitors were another World War One French ace, Charles Nungesser, and his navigator...", "answer": "The document identifies Charles Nungesser as a World War One French ace who was a competitor for the Orteig Prize alongside Charles Lindbergh in an attempt to fly the Atlantic Ocean."}}
{"agent_113": {"evidence": "Charles Eugène Jules Marie Nungesser, MC (15 March 1892 – presumably on or after 8 May 1927) was a French ace pilot and...", "answer": "Charles Eugène Jules Marie Nungesser, MC"}}
{"agent_146": {"evidence": "Levasseur built the aircraft in 1927, specifically for pilots Charles Nungesser and François Coli for a transatlantic attempt to...", "answer": "Charles Nungesser was a French pilot who co-piloted a Levasseur PL.8 aircraft for a transatlantic attempt to win the Orteig Prize."}}
{"agent_167": {"evidence": "L'Oiseau Blanc (commonly known in the English-speaking world as The White Bird) was a French Levasseur PL.8 biplane that...", "answer": "Charles Nungesser was a French World War I aviation hero who flew the aircraft L'Oiseau Blanc in an attempt to make the first non-stop transatlantic flight."}}

{"agent_20": {"evidence": "François Coli (June 5, 1881 – presumably on or after May 8, 1927) was a French pilot and navigator best known as the one-eyed...", "answer": "François Coli was a French pilot and navigator, best known as the flying partner of Charles Nungesser in their attempt to fly the Atlantic Ocean."}}
{"agent_80": {"evidence": "On the Paris side of the Atlantic their competitors were another World War One French ace, Charles Nungesser, and his navigator...", "answer": "No, François Coli was a navigator, not an ace pilot adventurer."}}
{"agent_146": {"evidence": "Levasseur built the aircraft in 1927, specifically for pilots Charles Nungesser and François Coli for a transatlantic attempt to...", "answer": "François Coli was a French pilot who flew a Levasseur PL.8 aircraft in a transatlantic attempt to win the Orteig Prize."}}
{"agent_167": {"evidence": "The aircraft was flown by French World War I aviation heroes, Charles Nungesser and François Coli.", "answer": "François Coli was a French World War I aviation hero."}}

Step 3

Assistant:
<think> Perfect! I found the answer. According to agent_113, "Charles Eugène Jules Marie Nungesser, MC (15 March 1892 – presumably on or after 8 May 1927) was a French ace pilot and adventurer, best remembered as a rival of Charles Lindbergh."
This matches exactly with the question asking for "Which French ace pilot and adventurer fly L'Oiseau Blanc".
So the answer is Charles Nungesser (specifically Charles Eugène Jules Marie Nungesser). </think>
Based on my research, I found that Charles Nungesser (Charles Eugène Jules Marie Nungesser) was the French ace pilot and adventurer who flew L'Oiseau Blanc. According to the evidence, Charles Nungesser was "a French ace pilot and adventurer, best remembered as a rival of Charles Lindbergh." He flew the L'Oiseau Blanc aircraft along with his navigator François Coli in their attempt to make the first non-stop transatlantic flight from Paris to New York City in 1927.
<answer>Charles Eugène Jules Marie Nungesser</answer>

Figure 10: Success case 4 (PARSER-9B; HotpotQA, 6400-paragraph).

Question

User:
Who is the husband of Princess Elene Of Georgia?

Step 1

Assistant:
<think> This is a multi-hop question that requires me to find information about Princess Elene Of Georgia and her husband. I need to query the agents to find information about Princess Elene Of Georgia and then find information about her husband.
Let me start by querying the agents with a question about Princess Elene Of Georgia. </think>

```
{"name": "query_agents", "arguments": {"query": "Princess Elene Of Georgia"}}
```

Observations:

```
{"agent_48": {"evidence": "Elene (1753 – 17 June 1786) was a Georgian princess royal ("batonishvili"), a daughter of Heraclius II, King of Kartli and Kakheti. She was the mother of Solomon II of Imereti, the last king to have reigned in the Georgian polities.", "answer": "Princess Elene of Georgia was a Georgian princess royal, daughter of Heraclius II, and mother of Solomon II of Imereti."}}
```

Step 2

Assistant:
<think> I found information about Princess Elene of Georgia, but this doesn't mention her husband. The information shows she was a daughter of Heraclius II and mother of Solomon II of Imereti, but doesn't mention who her husband was.
I need to query again to find information about her husband. Let me try a different query focused on her husband or spouse. </think>

```
{"name": "query_agents", "arguments": {"query": "husband of Princess Elene Of Georgia"}}
```

Observations:

```
{"agent_132": {"evidence": "Her husband was Prince Nicholas of Greece and Denmark", "answer": "Prince Nicholas of Greece and Denmark"}}
{"agent_152": {"evidence": "Solomon II (1772 – February 7, 1815), of the Bagrationi Dynasty, was the last King of Imereti (western Georgia) from 1789 to 1790 and from 1792 until his deposition by the Imperial Russian government in 1810. He was born as David to Prince Archil of Imereti, brother of King Solomon I of Imereti, by his wife Helene, daughter of King Heraclius II of Georgia.", "answer": "Solomon II"}}
```

Step 3

Assistant:
<think> Great! I found the answer. The second query returned information from agent_132 that states "Her husband was Prince Nicholas of Greece and Denmark". This directly answers the question "Who is the husband of Princess Elene Of Georgia?"
The answer is Prince Nicholas of Greece and Denmark. </think>
<answer>Prince Nicholas of Greece and Denmark</answer>

Figure 11: Failure case (PARSER-4B; 2WikiMultiHopQA, 6400-paragraph): a misleading local finding (`agent_132`) overrides identity-grounded evidence (`agent_152`) that supports the correct husband, Prince Archil of Imereti.

MemAgent:

...

Step 12 (after Paragraph 483)

Memory:

– **Problem:** Which film has the director who died earlier, Everything's Ducky or Karthika (Film)?
– Paragraph 11 states that Kim Jee-woon (born July 6, 1964) is a South Korean film director and screenwriter.
– Paragraph 7 states that the film *Dharmakshetre Kurukshetre* is a 1975 Indian Malayalam film, directed and produced by Kunchacko.
– Paragraph 14 states that Wallace Fox (March 9, 1895 – June 30, 1958) was an American film director. He directed 84 films between 1927 and 1953. He was born in Purcell, Oklahoma, and died in Hollywood, California.
– Paragraph 87 states that Anna of Trebizond is a film directed by a character who died in 1342.
– Paragraph 88 states that the film *A Woman of the Sea* is a 1926 film directed by Josef von Sternberg.
– Paragraph 90 states that Tom DiCillo (Thomas A. "Tom" DiCillo) is an American film director, screenwriter and cinematographer.
– Paragraph 91 states that Annemarie Sörensen (19 December 1913 – December 1993) was a German singer and film actress. She played the female lead in the 1934 comedy "Heinz in the Moon". She was a Protestant, but due to what was described as her uncertain racial origin she faced pressure following the Nazi takeover. In 1935 she emigrated from Germany and eventually went to live in Britain where she lived for the remainder of her life, although her attempt to break into British films failed.
– Paragraph 92 states that the film *The Courtship of Andy Hardy* is a 1942 film, part of the Andy Hardy series. It gave an early role to Donna Reed although Mickey Rooney had lobbied for his then-wife Ava Gardner to have her part. Within a few months of the film's release, she filed for divorce. This was the 12th entry in the long-running "Andy Hardy" series.
Paragraph 483 (M. Krishnan Nair, died 2001) is not written into memory.

...

Step 98 (after Paragraph 3911)

Memory:

– **Problem:** Which film has the director who died earlier, Everything's Ducky or Karthika (Film)?

– Paragraph 11 states that Kim Jee-woon (born July 6, 1964) is a South Korean film director and screenwriter.

– Paragraph 7 states that the film *Dharmakshetre Kurukshetre* is a 1975 Indian Malayalam film, directed and produced by Kunchacko.

– Paragraph 1977 states that Louis Marie Malle (30 October 1932 – 23 November 1995) was a French film director, screenwriter and producer. His film "Le Monde du silence" won the Palme d'Or in 1956 and the Academy Award for Best Documentary in 1957, although he was not credited at the ceremony; the award was instead presented to the film's co-director Jacques Cousteau. Later in his career he was nominated multiple times for Academy Awards. Malle is also one of only four directors to have won the Golden Lion twice. Malle worked in both French cinema and Hollywood, and he produced both French and English language films. His most famous films include the crime film "Ascenseur pour l'échafaud" (1958), the World War II drama "Lacombe, Lucien" (1974), the romantic crime film "Atlantic City" (1980), the comedy-drama "My Dinner with Andre" (1981), and the autobiographical film "Au revoir les enfants" (1987).

– Paragraph 1978 states that Claude Autant-Lara (5 August 1901 – 5 February 2000) was a French film director and later Member of the European Parliament (MEP).

– Paragraph 1979 states that Terror Train is a 1980 Canadian slasher film directed by Roger Spottiswoode in his directorial debut and starring Jamie Lee Curtis, Ben Johnson, and Hart Bochner. Set aboard a moving train on New Year's Eve, the film follows a group of medical school students holding a costume party who are targeted by a killer who steals their costumes after murdering them to avoid being caught. It features supporting performances from Sandee Currie, Anthony Sherwood, and David Copperfield. The concept for the film was based on an idea by executive producer Daniel Grodnik, who sought to "make Halloween on a train." A full-length script for the film was composed by T. Y. Drake, and production was initiated within four months. The film was shot in Montreal between late November and late December.

– Paragraph 2311 states that Russ Meyer (Russell Albion Meyer) was an American film director, producer, screenwriter, cinematographer, film editor, actor, and photographer. Meyer is known primarily for writing and directing a series of successful sexploitation films that featured campy humor, sly satire and large-breasted women, such as "Faster, Pussycat! Kill! Kill!" He was born March 21, 1922, and died September 18, 2004.

– Paragraph 2317 states that Helmut Käutner (born 25 March 1908 in Düsseldorf, Germany; died 20 April 1980 in Castellina in Chianti, Italy) was a German film director active mainly in the 1940s and 1950s.

– Paragraph 2765 states that the film *The Man with a Cloak* is a 1951 American film noir crime-thriller-drama directed by Fletcher Markle and starring Joseph Cotten, Barbara Stanwy.

– Paragraph 3591 states that the film *The Devil's Hairpin* is a 1957 feature film about car racing, filmed in Technicolor and VistaVision, written and directed by Cornel Wilde, who also stars.

– Paragraph 3605 states that the film *Sunshine Sally* is a 1922 Australian silent film directed by Lawson Harris set in the Sydney suburb of Woolloomooloo. Most of the movie survives today.

– Paragraph 3706 states that the film *Toen 't licht verdween* is a 1918 Dutch silent drama film directed by Maurits Binger.

Paragraph 3911 (Don Taylor, died 1998) is not written into memory. The memory continues to store directors of other films.

...

Step 140 (after Paragraph 5527)

Memory:

– **Problem:** Which film has the director who died earlier, Everything's Ducky or Karthika (Film)?

– Paragraph 5527 states that the film *Karthika* is a 1968 Indian Malayalam film, directed by **M. Krishnan Nair** and produced by V. M. Sreenivasan and A. R. Divakar. The film stars Sathyan, Sharada, Adoor Bhasi, Ummar and Manavalan Joseph in the lead roles. The film had musical score by M. S. Baburaj.

– Paragraph 11 states that Kim Jee-woon (born July 6, 1964) is a South Korean film director and screenwriter.

– Paragraph 7 states that the film *Dharmakshetre Kurukshetre* is a 1975 Indian Malayalam film, directed and produced by Kunchacko.

– Paragraph 1977 states that Louis Marie Malle (30 October 1932 – 23 November 1995) was a French film director, screenwriter and producer. His film "Le Monde du silence" won the Palme d'Or in 1956 and the Academy Award for Best Documentary in 1957, although he was not credited at the ceremony; the award was instead presented to the film's co-director Jacques Cousteau. Later in his career he was nominated multiple times for Academy Awards. Malle is also one of only four directors to have won the Golden Lion twice...

– Paragraph 1978 states that Claude Autant-Lara (5 August 1901 – 5 February 2000) was a French film director and later Member of the European Parliament (MEP).

– Paragraph 1979 states that Terror Train is a 1980 Canadian slasher film directed by Roger Spottiswoode in his directorial debut and starring Jamie Lee Curtis, Ben Johnson, and Hart Bochner. Set aboard a moving train on New Year's Eve, the film follows a group of medical school students holding a costume party who are targeted by a killer who steals their costumes after murdering them to avoid being caught. It features supporting performances from Sandee Currie, Anthony Sherwood, and David Copperfield...

– Paragraph 2311 states that Russ Meyer (Russell Albion Meyer) was an American film director, producer, screenwriter, cinematographer, film editor, actor, and photographer. Meyer is known primarily for writing and directing a series of successful sexploitation films that featured campy humor, sly satire and large-breasted women, such as "Faster, Pussycat! Kill! Kill!" He was born March 21, 1922, and died September 18, 2004.

– Paragraph 2317 states that Helmut Käutner (born 25 March 1908 in Düsseldorf, Germany; died 20 April 1980 in Castellina in Chianti, Italy) was a German film director active mainly in the 1940s and 1950s.

– Paragraph 2765 states that the film *The Man with a Cloak* is a 1951 American film noir crime-thriller-drama directed by Fletcher Markle and starring Joseph Cotten, Barbara Stanwy.

– Paragraph 3591 states that the film *The Devil's Hairpin* is a 1957 feature film about car racing, filmed in Technicolor and VistaVision, written and directed by Cornel Wilde, who also stars.

The film–director mapping is stored, but M. Krishnan Nair's death date is still absent.

...

Step 142 (after Paragraph 5631)

Memory:

– **Problem:** Which film has the director who died earlier, Everything's Ducky or Karthika (Film)?

– Paragraph 5631 states that the film *Everything's Ducky* is a 1961 comedy film directed by **Don Taylor** and written by Benedict Freedman and John Fenton Murray. The film stars Mickey Rooney, Buddy Hackett, Jackie Cooper, Joanie Sommers, Roland Winters and Elizabeth MacRae. The film was released on December 20, 1961, by Columbia Pictures.

– Paragraph 5527 states that the film *Karthika* is a 1968 Indian Malayalam film, directed by **M. Krishnan Nair** and produced by V. M. Sreenivasan and A. R. Divakar. The film stars Sathyan, Sharada, Adoor Bhasi, Ummar and Manavalan Joseph in the lead roles. The film had musical score by M. S. Baburaj.

– Paragraph 11 states that Kim Jee-woon (born July 6, 1964) is a South Korean film director and screenwriter.

– Paragraph 7 states that the film *Dharmakshetre Kurukshetre* is a 1975 Indian Malayalam film, directed and produced by Kunchacko.

– Paragraph 1977 states that Louis Marie Malle (30 October 1932 – 23 November 1995) was a French film director, screenwriter and producer. His film "Le Monde du silence" won the Palme d'Or in 1956 and the Academy Award for Best Documentary in 1957, although he was not credited at the ceremony; the award was instead presented to the film's co-director Jacques Cousteau. Later in his career he was nominated multiple times for Academy Awards. Malle is also one of only four directors to have won the Golden Lion twice. Malle worked in both French cinema and Hollywood, and he produced both French and English language films. His most famous films include the crime film "Ascenseur pour l'échafaud" (1958), the World War II drama "Lacombe, Lucien" (1974), the romantic crime film "Atlantic City" (1980), the comedy-drama "My Dinner with Andre" (1981), and the autobiographical film "Au revoir les enfants" (1987).

– Paragraph 1978 states that Claude Autant-Lara (5 August 1901 – 5 February 2000) was a French film director and later Member of the European Parliament (MEP).

– Paragraph 1979 states that Terror Train is a 1980 Canadian slasher film directed by Roger Spottiswoode in his directorial debut and starring Jamie Lee Curtis, Ben Johnson, and Hart Bochner. Set aboard a moving train on New Year's Eve, the film follows a group of medical school students holding a costume party who are targeted by a killer who steals their costumes after murdering them to avoid being caught. It features supporting performances from Sandee Currie, Anthony Sherwood, and David Copperfield. The concept for the film was based on an idea by executive producer Daniel Grodnik, who sought to "make Halloween on a train." A full-length script for the film was composed by T. Y. Drake, and production was initiated within four months. The film was shot in Montreal between late November and late December.

– Paragraph 2311 states that Russ Meyer (Russell Albion Meyer) was an American film director, producer, screenwriter, cinematographer, film editor, actor, and photographer. Meyer is known primarily for writing and directing a series of successful sexploitation films that featured campy humor, sly satire and large-breasted women, such as "Faster, Pussycat! Kill! Kill!" He was born March 21, 1922, and died September 18, 2004.

– Paragraph 2317 states that Helmut Käutner (born 25 March 1908 in Düsseldorf, Germany; died 20 April 1980 in Castellina in Chianti, Italy) was a German film director active mainly in the 1940s and 1950s.

Both target director names are stored, but neither death date is present.

. . .

Step 161 (after all paragraphs)

Memory:

– **Problem:** Which film has the director who died earlier, Everything's Ducky or Karthika (Film)?

– Paragraph 5631 states that the film *Everything's Ducky* is a 1961 comedy film directed by **Don Taylor** and written by Benedict Freedman and John Fenton Murray. The film stars Mickey Rooney, Buddy Hackett, Jackie Cooper, Joanie Sommers, Roland Winters and Elizabeth MacRae. The film was released on December 20, 1961, by Columbia Pictures.

– Paragraph 5527 states that the film *Karthika* is a 1968 Indian Malayalam film, directed by **M. Krishnan Nair** and produced by V. M. Sreenivasan and A. R. Divakar. The film stars Sathyan, Sharada, Adoor Bhasi, Ummar and Manavalan Joseph in the lead roles. The film had musical score by M. S. Baburaj.

– Paragraph 1977 states that Louis Marie Malle (30 October 1932 – 23 November 1995) was a French film director, screenwriter and producer. His film "Le Monde du silence" won the Palme d'Or in 1956 and the Academy Award for Best Documentary in 1957, although he was not credited at the ceremony; the award was instead presented to the film's co-director Jacques Cousteau. Later in his career he was nominated multiple times for Academy Awards. Malle is also one of only four directors to have won the Golden Lion twice. Malle worked in both French cinema and Hollywood, and he produced both French and English language films. His most famous films include the crime film "Ascenseur pour l'échafaud" (1958), the World War II drama "Lacombe, Lucien" (1974), the romantic crime film "Atlantic City" (1980), the comedy-drama "My Dinner with Andre" (1981), and the autobiographical film "Au revoir les enfants" (1987).

– Paragraph 1978 states that Claude Autant-Lara (5 August 1901 – 5 February 2000) was a French film director and later Member of the European Parliament (MEP).

– Paragraph 1979 states that Terror Train is a 1980 Canadian slasher film directed by Roger Spottiswoode in his directorial debut and starring Jamie Lee Curtis, Ben Johnson, and Hart Bochner. Set aboard a moving train on New Year's Eve, the film follows a group of medical school students holding a costume party who are targeted by a killer who steals their costumes after murdering them to avoid being caught. It features supporting performances from Sandee Currie, Anthony Sherwood, and David Copperfield. The concept for the film was based on an idea by executive producer Daniel Grodnik, who sought to "make Halloween on a train." A full-length script for the film was composed by T. Y. Drake, and production was initiated within four months. The film was shot in Montreal between late November and late December.

– Paragraph 2311 states that Russ Meyer (Russell Albion Meyer) was an American film director, producer, screenwriter, cinematographer, film editor, actor, and photographer. Meyer is known primarily for writing and directing a series of successful sexploitation films that featured campy humor, sly satire and large-breasted women, such as "Faster, Pussycat! Kill! Kill!" He was born March 21, 1922, and died September 18, 2004.

– Paragraph 2317 states that Helmut Käutner (born 25 March 1908 in Düsseldorf, Germany; died 20 April 1980 in Castellina in Chianti, Italy) was a German film director active mainly in the 1940s and 1950s.

– Paragraph 6215 states that Rajkumar Kohli (born 14 September 1930 in Lahore, British India) is an Indian film director. He was famous for directing several popular Bollywood films in the 1970s like Lootera starring Dara Singh and Nishi, who married Rajkumar Koli and 1980s with ensemble cast such as "Nagin" (1976).

The final memory still has only the two director names, not their death dates.

Final answer

Based on the provided memory, we can compare the directors of the films *Everything's Ducky* and *Karthika*:

1. The film *Everything's Ducky* (1961) is directed by **Don Taylor**.
2. The film *Karthika* (1968) is directed by **M. Krishnan Nair**.
3. The memory explicitly states that **Louis Marie Malle** (30 October 1932 – 23 November 1995) was a French film director, screenwriter and producer. His film "Le Monde du silence" won the Palme d'Or in 1956 and the Academy Award for Best Documentary in 1957, although he was not credited at the ceremony; the award was instead presented to the film's co-director Jacques Cousteau. Later in his career he was nominated multiple times for Academy Awards. Malle is also one of only four directors to have won the Golden Lion twice. Malle worked in both French cinema and Hollywood, and he produced both French and English language films. His most famous films include the crime film "Ascenseur pour l'échafaud" (1958), the World War II drama "Lacombe, Lucien" (1974), the romantic crime film "Atlantic City" (1980), the comedy-drama "My Dinner with Andre" (1981), and the autobiographical film "Au revoir les enfants" (1987).
4. The memory also states that **Claude Autant-Lara** (5 August 1901 – 5 February 2000) was a French film director and later Member of the European Parliament (MEP).
5. The memory states that **Russ Meyer** (Russell Albion Meyer) was an American film director, producer, screenwriter, cinematographer, film editor, actor, and photographer. Meyer is known primarily for writing and directing a series of successful sexploitation films that featured campy humor, sly satire and large-breasted women, such as "Faster, Pussycat! Kill! Kill!" He was born March 21, 1922, and died September 18, 2004.
6. The memory states that **Helmut Käutner** (born 25 March 1908 in Düsseldorf, Germany; died 20 April 1980 in Castellina in Chianti, Italy) was a German film director active mainly in the 1940s and 1950s.
7. The memory states that **Rajkumar Kohli** (born 14 September 1930 in Lahore, British India) is an Indian film director. He was famous for directing several popular Bollywood films in the 1970s like Lootera starring Dara Singh and Nishi, who married Rajkumar Koli and 1980s with ensemble cast such as "Nagin" (1976), "Jaani Dushman".
8. The memory states that **Rajkumar Kohli** was famous for directing several popular Bollywood films in the 1970s like Lootera starring Dara Singh and Nishi, who married Rajkumar Koli and 1980s with ensemble cast such as "Nagin" (1976), "Jaani Dushman".

...

The two target death dates were never written, so the comparison cannot be completed.

Figure 12: MemAgent (4B)'s memory updates on the reverse-evidence sample. Death dates in paragraphs 483 and 3911 are skipped because their link to the question is not yet known; later updates keep only the two director names. Intermediate steps are omitted ("...").