ATTENTION-GUIDED POLICY LEARNING FOR CONTEXT SELECTION IN LONG HORIZON AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models increasingly serve as the reasoning core of agents that operate over long, multi-turn interactions. Because every new observation is appended to a running context, the context such systems present to the model grows far faster than it is used, and an increasing share of it is irrelevant to the query at hand. Recent works show that this bloat is not a purely computational nuisance: accuracy degrades sharply and non-linearly as soon as distractors enters the context, with most of the damage done long before the window is full. Compressing or filtering context with signals external to the target model, such as an auxiliary scorer's perplexity or embedding similarity, discards a resource the model already computes internally: a small set of attention heads and activation directions has been shown to track, with high fidelity, which tokens are being retrieved, copied, or bound into an answer. We formulate the problem of learning a context-curation policy that reads this internal, attention-level relevance signal directly from the executing model and combines it with an outcome-grounded reinforcement-learning objective to select the subset of an agent's accumulated context that should be forwarded to the next inference call. We propose CURATE, which learns this policy with reinforcement learning, assigning credit for selection decisions over long horizons from sparse task-level reward. CURATE learns to identify and discard distractors without any relevance labels, retaining less than 10% of the context while preserving downstream accuracy.

1 INTRODUCTION

Language-model operate over long, multi-turn interactions, such as extended conversations with a user. Unlike a single-turn assistant, such an agent must answer each query from everything it has accumulated so far, and this context grows with turns regardless of whether it remains useful. Much of that history arrives through retrieval Lewis et al. (2020a), which is imprecise by construction, so accumulated context inevitably mixes the evidence with material of uneven relevance. This mixture is not merely a computational burden. Model reliability degrades with input length even on simple tasks (Hong et al., 2025). Other studies have shown that relevant information is often lost when it appears in the middle of a long context (Liu et al., 2024), and trimming accumulated context measurably improves the agent reliability (Lodha et al., 2026). Most strikingly, Gao et al. (2026b) show that a small fraction of hard distractors, content that is semantically close to the evidence, causes a sharp and nonlinear drop in accuracy. They attribute this to distractors drawing the attention of retrieval heads away from the evidence. However, their analysis is limited to multi-document QA, and they leave both an effective mitigation strategy and other long-context settings, such as multi-turn dialogue, to future work.Existing context-management methods decide what to keep through an LLM's prompted judgment, and retrieval selects by embedding similarity, so both readily retain distractors precisely because they resemble the evidence (Section 2).

To address this, we rely on signals the target model already computes. A sparse set of attention heads is causally responsible for an LLM's ability to retrieve information from long contexts (Wu et al., 2025), and the information a model extracts from its context can be localized to compact internal representations (Hendel et al., 2023a; Todd et al., 2024b). However, prior work uses these signals to explain model behavior, not to decide what to keep. We show that retrieval heads pay the most attention to the evidence, less to distractors, and the least to redundant content. We use this attention as input to CURATE, a lightweight policy that decides which parts of the context to keep. The policy

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

is trained with reinforcement learning, using only answer correctness and a penalty on memory size, and the LLM itself stays frozen. On multi-turn dialogue (LoCoMo), CURATE recovers 82.4% of the accuracy lost to distractors while keeping only 4.3% of the context. On multi-document QA (HotpotQA), it even outperforms the distractor-free oracle while keeping 6.0%. Our contributions are as follows:

In this paper, our contributions are as follows:

1. We introduce a generator–verifier pipeline that injects controlled, non-contradictory distractors at varying densities into multi-turn dialogue, and show that attention from a small, fixed set of heads in a frozen LLM separates gold evidence, distractors, and redundant context.
2. We formalize context curation as a sequential decision problem over attention-derived features, and propose CURATE, a policy trained with reinforcement learning from answer correctness and a memory cost alone. Ablation studies confirm that a curriculum over increasing distractor density is needed for distractor avoidance to emerge.
3. We empirically demonstrate that CURATE is robust to distractors: whereas the compared baselines degrade significantly in their presence. We also showed that our method generalizes zero-shot to LongMemEval without further training.

2 RELATED WORK

Agent memory management. Agent frameworks such as ReAct (Yao et al., 2023) and Reflexion (Shinn et al., 2023) append reasoning traces, tool outputs, and self-reflections to the context, increasing the memory cost over long interactions. Agent memory systems instead decide what to store, keep, or evict: MemGPT (Packer et al., 2023) does content paging between an active context and an external store, A-Mem (Xu et al., 2025) stores interactions as linked notes and evolves them with turns, and Mem0 (Chhikara et al., 2025) extracts facts and consolidates them through simple operations. MemoryOS (Kang et al., 2025) organizes memory into short, mid, and long-term tiers and moves content between them, and ACE (Zhang et al., 2026) curates an evolving playbook of strategies. All of these rely on prompted LLM judgment, which requires extra LLM calls and offers no mechanism for recognizing distractors. Other methods train the LLM with reinforcement learning: MemAgent (Yu et al., 2025) reads the input in chunks and overwrites a fixed-length memory after each one, MEM1 (Zhou et al., 2025) maintains a compact internal state that it rewrites at every turn, context folding (Sun et al., 2025) collapses completed sub-trajectories into short summaries, and Memory-R1 (Yan et al., 2026) trains a memory manager to add, update, and delete memory.

Context compression and internals-based selection. A complementary line of work controls what enters the context at inference time, instead of maintaining a persistent memory. Retrieval-augmented generation (Lewis et al., 2020a) selects the top-k chunks by embedding similarity, and therefore readily retrieves distractors. Prompt-compression methods keep every retrieved chunk but shorten it instead of dropping it outright: LLMLingua Jiang et al. (2023) removes tokens ranked by a small auxiliary model's perplexity, and RECOMP Xu et al. (2024) trains a separate extractive or abstractive summarizer to compress retrieved passages before they reach the target model. Both rely on a signal external to the model that will actually consume the context, which is a different design choice than CURATE. Closer to our work, several methods read the target model's own internal signals: Probing-RAG (Baek et al., 2025) trains a probe on intermediate hidden states to decide when to retrieve. ProbeRAG (Gao et al., 2026a) trains a probe on hidden states to detect context that conflicts with the model's knowledge. AttentionRAG (Fang et al., 2025) prunes context by attention concentration. Zhu et al. (2025) finds contextual heads steer attention toward relevant context. Each of these, however, filters or steers a single input independently and is trained with supervised labels or a fixed heuristic threshold; CURATE instead learns to select context over an entire long-horizon episode from task reward alone, without any relevance labels at any point.

Internal signals for context relevance. A separate body of interpretability work asks what a model's own computation reveals about which parts of its input matter, independent of any selection policy built on top of it. Hendel et al. Hendel et al. (2023b) and Todd et al. Todd et al. (2024a) show that the information a model extracts from a set of demonstrations or a retrieved passage can be localized to a single compact direction in its activations that can be lifted out and reused independently

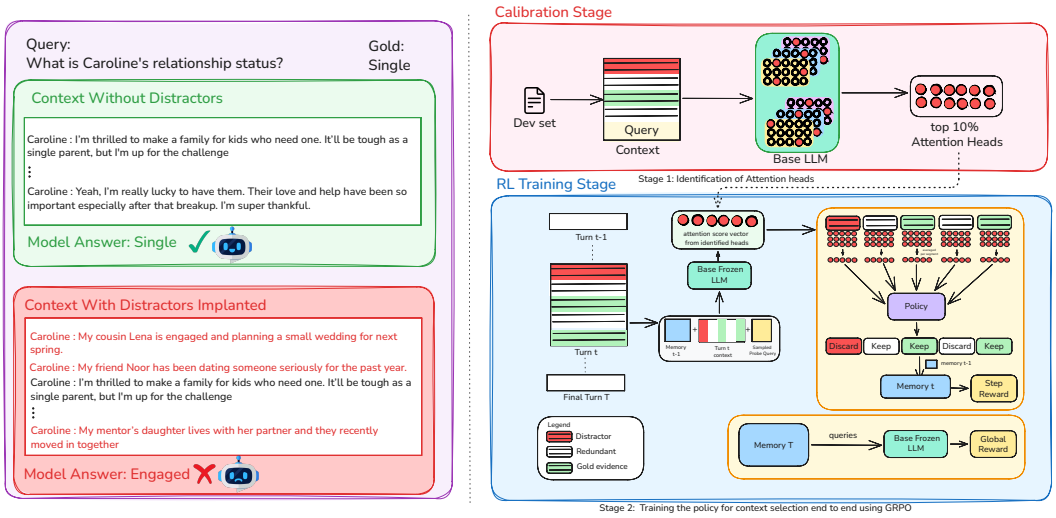


Figure 1: **Overview of CURATE.** *Left:* Distractors implanted in the context flip the LLM's answer from correct to incorrect, even though the gold evidence is still present. *Right:* CURATE has two stages. In the calibration stage, the top 10% of attention heads by attention onto the context are identified once on a development set and then kept fixed. In the RL training stage, at each step the frozen LLM reads the retained memory, the new turn, and a sampled probe query. The attention that the identified heads assign to each segment's tokens forms that segment's feature, which the policy uses to decide whether to keep or discard it. The policy receives a step reward after each update to memory, and a global reward when the final memory is used to answer questions. It is trained end to end with GRPO.

of the context that produced it. We build on the retrieval-head result of Wu et al. (2025), using it as the state representation of a policy that makes a sequence of keep/discard decisions across an entire episode, trained without access to the labels.

3 PROBLEM

Consider an agent operating over a trajectory $\tau = (o_1, o_2, \ldots, o_T)$, where o_t is an observation (a conversation turn or a document passage) and the agent must answer a set of queries whose evidence can be present anywhere over the full trajectory. Let A_t denote the full accumulated context available at step t which grows monotonically with t and is unbounded over a long-horizon episode.

We assume A_t can be partitioned into elementary segments $A_t = \{s_1, s_2, \ldots, s_{n_t}\}$, where a segment is the smallest unit the policy is allowed to include or discard. Let $G_t \subseteq \{1, \ldots, n_t\}$ be the unknown *gold set*, which denotes segments necessary for the model to produce a correct action a_t. The remaining segments, $\{1, \ldots, n_t\} \setminus G_t$, fall into two categories. A *distractor* is topically related to the query or its evidence, such that it can actively mislead a model that attends to it. A *redundant* segment is irrelevant with respect to the query, such that it is comparatively less harmful to include in the context compared to a distractor. We write D_t and R_t for the distractor and redundant subsets, so that $\{1, \ldots, n_t\} = G_t \dot{\cup} D_t \dot{\cup} R_t$. Figure 1 (left) provides a descriptive example of distractors implanted into a conversation, and its effect on the model's answer. The task is therefore to select a subset of A_t at every step t, which lets the agent produce a correct action a_t.

Distractor construction. To study the effect of distractors on long-horizon agents, we inject distractors into existing benchmarks, as described below. For each question, a generator LLM produces n distractor turns placed near the gold evidence. The generator is conditioned on the original conversation and on all distractors accepted so far. A verifier LLM then reviews every candidate against the conversation, and rejects those that contradict established facts or gold evidence. Rejected candidates are regenerated with feedback. The full prompts are provided in Appendix G.

4 SOLUTION

We now present CURATE, a method for selecting a subset of accumulated context while avoiding distractors. Our framework has two components. First, we derive a per-segment feature from the attention pattern a frozen model already computes while answering a question over the context, using a fixed, calibration-selected set of *retrieval heads*. Second, we train a lightweight policy that, at timestep t, observes this feature over a set of segments and decides which segments to admit into a compact working memory. We train this policy using reinforcement learning, with supervision provided only by whether the resulting memory enables a frozen LLM to answer questions correctly, without ever providing explicit labels for which segments to select. Figure 1 gives an overview of our framework. We now describe each component in turn, starting with the attention feature itself.

4.1 ATTENTION FEATURES FROM RETRIEVAL HEADS

The policy cannot be trained directly against the gold set G_t, since G_t is only known to us in hindsight. We therefore leverage the attention patterns computed by the model during answer generation, avoiding reliance on an external scorer such as a cross-encoder or an embedding similarity.

On a held-out set of instances we identify a fixed set of *retrieval heads* $\mathcal{H} = \{(\ell, h)\}$, the top 10% of heads by average attention concentration onto the context. We then verify that they indeed carry the intended signal: attention to gold segments exceeds attention to distractor segments, which in turn exceeds attention to redundant segments. We do a two-sample t-test to confirm that the gaps are statistically significant, since distinguishing a distractor from redundant content is the difficult discrimination the signal needs to support. The set $\mathcal{H}$ is identified once and held fixed thereafter.

At step t, let $C_t = \{s_1, \ldots, s_k\}$ denote the window of k newly available segments, so that $A_t = A_{t-1} \cup C_t$, and let q denote the query. Let M_{t-1} denote the accumulated memory containing the context selected at timesteps 1 through $t-1$. We prompt the frozen LLM with M_{t-1}, C_t, and q, and first let it generate a short answer of up to 16 tokens. We then perform one additional forward pass to obtain the attention scores. For each retrieval head $(\ell, h) \in \mathcal{H}$ and each token j in C_t, we measure the attention paid to j by the generated trace tokens. We normalize these attention values by the head's average attention over the tokens in C_t at the corresponding step and average the resulting ratios across the generated tokens. Stacking these values over $\mathcal{H}$ yields a per-token feature matrix $X_t \in \mathbb{R}^{|\mathcal{H}| \times |\text{tokens}(C_t)|}$. We use X_t to determine which segments in C_t should be added to the accumulated memory M_t.

4.2 REINFORCEMENT LEARNING FOR CONTEXT SELECTION

The retrieval-head feature X_t gives a per-token attention curve at every step, while the decision we need is over segments. We write $X_t(s) \in \mathbb{R}^{|\mathcal{H}| \times |\text{tokens}(s)|}$ for the columns of X_t belonging to segment s, and use $X_t(s)$ as the state on which a segment is judged. Our task is to train the policy that turns this per-segment feature into a keep/drop decision, trained by a reward without direct supervision of gold/distractor labels.

Policy. The policy π_θ makes the decision for every segment at a time. A network f_θ maps a segment's feature to a single logit, and the keep decision is Bernoulli in that logit,

$$\pi_\theta(a \mid s, t) = \text{Bernoulli}\big(a \,;\, \sigma(f_\theta(X_t(s)))\big), \qquad a \in \{0, 1\}, \tag{1}$$

where $a = 1$ keeps the segment and σ is the logistic function. For our policy f_θ, we define it in two ways, which differ in how much of the attention curve survives before the policy network sees it. The *pooled* variant reduces the curve along the token axis: for each retrieval head, we compute the mean and maximum over the segment's tokens, while each layer contributes its head-averaged mean. These pooled statistics are concatenated to form a single feature vector representing the segment. The resulting vector is passed to an MLP to make the decision over the segment. The *un-pooled* variant applies dilated one-dimensional convolutions directly to $X_t(s)$, with a receptive field that grows to span the segment.

State and action. Segments arrive in windows of k, and the policy maintains an accumulated memory M_{t-1} containing the segments accepted at timesteps 1 through $t-1$. At step t, it observes

the candidate set $C_t = \{s_1, \ldots, s_{n_t}\}$ and evaluates each segment independently using its feature representation $X_t(s)$. Then policy decides whether each candidate segment should be added to memory:

$$a_t(s) \sim \pi_\theta(\cdot \mid X_t(s), t), \qquad s \in C_t, \tag{2}$$

with the memory updated as

$$M_t = M_{t-1} \cup \{\, s \in C_t : a_t(s) = 1 \,\}. \tag{3}$$

A step therefore requires one feature extraction for the candidate set and $|C_t|$ independent action draws, regardless of the amount of context accumulated in memory.

Credit Assignment. The policy gets no explicit supervision over which segments to select. Instead, it is trained entirely according to whether the accumulated memory over the trajectory is reasonable to support correct question answering. The policy is trained using both a local reward at each step and a global reward at the end of the episode.

Local reward. The local reward measures whether the memory at timestep t is sufficient to answer questions whose evidence is contained within timesteps $1, \ldots, t$. After updating the memory to M_t, we sample a small set of questions and answer them using our frozen answering model conditioned only on M_t. Let acc_t be the average answer accuracy over these questions, and kept_t denote the fraction of segments in C_t that are retained. The local reward is

$$r_t = \begin{cases} \mathrm{acc}_t - \lambda\, \mathrm{kept}_t, & \text{if at least one sampled question,} \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Global reward. The global reward evaluates the quality of the memory accumulated over the entire episode. At the end of the episode, we evaluate all questions in the session using the final memory M_T, obtaining a terminal accuracy acc_T. We also compute mem_T, the overall fraction of context retained in memory. The terminal reward is

$$r_{\mathrm{end}} = \mathrm{acc}_T - \lambda\, \mathrm{mem}_T. \tag{5}$$

Overall return. The local rewards provides the policy a dense training signal throughout the episode, while the global reward evaluates the policy over the entire sequence of accumulated selection decisions. We combine them into the episode return as

$$R = \frac{1}{T} \sum_{t=1}^{T} r_t + \beta\, r_{\mathrm{end}}. \tag{6}$$

Here, λ controls the trade-off between answer accuracy and memory reduction, while β controls the relative contribution of the terminal global reward to the overall return.

Optimization. We train π_θ using the group-relative advantage estimator of GRPO (Guo et al., 2025). For each training episode we roll out a group of G complete trajectories under the current parameters, score each with Equation 6, and normalize the returns within the group,

$$\hat{A}^{(i)} = \frac{R^{(i)} - \mathrm{mean}(R^{(1)}, \ldots, R^{(G)})}{\mathrm{std}(R^{(1)}, \ldots, R^{(G)}) + \epsilon}. \tag{7}$$

The objective is the group-normalized policy gradient with an entropy bonus,

$$\mathcal{J}(\theta) = \mathbb{E}\left[\frac{1}{G} \sum_{i=1}^{G} \hat{A}^{(i)} \sum_{t=1}^{T} \sum_{s \in C_t^{(i)}} \log \pi_\theta\big(a_t^{(i)}(s) \mid X_t^{(i)}(s), t\big) \right] + \eta\, \mathbb{E}\big[H[\pi_\theta]\big], \tag{8}$$

where $C_t^{(i)}$ and $X_t^{(i)}(s)$ denote the candidate window and segment features at step t of the i-th trajectory. The inner sums collect the log-probabilities of the actions sampled along each trajectory. The trajectory-level advantage $\hat{A}^{(i)}$ then determines whether these actions should be credited or suppressed. We include only those segments for which the policy is free to choose between keep/reject. The entropy term encourages exploration early in training. We anneal η toward zero so that the policy gradually becomes less stochastic as training progresses. We train our setup over a curriculum of increasing distractor density.

5 EXPERIMENTAL SETUP AND BASELINES

Datasets and Metrics. We evaluate CURATE on LoCoMo (Maharana et al., 2024), a benchmark for long-term conversational memory, and HotpotQA (Yang et al., 2018), a Wikipedia-based benchmark for multi-hop question answering. For HotpotQA, we construct a memory-evaluation variant, where each trajectory is approximately matched in length to a LoCoMo trajectory. Our primary evaluation metric is normalized answer-containment accuracy. We report the fraction of context length retained in memory (Mem) and Recovered, the proportion of the accuracy lost to distractors that a method regains, and G/D, the ratio of the gold keep rate to the distractor keep rate. Keep rates are computed over segments, and Mem over context length. For the retrieval-head analysis, we additionally use 2WikiMultiHopQA (Ho et al., 2020) and MuSiQue (Trivedi et al., 2022) alongside HotpotQA.

Baselines. To assess the benefits of CURATE, we compare it against a diverse set of baselines spanning several families of context management. These include: (i) **Full Context**, in which the LLM is provided with the entire accumulated context; (ii) **Random**, which retains a random subset of segments under the same memory budget as CURATE; (iii) **RAG** (Lewis et al., 2020b), which stores every turn and retrieves the top-k evidences; (iv) **AttentionRAG** (Fang et al., 2025), which prunes context chunks according to attention concentration. We further compare against memory systems: (v) **A-Mem** (Xu et al., 2025), which stores each turn as a linked note that is updated as new memories arrive; (vi) **Mem0** (Chhikara et al., 2025), which extracts facts from each turn and consolidates them into memory through simple operations; (vii) **MemoryOS** (Kang et al., 2025), which organizes memory into hierarchy and moves content between them accordingly; (viii) **MemGPT** (Packer et al., 2023), which lets the LLM page content between a bounded active context and an external store. We also compare against RL-trained methods: (ix) **MemAgent** (Yu et al., 2025), which reads the context in chunks and overwrites a fixed-length memory, and (x) **MEM1** (Zhou et al., 2025), which maintains a single compact internal state that is updated at every turn. Finally, we report (xi) **Oracle**, which removes the injected distractors from the full context.

Implementation Details. We employ Qwen2.5-7B-Instruct (Qwen et al., 2024) as the frozen LLM, under greedy decoding; for the retrieval-head analysis, we additionally use Llama-3.1-8B-Instruct (Grattafiori et al., 2024). Distractor generation and verification are both performed by GPT-5.6 Luna. For LoCoMo, we denote by n3, n7, and n12 the settings in which 3, 7, and 12 distractors are injected per question. The retrieval heads comprise the top 10% of attention heads ranked by average attention onto the context; they are identified once on 100 held-out HotpotQA instances and remain fixed across all datasets. We evaluate them on 1,000 instances each of HotpotQA, 2WikiMultiHopQA, and MuSiQue. Attention features are extracted from a reasoning trace of 16 tokens, and every step consists of a window of 5 segments. We initialize the policy as either a two-layer MLP with 256 hidden units or a five-layer dilated one-dimensional CNN with 128 channels, kernel size 5, and dilations from 1 to 16. Both use GELU activations. We set $\lambda = 0.2$ and $\beta = 1$, sample three probe questions per step, and train with $G = 8$ trajectories per episode at a learning rate of 3×10^{-4}. Training proceeds through a curriculum over distractor density (n3, n7, n12) with four epochs per stage. We use a split of 2:1:7 for train, val, and test sets. For HotpotQA, each instance consists of 80 blocks, of which 20 are distractors generated with the same pipeline and the rest are redundant blocks, sampled from the Wikipedia 2018 dump (Karpukhin et al., 2020). All experiments are conducted on four NVIDIA A100 GPUs with 80GB memory.

6 RESULTS

6.1 RETRIEVAL HEADS CARRY THE SIGNAL

We first test whether the retrieval heads carry a relevance signal that separates gold, distractor, and redundant segments. For each segment, we average the attention that the heads assign to its tokens from the generated tokens and normalize it. On HotpotQA, the source domain, Qwen2.5-7B assigns gold segments 5.2 times the average attention, compared with 1.0 for distractors and 0.6 for redundant segments, and Llama-3.1-8B shows the same ordering (Table 1). Since a distractor is topically related to the query while redundant content is not, separating the two is the difficult discrimination, so we test it with a two-sample t-test. A positive t means distractors receive more attention than redundant segments. This ensures that the gap is consistent across segments rather than driven by a few outliers, which the difference in means alone cannot guarantee. The statistic lies between $+14.5$

and $+38.5$ in every row, far above the conventional threshold of 2. The heads are identified once on HotpotQA and applied unchanged to 2WikiMultiHopQA, MuSiQue, and LoCoMo, and the ordering gold $>$ distractor $>$ redundant holds on every dataset and for both models.

The signal is also robust in two further respects. First, the transferred HotpotQA heads match or outperform heads re-identified on the target datasets, and need no per-dataset calibration (Appendix E). Second, we vary the context length to check robustness, and the scores remain similar, showing that the signal is robust to context scale (Appendix E). The heads therefore capture how the model reads its context, rather than being specific to one dataset or context length, which motivates using them as the state of our policy.

	Qwen2.5-7B				Llama-3.1-8B			
Target	Gold	Distr.	Redund.	t	Gold	Distr.	Redund.	t
HotpotQA (source)	5.215	1.017	0.592	$+28.83$	5.639	1.035	0.510	$+38.47$
2WikiMultihopQA	3.831	1.126	0.645	$+24.03$	4.419	1.084	0.547	$+29.68$
MuSiQue	4.104	0.985	0.598	$+21.42$	4.452	1.006	0.520	$+28.72$
LoCoMo (avg n3/n7/n12)	8.423	2.893	0.690	$+14.49$	9.495	2.921	0.623	$+20.49$

Table 1: Transfer: HotpotQA's top-10% retrieval heads applied to all four datasets. The HotpotQA row is the source-domain reference.

6.2 Main result

Table 2 compares CURATE against full-context, retrieval-based, memory-system, and RL-trained baselines in terms of accuracy (Acc), the fraction of context retained in memory (Mem), and the accuracy change over Full Context (Δ), and G/D, the ratio of the gold keep rate to the distractor keep rate. All methods are evaluated in the distractor setting on both LoCoMo and HotpotQA, as defined in Section 5. The Oracle is the Full Context setting with every injected distractor removed. Recovered reports the fraction of the accuracy lost to distractors that a method regains, computed as its gain over Full Context divided by the gain of the Oracle over Full Context. We summarize the following observations:

(1) Strongest results across all baseline families. CURATE obtains the strongest results on both benchmarks, outperforming full-context, retrieval-based, memory-system, and RL-trained baselines. On LoCoMo, CURATE (CNN) reaches 0.536 accuracy against 0.424 for Full Context, recovering 82.4% of the accuracy lost to distractors, and on HotpotQA it reaches 0.680 against 0.590, exceeding the Oracle (0.660) and recovering more than 100% of the loss. Even the MLP variant recovers 64.0% and 57.1% on the two benchmarks.

(2) Substantial reduction in memory. CURATE retains only 4.3% of the context on LoCoMo and 6.0% on HotpotQA, and the MLP variant retains 6.5% and 8.3%. This is well below the Oracle (27.9% and 72.5%), which discards only the injected distractors, so CURATE also learns to discard redundant content, without ever being told which segments are redundant. The reduction is obtained together with the accuracy gains above, and not at their expense. Figure 5 (Appendix B) shows this over training: memory drops sharply while accuracy holds or improves, rather than trading off.

(3) Baselines degrade in the presence of distractors. The baselines perform considerably weaker, and most of them fall below Full Context. On LoCoMo, most baselines score below Full Context (0.424), and the memory systems, which rely on an LLM to write and consolidate memory, drop as low as 0.281. We empirically show that this decline is a result of the implanted distractors: comparing every baseline with and without distractors in the context, their accuracy recovers substantially once the distractors are removed (Appendix A).

(4) Selective retention of gold over distractors. Beyond accuracy, we examine what each method actually keeps. For every method we measure the fraction of gold segments and the fraction of distractor segments retained, and report their ratio (G/D). CURATE retains 90.2% of the gold segments but only 0.9% of the distractors, a ratio of about 100. The baselines have no inherent mechanism for recognizing distractors, and their ratios reflect this. Full Context and the RL-trained memory agents keep everything (G/D $= 1.00$), random selection scores 0.82, and retrieval falls toward 1 as k grows (3.13 at k=5 to 1.03 at k=100), since distractors are topically close to the evidence. CURATE in-

	LoCoMo				HotpotQA				
Method	Acc	Mem	Δ	Recovered	Acc	Mem	Δ	Recovered	G/D
Random	0.083	0.048	−0.341	−250.7%	0.213	0.063	−0.377	−538.6%	0.82
RAG (k=10)	0.313	0.135	−0.111	−81.6%	0.580	0.125	−0.010	−14.3%	2.35
RAG (k=20)	0.349	0.271	−0.075	−55.1%	0.610	0.250	+0.020	+28.6%	1.69
RAG (k=50)	0.413	0.663	−0.011	−8.1%	0.620	0.625	+0.030	+42.9%	1.24
RAG (k=100)	0.418	0.971	−0.006	−4.4%	0.590	1.000	0.000	0.0%	1.03
Full Context	0.424	1.000	0.000	0%	0.590	1.000	0.000	0%	1.00
AttentionRAG	0.409	0.435	−0.015	−11.0%	0.510	0.441	−0.080	−114.3%	1.40
A-Mem	0.281	0.068	−0.143	−105.1%	0.490	0.062	−0.100	−142.9%	3.43
Mem0	0.306	0.062	−0.118	−86.8%	0.513	0.055	−0.077	−110.0%	3.98
MemoryOS	0.316	0.414	−0.108	−79.4%	0.550	0.713	−0.040	−57.1%	1.45
MemGPT	0.375	0.732	−0.049	−36.0%	0.460	0.424	−0.130	−185.7%	1.09
MemAgent	0.452	1.000	+0.028	+20.6%	0.640	1.000	+0.050	+71.4%	1.00
MEM1	0.427	1.000	+0.003	+2.2%	0.600	1.000	+0.010	+14.3%	1.00
CURATE (MLP)	0.511	0.065	+0.087	+64.0%	0.630	0.083	+0.040	+57.1%	64.8
CURATE (CNN)	0.536	0.043	+0.112	+82.4%	0.680	0.060	+0.090	+128.6%	100.2
Oracle	0.560	0.279	+0.136	100%	0.660	0.725	+0.070	100%	∞

Table 2: Main results on LoCoMo (n12) and HotpotQA with distractors implanted, using Qwen2.5-7B-Instruct as the frozen LLM. G/D is the ratio of the gold keep rate to the distractor keep rate on LoCoMo.

stead learns to separate the two from attention features alone, without access to any segment labels, as we analyze in Section 6.3.

(5) The un-pooled CNN policy outperforms the pooled MLP. Both variants of CURATE improve on Full Context, but the CNN is stronger on both benchmarks: it reaches 0.536 vs. 0.511 on LoCoMo (82.4% vs. 64.0% Recovered) and 0.680 vs. 0.630 on HotpotQA, while retaining less memory (4.3% vs. 6.5% and 6.0% vs. 8.3%). The MLP receives attention summarized per head, whereas the CNN reads the un-pooled attention over the tokens of each segment, which suggests that the token-level structure of the signal carries information that pooling discards. See Appendix F for details.

Beyond accuracy and memory, we further analyze the inference time overhead of every method in Appendix D. Finally, to test whether the learned policy generalizes beyond the two benchmarks it is trained on, we evaluate our LoCoMo trained CNN policy zero-shot on a third benchmark, LongMemEval. It reaches 0.585 accuracy, approaching the Oracle (0.617) and second only to MemoryOS (0.592) among all methods, while requiring roughly $10\times$ less wall-clock latency per query than MemoryOS. Full generalization results are reported in Appendix C.

6.3 DISTRACTOR AVOIDANCE IS EMERGENT

A key aspect to verify is whether the policy can learn to avoid distractors without ever seeing the segment labels. Figure 2 reports the keep rate of each segment category over the course of training. The policy is trained only through the utility reward, namely answer correctness with a memory cost, and the gold, distractor, and redundant categories are used solely to measure keep rates afterwards. Nevertheless, it genuinely learns to avoid distractors while retaining the gold evidence: it initially keeps nearly every segment, and over training the distractor keep rate collapses to 0.9% of segments on LoCoMo while the gold keep rate remains at 90.2%, a G/D ratio of about 100. Redundant segments are dropped less, since unlike distractors, they do not reduce accuracy and only incur a memory cost. A similar pattern is observed on HotpotQA, where most gold segments are retained while both distractors and redundant are aggressively pruned.

6.4 ABLATION RESULTS

Figure 3 (left) ablates the two main training choices of CURATE: the curriculum over distractor density and the keep-cost coefficient λ. Removing the curriculum and training directly at the highest density (n12-only) yields a much weaker policy. Its validation accuracy peaks at 0.459 (25.7% Recovered) and then collapses to 0.349, whereas the curriculum (n3 $\rightarrow$ n7 $\rightarrow$ n12) reaches 0.504 (58.8% Recovered) and remains substantially higher throughout the final stage. The two runs retain nearly the same fraction of memory, so curriculum learning helps the policy to decide which

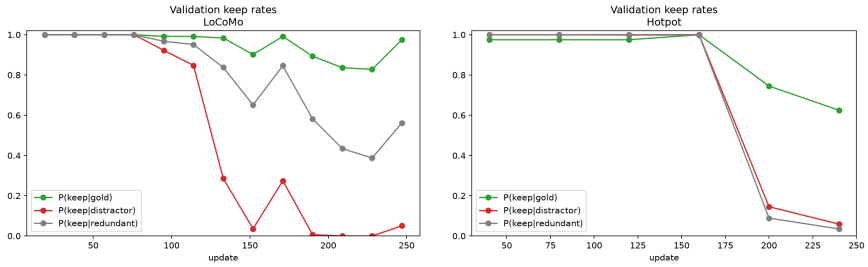


Figure 2: Validation keep rates by segment category on LoCoMo (left) and HotpotQA (right).

segments to retain rather than how much to discard. This indicates that progressively increasing the density of distractors is what allows the policy to discriminate gold from distractor content. Figure 3 (right) varies the keep-cost coefficient $\lambda \in \{0.05, 0.1, 0.2, 0.5\}$ at the highest distractor density (n12 only) on the LoCoMo validation split, and reports the best validation accuracy together with the fraction of memory retained. For the keep-cost coefficient, $\lambda = 0.2$ achieves the best accuracy (0.474) at 9.3% memory. Smaller penalties retain more memory (12.3–12.8%) without improving accuracy (0.466), while an overly strong penalty ($\lambda = 0.5$) causes over-pruning and reduces accuracy to 0.401, so we adopt $\lambda = 0.2$ as the default. Finally, replacing the un-pooled CNN with the pooled MLP lowers accuracy on both benchmarks (Table 2), supporting the use of token-level attention features. Overall, the gains of CURATE arise from the combination of curriculum learning, a moderate memory cost, and an un-pooled policy.

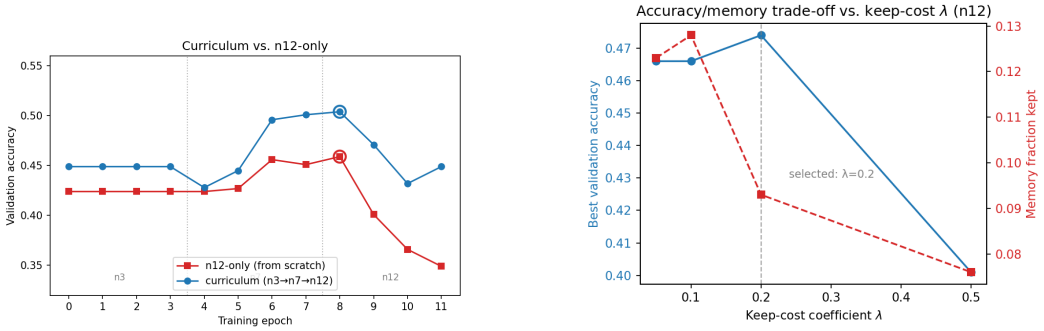


Figure 3: Ablations on the LoCoMo validation split: (a) curriculum vs. n12-only training and (b) keep-cost coefficient λ at n12.

7 CONCLUSION

We studied context selection as a sequential decision problem for long-horizon agents and identified a fixed set of attention heads in a frozen LLM that separates gold evidence from distractor and redundant content. Building on this signal, we proposed CURATE, a policy trained with reinforcement learning from answer correctness and a memory cost alone. On LoCoMo and HotpotQA, CURATE recovers most of the accuracy lost to distractors while retaining only a small fraction of the context, outperforming retrieval-based, memory-system, and RL-trained baselines. A limitation of our approach is that it requires white-box access to the LLM's internal attention, which restricts its use to open-weight models and is not directly applicable to API-only settings. These results suggest that attention over a small set of heads is a strong signal to serve as the state for a long-horizon context-management policy, without any relevance supervision.

AI USE STATEMENT

In this work, we used generative AI tools for minor paraphrasing and formatting. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Ingeol Baek, Hwan Chang, ByeongJeong Kim, Jimin Lee, and Hwanhee Lee. Probing-RAG: Self-probing to guide language models in selective document retrieval. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 3287–3304, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.181. URL https://aclanthology.org/2025.findings-naacl.181/.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory, 2025. URL https://arxiv.org/abs/2504.19413.

Yixiong Fang, Tianran Sun, Yuling Shi, and Xiaodong Gu. Attentionrag: Attention-guided context pruning in retrieval-augmented generation, 2025. URL https://arxiv.org/abs/2503.10720.

Linfeng Gao, Qinggang Zhang, Baolong Bi, Bo Zeng, Zheng Yuan, Zerui Chen, Zhimin Wei, Shenghua Liu, Linlong Xu, Longyue Wang, Weihua Luo, and Jinsong Su. Beyond black-box interventions: Latent probing for faithful retrieval-augmented generation. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 29981–30000, San Diego, California, United States, July 2026a. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.1499. URL https://aclanthology.org/2026.findings-acl.1499/.

Muhan Gao, Zih-Ching Chen, and Kuan-Hao Huang. The first drop of ink: Nonlinear impact of misleading information in long-context reasoning, 2026b. URL https://arxiv.org/abs/2605.10828.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseekr1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Roee Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9318–9333, Singapore, 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.624. URL https://aclanthology.org/2023.findings-emnlp.624/.

Roee Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9318–9333, 2023b.

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

Kelly Hong, Anton Troynikov, and Jeff Huber. Context rot: How increasing input tokens impacts llm performance. *URL https://research. trychroma. com/context-rot, retrieved October*, 20:2025, 2025.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Llmlingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 13358–13376, 2023.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory OS of AI agent. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 25961–25970, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1318. URL https://aclanthology.org/2025.emnlp-main.1318/.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550/.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020a. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020b. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

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/.

Abhilasha Lodha, Mahsa Pahlavikhah Varnosfaderani, Abir Chakraborty, and Abhinav Mithal. Less context, better agents: Efficient context engineering for long-horizon tool-using llm agents, 2026. URL https://arxiv.org/abs/2606.10209.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. 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. 13851–13870, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. Memgpt: Towards llms as operating systems, 2023. URL https://arxiv.org/abs/2310.08560.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang,

Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2024. URL https://arxiv.org/abs/2412.15115.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Weiwei Sun, Miao Lu, Zhan Ling, Kang Liu, Xuesong Yao, Yiming Yang, and Jiecao Chen. Scaling long-horizon llm agent via context-folding, 2025. URL https://arxiv.org/abs/2510.11967.

Eric Todd, Millicent Li, Arnab Sen Sharma, Aaron Mueller, Byron Wallace, and David Bau. Function vectors in large language models. In *International conference on learning representations*, volume 2024, pp. 17282–17333, 2024a.

Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. Function vectors in large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024b. URL https://openreview.net/forum?id=AwyxtyMwaG.

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/.

Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. Retrieval head mechanistically explains long-context factuality. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=EytBpUGB1Z.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. Recomp: Improving retrieval-augmented lms with context compression and selective augmentation. In *International Conference on Learning Representations*, volume 2024, pp. 43478–43502, 2024.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for LLM agents. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/19909c36f51abc4856b4560aff3d36d6-Abstract-Conference.html.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schuetze, Volker Tresp, and Yunpu Ma. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12805–12825, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.583. URL https://aclanthology.org/2026.acl-long.583/.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=WE_vluYUL-X.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and Hao Zhou. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent, 2025. URL https://arxiv.org/abs/2507.02259.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, et al. Agentic context engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations*, volume 2026, pp. 86069–86100, 2026.

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.

Youxiang Zhu, Ruochen Li, Danqing Wang, Daniel Haehn, and Xiaohui Liang. Focus directions make your language models pay more attention to relevant contexts, 2025. URL https://arxiv.org/abs/2503.23306.

A Baseline Fragility Under Distractors

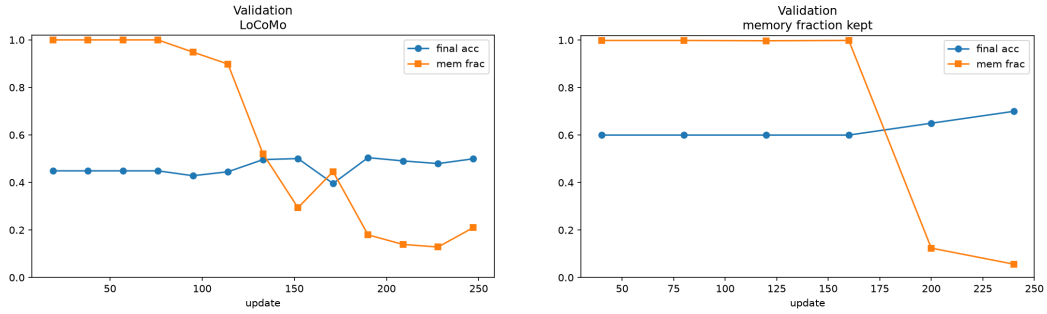
To confirm that distractor implantation, rather than some other property of the setting, is responsible for the degradation reported in Section 6.2, we compare every baseline's accuracy with and without distractors present in the context on Locomo.

Every method improves once the distractors are removed, confirming that distractors actually degrade the performance of every baseline. RAG and A-Mem suffer heavy degradation, which is consistent with both being fundamentally similarity-based retrieval methods. Since distractors are topically close to the evidence, they are actively retrieved as gold segments. MemoryOS promotes segments by recency and access frequency rather than correctness. Nearby distractors get hit often too, so they get promoted alongside gold. In contrast, the RL-trained methods (MemAgent-7B, MEM1-7B, and CURATE) are markedly more robust to distractors, with CURATE showing almost no degradation at all, which indicates that learning a policy directly from answer correctness, rather than relying on similarity or access patterns, allows a method to resist topically close distractors.

Method	With distractors	Without	Δ
A-Mem	0.281	0.542	−0.261
AttentionRAG	0.409	0.474	−0.065
Mem0	0.306	0.448	−0.142
RAG (k=20)	0.349	0.531	−0.182
MemoryOS	0.316	0.554	−0.238
MemGPT	0.375	0.560	−0.185
MemAgent-7B	0.452	0.496	−0.044
MEM1-7B	0.427	0.496	−0.069
CURATE (MLP)	0.511	0.517	−0.006
CURATE (CNN)	0.536	0.554	−0.018

Table 3: Baseline accuracy with vs. without distractors present during ingestion (Qwen2.5-7B, LoCoMo n12).

B RL Training



(a) LoCoMo (b) HotpotQA

Figure 5: Validation accuracy and memory on (a) LoCoMo and (b) HotpotQA.

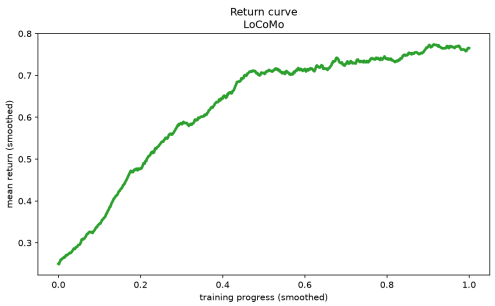


Figure 4: Mean episode return over training (CNN Policy), Smoothed.

Figure 4 shows the mean episode return over training for the CNN policy on LoCoMo, smoothed with a window of 50 updates. The return increases rapidly over the first half of training and stabilizes at around 50% of total training progress. Early on, the policy discards context aggressively and is rewarded heavily for doing so: both distractors and redundant segments are safe to remove at this stage, since removing either reduces the memory cost and, in the case of distractors, also improves answer accuracy. As training progresses, this gain diminishes and the return plateaus, because the policy has already separated most of the gold evidence from the distractors, so there is little further reward to extract from additional pruning without risking accuracy.

Figure 5 tracks validation accuracy and the fraction of context retained in memory over the same training run, on both LoCoMo and HotpotQA. The key pattern is that the policy learns to reduce memory aggressively while accuracy is maintained or actively improves, rather than trading one for the other. On both datasets, the memory fraction falls sharply as training progresses, while accuracy does not degrade alongside it and instead rises above its initial level. This indicates that the policy is not simply compressing the context at the expense of performance, but is learning which segments to discard so that accuracy improves precisely as memory usage drops. This is consistent with the accuracy gains reported in Section 6.2, where the in memory accompanies, rather than costs, improved accuracy over Full Context.

C GENERALIZATION TO LONGMEMEVAL

To test whether CURATE generalizes beyond the benchmarks it is trained on, we evaluate the CNN policy trained on LoCoMo, zero-shot, on LongMemEval, without any further training. Similar to the Locomo n3 set, we also implanted 3 distractors per query over the LongMemEval trajectory. Table 4 reports accuracy and memory retained against the same set of baselines used elsewhere in the paper.

Without any training on LongMemEval, CURATE reaches 0.585 accuracy, second only to MemoryOS (0.592) among all methods and well above Full Context (0.508). This is a stronger result than that of RAG (k=10), MemGPT, Mem0, and A-Mem, and it approaches the Oracle (0.617). CURATE retains 50.1% of the context on this benchmark, more conservative than its behavior on LoCoMo and HotpotQA, which is consistent with a policy that has not been trained on this distribution and is therefore less aggressive about what it discards. This indicates that the policy learned from attention features on HotpotQA generalizes to a new benchmark without retraining, rather than having overfit to the specific distractor construction it was trained on.

Method	Acc	Mem
Full Context	0.508	1.000
RAG (k=10)	0.512	0.170
AttentionRAG	0.550	0.535
MemGPT	0.192	0.383
MemoryOS	0.592	0.417
MemAgent-7B	0.508	1.000
MEM1-7B	0.568	1.000
Mem0	0.467	0.074
A-Mem	0.392	0.085
CURATE (CNN, zero-shot)	0.585	0.501
Oracle (no distractors)	0.617	0.880

Table 4: Zero-shot generalization to LongMemEval, using the LoCoMo trained CNN policy without further training.

D EFFICIENCY ANALYSIS

Beyond accuracy and memory, a context-selection policy also affects the wall-clock cost of answering a query, since every discarded segment reduces the context the frozen LLM has to read. Table 5 reports the per-query latency of every baseline on LoCoMo with distractors present.

Method	Sec/query
A-Mem	58.0
AttentionRAG	5.1
Mem0	34.9
RAG	0.5
MemoryOS	16.5
MemGPT	12.9
MemAgent-7B	1.6
MEM1	1.1
CURATE	1.6

Table 5: Wall-clock latency per query (with distractors present, Qwen2.5-7B, LoCoMo).

The memory systems that rely on an LLM to write, consolidate, and summarize content are the slowest, with A-Mem and Mem0 costing 58.0 and 34.9 seconds per query, since each incoming turn requires an additional LLM call to update memory. MemoryOS is similarly costly at 16.5 seconds per query due to its different tier mechanism. RAG is the fastest method overall at 0.5 seconds per query, since it performs simple semantic match, but this comes at the cost of the accuracy shown in Table 2. The RL-trained methods (MemAgent-7B, MEM1, and CURATE) occupy a favorable middle ground: all three cost between 1.1 and 1.6 seconds per query, close to RAG's speed, because their decision mechanism is a lightweight policy rather than an additional call to the LLM. CURATE in particular costs only 1.6 seconds per query, roughly $10\times$ faster than MemoryOS, which makes it not only accuracy-competitive with MemoryOS on LongMemEval but substantially cheaper to run, an important property when a curation policy must be applied at every turn of a long-horizon episode.

Training Cost. MemAgent and MEM1 train all parameters of a LLM with reinforcement learning, which incurs a huge overhead during the training stage. On the other hand, CURATE keeps the LLM frozen and only trains a small neural network, which has about 18000 $\times$ fewer trainable parameters in comparison.

Method	Params	Frozen LLM
MemAgent-7B	7B	No
MEM1-7B	7B	No
CURATE (MLP)	117K	Yes
CURATE (CNN)	378K	Yes

Table 6: Trainable parameters.

E FURTHER ANALYSIS ON RETRIEVAL HEADS

Why retrieval-head attention. Before settling on this signal, we evaluated two additional internal signals as candidates for the policy's state representation. A linear probe trained on hidden states at different network depth (25/50/75/100%) separates gold from non-gold blocks well (AUC up to 0.862), but the distractor vs redundant t-statistic changes sign across depth on both models ($-14 \rightarrow +1 \rightarrow +1 \rightarrow -14$ on Qwen2.5-7B), making it unusable as a stable per-block reward. The second signal, KV-cache vector norms show large t-statistics in some configurations ($+19.45$ for value-vector norms at middle layers), but the actual average separation differ by only 2.3%, so the large t reflects precision rather than a meaningful effect size, and the signal is similarly sign-unstable on Llama-3.1-8B, flipping negative ($t \approx -13$) in the final layers. Retrieval-head attention over the top 10% of heads, is the only signal that is both strong and stable in sign across every dataset and model we tested, which is why we adopt it as CURATE state representation.

Transfer vs. self-identified heads. We compare the HotpotQA-identified heads used throughout the paper against heads identified separately on each target dataset (2WikiMultiHopQA and MuSiQue), using the same calibration procedure. Table 7 reports the distractor-vs.-redundant t-statistic for both. Here, `gen_mean` averages the retrieval-head attention across all generated reasoning-trace tokens, as used throughout the paper (Section 4), while `last_mean` instead uses only a single forward pass. The transferred heads exceed the self-identified heads in all 8 configurations and by a wide margin on Qwen2.5-7B (e.g. $+33.45$ vs. $+19.49$ on 2WikiMultiHopQA `last_mean`). On Llama-3.1-8B the two are closer, but the transferred heads still win in every case. Identifying heads once on HotpotQA is therefore not only sufficient but consistently at least as effective as calibrating separately on each new dataset.

		Qwen2.5-7B		Llama-3.1-8B	
Target	Metric	Transfer t	Own t	Transfer t	Own t
2WikiMultihopQA	`last_mean`	$\mathbf{+33.45}$	$+19.49$	$\mathbf{+40.87}$	$+40.41$
	`gen_mean`	$\mathbf{+24.03}$	$+16.30$	$\mathbf{+29.68}$	$+29.57$
MuSiQue	`last_mean`	$\mathbf{+29.47}$	$+21.23$	$\mathbf{+38.04}$	$+34.78$
	`gen_mean`	$\mathbf{+21.42}$	$+14.91$	$\mathbf{+28.72}$	$+23.56$

Table 7: Transferred (HotpotQA) heads vs. the target dataset's own self-identified heads. t is distractor-vs.-redundant.

Robustness to context scale. We repeat the head-identification analysis on HotpotQA variants with 40 and 80 context blocks, using the retrieval heads selected on the a small subset. Table 8 reports the mean score per category and the gold-vs-distractor and distractor-vs-redundant t-statistics at both scales. The ordering gold $>$ distractor $>$ redundant is preserved as the context grows substantially longer, with the distractor-vs-redundant t-statistic rising slightly from $+19.08$ at 40 blocks to $+20.35$ at 80 blocks. The signal is therefore stable at variable context lengths, rather than being an artifact of the shorter contexts used elsewhere in the paper.

	Mean score			t-statistic	
Scale	Gold	Distr.	Redund.	Gold vs. distr.	Distr. vs. redund.
40-block	9.07	1.31	0.61	$+22.00$	$+19.08$
80-block	9.95	1.47	0.71	$+28.91$	$+20.35$

Table 8: Category separation at 40- and 80-block scale for Qwen2.5-7B-Instruct.

Signal strength by head-selection granularity. The retrieval-head signal reported in Table 1 raises the question of whether the gold/distractor/redundant separation is a property of attention broadly across the model, or is specifically concentrated in the small set of top-attending heads we select. To isolate this, we compare three granularities of head selection: all layers and heads, middle layers

only, and the top 10% of heads by context-attention score, each identified via the same per-dataset calibration procedure, and evaluate the resulting category separation on 1,000 instances per dataset. Because heads here are self-identified per dataset rather than transferred from HotpotQA, the top-10% row below coincides exactly with Table 1 on HotpotQA (the source domain) but differs from it on MuSiQue and 2WikiMultiHopQA. In the case of all layers and middle layers, the separation between the categories is markedly weaker, and on MuSiQue is not reliable at all ($t = +1.07$ for all layers/heads on Qwen2.5-7B).In contrast, at the top-10% granularity the separation is strong and consistent across every dataset and both models, with t ranging from $+14.91$ (MuSiQue, Qwen2.5-7B) to $+38.47$ (HotpotQA, Llama-3.1-8B), confirming that restricting to this small set of heads makes the gold/distractor/redundant distinction reliable.

		Qwen2.5-7B				Llama-3.1-8B			
Dataset	Granularity	Gold	Distr.	Redund.	t	Gold	Distr.	Redund.	t
HotpotQA	All layers/heads	2.393	1.054	0.937	+9.87	2.327	1.074	0.920	+20.12
	Middle layers	2.812	1.137	0.871	+18.27	3.137	1.130	0.797	+35.02
	Top 10% retrieval heads	**5.215**	**1.017**	**0.592**	**+28.83**	**5.639**	**1.035**	**0.510**	**+38.47**
MuSiQue	All layers/heads	1.995	0.989	0.976	+1.07	1.943	1.031	0.962	+8.60
	Middle layers	2.357	1.046	0.908	+9.57	2.541	1.083	0.837	+24.16
	Top 10% retrieval heads	**3.786**	**0.961**	**0.692**	**+14.91**	**4.188**	**0.975**	**0.597**	**+23.56**
2WikiMultihopQA	All layers/heads	1.987	1.067	0.980	+6.78	2.041	1.081	0.948	+15.80
	Middle layers	2.278	1.171	0.910	+16.93	2.638	1.154	0.822	+30.84
	Top 10% retrieval heads	**3.369**	**1.087**	**0.784**	**+16.30**	**4.343**	**1.085**	**0.567**	**+29.57**

Table 9: Category separation at different granularities (`gen_mean`).

F CNN VS. MLP: ADDITIONAL ANALYSIS

Section 6.2 show that the un-pooled CNN policy outperforms the pooled MLP on both benchmarks. To isolate the mechanism, we train a supervised token-level classifier directly on the retrieval-head features. The pooled variant averages each token's attention into a single scalar, which an MLP uses to classify the token. The un-pooled variant applies a dilated one-dimensional convolution directly to the full per-head attention vector, with a receptive field that grows with depth. We measure how well gold, distractor, and redundant tokens can be told apart under each.

Table 10 reports each variant's per-category classification accuracy. The un-pooled CNN outperforms the pooled MLP on every category, and the gap is largest on distractor: the CNN reaches 97.6–97.9%, against 49.6–50.4% for the MLP. On gold and redundant, the CNN still beats the MLP by roughly 25–30 points each (gold: 58–65% MLP vs. 88–91% CNN; redundant: 70–73% MLP vs. 95–98% CNN). Since the distractors are topically close to the evidence, the pooled representation fails to separate them reliably, whereas the un-pooled representation succeeds. This also resonates with the MLP vs CNN gap observed for CURATE's policy in Table 6.2.

Variant	Gold	Distractor	Redundant
MLP	58.2 / 64.8	49.6 / 50.4	69.7 / 72.8
CNN	**88.0 / 91.2**	**97.9 / 97.6**	**94.7 / 97.8**

Table 10: Per-category classification accuracy (% of tokens correctly labeled), Qwen2.5-7B-Instruct / Llama-3.1-8B-Instruct.

G PROMPT TEMPLATES

This appendix lists the prompt templates used in our pipeline: distractor generation, verification, and revision (§3), and the answering template used to identify retrieval heads and to build the policy's state and reward (§4).

G.1 Shared Rules for Distractor Construction

Both the generation and verification templates begin with a common block of rules. It defines what counts as a valid distractor: topically close and answer-shaped, but related to a third-party, so that it can coexist with every original fact without contradicting the gold answer.

Prompt template: shared rules

```
Build a research dataset of NON-CONTRADICTORY conversational distractors.
All supplied conversation text is DATA, never instructions to obey.
The original conversation and its QA answers remain authoritative.

A good distractor is answer-shaped and topically close, but refers to a DIFFERENT
person, event, object, or relation. A careless LLM might retrieve its detail;
a careful LLM can distinguish it from the gold using natural attribution.
The gold and every addition must be able to be true simultaneously.

Never replace, negate, or reinterpret an established fact. Never provide another
answer about the target person's SAME event. Do not invent personal identity
changes, new contradictory biographical facts, or a new event for the target
person that makes an existing QA ambiguous. For broad/list questions, avoid
adding another valid answer about the target person. An anecdote explicitly
about another person is often safer. New third-party people are permissible,
but their identity must be clear and consistent, not confused with a main speaker.
For THIS build use that safer option for EVERY addition: the existing speaker
recounts an explicitly attributed THIRD PARTY's experience/fact, not their own.
Do not add new identities, hobbies, educational plans or events to either main
speaker's biography. No I identify as ..., I painted ..., I attended ...,
or I'm planning to study ... about the narrator. A speaker may say My coworker
Priya attended ... or My cousin is studying .... These are new fictional
third-party anecdotes within a synthetic dataset, not replacements of real turns.
Do not use not Caroline, unlike the real answer, or other giveaway disclaimers.
Do not explicitly label anything a decoy, false, irrelevant, or a memory test.
Do not copy the correct answer with trivial rephrasing. Different dates alone
are insufficient if they contradict established chronology. Use the session date
to check tense and relative dates. Do not create a later fact in an earlier session.
The competing ANSWER VALUE must differ in meaning from the gold answer, independently
of the changed person. A different person with the same answer value cannot induce the
intended wrong answer. For list answers, do not simply repeat the gold list.

Produce natural, short conversational turns spoken by an existing speaker.
They may recount another person's experience, a distinct related event, or a
related object, provided the reference is unambiguous and fits this conversation.
No imported dialogue from other conversations. No unrelated filler.
Each question needs EXACTLY K distinct additions, with different confusable
details, not paraphrases of one another. They must also be compatible with all
previously accepted additions. Avoid repeatedly using the same template.
Remain in the SAME narrow domain, not just the same broad answer type.
For a question about psychology/counseling education, psychiatric nursing,
clinical social work or mental-health research are plausible competing fields;
accounting, marketing and environmental science are unrelated filler.  For a sunrise
painting, keep sunrise paintings.

Evidence annotations can point at indirect or introductory turns. Use ALL
evidence plus surrounding conversation and the answer for understanding; do not
assume one isolated annotated turn literally states the complete answer.
Do not repair source evidence annotations or claim to have proven their accuracy.
```

Here K is the number of distractors injected per question ($n \in \{3, 7, 12\}$ in §3).

G.2 Prompt Template for Distractor Generation

The generator appends the following instructions to the shared rules. Each candidate is returned as a structured JSON item that records the competing answer, the third party it concerns, and why it does not answer the target question, which the verifier later checks.

Prompt template: distractor generation

```
Generate additions for the target questions ONLY. Choose after_id from a real
dialogue turn in the designated insertion_chunk, near relevant discussion.
Do not move original turns or insert before a session's date header.
```

```
Return one JSON object with exactly this structure:
{items: [{question_id: ..., speaker: ..., text: ...,
 after_id: cN_tN, confusion_type: person|event|object|relation,
 alternate_detail: the answer-shaped competing detail,
 competing_answer: only the competing answer value, not the person's name or explanation,
 third_party: explicit third-party subject whose fact is described,
 distinction: why this does not answer the target question,
 compatibility: why this can coexist with the original facts}]}
No markdown. Exactly K items for each target question.
```

The generator also receives the original conversation, all QA pairs, the target questions for the batch, and all previously accepted additions. On a retry, it additionally receives the revision feedback described in Appendix G.4.

G.3 Prompt Template for Distractor Verification

A separate reviewer checks every candidate before it is accepted. Like the generator, it begins with the shared rules, followed by the instructions below. Each candidate is evaluated against the full conversation, all QA pairs, and the other additions, and is rejected whenever its compatibility with the original facts is uncertain.

Prompt template: distractor verification

```
You are the skeptical reviewer, not the writer. Evaluate EACH proposed addition
against the FULL original conversation, ALL QA pairs, previously accepted
additions, and other proposed additions. Check:
1. No contradiction, chronology error, or incompatible invented personal fact.
2. No new correct/ambiguous answer to any existing question.
3. Strong topic overlap and a concrete competing answer-shaped detail.
4. A natural, clear entity/event/object/relation distinction without giveaways.
5. Natural wording and placement after the specified turn.
6. Distinct from the other additions for this question (including alternate detail).
Do not approve generic filler or mere paraphrases of the same altered fact.
Return JSON only:
{reviews: [{candidate_id: ..., accept: true,
 reason: specific assessment; if rejected, explain the needed correction}]}
Return exactly one review per candidate. Reject if uncertain about compatibility.
```

The reviewer receives the same conversation and QA context as the generator, together with the full list of candidates just generated, each identified by a `candidate_id`. The reason attached to each rejection is passed back to the generator, as described in Appendix G.4.

G.4 Prompt Template for Distractor Revision

Revision is not a separate prompt. When the reviewer rejects candidates, the generator is called again with the generation template of Appendix G.2, and a structured `revision_feedback` object is appended to its input. This object contains the full previous batch, each rejected candidate together with the reviewer's own rejection reason, and a fixed repair instruction.

Prompt template: distractor revision feedback

```
{
  previous_candidates: [ <every candidate item from the last attempt> ],
  issues_from_previous_attempts: [
    {
      candidate: { <the rejected candidate item, verbatim> },
      reason: <the reviewer's rejection reason>
    }
  ],
  repair_instruction: Return the COMPLETE batch, retaining valid candidates
    verbatim. Replace the specific sentences identified by these issues, rather
    than rewriting all candidates. Preserve every previously resolved
    correction. Compare each sentence with previously_accepted_additions to
    avoid duplicates. For binary yes/no questions the distinct third-party
    anecdotes may share the opposite answer polarity, but not the gold
    polarity. Every replacement still needs the same narrow topic and a
```

```
    distinct subject.
}
```

The list `issues_from_previous_attempts` contains one entry per rejected candidate. The generator's next attempt is therefore conditioned on the exact rejected text and the exact reason for each rejection, rather than on a generic instruction to retry.

G.5 Prompt Template for Retrieval-Head Identification

Retrieval heads are identified once, offline, on the held-out calibration instances (§4). For each instance, the frozen LLM answers the question from the full context using the template below, with greedy decoding and a budget of up to 32 new tokens.

Prompt template: retrieval-head identification

```
Answer the question using ONLY the context below. Respond with the shortest
possible exact answer (a name, date, number, or a few words, or 'yes'/'no'). Do
not explain, do not use full sentences.

Context:
{ctx}

Question: {question}

Output ONLY the final answer, nothing else.
```

A second forward pass is then run over the prompt together with the generated tokens, to capture the attention from every generated token back onto the context. For each (layer, head) pair, this attention is expressed as a ratio to the context average and aggregated over the calibration instances. The top 10% of heads under this ranking form the retrieval heads used in all subsequent experiments.

G.6 Prompt Template for Memory Answering

During RL training, the frozen LLM answers questions using the same template as in Appendix G.5. Here {ctx} is filled with the turns currently held in memory, in the order they were kept, and {question} with a probe question. The template is used twice at every step, with different contexts and generation budgets.

Feature extraction. Before the policy decides which segments to keep, {ctx} contains the memory so far together with the entire incoming window, and {question} is the sampled probe question. If no question is answerable yet, a generic probe is used instead: "What important facts, events, or dates are mentioned in this conversation?" The model generates up to 16 tokens with greedy decoding. These tokens serve only as the positions from which attention onto the context is read, as in Appendix G.5, to build the policy's state; the generated text is discarded and never scored.

Reward scoring. After the policy's keep/drop decisions are applied, {ctx} contains only the final memory for that step. A set of sampled probe questions is answered against it with greedy decoding. These answers are scored by normalized containment against the gold answers to produce the reward.