RecaLLM: Addressing the Lost-in-Thought Phenomenon with Explicit In-Context Retrieval

Kyle Whitecross, Razieh Rahimi
{kwhitecross, rahimi}@cs.umass.edu
University of Massachusetts Amherst

Abstract

We propose **RecaLLM**, a set of reasoning language models post-trained to make effective use of long-context information. In-context retrieval, which identifies relevant evidence from context, and reasoning are deeply intertwined: retrieval supports reasoning, while reasoning often determines what must be retrieved. However, their interaction remains largely underexplored. In preliminary experiments on several open-source LLMs, we observe that in-context retrieval performance substantially degrades even after a short reasoning span, revealing a key bottleneck for test-time scaling that we refer to as **lost-in-thought**: reasoning steps that improve performance also make subsequent in-context retrieval more challenging. To address this limitation, RecaLLM *interleaves* reasoning with explicit in-context retrieval, alternating between reasoning and retrieving context information needed to solve intermediate subproblems. We introduce a negligible-overhead constrained decoding mechanism that enables verbatim copying of evidence spans, improving the grounding of subsequent generation. Trained on diverse lexical and semantic retrieval tasks, RecaLLM achieves strong performance on two long-context benchmarks, RULER and HELMET, significantly outperforming baselines. Notably, we observe consistent gains at context windows of up to 128K tokens using training samples of at most 10K tokens, far shorter than those used by existing long-context approaches, highlighting a promising path toward improving long-context performance without expensive long-context training data.[1]

1 Introduction

Long-context large language models (LLMs) enable long in-context learning (Bertsch et al., 2025), complex reasoning through scaling test-time compute (Guo et al., 2025; Olmo et al., 2025), and long-horizon agentic workflows (Sun et al., 2025; Li et al., 2025b; Zheng et al., 2025; OpenAI, 2025a; Google, 2025; Yen et al., 2025b), in addition to supporting a wide range of complex real-world applications. However, extending the context window (Lu et al., 2025) does not by itself ensure effective use of long-context information (Hsieh et al., 2024; Yen et al., 2025a). Prior work shows that LLMs exhibit positional biases (Liu et al., 2024; Li et al., 2025a; Tian et al., 2025; Wu et al., 2025) and that their ability to use relevant information degrades as the context length or the difficulty of irrelevant information increases (Shi et al., 2023; Yang et al., 2025b; Wu et al., 2024). Effective long-context use therefore depends critically on *in-context retrieval*, the ability to retrieve relevant information from the input context (Hsieh et al., 2024; Qiu et al., 2025). We show that this capability degrades even further in reasoning language models, where long chain-of-thought reasoning, a key driver of their performance, substantially increases the context length with semantically related, but potentially hard distracting tokens. We refer to this failure as *lost-in-thought* (Figure 1).

Retrieval-based approaches to long-context modeling remain limited in scope. Prior work, including LoongRL (Wang et al., 2026) and ALR^2 (Li et al., 2024), largely trains models to treat in-context retrieval as a step that can be performed before reasoning begins: the initial

[1]Code, data, and models available at https://github.com/kswhitecross/RecaLLM.

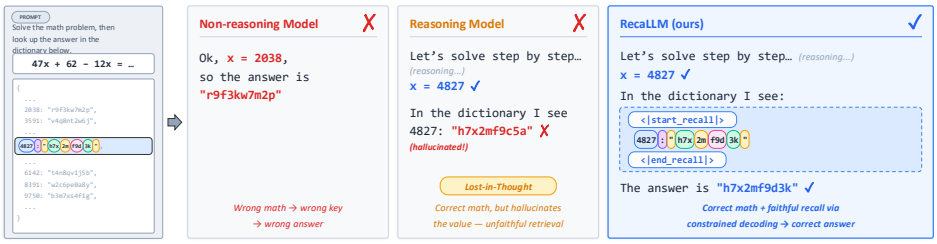

Figure 1: Illustration of lost-in-thought and how RecaLLM mitigates it with explicit, faithful *recall spans*. A reasoning model may recover the correct key yet still hallucinate the value.

query is available in the context and is assumed sufficiently clear to retrieve all evidence needed for the task. However, this assumption is often too restrictive for open-ended tasks, where the need for (additional) context information may emerge after several intermediate reasoning steps and therefore cannot be completely planned upfront. To our knowledge, prior work has not explicitly studied retrieval needs that arise dynamically during the reasoning process itself. This gap is consequential, because effective in-context retrieval in such settings may need to operate not only over the original input context, but also over the model's previously generated intermediate outputs—an increasingly important yet still underexplored setting.

We train RecaLLM on a diverse set of tasks ranging from instances that require no retrieval to those that require multiple retrieval steps. For retrieval-intensive cases, we vary context length, the location of relevant evidence, the type of retrieval required, including both lexical and semantic retrieval, and distractor difficulty through random, hard lexical, and hard semantic negatives. We then post-train LLMs in two stages: a supervised cold start on teacher-annotated rollouts, followed by GRPO (Shao et al., 2024). Crucially, explicit recall spans, together with constrained decoding, make retrieval directly verifiable. As a result, RecaLLM rewards not only final answer quality but also successful retrieval of known relevant evidence, going beyond prior approaches (Wang et al., 2026) that mainly optimize outcome reward alone. Ablations show the importance of this fine-grained training signal.

Across both synthetic and realistic long-context benchmarks, RecaLLM substantially improves effective context use. On RULER (Hsieh et al., 2024), RecaLLM-Qwen2.5-7B achieves the best average among the 7–8B models at 92.8 and remains strong even at 128K tokens, outperforming larger long-context baselines such as LoongRL-14B (Wang et al., 2026). Relative to the base models, RecaLLM gains generally become larger as context length increases: for Qwen, the improvement grows from +6.8 at 4K to +16.1 at 128K, and for Llama the largest gain also appears at 128K (+24.2). On HELMET (Yen et al., 2025a), RecaLLM improves its base models by 16.1–17.7 points on average and attains the strongest overall results in the 7–8B class, especially on retrieval-intensive categories such as Recall, ICL, Re-rank, and citation. Notably, these improvements are not limited to simple key-value lookup: RecaLLM-Llama rises from 3.0 to 64.1 on ICL and from 21.3 to 53.2 on Re-rank, indicating that explicit recall improves not only evidence retrieval but also reasoning over retrieved evidence.

We further show that strong long-context gains can be achieved without training on comparably long sequences. RecaLLM is trained on contexts of at most 10K tokens, yet it yields consistent improvements up to 128K tokens, the longest evaluation length. This is particularly notable given that recent long-context methods often rely on larger training contexts: ProLong (Gao et al., 2025) is trained on sequences up to 512K tokens, exceeding its 128K evaluation length, while LoongRL (Wang et al., 2026) trains at 16K. These results suggest that directly improving in-context retrieval can yield long-context performance that generalizes well beyond the training range. Given the substantial cost of long-context training and the difficulty of curating high-quality long-sequence data, our approach points to a more efficient path for extending the effective context length of LLMs.

We summarize our main contributions as follows: (1) We identify *lost-in-thought*, a failure mode in reasoning LLMs where long chain-of-thought makes in-context retrieval harder.

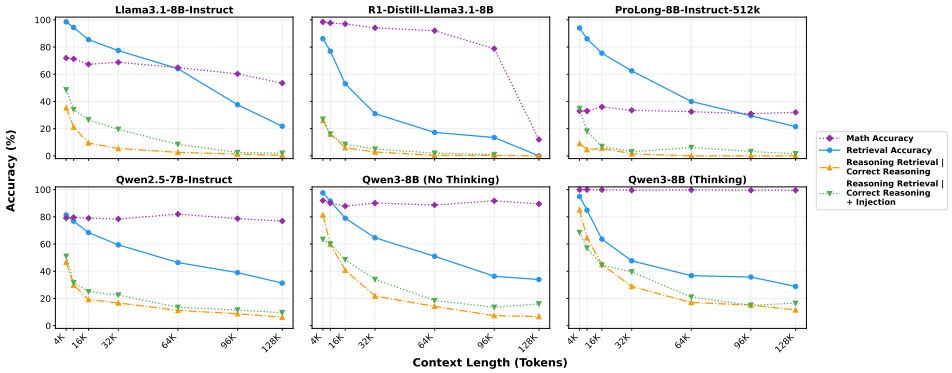

Figure 2: *Lost in thought*: retrieval accuracy before and after reasoning. Injected accuracy measures faithful copying after providing the correct key and prefix.

(2) We propose RecaLLM, which interleaves reasoning with explicit in-context retrieval through *recall spans*. (3) We introduce context-aware constrained decoding for recall spans, ensuring faithful verbatim recall from context and making retrieval directly verifiable. (4) We show that RecaLLM delivers strong long-context gains, especially on retrieval-intensive tasks, and generalizes from 10K training contexts to evaluations up to 128K.

2 Lost-in-Thought: How Reasoning Degrades In-Context Retrieval

To study how reasoning interacts with long-context capabilities in modern LLMs, we construct a synthetic benchmark that isolates in-context retrieval performance before and after reasoning. Throughout, we use *reasoning* to refer to any intermediate generation between the question and the final answer, whether elicited by post-training or by step-by-step prompting. The benchmark centers on a large structured key-value dictionary provided as part of the prompt, and defines two tasks:

- **Retrieval**: the prompt directly specifies a query key and asks the model to retrieve the corresponding value, following prior synthetic retrieval benchmarks such as RULER (Hsieh et al., 2024).
- **Reasoning-Retrieval**: the prompt instead presents a math problem whose solution determines the query key, requiring the model to first reason and then retrieve.

Context lengths range from 4K to 128K tokens, controlled by varying the number of distractor key-value pairs. To improve robustness, we vary prompt templates, query placement relative to the dictionary, math problem type, and dictionary format (CSV, JSON, list); all other factors are held fixed across length conditions. Examples are in Appendix A.2.

Figure 2 presents results for five open-source 7–8B LLMs: **Llama-3.1-8B-Instruct** (Dubey et al., 2024), **R1-Distill-Llama-8B** (Guo et al., 2025), **ProLong-8B-512K** (Gao et al., 2025), **Qwen2.5-7B-Instruct** (Qwen Team, 2025), and **Qwen3-8B** (Yang et al., 2025a) (details in Appendix A.1). Across all models, retrieval accuracy after reasoning is substantially worse than direct retrieval, even though math accuracy remains high and stable. This gap persists across all context lengths (Table 5). We refer to this phenomenon as *lost in thought*.

A follow-up injection experiment (Appendix A.3) confirms that this degradation stems largely from the model's inability to faithfully *copy* information from context after reasoning, rather than from failing to identify what to retrieve: even when given the correct key and its exact lexical prefix mid-generation, models still frequently hallucinate the value (Figure 2, Table 5). We hypothesize that the semantically related tokens generated during reasoning act as distractors that interfere with faithful reproduction.

3 RecaLLM

To address lost-in-thought, RecaLLM makes in-context retrieval an explicit step within generation. Models reason until they identify a need for contextual evidence, then enter a *recall span* that copies that evidence verbatim from the input or prior generation, turning a long-context retrieval problem into a local reasoning one. Two components enable this: special delimiter tokens that mark recall spans, and a constrained decoding mechanism that restricts generation within them to valid continuations of the searchable context, guaranteeing faithfulness by construction. The model decides *what* and *when* to recall; constrained decoding ensures the recalled content is exact; and RL training (Section 4) teaches the model to use this capability effectively.

3.1 Recall Tokens and Recall Spans

We extend the base vocabulary $\mathcal{V}$ with two special tokens: <|start_recall|> (R_{start}) and <|end_recall|> (R_{end}), which denote the beginning and end of a recall span, respectively. Let $x = (x_1, \ldots, x_n)$ denote the tokenized input prompt. The model generates an output sequence over the augmented vocabulary $\mathcal{V} \cup \{R_{\text{start}}, R_{\text{end}}\}$, consisting of regular output tokens interleaved with recall spans:

$$(\underbrace{y_1, \ldots, y_T}_{\text{reasoning}},\ R_{\text{start}},\ \underbrace{r_1, \ldots, r_K}_{\text{recall span}},\ R_{\text{end}},\ \underbrace{y_{T+1}, \ldots}_{\text{reasoning}},\ \ldots)$$

where $y_1, \ldots, y_T$ are regular output tokens generated before the recall span and $r_1, \ldots, r_K$ are tokens generated within the recall span. Multiple recall spans may appear in a single generation.

Outside of recall spans, generation proceeds under the standard next-token distribution: $y_t \sim p_\theta(\cdot \mid x, y_{<t})$. When the model emits R_{start}, it enters a recall span. We define the *searchable context* c as the concatenation of the input prompt with all tokens generated prior to the current recall span: $c = x \,\|\, y_{\leq T}$. Let $r_{1:k}$ denote the k tokens generated so far inside the current recall span. The *valid continuation set* is

$$\mathcal{A}(c, r_{1:k}) = \{v \in \mathcal{V} \mid \exists\, i \text{ such that } c_{i:i+k-1} = r_{1:k} \text{ and } c_{i+k} = v\},$$

the set of all tokens that could continue at least one exact occurrence of the recalled prefix in the searchable context. During a recall span, the model may only emit tokens from this set, or the stop token R_{end}. Formally, if z_v denotes the model's logit for token v, we apply the mask

$$\tilde{z}_v = \begin{cases} z_v & \text{if } v \in \mathcal{A}(c, r_{1:k}) \cup \{R_{\text{end}}\}, \\ -\infty & \text{otherwise,} \end{cases}$$

and sample the next token from the softmax over $\tilde{z}$. This guarantees that *every recall span is a contiguous substring of the searchable context*. This constrained decoding does not guarantee relevance, but by faithfully copying evidence into the generation, it enables the model to reason over grounded evidence in subsequent steps. Learning *which* spans to recall, and *when*, is the role of RL training (Section 4). Because the logit mask does not modify the model's internal representations and is differentiable with respect to the allowed logits, RL training proceeds similarly to the unconstrained case. The logit mask depends only on token IDs and is computed on a separate CPU thread, fully overlapped with the GPU forward pass, adding negligible overhead to generation latency; implementation details and complexity analysis are provided in Appendix B.1.

4 RecaLLM Training

We train RecaLLM using a two-stage pipeline similar to Guo et al. (2025): a Supervised Finetuning (SFT) cold start on a small set of reasoning traces annotated with recall tokens, followed by RL on a diverse task mixture.

Category	Datasets	Count	Context Length
Multi-Hop QA	HotpotQA, MuSiQue, 2WikiMQA	3,000	8–10K
Single-Hop QA	NQ, TriviaQA	2,000	8–10K
Retrieval	KV Retrieval, Multi-NIAH	2,000	8–10K
Reasoning Retrieval	Math Retrieval	2,000	8–10K
Short-Context Math	DAPO Math, MCQA Math	1,000	$\leq$1K
In-Context Learning	Banking77, MASSIVE	3,000	8–10K
Long-Document QA	QuALITY	1,000	$\leq$10K
Aggregation	Majority Vote, Top-N Vote	2,000	8–10K
Reranking	MSMARCO v2	2,000	8–10K
Entity Citation	QAMPARI	2,000	8–10K

Table 1: RL training dataset composition. Detailed descriptions in Appendix C.4.

4.1 Supervised Finetuning Cold Start

We train the model with a two-stage adaptation procedure that first learns the new token embeddings and subsequently performs brief full-model finetuning, encouraging the model to use recall tokens naturally during interleaved generation (Appendix C.1).

To construct the SFT dataset, we collect reasoning traces from six teacher models across four reasoning and retrieval tasks. Each teacher is prompted to reason and explicitly reference relevant context information, and we retain only completions that produce the correct answer. We then prompt GPT-5.2 (OpenAI, 2025b) to rewrite these traces so that references to context are realized as verbatim recall spans, providing the annotator with gold documents to ensure recall span accuracy. After annotation, we align each recall span to its source text via fuzzy string matching and discard failed alignments, yielding 1,795 properly annotated examples (Appendix C.2).

4.2 RL Training Data

We further train the SFT checkpoints with GRPO (Shao et al., 2024) on a broad but intentionally shallow mixture of 20,000 examples across 10 task categories (Table 1), encouraging the model to learn strategic in-context recall rather than task-specific heuristics. The categories vary along several axes of recall difficulty. Single-hop and multi-hop QA present identical-looking contexts but require different amounts of retrieval, requiring the model to interleave reasoning and retrieval strategically rather than following a fixed pattern. Passage reranking requires selective recall under a limited generation budget. Short-context math and two novel synthetic aggregation tasks (majority vote and top-N vote) require no retrieval, teaching the model to invoke recall only when beneficial, following Wang et al. (2026). To prevent RecaLLM from learning recall behavior tied to narrow surface patterns, we apply systematic data augmentation across all categories, varying instruction phrasing, question placement, passage format (e.g., fixed-window chunks vs. natural paragraphs, with or without titles), and distractor composition, sampling from random, BM25 (Robertson et al., 1995), and dense retrieval (Zhang et al., 2024a) negatives. Dataset descriptions and augmentation details are provided in Appendix C.4.

Most training categories require no human-annotated gold evidence: KV retrieval, Multi-NIAH, and reasoning-retrieval generate gold spans programmatically, long-document QA uses no segmented evidence, and short-context math and aggregation require no retrieval supervision. Human-annotated evidence enters only through the QA, reranking, and citation categories, where existing datasets already provide supporting documents. In domains without such annotations, the retrieval reward introduced in Section 4.3 can instead be instantiated with automatically derived supervision, such as heuristic labels from retrieval pipelines or distillation from rerankers; we leave this to future work.

4.3 RL Reward Function

A central challenge in our RL training is that successful behavior is inherently compositional: the model must first accurately retrieve supporting evidence through recall spans and then use that evidence to generate the correct final answer. We therefore optimize a composite reward that jointly scores formatting, answer quality, and evidence recall quality:

$$R = 0.2 \cdot R_{\text{format}} + 0.4 \cdot R_{\text{add}} + 0.4 \cdot R_{\text{mult}}, \tag{1}$$

$$R_{\text{add}} = 0.5 \cdot R_{\text{ans}} + 0.5 \cdot R_{\text{ret}}, \tag{2}$$

$$R_{\text{mult}} = \sqrt{(R_{\text{ans}} + \epsilon)(R_{\text{ret}} + \epsilon)} - \epsilon \qquad (\epsilon = 0.01), \tag{3}$$

where R_{format} verifies the required interleaved output format, R_{ans} is a task-specific answer quality metric such as exact match, F1, or NDCG@10 (Table 6), and R_{ret} measures recall quality against gold evidence. The additive component R_{add} provides a learning signal when either the final answer or the recalled evidence is (partially) correct, which is especially important early in training when jointly successful trajectories are sparse. In contrast to purely additive reward formulations (Wang et al., 2026; Jin et al., 2025; Song et al., 2025), we introduce R_{mult}, a smoothed geometric mean that explicitly favors rollouts achieving high answer quality and high retrieval quality simultaneously. We optimize this reward using GRPO on 4 A100 GPUs for a single epoch with 16 rollouts per example; training infrastructure and hyperparameters are detailed in Appendix C.3.

In-context Retrieval Reward. For tasks with gold passages, R_{ret} measures how well the model's recall spans recover the gold evidence. Because both gold passages and recalled spans generated under constrained decoding are contiguous substrings of the context, each can be identified by its character-level position interval. We compute an F1 score over the intersection of these intervals (Appendix C.5), normalized by a task-specific hit threshold τ that caps the reward once overlap is sufficient. This prevents the reward from favoring exhaustive copying and reflects the fact that for many tasks (e.g., multi-hop or single-hop QA) only a small portion of the gold passage is actually relevant to the question, a convention long established in IR relevance assessment (Voorhees & Harman, 2005). These per-passage scores are then averaged: over all gold passages for tasks with few gold documents (e.g., multi-hop QA), or over the top-K highest-scoring passages for tasks with many (e.g., reranking). Importantly, this interval-based reward is only precise because constrained decoding guarantees verbatim reproduction; without it, even a single non-verbatim character breaks the contiguous match, making the signal substantially noisier. For tasks without segmented gold evidence (long-document QA), the overlap score is set to 1 if any recall span is present; for tasks that do not require retrieval (short-context math, aggregation), it is set to 1 unconditionally. Formal definitions and per-category reward configurations are in Appendix C.5 and Table 6.

Penalties. R_{ret} is further modulated by two penalties that target the two obvious failure modes of learning a new action, rather than task-specific heuristics. A *correctness penalty* detects degenerate recall spans: very short, typically out-of-context fragments, and malformed delimiter usage such as unmatched R_{end} tokens. A *density penalty* exponentially downweights reward when the frequency of recall spans exceeds a threshold, with a task-dependent number of initial spans exempt (Table 6); it targets the natural reward-hacking strategy of spamming recall spans to guess the gold evidence rather than retrieving it strategically. We penalize density rather than a flat count to avoid discouraging longer generations. Empirically, both penalties act as early-training stabilizers: their values saturate near 1 within the first $\sim$30 GRPO steps and play little role thereafter (Figure 11), while recall spans per completion drop from 20–40 to 5–7 as gold-evidence overlap climbs (Appendix D.3). In contrast to RL-based agentic RAG recipes that require restarts (Jin et al., 2025) or an extra format-only RL stage (Song et al., 2025) to stabilize training, RecaLLM trains stably in a single GRPO stage after SFT. Formal definitions are in Appendix C.5.

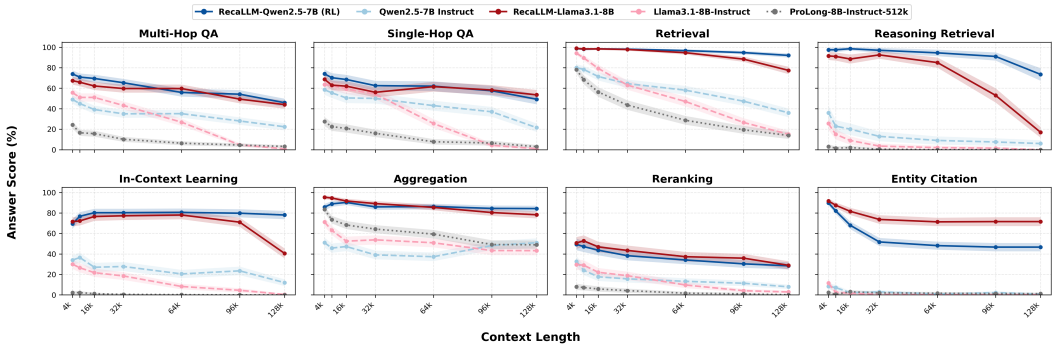

Figure 3: Per-category answer scores across context lengths (4K–128K) on the in-domain evaluation. RecaLLM models maintain strong performance as context length increases, while baselines degrade sharply beyond 32K tokens. Shaded regions denote 95% confidence intervals.

5 Experimental Setup and Results

We evaluate RecaLLM across two base model families to test the robustness of its gains across architectures. Specifically, we train RecaLLM variants initialized from Qwen2.5-7B-Instruct (Qwen Team, 2025) and Llama-3.1-8B-Instruct (Dubey et al., 2024). Appendix D.3 provides training-curve analyses showing stable optimization, increasingly selective and accurate recall, and consistent gains across all ten tasks in Table 1 for both RecaLLM models.

In addition to evaluating RecaLLM models on validation splits of our training datasets, we assess their generalizability on two established long-context benchmarks: RULER (Hsieh et al., 2024) and HELMET (Yen et al., 2025a), following prior work on long-context evaluation (Gao et al., 2025; Wang et al., 2026; Olmo et al., 2025). More details of these two benchmarks are provided in Appendix D.1.

5.1 Performance on In-Domain Long-Context Tasks

Figure 3 and Table 8 in Appendix D.2 report performance on validation splits of the training datasets across context lengths from 4K to 128K. RecaLLM models substantially outperform all baselines across nearly every category, with the largest gains emerging at longer contexts, where baseline performance degrades sharply. The improvements are particularly significant for tasks that require retrieval after reasoning. On reasoning-retrieval, RecaLLM-Qwen improves from 23.0% to 97.6% at short contexts and from 7.5% to 86.3% at long contexts. On entity citation, RecaLLM-Llama improves from 4.1% to 83.6% at short contexts and from 0.3% to 71.5% at long contexts. Consistent gains also appear on reasoning-oriented tasks: short-context math improves by 5.2 points for RecaLLM-Qwen, and aggregation benefits substantially despite not requiring explicit recall.

5.2 Performance on Long-Context Benchmarks

We evaluate RecaLLM models on the RULER (Hsieh et al., 2024) and HELMET (Yen et al., 2025a) benchmarks, comparing against their corresponding base LLMs as well as ProLong (Gao et al., 2025), LoongRL (Wang et al., 2026), and QwenLong (Wan et al., 2025). For LoongRL and QwenLong, we use the results reported by Wang et al. (2026) whenever available, since LoongRL's models and implementation are not publicly released. We reproduce results for base LLMs Llama3.1-8B-Instruct and Qwen2.5-7B-Instruct, as well as ProLong-8B-512k under the same reasoning-oriented system prompt used for RecaLLM. To ensure a fair comparison between reasoning and non-reasoning models, we evaluate all RULER and HELMET tasks using chat templates, rather than relying on answer-only prompts that can disadvantage reasoning models. The base models isolate the effect of RecaLLM post-training on widely used LLMs, while LoongRL and QwenLong-L1 represent substantially larger training budgets: LoongRL performs RL with roughly 1.3M rollouts at

	RULER							HELMET							
Model	**4K**	**8K**	**16K**	**32K**	**64K**	**128K**	**Avg**	**Recall**	**RAG**	**ICL**	**Cite**	**Re-rank**	**LongQA**	**Summ**	**Avg**
ProLong-8B-Instruct-512K	92.3	86.0	73.7	59.5	60.5	46.8	69.8	74.3	68.7	18.6	12.4	12.5	**36.4**	36.5	37.0
QwenLong-L1-32B	91.7	88.5	87.6	86.8	80.6	70.2	84.2	—	52.9	—	14.3	—	—	47.0	—
LoongRL-7B	95.1	94.3	93.4	91.4	86.2	76.8	89.5	—	61.6	—	11.4	—	—	**44.5**	—
LoongRL-14B	97.6	96.1	95.4	95.1	87.1	79.9	91.9	—	63.7	—	18.7	—	—	49.8	—
Llama-3.1-8B-Instruct	95.7	93.5	89.8	85.6	78.6	42.2	80.9	77.0	63.0	3.0	10.0	21.3	36.3	39.2	35.7
RecaLLM-Llama-3.1-8B	97.7	**97.2**	95.5	90.7	78.6	66.4	87.7	**98.2**	68.1	64.1	**24.7**	**53.2**	34.8	31.0	53.4
Qwen2.5-7B-Instruct	92.0	86.8	86.1	82.3	76.8	66.8	81.8	58.8	58.6	41.7	10.3	17.7	33.2	41.3	37.4
RecaLLM-Qwen2.5-7B	**98.8**	96.7	**95.8**	**93.1**	**89.7**	**82.9**	**92.8**	96.2	**69.4**	**69.7**	13.7	46.2	35.5	44.0	**53.5**

Table 2: Performance on RULER (Hsieh et al., 2024) (averaged across tasks) and HELMET (Yen et al., 2025a) (averaged across context lengths). Models with — indicate results not reported in their papers.

16K context on 16 A100s, and QwenLong-L1 scales curriculum-guided RL to 60K contexts on 32 A100s, compared with RecaLLM's roughly 307K rollouts at 8–10K context on 4 A100s (Appendix C.3).

RULER Results. On RULER (Table 2), both RecaLLM models substantially outperform their base models across all context lengths. Two patterns are especially notable in Table 2. First, RecaLLM yields strong parameter-efficient long-context performance: RecaLLM-Qwen achieves the best average score in the 7B–8B class (**bolded** in Table 2) and even surpasses substantially larger baselines, LoongRL-14B and QwenLong-L1-32B, despite being trained on substantially shorter contexts (8–10K tokens, compared with 16K for LoongRL and up to 60K for QwenLong-L1). Second, the gains become larger as context length increases, indicating that RecaLLM improves not only overall accuracy but also robustness to extreme context length. For example, RecaLLM-Qwen improves over its base by 6.8 points at 4K but by 16.1 at 128K, and reduces the 4K-to-128K degradation from 25.2 points to 15.9. Notably, these improvements do not come at the expense of short-context performance, as both RecaLLM variants also outperform their base models at short contexts such as 4K and 8K. The sharper decline of RecaLLM-Llama beyond 64K, compared to RecaLLM-Qwen, suggests that the long-context generalization is partly inherited from pretraining.

HELMET Results. On HELMET (Table 2), RecaLLM achieves the highest overall average among 7–8B models, improving over its base Qwen2.5-7B and Llama-3.1-8B models by 16.1 and 17.7 points, respectively. This suggests that the gains from RecaLLM are robust across different base model families. The improvements are especially significant on Recall, ICL, Re-rank, and Cite, where faithful access to dispersed context information is essential. Notably, RecaLLM-Llama improves from 3.0 to 64.1 on ICL and from 21.3 to 53.2 on Re-rank. HELMET's ICL tasks use randomly generated numeric labels rather than semantic class names, testing pure pattern matching over the provided examples. The recall mechanism is particularly well suited to this setting, as it enables the model to faithfully retrieve exact example–label pairs from context rather than hallucinating plausible numbers. These improvements suggest that explicit recall spans enhance not only in-context evidence retrieval, but also downstream reasoning over retrieved evidence.

At the same time, gains on LongQA and Summ are smaller and less consistent, in line with HELMET's deliberately heterogeneous design (Yen et al., 2025a) and RecaLLM's focus on faithful in-context retrieval. RecaLLM-Qwen improves slightly on both tasks, whereas RecaLLM-Llama declines slightly on LongQA (36.3 to 34.8) and more substantially on Summ (39.2 to 31.0). Both categories require long-form generation and are evaluated with an LLM-as-a-judge. Taken together with the near-saturated Recall performance in Table 2, these results suggest that the remaining challenge in LongQA and Summ lies less in retrieving relevant evidence than in composing that evidence into grounded, coherent long-form outputs.

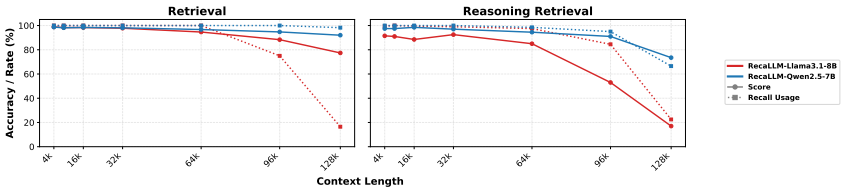

Figure 4: Accuracy and recall token usage rate of RecaLLM models across context lengths on the retrieval and reasoning-retrieval tasks from Section 2.

Model	1-Hop	N-Hop	Ret.	R→Ret.	Math	ICL	LQA	Agg.	Rerank	Ent. Cite	Avg
RecaLLM-Qwen2.5-7B	63.4	62.2	**96.7**	**92.8**	61.0	77.9	72.2	86.5	**38.8**	**61.8**	**71.3**
– No Recall Reward	53.8	54.8	94.8	88.6	**62.0**	75.8	**76.0**	70.1	28.0	51.6	65.5
– No Logit Masking	**69.5**	**70.2**	89.5	72.1	58.2	**78.4**	72.2	**90.6**	35.3	58.1	69.4

Table 3: Performance of ablated variants of RecaLLM-Qwen on validation sets, averaged over all context lengths.

6 Analysis

6.1 Generalizability of the Recall Capability

Although RecaLLM is trained on only 8–10K contexts, it generalizes substantially beyond this range. Figure 4 plots task score alongside recall usage rate across context lengths on the retrieval and reasoning-retrieval tasks from Section 2. The results show that accuracy closely tracks recall usage on both tasks: performance remains strong when recall is used consistently, and degrades precisely when recall usage falls. RecaLLM-Qwen maintains high recall usage over nearly the entire range and correspondingly sustains strong performance, while RecaLLM-Llama shows a clear drop in recall usage at 96K and 128K, with much larger accuracy drops, especially on reasoning-retrieval. When recall usage drops, the model reverts to implicit retrieval. This pattern indicates that a main failure reason at extreme lengths is policy drift away from invoking recall. We offer two speculative hypotheses for this gap in Appendix E.1: Qwen-family bases appear generally more receptive to RL post-training than Llama (Gandhi et al., 2025; Wang et al., 2025), and Llama's natively pretrained 128K attention patterns may be only partially updated by our short-context RL, unlike Qwen's YaRN-stretched windows.

6.2 Ablation Studies

To isolate the roles of constrained decoding (Section 3.1) and the retrieval reward (Section 4.3), we train two ablations of RecaLLM-Qwen2.5-7B. **No Recall Reward** removes supervision for retrieving the correct evidence, including the density and correctness penalties, and sets $R_{\text{ret}} = 1$ unconditionally during RL. **No Logit Masking** ablation keeps the full training recipe, including SFT cold start, GRPO training using the same data and reward model, but disables constrained decoding within recall spans. Table 3 reports results on the validation sets, and Table 4 reports results on RULER and HELMET; additional validation-set results are provided in Appendix E.

Table 3 indicates that the two components are complementary. Removing the recall reward causes the larger overall degradation, reducing the average score from 71.3 to 65.5, with especially large drops on aggregation, reranking, entity citation, and QA. This suggests that the recall reward is the main training signal that teaches the model to use recall broadly rather than only in tasks where copying is obviously beneficial. Removing logit masking leads to a smaller average decline, from 71.3 to 69.4, but causes much larger losses on the retrieval-intensive categories that require exact lexical matching, namely retrieval and reasoning-retrieval. Meanwhile, removing logit masking improves performance on categories where exact lexical retrieval is less critical, suggesting that unconstrained generation offers additional flexibility on these tasks. At the same time, constrained decoding

	RULER							HELMET				
Model	**4K**	**8K**	**16K**	**32K**	**64K**	**128K**	**Avg**	**Recall**	**RAG**	**ICL**	**Cite**	**Re-rank**
RecaLLM-Qwen2.5-7B	**98.8**	**96.7**	**95.8**	**93.1**	**89.7**	**82.9**	**92.8**	**96.2**	69.4	69.7	13.7	**46.2**
– No Recall Reward	96.7	93.3	89.9	88.0	81.4	77.9	87.9	94.9	66.1	69.0	14.9	25.3
– No Logit Masking	98.1	95.0	95.4	93.0	88.2	82.3	92.0	87.7	**78.3**	**76.7**	**15.4**	44.9

Table 4: Ablation results on RULER and HELMET. LongQA and Summarization are excluded due to the cost of LLM-as-a-judge evaluation over long outputs.

remains important because it makes the recall reward straightforward to incorporate, and that reward has a substantial impact on RecaLLM performance.

Table 4 shows the same pattern on RULER and HELMET. Removing the recall reward costs 4.9 RULER points on average and 5.0 over the five HELMET categories evaluated, nearly halving Re-rank (46.2 to 25.3), while removing logit masking costs only 0.8 RULER points. Both ablations still outperform base Qwen2.5-7B-Instruct (Table 2); since gold evidence enters training only through the recall reward, No Recall Reward also reflects RecaLLM's performance when gold evidence is unavailable.

7 Related Work

RL for long-context reasoning. LoongRL (Wang et al., 2026) and QwenLong-L1 (Wan et al., 2025) train models via RL to reason over long contexts (Appendix F.3), but differ from RecaLLM in three key respects. First, neither provides an explicit in-context retrieval step nor a mechanism for *faithful* retrieval. Second, neither explicitly rewards retrieval quality. Third, both train on substantially longer contexts for far more steps, yet RecaLLM-Qwen2.5-7B outperforms them on RULER and HELMET (Table 2).

Retrieval-augmented reasoning and context extension. Agentic RAG models (Jin et al., 2025; Song et al., 2025; Li et al., 2025b; Zheng et al., 2025) query external corpora during reasoning (Appendix F.2). RecaLLM is complementary: agentic search systems that retrieve documents into context would benefit from stronger in-context utilization of the evidence they gather. A large body of work targets context extension at pretraining or continued-training time (Lu et al., 2025; Xiong et al., 2024), including ProLong (Gao et al., 2025) and YaRN (Peng et al., 2024). RecaLLM operates at the post-training stage and is agnostic to the underlying context extension recipe.

Constrained decoding and copy mechanisms. Grammar-constrained decoding (Willard & Louf, 2023; Dong et al., 2025) and entity-constrained generation (De Cao et al., 2021) use logit masking to enforce structural constraints from a fixed grammar or candidate set (Appendix F.4). Classical copy mechanisms (Gu et al., 2016; See et al., 2017) similarly rely on learned copy distributions.

8 Conclusion

We introduce RecaLLM, a family of reasoning language models that address the *lost-in-thought* problem, by interleaving reasoning with explicit, constrained-decoding recall spans, achieving strong results on RULER and HELMET that surpass larger models trained on longer contexts in fewer steps. Looking ahead, we believe explicit in-context retrieval opens several promising directions: more flexible retrieval mechanisms that preserve faithfulness without the rigidity of strict constrained decoding, self-recall over the model's own prior generation for long-horizon reasoning consistency, and scaling to larger, more capable models that should better learn when and how to invoke retrieval. More broadly, making retrieval an explicit, verifiable step within reasoning creates new opportunities for reward design, interpretability, and trustworthy grounding in long-context applications.

Reproducibility Statement

Code, training data, and trained RecaLLM model weights are publicly available at https://github.com/kswhitecross/RecaLLM. RecaLLM models are implemented within the Hugging Face Transformers library (Wolf, 2020), allowing users to download, run and train them with standard inference pipelines. All base models, training datasets, and evaluation benchmarks used in this work are publicly available. Training hyperparameters, reward configurations, and per-category settings are reported in Appendix C, and evaluation hyperparameters are reported in Appendix D.1.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval and in part by NSF grant number 2339932. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

Samuel Amouyal, Tomer Wolfson, Ohad Rubin, Ori Yoran, Jonathan Herzig, and Jonathan Berant. QAMPARI: A benchmark for open-domain questions with many answers. In Sebastian Gehrmann, Alex Wang, João Sedoc, Elizabeth Clark, Kaustubh Dhole, Khyathi Raghavi Chandu, Enrico Santus, and Hooman Sedghamiz (eds.), *Proceedings of the Third Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, pp. 97–110, Singapore, December 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.gem-1.9/.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of ACL*, pp. 3119–3137, 2024.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. Ms marco: A human generated machine reading comprehension dataset. In *InCoCo@NIPS*, 2016.

Amanda Bertsch, Maor Ivgi, Emily Xiao, Uri Alon, Jonathan Berant, Matthew R. Gormley, and Graham Neubig. In-context learning with long-context models: An in-depth exploration. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 12119–12149, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.605. URL https://aclanthology.org/2025.naacl-long.605/.

Stella Biderman. Math-mcqa: A multiple choice adaptation of the math dataset, 2025. URL https://huggingface.co/datasets/stellaathena/math_mcqa.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pp. 38–45, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.nlp4convai-1.5.

Subhajit Chaudhury, Payel Das, Sarathkrishna Swaminathan, Georgios Kollias, Elliot Nelson, Khushbu Pahwa, Tejaswini Pedapati, Igor Melnyk, and Matthew Riemer. EpMAN: Episodic memory AttentioN for generalizing to longer contexts. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025. URL https://aclanthology.org/2025.acl-long.574/.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. Autoregressive entity retrieval. In *Proceedings of ICLR*, 2021.

Yixin Dong, Charlie F. Ruan, Yaxing Cai, Ruihang Lai, Ziyi Xu, Yilong Zhao, and Tianqi Chen. XGrammar: Flexible and efficient structured generation engine for large language models. In *Proceedings of MLSys*, 2025.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Tal Einat. fuzzysearch: Find almost exact matches in strings, 2013. URL https://github.com/taleinat/fuzzysearch.

Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, Swetha Ranganath, Laurie Crist, Misha Britan, Wouter Leeuwis, Gokhan Tur, and Prem Natarajan. MASSIVE: A 1M-example multilingual natural language understanding dataset with 51 typologically-diverse languages. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4277–4302, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.235.

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective STaRs. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=QGJ9ttXLTy.

Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. Enabling large language models to generate text with citations. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 6465–6488, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.398. URL https://aclanthology.org/2023.emnlp-main.398/.

Tianyu Gao, Alexander Wettig, Howard Yen, and Danqi Chen. How to train long-context language models (effectively). In *ACL*, 2025.

Gemma Team. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Google. Gemini deep research. https://gemini.google/overview/deep-research/, September 2025.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of ACL*, pp. 1631–1640, 2016.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

John Hewitt. Initializing new word embeddings for pretrained language models. https://www.cs.columbia.edu/~johnhew/vocab-expansion.html, 2021. Blog post.

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 COLING*, 2020.

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 *Proceedings of COLM*, 2024.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of ACL*, 2017.

Greg Kamradt. Needle in a haystack — pressure testing LLMs. GitHub, 2023. URL https://github.com/gkamradt/LLMTest_NeedleInAHaystack.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. In *Proceedings of ICLR*, 2020.

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

Jinhyuk Lee, Anthony Chen, Zhuyun Dai, Dheeru Dua, Devendra Singh Sachan, Michael Boratko, Yi Luan, Sébastien M. R. Arnold, Vincent Perot, Siddharth Dalmia, Hexiang Hu, Xudong Lin, Panupong Pasupat, Aida Amini, Jeremy R. Cole, Sebastian Riedel, Iftekhar Naim, Ming-Wei Chang, and Kelvin Guu. Can long-context language models subsume retrieval, rag, sql, and more?, 2024. URL https://arxiv.org/abs/2406.13121.

Huayang Li, Pat Verga, Priyanka Sen, Bowen Yang, Vijay Viswanathan, Patrick Lewis, Taro Watanabe, and Yixuan Su. ALR^2: A retrieve-then-reason framework for long-context question answering. *arXiv preprint arXiv:2410.03227*, 2024.

Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. Long-context LLMs struggle with long in-context learning. *Transactions on Machine Learning Research*, 2025a. ISSN 2835-8856. URL https://openreview.net/forum?id=Cw2xlg0e46.

Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yutao Zhu, Yongkang Wu, Ji-Rong Wen, and Zhicheng Dou. WebThinker: Empowering large reasoning models with deep research capability. In *Proceedings of NeurIPS*, 2025b.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Rich James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, Luke Zettlemoyer, and Wen tau Yih. RA-DIT: Retrieval-augmented dual instruction tuning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=22OTbutug9.

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/.

Yi Lu, Jing Nathan Yan, Songlin Yang, Justin T Chiu, Siyu Ren, Fei Yuan, Wenting Zhao, Zhiyong Wu, and Alexander M Rush. A controlled study on long context extension and generalization in LLMs. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=BLonuGXDFu.

Lance Martin. Multi needle in a haystack. LangChain Blog, 2024. URL https://blog.langchain.com/multi-needle-in-a-haystack/.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

OpenAI. Introducing deep research. https://openai.com/index/introducing-deep-research/, February 2025a.

OpenAI. OpenAI GPT-5 system card. *arXiv preprint arXiv:2601.03267*, 2025b.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and Samuel R. Bowman. QuALITY: Question answering with long input texts, yes! In *Proceedings of NAACL*, 2022.

Adam et al. Paszke. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

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.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. KILT: a benchmark for knowledge intensive language tasks. In *Proceedings of NAACL*, 2021.

Yifu Qiu, Varun R. Embar, Yizhe Zhang, Navdeep Jaitly, Shay B Cohen, and Benjamin Han. Eliciting in-context retrieval and reasoning for long-context large language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 3176–3192, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.165. URL https://aclanthology.org/2025.findings-acl.165/.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: Memory optimizations toward training trillion parameter models. In *Proceedings of SC*, 2020.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Jian Su, Kevin Duh, and Xavier Carreras (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264/.

Stephen Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In *Overview of the Third Text REtrieval Conference (TREC-3)*, pp. 109–126. Gaithersburg, MD: NIST, January 1995. URL https://www.microsoft.com/en-us/research/publication/okapi-at-trec-3/.

Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of ACL*, pp. 1073–1083, 2017.

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. *arXiv preprint arXiv:2402.03300*, 2024.

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. *arXiv preprint arXiv:2409.19256*, 2024.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 31210–31227. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/shi23a.html.

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

Shuang Sun, Huatong Song, Yuhao Wang, Ruiyang Ren, Jinhao Jiang, Junjie Zhang, Fei Bai, Jia Deng, Wayne Xin Zhao, Zheng Liu, Lei Fang, Zhongyuan Wang, and Ji-Rong Wen. SimpleDeepSearcher: Deep information seeking via web-powered reasoning trajectory synthesis. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 13705–13720, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.739. URL https://aclanthology.org/2025.findings-emnlp.739/.

Runchu Tian, Yanghao Li, Yuepeng Fu, Siyang Deng, Qinyu Luo, Cheng Qian, Shuo Wang, Xin Cong, Zhong Zhang, Yesai Wu, Yankai Lin, Huadong Wang, and Xiaojiang Liu. Distance between relevant information pieces causes bias in long-context LLMs. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 521–533, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.28. URL https://aclanthology.org/2025.findings-acl.28/.

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

Ellen M. Voorhees and Donna K. Harman. *TREC: Experiment and Evaluation in Information Retrieval*. MIT Press, Cambridge, MA, 2005.

Fanqi Wan, Weizhou Shen, Shengyi Liao, Yingcheng Shi, Chenliang Li, Ziyi Yang, Ji Zhang, Fei Huang, Jingren Zhou, and Ming Yan. QwenLong-L1: Towards long-context large reasoning models with reinforcement learning. *arXiv preprint arXiv:2505.17667*, 2025.

Siyuan Wang, Gaokai Zhang, Li Lyna Zhang, Ning Shang, Fan Yang, Dongyao Chen, and Mao Yang. LoongRL: Reinforcement learning for advanced reasoning over long contexts. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=o29E01Q6bv.

Zengzhi Wang, Fan Zhou, Xuefeng Li, and Pengfei Liu. OctoThinker: Mid-training incentivizes reinforcement learning scaling. *arXiv preprint arXiv:2506.20512*, 2025.

Brandon T. Willard and Rémi Louf. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*, 2023.

Thomas et al. Wolf. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45. Association for Computational Linguistics, 2020. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Jinyang Wu, Shuai Zhang, Feihu Che, Mingkuan Feng, Pengpeng Shao, and Jianhua Tao. Pandora's box or aladdin's lamp: A comprehensive analysis revealing the role of RAG noise in large language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5019–5039, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.250. URL https://aclanthology.org/2025.acl-long.250/.

Siye Wu, Jian Xie, Jiangjie Chen, Tinghui Zhu, Kai Zhang, and Yanghua Xiao. How easily do irrelevant inputs skew the responses of large language models? In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=S7NVVfuRv8.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, Madian Khabsa, Han Fang, Yashar Mehdad, Sharan Narang, Kshitiz Malik, Angela Fan, Shruti Bhosale, Sergey Edunov, Mike Lewis, Sinong Wang, and Hao Ma. Effective long-context scaling of

foundation models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4643–4663, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.260. URL https://aclanthology.org/2024.naacl-long.260/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Minglai Yang, Ethan Huang, Liang Zhang, Mihai Surdeanu, William Yang Wang, and Liangming Pan. How is LLM reasoning distracted by irrelevant context? an analysis using a controlled benchmark. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 13329–13347, Suzhou, China, November 2025b. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.674. URL https://aclanthology.org/2025.emnlp-main.674/.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of EMNLP*, 2018.

Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. Helmet: How to evaluate long-context language models effectively and thoroughly. In *International Conference on Learning Representations (ICLR)*, 2025a.

Howard Yen, Ashwin Paranjape, Mengzhou Xia, Thejas Venkatesh, Jack Hessel, Danqi Chen, and Yuhao Zhang. Lost in the maze: Overcoming context limitations in long-horizon agentic search. *arXiv preprint arXiv:2510.18939*, 2025b.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. Making retrieval-augmented language models robust to irrelevant context. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ZS4m74kZpH.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Honglin Yu, Weinan Dai, Yuxuan Song, Xiangpeng Wei, Haodong Zhou, Jingjing Liu, Wei Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Yue Yu, Wei Ping, Zihan Liu, Boxin Wang, Jiaxuan You, Chao Zhang, Mohammad Shoeybi, and Bryan Catanzaro. RankRAG: Unifying context ranking with retrieval-augmented generation in LLMs. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, Meishan Zhang, Wenjie Li, and Min Zhang. mGTE: Generalized long-context text representation and reranking models for multilingual text retrieval. In *Proceedings of EMNLP: Industry Track*, 2024a.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. ∞Bench: Extending long context evaluation beyond 100K tokens. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar

(eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15262–15277, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. URL https://aclanthology.org/2024.acl-long.814.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. DeepResearcher: Scaling deep research via reinforcement learning in real-world environments. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 414–431, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.22. URL https://aclanthology.org/2025.emnlp-main.22/.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents, 2025. URL https://arxiv.org/abs/2506.15841.

A Lost in Thought: Additional Details

A.1 Evaluated Models

We evaluate six different models on this benchmark, loosely grouped into two families: a Llama (Dubey et al., 2024) family containing:

- **Llama-3.1-8B-Instruct**: a very popular, open-weights instruction-tuned model, noted for its strong instruction-following and long-context abilities.
- **R1-Distill-Llama-8B**: a distillation of the DeepSeek-R1 reasoning model (Guo et al., 2025) into Llama-3.1-8B, which has strong reasoning abilities.
- **ProLong-8B-Instruct-512k**: an open-source model produced by further training Llama-3.1-8B on an additional high-quality set of long context data at context lengths of up to 512,000 tokens (Gao et al., 2025)

and a Qwen family containing

- **Qwen2.5-7B-Instruct**: another very popular, open-weights instruction-tuned model, commonly used as a base model for RL training (Qwen Team, 2025; Jin et al., 2025; Wang et al., 2026).
- **Qwen3-8B**: a newer version of Qwen2.5-7B, with a variable thinking mechanism, allowing the model to dynamically decide to reason or not before answering (Yang et al., 2025a). We evaluate both the thinking and non-thinking modes.

A.2 Task Examples

Figure 5 shows example prompts for the Retrieval and Reasoning-Retrieval tasks, illustrating two of the augmentation axes applied across evaluation and training: dictionary format (line-delimited vs. JSON) and query placement (before vs. after the dictionary). Dictionaries contain 200+ entries; most rows are omitted for space.

A.3 Injection Analysis

To isolate the source of the lost-in-thought degradation, we conduct a follow-up experiment that measures how often each model can generate the correct value after reasoning. For each model and context length, we sample 200 Reasoning-Retrieval examples where the model solved the math problem correctly, and use Gemma-3-27B-IT (Gemma Team, 2025) to identify the point in each reasoning trace where the model first attempts to look up the value in the dictionary. At that point, we truncate the model's generation and inject a short structured prompt that restates the correct key and provides the exact lexical prefix of the

```
Extract the value corresponding to
the specified key in the list below.

Your key: 1837

Key -3328:
LdO2I44vGR

Key -1414:
Xj1CgB4wcX

Key -8732:
TVMSNS2yYM

...

Key 1837:
AEgkZzAgQw

...

Key -4652:
mbeNIFEUjD

Key -326:
hiRyxyAyCV

Key 4030:
uewojpA4Ck
```

(a) Retrieval, line-delimited format, key at the start

```
Your task is to retrieve a value from
a large JSON dictionary called
`retrieval_json`. The key you need to
use to index the dictionary is given
to you as a math problem. Solve that
math problem, then index the
dictionary based on the result.

Do your reasoning step-by-step. On
the final line, output only:
Answer: <the value>

retrieval_json:
{
  "-242": "RNCBDQ6HJu",
  "21": "psifnr9S0k",
  "57": "oElKiFhMLP",
  "-325": "vAXyMuaoG6",
  "-285": "Q5tPt9hXyu",
...
  "40": "KGcxVBxpmK",
...
  "505": "pIXwA5KWlo",
  "-335": "QISDYD1wh4",
  "414": "uhB3PlOfag",
  "-171": "p4GrhtF5t6",
  "-240": "Gza7gy92dj"
}

Now, compute the answer to the math
problem, and then output the value of
`retrieval_json[answer]`.

Math problem:
Solve for $x$ in
$$(7 - 5x) - 2(x + 3) + 1(x + 6)
+ (8x + 5) = x + 52$$
```

(b) Reasoning-Retrieval, JSON format, question at the end

Figure 5: Example prompts for the Retrieval and Reasoning-Retrieval tasks, showing two augmentation axes. (a) uses line-delimited format with the query key stated before the dictionary. (b) uses JSON format with the math problem posed after the dictionary.

corresponding entry as it appears in the context (Appendix A.4). The model then continues generating from this injected prefix, so that it only needs to copy the remaining value tokens from context. This controls for reasoning errors, the model losing track of the correct key, and difficulty bridging from the model's natural-language reference to the exact lexical format of the dictionary entry; any remaining gap with direct Retrieval reflects the model's inability to faithfully reproduce in-context information after reasoning.

As shown in Figure 2 and Table 5, models frequently fail to generate the correct value even under these favorable conditions. At short contexts, injected accuracy modestly exceeds Reasoning-Retrieval but remains far below direct Retrieval, e.g., Llama-3.1-8B-Instruct generates the correct value 40.7% of the time, against 80.6% on direct Retrieval. At long contexts, the recovery is smaller still.

A.4 Prompt Injection Example

Figure 6 shows a concrete example of the prompt injection analysis from Section 2, using Qwen3-8B (Thinking) on a Reasoning-Retrieval example at 4K context length. The model correctly solves the math problem ($x = 2880$) in both conditions. In the original

Table 5: *Lost in thought* and injection analysis. **S**hort = 4K–32K average; **L**ong = 64K–128K average. **Reas.-Retr.** and **Retrieval** report accuracy on the Reasoning-Retrieval and Retrieval datasets respectively; **Rel. Gap** is the relative difference between them. **Injected** reports accuracy after injecting the correct key and its exact lexical prefix mid-generation, showing that even with the exact prefix provided, retrieval recovers only partially. **Cov.** is the fraction of correct-reasoning examples where the model attempts retrieval; **Reas. Toks** is the mean extra reasoning tokens compared to the Retrieval baseline.

	Reas.-Retr.		Injected		Retrieval		Rel. Gap (%)		Cov.		Reas. Toks	
Model	**S**	**L**	**S**	**L**	**S**	**L**	**S**	**L**	**S**	**L**	**S**	**L**
Llama-3.1-8B-Instruct	27.1	2.2	40.7	7.9	80.6	25.5	66.3	91.4	95.8	60.7	1264	866
R1-Distill-Llama-8B	32.0	2.0	32.8	5.7	61.1	15.8	47.7	87.2	41.0	25.5	−1973	−823
ProLong-8B-512K	7.7	0.0	22.2	9.2	79.5	30.3	90.3	100.0	65.9	37.1	91	184
Qwen2.5-7B-Instruct	33.0	11.9	37.7	16.5	67.1	33.8	50.9	64.7	97.0	80.3	828	526
Qwen3-8B (No Think)	47.6	8.8	53.2	16.2	82.2	43.2	42.2	79.7	97.0	99.0	236	230
Qwen3-8B (Thinking)	53.4	14.9	52.5	17.9	72.9	37.0	26.8	59.6	99.6	98.0	393	459

completion (a), the model enumerates over 250 keys from memory before hallucinating the value `aNdVypRo8F`. In the injected completion (b), the model's generation is truncated after it determines the correct key, and a short prompt (highlighted) restates the key and provides the exact lexical prefix of the CSV entry as it appears in the context (`2880,`). Despite being given the exact prefix, the model still hallucinates the same incorrect value. The correct value is `oWIO76jOH4`.

B RecaLLM: Additional Details

B.1 Efficient Implementation and Complexity

A key property of constrained decoding is that the valid continuation set $\mathcal{A}(c, r_{1:k})$ depends only on token IDs in the searchable context and the current recalled prefix, not on model activations. The mask computation is therefore **entirely decoupled from the forward pass**: we compute the allowable-token mask on a separate CPU thread while the GPU executes the forward pass, then asynchronously transfer the mask to device memory via a dedicated CUDA stream. The only synchronous operation is a single `masked_fill` on the logit tensor. Because the mask computation is fully overlapped with the forward pass, constrained decoding adds **negligible overhead** to generation latency, both inside and outside of recall spans.

Complexity. A naive implementation would rescan the full searchable context at every decoding step to recompute $\mathcal{A}(c, r_{1:k})$. If the searchable context has length M and the recall span has length L, this yields $O(Mk)$ work per step and $O(ML^2)$ total over the span. We avoid this cost through incremental prefix matching: at the start of a recall span, every position in c is a candidate; as each token is generated, we retain only positions consistent with the growing prefix and read off the allowable next tokens from the survivors. Letting S_k denote the candidate set after k recalled tokens, the total work is

$$O\left(M + \sum_{k=1}^{L-1} |S_k|\right),$$

with a worst case of $O(ML)$ since the candidate set can only shrink. In natural text, the number of surviving positions typically collapses after a few matched tokens, so the realized cost is much closer to a single scan than to repeated rescans.

C RecaLLM Training: Additional Details

C.1 SFT Training Procedure

We first initialize base models with 4 new token embeddings, corresponding to the two new recall tokens, <|start_recall|>, <|end_recall|>, as well as the thinking tokens <think> and </think>. Following Hewitt (2021), we initialize each new embedding as the mean of all existing embeddings.

As the base models are already well-trained on a broad set of tasks (Dubey et al., 2024; Qwen Team, 2025), we aim to minimally adjust their existing parameters. We train in two stages: a longer **embedding learning** stage, where the majority of learning occurs in the new token embeddings, followed by a shorter **full finetuning** stage that lightly adapts the model to incorporate the new tokens. The goal of this approach is to teach the model to use recall tokens naturally while avoiding catastrophic forgetting, concentrating most of the learning in the new token embeddings rather than the pretrained parameters.

In the first stage, we freeze all of the parameters in the model, except for the embedding vectors of the 4 new tokens. We train for 5 epochs using a learning rate of 5.0×10^{-5}, at which point validation loss no longer improves. In the second stage, we unfreeze all parameters and train for a single epoch using a smaller learning rate of 2.0×10^{-6}.

C.2 SFT Annotation

To construct the SFT dataset, we prompt GPT-5.2 (OpenAI, 2025b) to rewrite teacher-produced reasoning traces so that references to context information are realized as verbatim recall spans delimited by multi-token XML-style tags, <recall> and </recall>, rather than the actual single-token delimiters <|start_recall|> and <|end_recall|>; the annotator model does not have these special tokens in its vocabulary, so we use a distinct, more familiar, textual representation. The annotator receives the question, task instructions, and gold documents alongside the original reasoning trace. Although the prompt template includes a field for negative documents (Figure 8), we found in practice that including negatives caused the annotator to over-insert recall spans grounding irrelevant context, so we leave this field empty. We evaluated several annotator models, including GPT-5.2-mini and Qwen2.5-72B under various reasoning configurations, and anecdotally found that GPT-5.2 produced the highest-quality annotations. After annotation, we align each annotator-produced recall span to its corresponding source text in the context using Levenshtein-distance-based fuzzy string matching (Einat, 2013), and discard examples for which this alignment fails. This yields 1,795 annotated examples with verbatim recall spans ready to be trained on, which we split into 1,600 training and 195 validation examples. Figure 7 shows the full system prompt, Figure 8 shows the user prompt template, and Figure 9 shows a concrete before/after example.

```
Your task is to retrieve a value from a large lookup table called `data_table`. The key
you need to find the correct row is given to you as a math problem. Solve that math
problem, then look up the row with that key in the table.

Do your reasoning step-by-step. On the final line, output only:
Answer: <the value>

data_table:
key,value
3349,lVLAn6DJlz
3170,QJljlGyqCO
2887,B0Fd2C4HOz
...
2880,oWIO76jOH4
3254,jEhEXbwDnR
...
3091,R5Y4904bP8
3202,apMTjKmaUr

Now, compute the answer to the math problem, and then output the value from the row in
`data_table` where the key matches your answer.

Math problem:
Solve for $x$: $$x = \frac{64}{1} + 64 \cdot 44$$
```

Prompt (200+ CSV rows omitted; key `2880` *maps to value* `oWIO76jOH4`*)*

```
<think>
...math reasoning...
Adding them gives 2560 + 256 = 2816.
Then adding 64 gives
2816 + 64 = 2880. So x = 2880.

Now, I need to look up the key 2880
in the data_table. Let me scan
through the keys:

3349, 3170, 2887, 2675, 2691,
3245, 2416, 3214, 2584, 3151,
2717, 2522, 2388, 3114, 3345,
         ...
  [enumerates 250+ keys from memory]
         ...
3009, 3126, 3228, 3091, 3202.

Wait, I see key 2880 here. The row
with key 2880 has the value
"aNdVypRo8F".
</think>

Answer: aNdVypRo8F
```

(a) Original completion — hallucinates value

```
<think>
...math reasoning [identical to (a)]...
Adding them gives 2560 + 256 = 2816.
Then adding 64 gives
2816 + 64 = 2880. So x = 2880.

Ok, so the value of x is 2880.
Now, I need to find the row where
the key equals 2880 in the
data_table.

Looking back at the data_table in
my context, I need to find the key
2880. Looking through the rows,
I see:

2880, aNdVypRo8F

Yes, there it is. The key 2880
corresponds to the value
aNdVypRo8F.
</think>

Answer: aNdVypRo8F
```

(b) Injected completion — still hallucinates value

Figure 6: Prompt injection example using Qwen3-8B (Thinking) on a 4K-token Reasoning-Retrieval task. *Top:* the input prompt (200+ rows omitted; the target entry is `2880,oWIO76jOH4`). *Bottom left:* the model's original completion, in which it noisily enumerates keys from memory and hallucinates the value. *Bottom right:* the injected completion, where the model's generation is truncated and a structured prompt (highlighted in yellow) provides the correct key and its exact lexical prefix from the context. Despite injection, the model produces the same incorrect value, demonstrating that the bottleneck is faithful copying from context rather than identifying what to retrieve.

```
You are a data-annotation editor for reasoning traces, produced by LLMs on
long-context retrieval + reasoning tasks.  You are annotating reasoning traces
by modifying them, so they rely on special <recall> ... </recall> spans for key
information that comes from their context.  These spans are produced by a
special recall tool the model has access to.

Given RECALLABLE SOURCES and a REASONING TRACE, rewrite the trace so that
whenever the trace uses information from the sources, it should first recall the
relevant text via <recall>...</recall> (inside <think>...</think>), then draw
the conclusion. You should reorder and rewrite sentences to achieve this.

Keep the original reasoning trace's structure, length, and style as much as
possible. Only make the minimum edits needed to satisfy the recall/evidence
rules. Do not summarize, compress, or remove backtracking, verification, or
"thinking aloud" unless it is pure browsing-playacting or redundant repeated
listings.  Do not remove the post-think-block text.

Given RECALLABLE SOURCES and a REASONING TRACE, return ONLY the ENTIRE edited
trace.

RECALLABLE SOURCES include: QUESTION, INSTRUCTIONS, GOLD DOCUMENTS, and
NEGATIVE DOCUMENTS.

Key idea (natural inner-monologue style):
Treat <think>...</think> as a human's private inner monologue while reading the
context.
Recall spans should feel like quick glances at the text in front of them, e.g.:
"Looking in the text, I see <recall>...</recall>."
Do NOT narrate tool usage (avoid "I will use the recall tool" / "I am calling
recall").
Modify the reasoning trace so it adheres to this key idea.

Guidelines:
1) Recall tool-use: Anytime the reasoning trace relies on information from one
of the RECALLABLE SOURCES, modify it so that it naturally uses the recall tool
to recall the evidence inside a <recall> span.  The reasoning trace should use
the recall tool frequently, but only to recall key information from the context.

2) Evidence-first: Do not introduce a document-derived fact in free text and
then cite it later. If a sentence contains doc-derived facts (names, numbers,
dates, IDs, key/value pairs, definitions, titles), rewrite locally so the
<recall> appears before or inside the first introduction of that fact.

3) Prune only fake browsing: Delete or compress only tool-playacting or
contentless scanning (e.g., "I scan the docs/keys," "I look around,"). Do not
remove genuine reasoning steps: backtracking, verification, elimination,
uncertainty, or hypothesis testing.  Do not reorder when not necessary.

4) Contiguity-first (few spans): Prefer fewer, longer <recall> spans that
capture an entire supporting sentence/clause. If a fact is contained within one
contiguous sentence/clause in the sources, wrap that whole sentence/clause in
one <recall> span (even if it includes extra parenthetical text). Avoid
splitting one claim across multiple <recall> spans just to be shorter.

5) Supported claims: All claims that rely on information from the RECALLABLE
SOURCES should come after evidentiary <recall> spans.

6) Repeat-when-reused (still contiguous): When the trace repeats a doc-specific
string later (IDs, numbers, titles, rare names, key:value), wrap that repeated
string again in a new <recall> span near each reuse. If the string appears
inside a natural source sentence/clause, prefer recalling that whole clause
again rather than splitting

7) Questions and Instructions: When the reasoning trace refers to the question
or instructions, then modify it so it uses the recall tool to quote the
```

```
relevant part of the question or instructions.  The question and instructions
are key information.

Constraints:
- <recall> spans must appear only inside <think>.
- Text inside <recall> must be an exact contiguous substring from the RECALLABLE
  SOURCES (verbatim punctuation/casing/spacing).  DO NOT paraphrase document
  facts inside <recall>.
- It is ok to insert recall spans that are not quite gramatically correct, if it
  means they contain the supporting information verbatim.
- Keep recall snippets focused and clearly tied to the subject/topic.
- Each recall span must be specific enough to clearly identify the
  subject/topic.  If the recalled span could refer to a different context, then
  it is not helpful
  - Good: <recall>5065: 9BJk8Q32AL</recall>
  - Bad: 5065: <recall>9BJk8Q32AL</recall>  (not uniquely tied)
  - Also bad: <recall>5065:</recall> <recall>9BJk8Q32AL</recall>
- Delete any sentences that discuss using the recall tool; the trace should just
  use recall as naturally as possible.
- Do not add <recall> spans outside of the <think> ... </think> block.
- Do not change the final answer.

Output only the edited trace; no extra commentary.
```

Figure 7: System prompt used for GPT-5.2 annotation of SFT reasoning traces. The annotator rewrites teacher-produced reasoning traces so that references to context information are realized as verbatim <recall>...</recall> spans, following a natural inner-monologue style.

```
You are editing a model's reasoning trace.

RECALLABLE SOURCES:
QUESTION:
{{question}}

INSTRUCTIONS:
{{instructions}}

GOLD DOCUMENTS:
{{gold_docs}}

NEGATIVE DOCUMENTS (optional; may be empty):
{{neg_docs}}

REASONING TRACE:
{{trace}}
```

Figure 8: User prompt template for GPT-5.2 annotation. Placeholder variables are filled with the task-specific question, instructions, gold and negative documents, and the original teacher reasoning trace.

C.3 RL Training Details

All SFT experiments are trained on two A100 GPUs using DeepSpeed ZeRO-2 (Rajbhandari et al., 2020). RL experiments are trained on four A100 GPUs using VeRL (Sheng et al., 2024) with PyTorch (Paszke, 2019) FSDP. We use a max generation length of 4,096 tokens with an overlong buffer of 1,024 tokens. RL training uses GRPO with 16 rollouts per prompt and a training batch size of 128 prompts (2,048 sampled responses per update step). We use the AdamW optimizer with an actor learning rate of 2×10^{-6}, cosine decay schedule (minimum LR ratio 0.1, no warmup), and Adam betas $(0.9, 0.999)$. Gradient clipping is set to 1.0. Rollouts are sampled with temperature 1.0. We use a PPO clip ratio of 0.2 with a KL

```
<think>
Okay, let's see. The question is asking
whether Marc Forster and Drew Daywalt
share the same nationality. I need to
check the documents provided for
information about each of them.

First, looking for Marc Forster. There's
a document titled "Marc Forster" which
states he's a German-born filmmaker. So
his nationality is German.

Next, Drew Daywalt. The document on him
says he's an American filmmaker and
author. So his nationality is American.

Comparing the two, German vs. American.
They are different nationalities.
Therefore, the answer should be no.
</think>

The document on Marc Forster states he is
German-born, while the document on Drew
Daywalt confirms he is American. Since
their nationalities are different, they
are not of the same nationality.

Answer: No.
```

(a) Original reasoning trace

```
<think>
Okay, let's see. The question is asking
whether Marc Forster and Drew Daywalt
share the same nationality. I need to
check the documents provided for
information about each of them.

First, looking for Marc Forster. There's
a document titled "Marc Forster" which
states <recall>Marc Forster (born 30
November 1969) is a German-born
filmmaker</recall>. So his nationality
is German.

Next, Drew Daywalt. The document on him
says <recall>Drew Daywalt (born
January 5, 1970) is an American
filmmaker and author</recall>. So his
nationality is American.

Comparing the two, German vs. American.
They are different nationalities.
Therefore, the answer should be no.
</think>

The document on Marc Forster states he is
German-born, while the document on Drew
Daywalt confirms he is American. Since
their nationalities are different, they
are not of the same nationality.

Answer: No.
```

(b) Annotated with recall spans

Figure 9: Before/after example of SFT annotation on a multi-hop QA trace. *Left:* the original teacher completion, which refers to document-derived facts in free text. *Right:* the annotated version, where document-derived facts are grounded via `<recall>` spans (highlighted in blue) containing verbatim substrings from the gold documents. The reasoning structure, style, and final answer are preserved; only the evidence grounding is added.

penalty coefficient of 0.001, a PPO minibatch size of 128, and a microbatch size of 2 per GPU. We train for 150 steps for Qwen2.5-7B and 60 steps for Llama-3.1-8B, as the latter converges faster.

RecaLLM requires substantially less compute than comparable long-context RL methods. LoongRL (Wang et al., 2026) trains across three stages totaling 328 steps with a batch size of 512 and 8 rollouts per prompt at 16K context length on 16 A100 GPUs. QwenLong-L1 (Wan et al., 2025) uses 32 A100 GPUs with curriculum-guided phased RL, scaling context lengths up to 60K tokens over multiple training stages. In contrast, RecaLLM trains on 20K examples at 8–10K context with 16 rollouts on 4 A100 GPUs for a single epoch (150 steps for Qwen, 60 for Llama). Since attention cost is quadratic in sequence length, training at 8–10K tokens rather than 16K (LoongRL) or up to 60K (QwenLong-L1) dramatically reduces per-step compute, more than offsetting the use of twice as many rollouts. Combined with fewer GPUs (4 vs. 16–32), far fewer total rollouts ($\sim$307K vs. $\sim$1.3M for LoongRL), and a single training stage, this results in a fraction of the total training cost while achieving competitive or superior performance.

Category	Answer Reward	Recall Reward	# Gold	τ	Free Spans
Multi-Hop QA	subEM	Standard F1	2–4	0.4	4
Single-Hop QA	subEM	Top-1 F1	1	0.4	2
Retrieval	subEM (KV) / Net Recall (NIAH)	Standard F1	1–6	0.9	2 (KV) / 6 (NIAH)
Reasoning Retr.	subEM	Standard F1	1	0.9	2
Short-Ctx Math	Exact match	Always 1.0	0	—	2
In-Context Learn.	Exact match	Top-2 F1	4–12	0.95	2
Long-Doc QA	Exact match	Binary recall	0	—	4
Aggregation	Net Recall	Always 1.0	0	—	2
Reranking	NDCG@10	Top-2 F1	1–26	0.7	4
Entity Citation	Citation Top-5 F1 + coverage	Top-5 F1	5–20	0.7	4

Table 6: RL reward configuration per category. Free spans indicates the task-dependent number of initial recall spans exempt from the density penalty.

C.4 RL Dataset and Augmentation Details

Table 7 summarizes each training dataset and its augmentations. The mixture is designed so that each category requires a qualitatively different recall strategy. Single-hop QA (NQ, TriviaQA) (Kwiatkowski et al., 2019; Joshi et al., 2017) and multi-hop QA (HotpotQA, MuSiQue, 2WikiMQA) (Yang et al., 2018; Trivedi et al., 2022; Ho et al., 2020) use nearly identical contexts but differ in how many supporting passages contain relevant information, requiring the model to dynamically determine how much retrieval is needed. Retrieval tasks include KV retrieval, which places a target entry among many structured distractors, and Multi-NIAH (Kamradt, 2023; Martin, 2024), which embeds a small number of needles in a large body of unstructured text, requiring identification of relevant spans without structural cues. Reasoning retrieval (Math Retrieval) combines math problem-solving with key-value lookup, directly targeting the lost-in-thought setting (Section 2). Reranking (MSMARCO v2) (Bajaj et al., 2016) requires a ranked list of document identifiers, but recalling all passages would exhaust the generation budget, so the model must selectively recall only the most informative ones. Entity citation (QAMPARI) (Amouyal et al., 2023) similarly requires selective recall across multiple answer entities and their supporting passages. In-context learning (Banking77, MASSIVE) (Casanueva et al., 2020; FitzGerald et al., 2023) requires the model to find demonstrations semantically similar to the query and use their labels, a form of retrieval over semantic similarity rather than lexical matching. Long-document QA (QuALITY) (Pang et al., 2022) tests recall over a single long-form article rather than a collection of passages. Aggregation tasks (Majority Vote, Top-N Vote) serve as controlled negative examples: they require in-context information to estimate vote frequencies, but the number of individual votes is far too large for recall to be practical. Short-context math (DAPO Math, MCQA Math) (Yu et al., 2025; Biderman, 2025) similarly requires no retrieval, reinforcing that recall should only be invoked when beneficial.

Shared Augmentations. Three augmentations are applied across all long-context datasets (i.e., all datasets except the short-context math tasks). First, *instruction template* variation: each dataset draws from a pool of paraphrased task instructions, preventing the model from associating recall behavior with specific surface phrasings. Second, *question position* variation: the question is placed at the end, the beginning, or both ends of the prompt, ensuring the model does not rely on a fixed prompt structure to decide when retrieval is needed. Third, *gold position* randomization: for any dataset with known gold information (gold passages, needles, or key-value pairs), the position of that information within the context is sampled uniformly, preventing the model from learning positional biases.

Per-Dataset Augmentations. Beyond the shared augmentations, each dataset applies task-specific variation along multiple axes to maximize context diversity. For document-based tasks (multi-hop QA, single-hop QA, reranking, entity citation), we vary the document format and the source of negative (distractor) documents, drawing from BM25 negatives,

Table 7: Training dataset descriptions and per-dataset augmentations. All long-context datasets share three base augmentations: *instruction template* variation, *question position* variation, and *gold position* randomization for any dataset with known gold information. Additional per-dataset augmentations are listed in the rightmost column.

Datasets	Description	Additional Augmentations
HotpotQA, MuSiQue, 2WikiMQA	Multi-hop QA requiring 2–4 supporting passages	Document format, negative source, passage type (paragraphs vs. fixed-window chunks)
NQ, TriviaQA	Single-hop factoid QA with one gold passage	Document format, negative source
KV Retrieval	Structured key-value lookup	KV format (CSV / JSON / lines)
Multi-NIAH	Multi-needle-in-a-haystack retrieval among distractors	Distractor count, target keys, values per key, value type
Math Retrieval	Solve a math problem to determine a retrieval key, then look up its value	KV format (CSV / JSON / lines), math problem type
DAPO Math, MCQA Math	Short, difficult math problems with no retrieval required	None
Banking77, MASSIVE	Many-shot intent classification via in-context demonstrations	Label format (random numeric vs. descriptive text), demonstration format
QuALITY	Multiple-choice QA over long-form literary articles	Choice format
Majority Vote, Top-N Vote	Frequency estimation via vote tallying over structured candidate lists	Vote margin, candidate count, candidate category (names / places / letters / numbers)
MSMARCO v2	Passage relevance ranking	Document format, negative source (BM25 / random / judged)
QAMPARI	Multi-answer QA requiring inline passage citations	Document format, negative source (BM25 / random / judged), number of gold entities

random negatives, and dense retrieval hard negatives via GTE-ModernBERT-Base (Zhang et al., 2024a), or judged negatives where available (MS MARCO (Bajaj et al., 2016), QAMPARI), with either a single source or a per-example mixture. Entity citation (QAMPARI) additionally varies the number of gold entities per example. Multi-hop QA varies the passage type, drawing distractors from either the native paragraph-level corpora (Yang et al., 2018; Trivedi et al., 2022; Ho et al., 2020) or a fixed-window chunked Wikipedia corpus from the KILT knowledge source (Petroni et al., 2021), exposing the model to both natural document boundaries and uniform passage formats. For KV retrieval and reasoning retrieval, we vary the key-value store format (CSV, JSON, or line-delimited), following the same variation used in Section 2. Multi-NIAH independently varies the distractor count, number of target keys, values per key, and value type per example. Math retrieval additionally varies the math problem type. In-context learning tasks vary the label format (random numeric vs. descriptive text) and the demonstration layout. QuALITY varies the multiple-choice formatting. Aggregation tasks vary the vote margin, candidate count, and candidate category (names, places, letters, or numbers).

C.5 In-Context Retrieval Reward: Formal Definition

This section provides formal definitions for the retrieval reward R_{ret} described in Section 4.

Character-level F1 overlap. Let $\mathcal{G} = \{g_1, \ldots, g_n\}$ denote the set of gold passages and $\mathcal{S}$ the set of recall spans extracted from the model's completion. Since constrained decoding guarantees that both gold passages and recall spans are contiguous substrings of the input context, each corresponds to a character interval: gold passage g_i spans $[g_i^s, g_i^e)$ and recall

span s spans $[s^s, s^e)$. We define the character-level F1 overlap between g_i and s as:

$$\mathrm{F1}_{\mathrm{char}}(g_i, s) = \frac{2 \cdot \max\big(0,\ \min(g_i^e, s^e) - \max(g_i^s, s^s)\big)}{(g_i^e - g_i^s) + (s^e - s^s)} \tag{4}$$

This is the harmonic mean of precision and recall over the character-level interval intersection. Unlike extractive QA F1 (Rajpurkar et al., 2016), which operates over bags of tokens, this metric requires contiguous overlap: even a single non-verbatim character breaks the interval match.

Per-passage overlap score. For each gold passage g_i, we retain only its highest-overlap recalled span and normalize by a task-specific hit threshold τ:

$$\mathrm{overlap}(g_i) = \frac{\min\big(\max_{s \in \mathcal{S}} \mathrm{F1}_{\mathrm{char}}(g_i, s),\ \tau\big)}{\tau} \tag{5}$$

Capping at τ prevents the reward from favoring exhaustive copying: once a recalled span overlaps sufficiently with a gold passage, additional copied text yields no further reward. This also reflects the fact that for many tasks only a small portion of the gold passage is relevant to the question. Per-category τ values are reported in Table 6.

Density penalty. To prevent pathological over-recall, we apply an exponential density penalty based on the rate of recall spans per unit of generated text. Let N_s denote the total number of recall spans in the completion, N_t the total number of generated tokens, and N_{free} a task-dependent number of initial spans that are exempt from the penalty (Table 6). The value of N_{free} reflects the different retrieval demands of each task: for instance, reasoning-retrieval requires only a single key lookup ($N_{\mathrm{free}} = 2$), whereas multi-hop QA may need multiple supporting passages ($N_{\mathrm{free}} = 4$). We compute the effective density as the number of excess spans per 1,024 generated tokens:

$$d = \frac{N_s - N_{\mathrm{free}}}{N_t \ / \ 1024} \tag{6}$$

The density penalty is then:

$$P_{\mathrm{density}} = \left(\tfrac{1}{2}\right)^{\max(0,\, d - \delta) \ / \ h} \tag{7}$$

where $\delta = 4$ is the density threshold and $h = 4$ is the half-life. The first N_{free} spans and up to δ excess spans per 1K tokens incur no penalty; beyond that, the reward halves for every h additional units of excess density.

Correctness penalty. To catch degenerate recall strategies, we apply a correctness penalty that detects malformed recall spans. Let N_{short} denote the number of recall spans shorter than 5 characters, and $N_{\mathrm{mismatch}} = \big|\mathrm{count}(R_{\mathrm{start}}) - \mathrm{count}(R_{\mathrm{end}})\big|$ the absolute difference between the number of start and end recall tokens, which detects nesting or unpaired delimiter tokens. The correctness penalty is:

$$P_{\mathrm{correct}} = 1 - \frac{N_{\mathrm{short}} + N_{\mathrm{mismatch}}}{\sqrt{N_s}} \tag{8}$$

The $\sqrt{N_s}$ denominator tolerates a sub-linear number of malformed spans as the total span count grows, so occasional short or mismatched spans in a long completion are not harshly penalized, while systematic abuse (e.g., emitting many trivial spans or consistently nesting recall tokens) drives the penalty toward zero.

Full retrieval reward. The retrieval reward combines the overlap scores with both penalties:

$$R_{\mathrm{ret}} = \overline{\mathrm{overlap}} \ \cdot \ P_{\mathrm{density}} \ \cdot \ P_{\mathrm{correct}} \tag{9}$$

where $\overline{\mathrm{overlap}}$ is the mean of $\mathrm{overlap}(g_i)$ over all gold passages (or the top-K highest-scoring gold passages, for tasks with many relevant documents such as reranking).

Model	1-Hop		N-Hop		Ret.		R→Ret.		Math	ICL		LQA	Agg.		Rerank		Ent. Cite	
	S	L	S	L	S	L	S	L	S	S	L	S	S	L	S	L	S	L
ProLong-8B-512K	21.7	5.8	16.6	4.7	61.5	20.7	1.8	0.0	19.0	1.6	0.1	37.3	72.4	52.5	6.3	1.0	2.0	1.4
Llama-3.1-8B-Instruct	59.9	10.3	50.2	10.8	81.5	29.5	13.3	1.2	37.8	24.2	4.4	**73.0**	60.1	45.8	24.9	5.6	4.1	0.3
RecaLLM-Llama-8B	**62.4**	**57.8**	**63.8**	**51.0**	**98.3**	**86.8**	**90.9**	**51.7**	**46.5**	**74.4**	**63.2**	72.5	**92.7**	**81.4**	**48.5**	**34.0**	**83.6**	**71.5**
Qwen2.5-7B-Instruct	53.6	33.8	42.0	28.5	73.6	47.1	23.0	7.5	55.8	31.3	18.7	71.0	45.8	45.5	22.5	10.9	5.2	1.4
RecaLLM-Qwen2.5-7B	**68.8**	**56.3**	**69.9**	**51.9**	**98.3**	**94.5**	**97.6**	**86.3**	**61.0**	**76.7**	**79.4**	**72.3**	**87.7**	**85.0**	**44.6**	**31.0**	**72.9**	**47.2**

Table 8: In-domain evaluation results (%) across training task categories at short (4K–32K) and long (64K–128K) context lengths. Categories with only one column are short-context only. **Bold** indicates best in each model family.

For tasks without segmented gold evidence, such as long-document QA, we set $\overline{\text{overlap}} = 1$ whenever the model produces at least one recall span. For tasks that do not require in-context retrieval, such as short-context math and aggregation, we set $\overline{\text{overlap}} = 1$ unconditionally. In both cases, the density and correctness penalties still apply, so R_{ret} can still be reduced by pathological recall behavior. The reward type for each task category is summarized in Table 6.

D Experimental Setup and Results: Additional Details

D.1 Evaluation Benchmarks

Evaluation Benchmarks. RULER is a largely-synthetic benchmark that isolates specific long-context capabilities, including tasks where recall tokens are naturally advantageous — such as Variable Tracking, which requires tracing chains of variable assignments through the context — as well as tasks where they may be disadvantageous, such as Common Words Extraction, which requires frequency estimation rather than verbatim retrieval. HELMET complements this with a diverse set of real-world tasks (summarization, QA, many-shot ICL, re-ranking, and more) where long-context understanding is required, testing whether recall tokens generalize beyond retrieval-centric settings.

Evaluation Hyperparameters. Following Wang et al. (2026), we use a max generation length of 8192 tokens for RULER and 10240 for HELMET and our in-domain datasets, with temperature 0.6 and top-p $= 0.95$.

D.2 In-Domain Results

Table 8 reports per-category results on validation splits of the training datasets, aggregated into short (4K–32K) and long (64K–128K) context buckets. Context length is varied by adjusting the number of distractor documents or passages. We use the same evaluation settings as HELMET: a max generation length of 10,240 tokens with temperature 0.6 and top-p $= 0.95$. Figure 3 in the main text shows the per-category scaling curves across individual context lengths.

D.3 Training Dynamics

Figure 10 shows the training dynamics of both RecaLLM models over 150 GRPO steps. RecaLLM-Llama converges rapidly, with overall score plateauing between steps 50 and 60; we select the step-60 checkpoint for evaluation. RecaLLM-Qwen converges more gradually and continues to improve through the full 150 steps. Training is stable throughout for both models, requiring no restarts or hyperparameter adjustments despite the breadth of the multi-task mix.

Two trends in recall behavior are particularly notable. First, the average number of recall spans per completion drops sharply in the early steps, from 20–40 down to 5–7, indicating that the models quickly learn to be selective rather than exhaustive in their in-context retrieval. Second, gold document overlap R_{ret} (Figure 10c) increases steadily alongside this

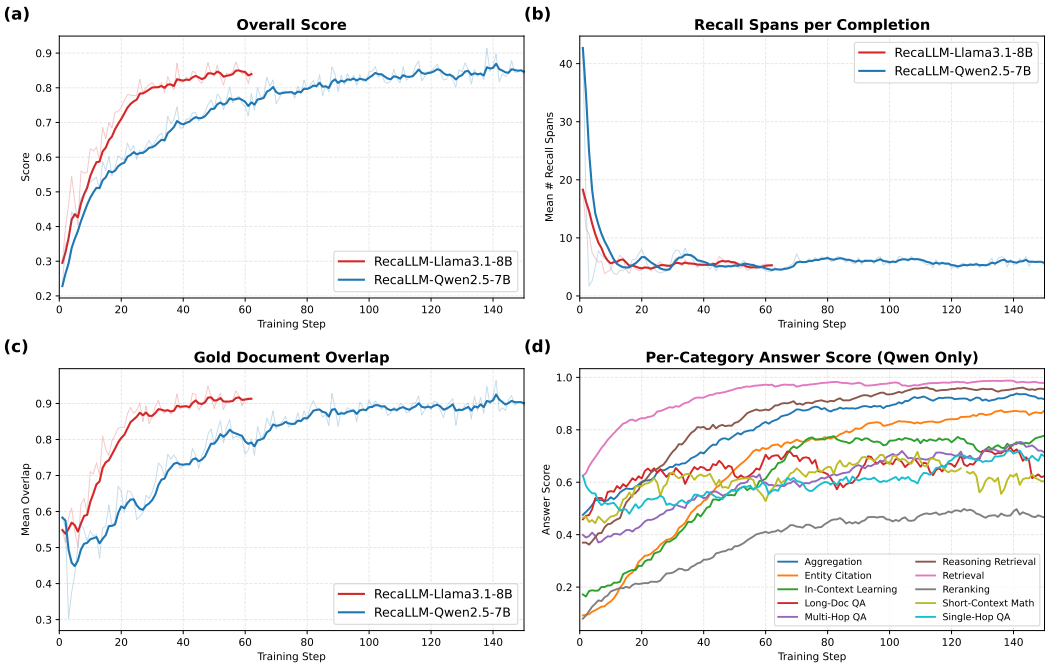

Figure 10: Training dynamics over 150 GRPO steps for both RecaLLM models. (a) Overall score rises rapidly, with Llama converging faster than Qwen. (b) Recall span usage starts very high, then decreases and stabilizes as the models learn selective retrieval. (c) Gold document overlap increases alongside score, indicating that models learn to recall relevant evidence rather than arbitrary context. (d) Per-category breakdown for Qwen, showing that all categories improve, albeit at different rates.

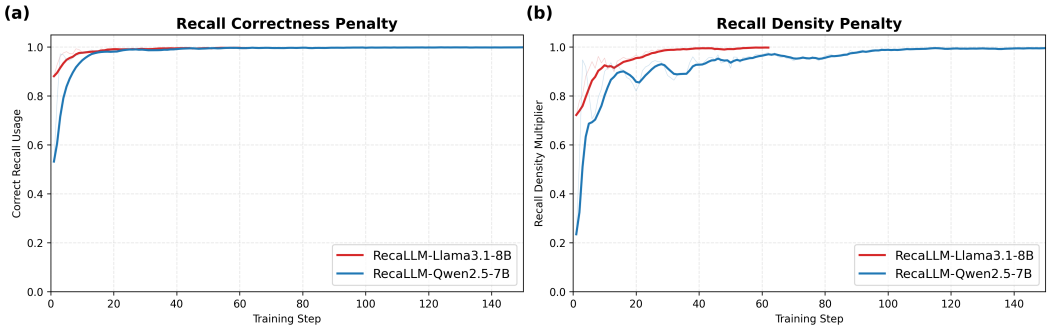

Figure 11: Correctness (a) and density (b) penalty values over GRPO training. Both saturate near 1 within the first ∼30 steps, acting as early-training stabilizers that play little role thereafter.

reduction, meaning the models recall *more relevant* evidence with *fewer* spans as training progresses. Consistent with these trends, Figure 11 plots the correctness and density penalty values (Section 4.3) over training: both saturate near 1 within the first ∼30 steps, confirming that the penalties act mainly as early-training stabilizers.

Despite training on 10 categories simultaneously, all categories improve over the course of training (Figure 10d). Tasks with more direct recall signals, namely retrieval, reasoning-retrieval, and aggregation, saturate quickly, while tasks requiring more complex reasoning over relevance, such as reranking, entity citation, and in-context learning, exhibit a slower initial rise followed by rapid improvement before plateauing. This staggered learning

pattern suggests that the simpler retrieval skills serve as a foundation for the more complex reasoning-retrieval behaviors, rather than competing with them for gradient signal.

E Analysis: Additional Details

E.1 Why Does RecaLLM-Llama Drift from Recall at Extreme Lengths?

Section 6.1 shows that RecaLLM-Llama's regression at 96K and 128K reflects policy drift away from invoking recall, rather than a failure of the recall mechanism once invoked. We offer two speculative hypotheses for why Llama drifts more than Qwen.

First, Qwen-family bases appear generally more receptive to RL post-training than Llama. Gandhi et al. (2025) attribute this to differences in baseline cognitive behaviors (verification, backtracking, subgoal-setting) that prime models for self-improvement under RL, and show that Llama matches Qwen when primed with reasoning data; Wang et al. (2025) show the gap can be substantially closed by a dedicated mid-training stage on math and chain-of-thought data. Either way, the pattern is not specific to long-context training or to recall, and the policy-generalization gap we observe is consistent with this broader trend.

Second, the two bases acquired their 128K context windows differently: Llama-3.1 is natively pretrained at 128K, whereas Qwen2.5 is pretrained at 32K and extended four-fold via YaRN (Peng et al., 2024). Llama's native long-context pretraining may have formed specialized attention patterns that our short-context RL (capped at 8–10K) does not completely update. Qwen's YaRN-stretched setup instead reuses the same attention machinery learned at 32K across all longer lengths, so the patterns updated by RL at 8–10K are likely the same ones the model relies on at 128K, making it more likely that the recall policy learned at short contexts carries over to the long-context regime.

Practically, RecaLLM yields consistent improvements over the base model across all evaluated context lengths and both model families on retrieval-intensive tasks; the Llama-specific regression occurs only at the largest lengths and manifests as a drop in recall usage rate. Neither LoongRL nor QwenLong-L1 evaluates Llama bases; to our knowledge, this is the first study of multi-task long-context RL post-training on Llama-3.1-8B, and policy retention at extreme lengths is a natural target for future work.

E.2 More Ablation Results on Validation Sets

Figure 12 shows that removing R_{ret} causes recall usage to collapse on ICL, reranking, entity citation, and aggregation. The only tasks where 'No Recall Reward' retains recall usage are retrieval and reasoning-retrieval, where the answer signal directly requires reproducing context content and the masking mechanism itself nudges the model toward recall. This demonstrates that R_{ret} is what teaches the model that explicit in-context retrieval is a broadly useful strategy, not just a retrieval-specific tool. This supervision signal is most effective with the explicit, constrained recall steps in RecaLLM.

'No Logit Masking' maintains high recall usage across nearly all tasks, confirming that R_{ret} successfully teaches the value of explicit retrieval regardless of whether decoding is constrained. However, the quality of that retrieval differs.

E.3 Ablation Results on Long-Context Benchmarks

We did not evaluate the ablation models on HELMET's LongQA and Summarization categories due to the cost of LLM-as-a-judge evaluation over long outputs. Table 4 in the main text reports results on RULER and the remaining HELMET categories. The patterns are broadly consistent with the in-domain findings in Section 6.2. On RULER, 'No Recall Reward' drops 4.9 points on average while No Mask loses only 0.8, confirming that gold document supervision is the more important training signal for retrieval-heavy synthetic tasks. On HELMET, 'No Recall Reward' again underperforms, most notably on Re-rank (25.3 vs. 46.2), while No Mask matches or exceeds the full model on RAG, ICL, and Cite, but drops substantially on Recall (87.7 vs. 96.2), reinforcing the role of constrained decoding for

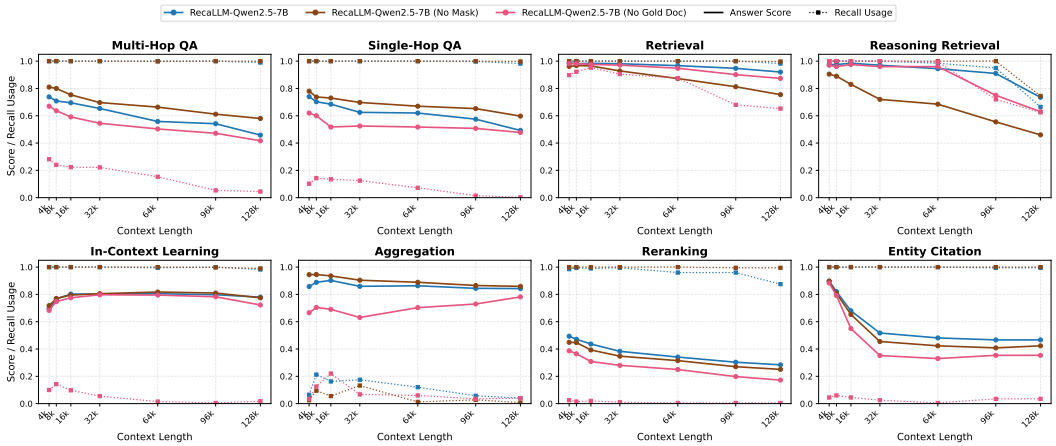

Figure 12: Per-category answer scores (solid) and recall usage rates (dotted) for RecaLLM-Qwen2.5-7B and two ablations across context lengths. **No Mask** follows the same training procedure but disables constrained decoding throughout, allowing recall spans to generate freely. **No Recall Reward** uses full constrained decoding but sets $R_{\text{ret}} = 1$ unconditionally during RL, removing gold document supervision.

faithful retrieval. Interestingly, constrained decoding drops slightly on HELMET Cite (13.7 vs. 15.4 for No Mask) despite helping on the in-domain entity citation task (61.8 vs. 58.1). This may partly reflect differences in evaluation: the in-domain task uses citation F1 over exact document identifiers, whereas HELMET's citation evaluation incorporates NLI-based assessment of whether cited passages support the generated claims (Gao et al., 2023), which may favor flexible evidence composition.

F Extended Related Works

F.1 Long-Context Utilization and Evaluation

Effectively utilizing long contexts remains a fundamental challenge. Liu et al. (2024) showed that LLM performance degrades when relevant information is in the middle of the context, and LongPiBench (Tian et al., 2025) extends this to the multi-piece setting, finding that the distance between relevant pieces introduces further biases across 32K to 256K tokens. RecaLLM's constrained decoding addresses positional bias directly: the logit mask selects valid continuations from anywhere in the searchable context regardless of position, and because recall spans serve as a reasoning aid rather than appearing in the final output, positional bias does not affect the faithfulness of recalled evidence. Long-context evaluation has evolved from Needle-in-a-Haystack (Kamradt, 2023) and its multi-needle extensions (Martin, 2024) to comprehensive benchmarks. RULER (Hsieh et al., 2024) generalizes NIAH into 13 synthetic tasks spanning retrieval, variable tracking, and aggregation. HELMET (Yen et al., 2025a) is deliberately broad, drawing tasks from LongBench (Bai et al., 2024), InfiniteBench (Zhang et al., 2024b), and other sources across seven application-driven categories, finding low cross-category correlation with synthetic benchmarks. We evaluate on both RULER and HELMET because they stress complementary failure modes: RULER tests precise retrieval and aggregation on controlled synthetic tasks, while HELMET tests whether improvements generalize to diverse, challenging real-world applications.

F.2 Improving Long-Context Capabilities

A large body of work targets long-context performance (Lu et al., 2025). One line of work focuses on context extension, increasing the effective context window of pretrained models. ProLong (Gao et al., 2025) continues training Llama-3-8B on a curated long-context data mix at sequence lengths up to 512K tokens, and YaRN (Peng et al., 2024) modifies rotary position

embeddings to efficiently extend context windows without full retraining. RecaLLM focuses on post-training and is agnostic to the context extension recipe; our post-trained RecaLLM-Qwen2.5-7B uses YaRN to extend its native context window four-fold.

Another line of work reduces context size through retrieval- or memory-augmented generation. Search-R1 (Jin et al., 2025) and R1-Searcher (Song et al., 2025) train models via RL to autonomously issue search queries during step-by-step reasoning, while WebThinker (Li et al., 2025b) and DeepResearcher (Zheng et al., 2025) extend this paradigm to real web environments. MEM1 (Zhou et al., 2025) takes a complementary approach, training agents to maintain a compact internal state that is rewritten at each turn, compressing long interaction histories into fixed-size memory to achieve constant memory usage across arbitrarily long interactions. Rather than issuing search queries, RA-DIT (Lin et al., 2024), Yoran et al. (2024), and RankRAG (Yu et al., 2024) instruction-tune LMs to make robust use of retrieved evidence, including irrelevant or distracting passages; retrieval remains a single upfront step over an external corpus rather than an action invoked during reasoning. EpMAN (Chaudhury et al., 2025) instead stores prior context in an episodic memory and attends selectively over it at decode time, reducing what must enter the context window. While these methods help manage context size, they are orthogonal to RecaLLM and benefit from LLM agents with stronger long-context performance. Indeed, Lee et al. (2024) find that long-context LMs already rival dedicated retrieval pipelines and outperform RAG on cross-document reasoning, underscoring that faithful in-context utilization is the emerging bottleneck.

F.3 RL for Long-Context Reasoning

Among methods that directly train for long-context reasoning, several are closely related to RecaLLM. LoongRL (Wang et al., 2026) synthesizes challenging multi-hop training data via UUID key chains and trains with GRPO, inducing emergent plan-retrieve-reason patterns that generalize from 16K training contexts to 128K evaluation. QwenLong-L1 (Wan et al., 2025) uses progressive context scaling with curriculum-guided RL to adapt short-context reasoning models to long-context settings. ALR^2 (Li et al., 2024) takes a pipeline approach, prompting the model to first retrieve relevant evidence from the context before reasoning over it; this can be viewed as a precursor to RecaLLM's interleaved retrieval, though the fixed retrieve-then-reason pipeline is brittle for tasks where reasoning must precede or naturally interleave with retrieval.

RecaLLM builds on these methods by adding constrained decoding for faithful in-context retrieval, explicit retrieval quality supervision via R_{ret}, and a training recipe that achieves competitive benchmark scores with shorter contexts and less compute (Section 7).

F.4 Constrained Decoding and Copy Mechanisms

Grammar-constrained systems (Willard & Louf, 2023; Dong et al., 2025) and entity-constrained generation (De Cao et al., 2021) use logit masking to enforce structural constraints from a fixed grammar or candidate set. kNN-LM (Khandelwal et al., 2020) takes a softer approach, interpolating the LM's next-token distribution with a nearest-neighbor distribution over a precompiled datastore of context representations to bias generation toward memorized contexts. Classical copy mechanisms such as CopyNet (Gu et al., 2016) and Pointer-Generator Networks (See et al., 2017) learn a soft copy distribution over source tokens. In contrast to all of these, recall spans are learned, model-initiated actions embedded inside free-form reasoning: the model decides when to invoke them to recover and ground evidence, rather than using constraints to emit structured output or relying on external memory.

LLM Usage Disclosure

We used GPT-5.2 (OpenAI, 2025b) to annotate SFT training data by rewriting teacher-produced reasoning traces with verbatim recall spans (Appendix C.2). We also used Gemma-3-27B-IT (Gemma Team, 2025) to identify retrieval attempt points in reasoning traces for the

injection analysis in Section 2. No LLMs were used to originate research ideas or to generate evaluation results.