PREVIEW-GUIDED MEMORY: LEARNING TO REREAD FOR LONG-CONTEXT REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM agents accumulate long contexts as they read documents, use tools, and carry out multi-step tasks. Memory-based methods handle these contexts by reading text in chunks and maintaining a compact memory. However, single-pass readers must decide what to retain before seeing later chunks, while still processing every chunk regardless of its relevance. We propose Preview-Guided Memory (PGM), which uses a preliminary preview to guide both reading and memory updates. In the first pass, PGM generates a short, question-independent preview of each chunk and groups neighboring previews into pages. A learned selector uses these pages and the question to choose which chunks to read in detail in the second pass. The reader then processes the selected chunks in order, using the corresponding preview page when updating its memory. These previews provide clues about upcoming content, while selection reduces the number of memory updates. We keep the reader fixed and train the selector with supporting-fact supervision followed by reinforcement learning, rewarding answer quality and evidence coverage, with a bonus for reduced reading when the answer is correct. PGM-Selective-7B achieved state-of-the-art on both HQA and longbench v2 with fewer tokens for reasoning.

1 INTRODUCTION

LLM agents often work with long contexts containing documents, tool outputs, and previous interactions (Yao et al., 2023; Packer et al., 2023). As a task continues, this context grows, and information collected early may still be needed much later. Keeping everything in the prompt is straightforward, but repeatedly processing it becomes expensive and eventually reaches the model's context limit. Larger context windows allow longer inputs (Ding et al., 2024; Yang et al., 2025b), but do not remove the cost of reading them. Long-context reasoning therefore calls for methods that can retain useful information without keeping all previous text in the active context.

Memory-based methods address this problem by compressing previously read text into a short memory. MemAgent (Yu et al., 2026), for example, divides a document into chunks and reads them sequentially. At each step, the model receives the question, the current chunk, and the previous memory, then writes an updated memory for the next step. After reading the document, it answers from the final memory. ReMemR1 (Shi et al., 2026) allows the reader to revisit historical memories during reading. These methods keep individual model calls within a limited context window while carrying information across much longer documents.

However, reading and compressing one chunk at a time has two limitations. First, the model must decide what to keep before it has seen the content that comes next. Although the question tells it what to look for, the role of a detail may only become clear after reading another passage. A name or relation that appears unimportant in the current chunk may later be needed to connect two pieces of evidence. If that detail was never stored, revisiting historical memories may not recover it. Second, reading every chunk in detail requires repeated memory-update calls, even when many chunks contribute little to the answer. Each call still includes the question and accumulated memory. The reader therefore makes compression decisions with incomplete information about what lies ahead, while spending repeated computation on content it may not need.

We propose Preview-Guided Memory (PGM), which first skims a document to produce short, question-independent previews and then uses them to guide a second pass of reading. The previews serve two purposes. They give the reader clues about surrounding content, including passages not

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.

yet read in detail, when deciding what to retain in memory. They also help a learned selector use the question to choose which chunks to read. The reader then processes the selected chunks in their original order, using the previews to guide its memory updates. Preview generation still covers the full document; selection reduces the subsequent reading and memory updates. We keep the reader fixed and train the selector with supporting-fact supervision followed by reinforcement learning from the reader's final answers.

We evaluate PGM on HQA and LongBench v2. PGM-Selective-7B achieves higher accuracy with fewer tokens across both benchmarks. These results demonstrate that preview-guided selective reading improves the effectiveness and efficiency of memory-based reasoning across long contexts and diverse tasks.

2 Related Work

Memory-based long-context reasoning. Memory-based agents organize past information through memory stores or compact states (Packer et al., 2023; Zhong et al., 2024; Chhikara et al., 2025; Zhou et al., 2025). For document reasoning, MemAgent incrementally updates a textual memory (Yu et al., 2026); ReMemR1 revisits historical memories (Shi et al., 2026); and InfMem and MemReread retrieve or reread source evidence (Wang et al., 2026; Ji et al., 2026). PGM gives memory updates a view of neighboring content before it is read in detail, helping the reader anticipate which information to retain.

Retrieval and multi-pass reading. Retrieval-augmented methods select evidence to support generation (Karpukhin et al., 2020; Lewis et al., 2020; Izacard & Grave, 2021; Borgeaud et al., 2022; Shi et al., 2024). Self-RAG learns retrieval and reflection, while RAPTOR retrieves over hierarchical summaries (Asai et al., 2024; Sarthi et al., 2024). Closer to PGM, DOS RAG preserves source order (Laitenberger et al., 2025), ReadAgent uses gists for raw-page lookup (Lee et al., 2024), and ReadTwice shares first-pass representations across segments (Zemlyanskiy et al., 2021). PGM reuses the same textual previews to select raw chunks and guide subsequent memory updates.

Learning reading decisions. RL has been used to optimize long-context reasoning and search policies (Wan et al., 2025; Jin et al., 2025). LycheeMemory selects compressed key–value blocks (Chen et al., 2026), while GRU-Mem learns when to update memory or stop reading (Sheng et al., 2026). PGM initializes its selector with supporting facts, then uses the final answers of a fixed reader to evaluate complete chunk selections.

3 Method

Sequential memory as a starting point. MemAgent (Yu et al., 2026) handles a long document by compressing previously read text into a bounded memory. Given a question q and ordered source chunks $D = (c_1, \ldots, c_N)$, its reading process can be written as

$$\begin{aligned} m_i^{\text{seq}} &= U_{\text{seq}}(q, c_i, m_{i-1}^{\text{seq}}), \quad i = 1, \ldots, N, \\ \hat{y}_{\text{seq}} &= F_{\text{seq}}(q, m_N^{\text{seq}}), \qquad |m_i^{\text{seq}}| \le B. \end{aligned} \tag{1}$$

Starting from empty memory, each update rewrites the accumulated information within a B-token budget, and the final answer uses only the question and the last memory. This keeps individual calls short. However, a fact needed later must survive every intervening rewrite, even before the reader knows how it connects to the question.

Preview, then read and remember. These limitations suggest that finding evidence and retaining it should be considered together. PGM first builds short, question-independent previews, giving reading decisions a view of the document before detailed memory updates begin. The same previews then guide *which chunks to read* and *what to retain* during those reads. Selection shortens the sequence of recurrent updates, while neighboring previews provide context for interpreting the current passage. Figure 1 presents the general workflow.

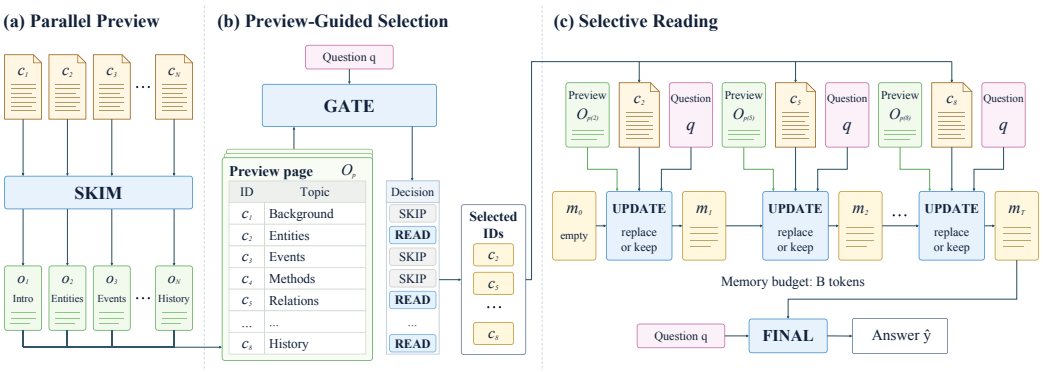

Figure 1: **Overview of Preview-Guided Memory (PGM).** (a) In the first pass, SKIM generates indexed, question-independent topical previews for all chunks in parallel. Neighboring previews are grouped into pages. (b) GATE uses the question and preview pages to select raw chunks before the second pass. (c) Starting from empty memory m_0, the reader processes the selected chunks in source order and updates memory within a B-token budget, using the corresponding preview page as context. The same previews guide both chunk selection and memory updates. FINAL produces $\hat{y}$ using only the question q and final memory m_T. Topic labels and selections are schematic; green arrows denote preview conditioning.

3.1 Preview before detailed reading

Let θ and ϕ denote the reader and selector parameters. Before updating memory, PGM builds a compact view of the source:

$$o_i = H_\theta(c_i), \qquad O_p = \{(i, o_i) : i \in I_p\}, \tag{2}$$

where o_i is a topical preview and I_p contains the indices of neighboring chunks grouped into page p. We write $p(i)$ for the page containing chunk i and O for all preview pages. Each page fits within a bounded input budget, so the preview representation can grow with the document without requiring a single long-context call.

Each preview describes a chunk's topics and points back to the raw source. Generating it without the question separates document previewing from task-specific selection. Neighboring previews reveal how topics develop across chunks; each preview depends only on its own chunk, so all first-pass calls can run in parallel.

3.2 Use the preview to select and remember

Select where to read. Once the previews are available, the question determines which parts warrant detailed reading. The selector receives one preview page at a time and scores its entries together in a single forward pass:

$$a_i = \sigma\big(\ell_\phi(q, O_{p(i)}, i)\big)\,, \tag{3}$$

where ℓ_ϕ is the selection logit and σ is the sigmoid function. These scores determine the selected set $S = \{s_1, \ldots, s_T\}$, with $s_1 < \cdots < s_T$. All page decisions are completed before the second pass. The selector therefore chooses the reading sequence from the question and previews, while the reader subsequently processes that sequence in source order.

Decide what to remember. Selection is only the first use of the previews. A relevant chunk may contain several details, and the reader still has to decide which ones deserve space in memory. PGM supplies the corresponding preview page again at each update:

$$\begin{aligned} m_t &= U_\theta(q, c_{s_t}, m_{t-1}, O_{p(s_t)}), \\ \hat{y} &= F_\theta(q, m_T), \qquad |m_t| \le B. \end{aligned} \tag{4}$$

Starting from empty memory m_0, the reader either replaces its memory with an updated version or keeps it unchanged. This single memory carries across preview pages; after the last selected chunk, the final answer is generated from the question and m_T.

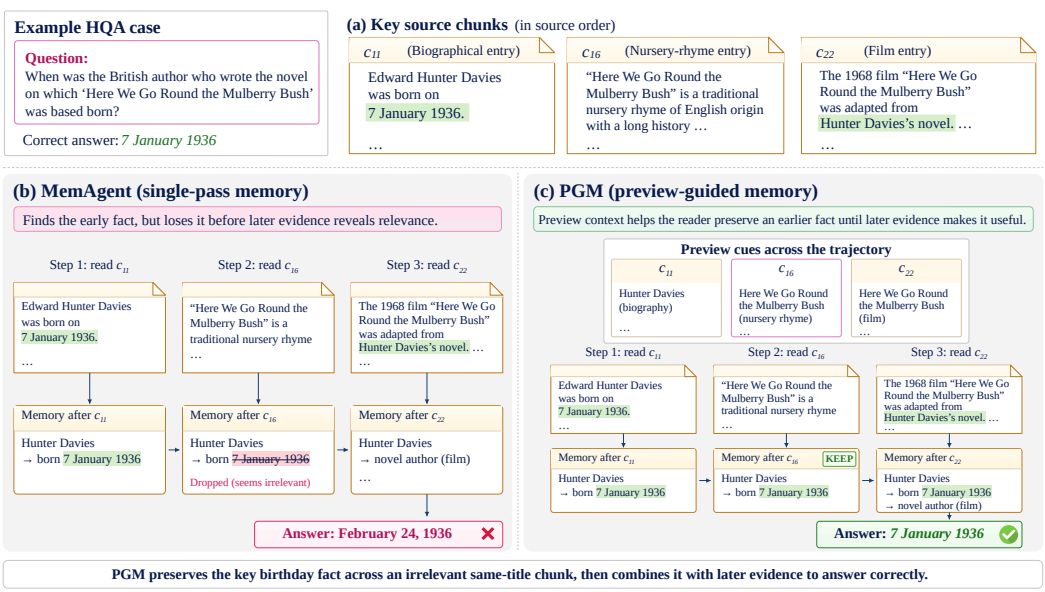

Figure 2: **Motivating HQA case.** (a) Key source chunks. (b) MemAgent loses an initially stored birth date before the film entry identifies the author. (c) PGM-Selective retains the date through the same-title distractor and combines it with the later author relation. Preview cues are summarized across pages, not reproduced from one page; the film preview supplies neither the author relation nor the date. Only key states are shown; intermediate updates are omitted. PGM reads seven of 23 chunks in the second pass. Appendix C gives the run settings and local rewrite control for this illustrative diagnostic pair.

The inputs have complementary roles: previews provide surrounding topic cues, raw chunks supply new evidence, and memory carries earlier facts across selected chunks and page boundaries. The update prompt treats previews as navigation rather than evidence: old facts must come from the previous memory, and new facts from the current raw chunk. A preview can therefore guide retention without itself supplying the answer or a missing relation.

We call this workflow PGM-Selective. PGM-ReadAll retains the same previews and memory-update procedure but reads every source chunk. This full-reading reference lets us examine the benefit of selection within the preview-guided workflow.

When relevance becomes clear too late. Figure 2 compares PGM with MemAgent on a real 112K-token HQA trajectory. The question asks for the birth date of the author whose novel was adapted into *Here We Go Round the Mulberry Bush*. MemAgent stores Hunter Davies's birth date, *7 January 1936*, at chunk 11, but drops it while processing a same-title nursery-rhyme entry at chunk 16. When chunk 22 identifies Davies as the novel's author, the date is absent and its final answer is incorrect. The fact was extracted, then lost before later evidence established its relevance.

On the same question and source document, PGM retains Davies's birth date, combines it with the later author relation, and answers correctly after reading seven of the 23 chunks in its second pass. Between chunks 11 and 22, MemAgent performs ten intervening updates, whereas PGM reads only chunks 13 and 16 before reaching chunk 22. PGM therefore succeeds while still processing the same-title distractor, rather than by excluding every distractor. Its preview page at chunk 16 lists the later film entry but not the author relation or birth date; those facts come from the raw source. Appendix C preserves the run settings and local rewrite control.

3.3 Training the Reader and Selector

From full reading to selective reading. We first train PGM-ReadAll with SFT followed by RL, then initialize selective training from this checkpoint. Selective SFT learns chunk selection from

supporting facts, followed by Selective RL using final-answer feedback. Thus, the Selective SFT-only ablation already inherits both ReadAll training stages.

Start by locating evidence. We align supporting facts with the source chunks and assign $y_i^{\mathrm{ev}} = 1$ to chunks that contain them. A weighted binary cross-entropy objective initializes the selector:

$$\mathcal{L}_{\mathrm{ev}} = -\frac{1}{N}\sum_{i=1}^{N}\left[w_+ y_i^{\mathrm{ev}} \log a_i + (1 - y_i^{\mathrm{ev}}) \log(1 - a_i)\right]. \tag{5}$$

The positive-class weight $w_+ > 1$ emphasizes retaining evidence. This provides a direct learning signal before the selector has learned to compare complete reading trajectories.

Evaluate a set through its final answer. For each training question, we sample $K > 1$ chunk sets. Each set produces its own source-ordered memory sequence and final answer through Equation 4. Writing $z_i = \mathbf{1}[i \in S]$, the log probability of a sampled selection is

$$\log \pi_\phi(S \mid q, O) = \sum_{i \in \mathcal{I}} \left[z_i \log p_i + (1 - z_i) \log(1 - p_i)\right], \tag{6}$$

where $\mathcal{I}$ contains the stochastic selection decisions and p_i is a smoothed version of a_i. This probability includes both reading and skipping decisions across preview pages. We optimize the expected reward $\mathcal{J}(\phi) = \mathbb{E}_{S\sim\pi_\phi}[R(S)]$, so feedback is attached to the complete selection rather than to isolated relevance scores.

The reward combines answer quality, evidence coverage, and economical reading. Let C be normalized answer exact-match correctness, F answer F1, and C_{all} the correctness of the paired PGM-ReadAll answer. For each of the J supporting facts, let $g_j(S)$ indicate whether the selected chunks contain at least one complete occurrence. We measure the fraction of facts covered, whether all facts are covered, and the fraction of source tokens skipped as

$$G(S) = \frac{1}{J}\sum_{j=1}^{J} g_j(S), \qquad E(S) = \prod_{j=1}^{J} g_j(S), \qquad s(S) = 1 - \frac{\sum_{i\in S}|c_i|}{\sum_{i=1}^{N}|c_i|}. \tag{7}$$

The trajectory reward is

$$\begin{aligned} R(S) = {} & \alpha C - \beta C_{\mathrm{all}}(1 - C) + \gamma(1 - C)F + \delta G(S) \\ & - \eta\big(1 - E(S)\big) + \lambda_u C\, s(S) - \zeta V, \end{aligned} \tag{8}$$

where V indicates an invalid answer format. The nonnegative coefficients weight correctness, regression from full reading, partial-answer credit, coverage, missing evidence, and invalid formatting, respectively; λ_u controls the reading bonus at training update u.

Each part addresses a different consequence of selection. Answer feedback measures whether the evidence remains usable after memory updates, with an additional penalty for losing an answer that full reading gets right. Coverage rewards retaining the links needed for multi-hop reasoning. Crucially, the reading bonus is multiplied by C: shorter reading is rewarded when it preserves a correct answer. The selector thus learns to remove unnecessary reading while retaining evidence that the downstream reader can use.

Compare alternative reading sequences. For rewards $R_1, \ldots, R_K$ from the same question, a leave-one-out baseline compares each selection with its alternatives (Williams, 1992; Ahmadian et al., 2024):

$$A_k = R_k - \frac{1}{K-1}\sum_{\ell \neq k} R_\ell. \tag{9}$$

A selection that outperforms the other samples receives a positive advantage; a weaker selection receives a negative one. We use these advantages as fixed policy-gradient weights in the regularized objective

$$\mathcal{L} = -\frac{1}{K}\sum_{k=1}^{K} A_k \log \pi_\phi(S^{(k)} \mid q, O) + \lambda_{\mathrm{KL}}\overline{D}_{\mathrm{KL}} + \lambda_{\mathrm{ev}}\mathcal{L}_{\mathrm{ev}} - \lambda_{\mathrm{ent}}\overline{\mathcal{H}}. \tag{10}$$

Here, $\overline{D}_{\mathrm{KL}}$ anchors the policy to the supervised selector and $\overline{\mathcal{H}}$ encourages exploration; both are averaged over chunks, with nonnegative regularization weights. Retaining $\mathcal{L}_{\mathrm{ev}}$ preserves the direct evidence signal while answer feedback refines set-level decisions. The trained selector changes which chunks enter the memory-update sequence, allowing the same reader to operate on a more useful selection. Appendix B gives the label alignment and training settings.

3.4 WHY PREVIEWING BEFORE COMPRESSION CAN HELP

The distinction between Equations 1 and 4 is the timing of information: a preview may reveal which earlier detail will become relevant before that detail is compressed. We formalize this through a delayed-reference task, using a classical random-access coding argument (Nayak, 1999). A B-token memory over a finite vocabulary has a finite state space; b below measures its capacity in bits.

Proposition 1 (A memory advantage from an early dependency cue)**.** *Let the first of two chunks contain d independent uniform bits $X = (X_1, \ldots, X_d)$, and let the second contain an independent uniform index $Z \in \{1, \ldots, d\}$. The fixed question asks for $Y = X_Z$. A single-pass reader retains a state W with at most 2^b possible values after the first chunk and has no other access to X. Its minimum average answer error e satisfies*

$$b \geq d\big[1 - h_2(e)\big], \tag{11}$$

where $h_2(e) = -e \log_2 e - (1-e)\log_2(1-e)$ is binary entropy. In contrast, if the preview page exposes Z before the first memory update, a preview-guided reader can answer exactly using only two memory states.

Proof. Let e_j be the optimal error for recovering X_j from W. Because Z is uniform and independent of (X, W), $e = d^{-1}\sum_j e_j$. Binary decoding gives $H(X_j \mid W) \leq h_2(e_j)$: given W, the predicted bit and its error indicator determine X_j. Independence of the source bits, conditional entropy subadditivity, and concavity of h_2 imply

$$\begin{aligned} b \geq H(W) \geq I(X; W) &= d - H(X \mid W) \\ &\geq d - \sum_{j=1}^{d} h_2(e_j) \geq d\big[1 - h_2(e)\big]. \end{aligned} \tag{12}$$

For the preview-guided construction, group the two previews on one page and let the second preview report $o_2 = Z$. When reading the first raw chunk, write X_Z into memory; keep that bit during the second update and output it at the end. The first preview need not retain any of the bits in X. □

With fewer than d bits of retained information, the single-pass reader cannot attain zero error on this task. The preview-guided construction retains only the needed bit: the preview identifies *which* detail matters, while its value still comes from the raw chunk. Both readers can process the same two chunks, so the separation isolates preview-conditioned memory writing from chunk selection. In PGM, a neighboring preview can play this role by exposing an entity or relation needed later. If that dependency cue is decoded incorrectly with probability at most ε, the same construction has error at most ε.

Accuracy takes precedence in the selection reward. The reading bonus also has an exact property that follows from the objective, independently of how the selector is optimized.

Proposition 2 (Correctness-prioritized trajectory reward)**.** *Suppose all reward coefficients are nonnegative and $\alpha > \gamma + \delta + \eta + \zeta$. For any two trajectories of the same question, with $C(S^+) = 1$ and $C(S^-) = 0$, Equation 8 satisfies*

$$R(S^+) - R(S^-) \geq \alpha - \eta - \zeta - \gamma - \delta > 0. \tag{13}$$

This ordering holds for every nonnegative reading-bonus weight λ_u.

Proof. For a correct answer, the regression and partial-answer terms vanish, and $R(S^+) \geq \alpha - \eta - \zeta$. For an incorrect answer, the reading bonus vanishes, the remaining penalties are nonpositive, and $R(S^-) \leq \gamma + \delta$. Subtracting proves the claim. □

The training coefficients in Appendix B satisfy this condition. Thus, a shorter failed trajectory cannot outrank a successful one through the reading bonus; among otherwise equivalent successful trajectories, an active bonus favors less reading.

3.5 How previewing changes the reading cost

Previewing adds an initial pass, but allows the second pass to contain fewer recurrent updates. To see this trade-off, consider MemAgent, which updates memory on all N source chunks (Yu et al., 2026). Let c_m be its representative input-plus-output token cost per update. For PGM, let c_s, c_g, and c_u be the corresponding costs of generating one preview, scoring one preview page, and updating memory on one selected chunk. With M preview pages and $T = |S|$ selected chunks,

$$\begin{aligned} C_{\text{MemAgent}} &\simeq Nc_m + C_f^{\text{MemAgent}}, \\ C_{\text{PGM}} &\simeq Nc_s + Mc_g + Tc_u + C_f^{\text{PGM}}, \end{aligned} \tag{14}$$

where C_f denotes final-answer cost. Each term includes its prompt and output tokens. Under fixed chunk, preview-page, memory, and generation budgets, both workflows scale as $O(N)$.

Writing $\rho = T/N$, and neglecting the final-answer terms for long documents, gives

$$\frac{C_{\text{PGM}}}{C_{\text{MemAgent}}} \simeq \frac{c_s}{c_m} + \frac{M}{N}\frac{c_g}{c_m} + \rho\frac{c_u}{c_m}. \tag{15}$$

The first two terms build and use the preview index; the last pays for detailed reading. PGM previews every chunk but concentrates recurrent updates on the selected sequence. The experiments measure the full cost.

The comparison with PGM-ReadAll further isolates the savings from selection. In the same representative-cost model, full reading uses $Nc_s + Nc_u + C_f^{\text{ReadAll}}$ tokens, so

$$C_{\text{ReadAll}} - C_{\text{PGM}} \simeq (N - T)c_u - Mc_g + C_f^{\text{ReadAll}} - C_f^{\text{PGM}}. \tag{16}$$

The shared preview pass cancels. With similar final-answer costs, selection saves tokens when the skipped updates outweigh the selector's Mc_g overhead. Savings thus come from reducing recurrent reading and rewriting after the initial scan.

4 Experiments

We evaluate PGM on HQA and LongBench v2 to study answer quality and end-to-end token use. We compare against long-context and memory-based baselines, analyze inference cost, and examine the effects of training and chunk selection.

4.1 Experimental setup

Benchmarks. We evaluate HQA at 28K, 112K, 224K, 448K, and 896K tokens, with 128 questions at each length (Yang et al., 2018; Yu et al., 2026). We report substring exact match (SubEM) and its equal-weight average over the five lengths. LongBench v2 contains 503 multiple-choice questions across six task categories (Bai et al., 2025). We report overall accuracy and breakdowns by task category, context length, and difficulty.

Baselines. We compare with Qwen2.5 and Qwen2.5-1M (Yang et al., 2025a;b), BM25 and dense DOS RAG (Laitenberger et al., 2025), and the memory-based methods ReadAgent, MemAgent, LycheeMemory, ReMemR1, InfMem, and MemReread, discussed in Section 2. PGM-ReadAll uses the same previews and memory-update procedure as PGM-Selective, but reads every chunk in the second pass. Implementation details are provided in Appendix A.

Token cost. We count all input and output tokens for preview generation, selection, memory updates, and final answering. HQA cost is averaged over the same five lengths as SubEM and normalized to Qwen2.5-7B-1M. In the ablations, the reading ratio is the fraction of raw-source tokens read in the second pass.

Table 1: HQA SubEM (%) across context lengths. Avg. and cost use the same five lengths (28K–896K); cost is normalized to Qwen2.5-7B-1M.

Method	28K	112K	224K	448K	896K	Avg.↑	Cost↓
Qwen2.5-7B	46.09	10.16	2.34	1.56	3.12	12.65	0.0499×
Qwen2.5-7B-1M	50.78	46.88	32.03	9.38	14.84	30.78	1.0000×
BM25-DOS RAG	46.09	40.62	17.97	9.38	0.78	22.97	0.0204×
Dense-DOS RAG	53.91	55.47	25.78	16.41	3.91	31.10	0.0214×
ReadAgent-7B	39.06	28.91	24.22	10.16	4.69	21.41	3.1363×
MemAgent-7B	79.69	75.78	75.78	72.66	**75.00**	75.78	0.8203×
LycheeMemory-7B	60.16	50.00	45.31	42.19	40.62	47.66	0.8569×
ReMemR1-7B	82.03	**81.25**	74.22	74.22	71.88	76.72	1.1280×
InfMem-4B	68.75	68.75	67.97	67.19	69.53	68.44	0.3641×
MemReread-4B	57.03	45.31	42.97	39.84	38.28	44.69	2.4484×
PGM-ReadAll-7B	82.03	80.47	75.78	67.97	68.75	75.00	1.8928×
PGM-Selective-7B	**82.81**	**81.25**	**76.56**	**78.13**	**75.00**	**78.75**	1.0549×

Table 2: LongBench v2 accuracy (%): overall and by task category, context length, and difficulty.

	Overall	Task category						Context length			Difficulty	
Method	All (n=503)	Single-doc QA (n=175)	Multi-doc QA (n=125)	Long-context learning (n=81)	Dialogue history (n=39)	Code repo. (n=50)	Structured data (n=33)	Short (n=180)	Medium (n=215)	Long (n=108)	Easy (n=192)	Hard (n=311)
Qwen2.5-7B	28.83	33.71	27.20	**33.33**	12.82	32.00	12.12	36.67	25.12	23.15	32.29	26.69
Qwen2.5-7B-1M	30.42	36.57	26.40	30.86	10.26	**40.00**	21.21	39.44	24.19	27.78	35.42	27.33
BM25-DOS RAG	28.23	36.57	25.60	27.16	10.26	28.00	18.18	30.56	27.91	25.00	32.81	25.40
Dense-DOS RAG	30.02	36.00	28.00	25.93	20.51	32.00	24.24	35.56	26.05	28.70	31.25	29.26
ReadAgent-7B	29.03	36.00	28.80	25.93	25.64	20.00	18.18	**41.11**	22.33	22.22	30.73	27.97
MemAgent-7B	29.03	32.00	24.80	17.28	35.90	32.00	45.45	29.44	28.37	29.63	**35.94**	24.76
LycheeMemory-7B	8.75	8.57	9.60	13.58	5.13	8.00	0.00	6.67	9.77	10.19	10.94	7.40
ReMemR1-7B	26.04	24.00	28.00	20.99	20.51	36.00	33.33	26.11	25.12	27.78	30.21	23.47
InfMem-4B	29.22	30.29	26.40	24.69	25.64	30.00	**48.48**	29.44	26.98	33.33	32.29	27.33
MemReread-4B	28.63	30.29	24.80	24.69	28.21	36.00	33.33	27.22	28.37	31.48	34.90	24.76
PGM-ReadAll-7B	34.59	**38.86**	27.20	32.10	33.33	36.00	45.45	35.00	32.56	**37.96**	35.42	**34.08**
PGM-Selective-7B	**34.79**	37.71	**30.40**	29.63	**41.03**	34.00	42.42	32.78	**35.81**	36.11	**35.94**	**34.08**

4.2 Main results

HQA. Table 1 shows that PGM-Selective-7B achieves 78.75 average SubEM, compared with 75.00 for PGM-ReadAll-7B, 76.72 for ReMemR1, and 75.78 for MemAgent. It matches or exceeds the strongest non-PGM baseline at every evaluated length. The gains over ReadAll are largest at 448K and 896K, reaching 10.16 and 6.25 percentage points. Selection thus improves answer quality while reducing the number of chunks passed through the memory-update sequence.

Across the 32-fold increase from 28K to 896K, Selective declines from 82.81 to 75.00 SubEM, compared with 82.03 to 68.75 for ReadAll. The advantage at the two longest lengths is consistent with prioritizing a useful reading sequence over repeatedly processing every chunk.

LongBench v2. In Table 2, PGM-Selective-7B obtains the highest overall accuracy among the evaluated methods, at 34.79%, compared with 34.59% for PGM-ReadAll-7B and 30.42% for Qwen2.5-7B-1M. Relative to ReadAll, it improves multi-document QA from 27.20% to 30.40%, dialogue history from 33.33% to 41.03%, and medium-length contexts from 32.56% to 35.81%. ReadAll remains stronger on single-document QA and long contexts; both configurations achieve 34.08% on hard questions. Across this broader task mix, selection preserves comparable overall accuracy with fewer tokens. Relative to MemAgent, the gain is concentrated in the hard subset: accuracy rises from 24.76% to 34.08%, while both methods obtain 35.94% on easy questions.

4.3 Token efficiency

Savings from selective reading. Relative to PGM-ReadAll-7B, PGM-Selective-7B reduces total token use by 44.27% on HQA and 34.76% on LongBench v2. On HQA, its cost is approximately $0.77\times$ that of ReadAll at 28K and 0.55–$0.56\times$ at 112K–896K. At 896K, Selective uses 1.65M tokens

per query instead of 2.98M, while increasing SubEM from 68.75 to 75.00. The reduced second-pass workload therefore yields substantial end-to-end savings as context length grows.

Comparison with memory-based readers. Across the five HQA lengths, Selective improves on ReMemR1 by 2.03 SubEM points while using 6.48% fewer tokens. ReMemR1 is cheaper at 28K–448K, whereas Selective uses 16.26% fewer tokens at 896K. Relative to MemAgent, Selective raises average HQA SubEM from 75.78 to 78.75 with 28.61% more tokens; on LongBench v2, it raises accuracy from 29.03% to 34.79% with 26.99% more tokens. PGM trades additional preview computation for higher answer quality, while learned selection reduces the cost of preview-guided reading.

4.4 Ablation studies

We study training and selection on HQA at 112K and 224K. Table 3 pairs the answer-quality results with reading ratios and score differences. Random-selection scores are means and standard deviations over five seeds.

Table 3: HQA ablations. Answer scores are SubEM (%); reading ratios measure second-pass source tokens. Score differences are in percentage points.

(a) Answer quality

Method	112K	224K
PGM-Selective		
SFT-only	78.13	74.22
SFT+RL	**81.25**	**76.56**
Random chunks	43.91 ± 3.64	41.72 ± 3.69
Golden chunks	87.50	85.94
PGM-ReadAll		
SFT-only	74.22	74.22
SFT+RL	**80.47**	**75.78**

(b) Reading amount and gains

Metric	112K	224K
Reading ratio (%)		
SFT+RL	46.22	44.16
Random	45.66	43.89
SubEM difference (pp)		
RL gain (Selective)	$+3.13$	$+2.34$
RL gain (ReadAll)	$+6.25$	$+1.56$
Learned − Random	$+37.34$	$+34.84$
Golden − Learned	$+6.25$	$+9.38$

Learning from answer feedback. With the reader fixed, RL improves the selector's SubEM from 78.13 to 81.25 at 112K and from 74.22 to 76.56 at 224K, gains of 3.13 and 2.34 percentage points. Final-answer feedback complements supporting-fact supervision by evaluating whether the selected chunks support a correct answer after memory updates.

Selecting useful content. SFT+RL reads 46.22% of source tokens at 112K and 44.16% at 224K, close to the random baseline's 45.66% and 43.89%. Yet its SubEM is higher by 37.34 and 34.84 points. Which chunks are read, rather than reading less alone, is central to the result. Selecting golden chunks yields 87.50 and 85.94 SubEM, leaving gaps of 6.25 and 9.38 points to learned selection.

Training the full reader. The PGM-ReadAll block in Table 3 compares its SFT-only and SFT+RL results. RL improves SubEM by 6.25 points at 112K and 1.56 points at 224K. Together, the two training comparisons show gains from answer feedback for both full reading and learned selection.

5 Conclusion

PGM learns which source chunks a bounded-memory reader should consolidate. Indexed previews support admission and memory updating, while terminal answer feedback evaluates the selected source set through a fixed reader. Relative to PGM-ReadAll, PGM-Selective improves HQA answer quality and reduces token use, while maintaining comparable accuracy at lower token cost on LongBench v2. Learning source admission is therefore an effective approach to allocating inference tokens for long-context reasoning.

AI USE STATEMENT

We used an AI assistant to support manuscript preparation and improve the clarity of the writing. The authors take full responsibility for the content and conclusions.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.662. URL https://aclanthology.org/2024.acl-long.662/.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/25f7be9694d7b32d5cc670927b8091e1-Abstract-Conference.html.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. 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. 3639–3664, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.183. URL https://aclanthology.org/2025.acl-long.183/.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 2206–2240. PMLR, 2022. URL https://proceedings.mlr.press/v162/borgeaud22a.html.

Zhuoen Chen, Dongfang Li, Meishan Zhang, Baotian Hu, and Min Zhang. Dynamic long context reasoning over compressed memory via end-to-end reinforcement learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8064–8083. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.365. URL https://aclanthology.org/2026.acl-long.365/.

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

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. LongRoPE: Extending LLM context window beyond 2 million tokens. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11091–11104. PMLR, 2024. URL https://proceedings.mlr.press/v235/ding24i.html.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://www.microsoft.com/en-us/research/publication/lora-low-rank-adaptation-of-large-language-models/.

Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European*

Chapter of the Association for Computational Linguistics: Main Volume, pp. 874–880. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.eacl-main.74. URL https://aclanthology.org/2021.eacl-main.74/.

Baibei Ji, Xiaoyang Weng, Juntao Li, Zecheng Tang, Yihang Lou, and Min Zhang. MemReread: Enhancing agentic long-context reasoning via memory-guided rereading. *arXiv preprint arXiv:2605.10268*, 2026. URL https://arxiv.org/abs/2605.10268.

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. URL https://arxiv.org/abs/2503.09516.

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

Alex Laitenberger, Christopher D. Manning, and Nelson F. Liu. Stronger baselines for retrieval-augmented generation with long-context language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 32559–32569. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.1656. URL https://aclanthology.org/2025.emnlp-main.1656/.

Kuang-Huei Lee, Xinyun Chen, Hiroki Furuta, John Canny, and Ian Fischer. A human-inspired reading agent with gist memory of very long contexts. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 26396–26415. PMLR, 2024. URL https://proceedings.mlr.press/v235/lee24c.html.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Ashwin Nayak. Optimal lower bounds for quantum automata and random access codes. *arXiv preprint arXiv:quant-ph/9904093*, 1999. URL https://arxiv.org/abs/quant-ph/9904093.

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

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2401.18059.

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

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. REPLUG: Retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8371–8384. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.naacl-long.463. URL https://aclanthology.org/2024.naacl-long.463/.

Yaorui Shi, Yuxin Chen, Siyuan Wang, Sihang Li, Hengxing Cai, Qi Gu, Xiang Wang, and An Zhang. Look back to reason forward: Revisitable memory for long-context LLM agents. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/7e0dc9ccba0f1333be13a3f9dc2b3138-Abstract-Conference.html.

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. URL https://arxiv.org/abs/2505.17667.

Xinyu Wang, Mingze Li, Peng Lu, Xiao-Wen Chang, Lifeng Shang, Jinping Li, Fei Mi, Prasanna Parthasarathi, and Yufei Cui. InfMem: Learning system-2 memory control for long-context agent. *arXiv preprint arXiv:2602.02704*, 2026. URL https://arxiv.org/abs/2602.02704v2. Accepted to COLM 2026.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

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. *arXiv preprint arXiv:2412.15115*, 2025a. URL https://arxiv.org/abs/2412.15115v2.

An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, Junyang Lin, Kai Dang, Kexin Yang, Le Yu, Mei Li, Minmin Sun, Qin Zhu, Rui Men, Tao He, Weijia Xu, Wenbiao Yin, Wenyuan Yu, Xiafei Qiu, Xingzhang Ren, Xinlong Yang, Yong Li, Zhiying Xu, and Zipeng Zhang. Qwen2.5-1M technical report. *arXiv preprint arXiv:2501.15383*, 2025b. URL https://arxiv.org/abs/2501.15383v1.

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 *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and Hao Zhou. MemAgent: Reshaping long-context LLM with multi-conv RL-based memory agent. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/4264ee4376776907c0b87ed70b959585-Abstract-Conference.html.

Yury Zemlyanskiy, Joshua Ainslie, Michiel de Jong, Philip Pham, Ilya Eckstein, and Fei Sha. ReadTwice: Reading very large documents with memories. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5189–5195. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.408. URL https://aclanthology.org/2021.naacl-main.408/.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: Enhancing large language models with long-term memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19724–19731, 2024. doi: 10.1609/aaai.v38i17.29946. URL https://ojs.aaai.org/index.php/AAAI/article/view/29946.

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. *arXiv preprint arXiv:2506.15841*, 2025. URL https://arxiv.org/abs/2506.15841.

A Implementation details

Prompt interfaces. Templates are included in `prompts/`. The preview prompt requests ordered topics without the question. The selector receives the question, page outline, focused entries, and decision slots; the update prompt receives the raw chunk, question, memory, local section index, and page outline. The final prompt uses only the question and memory and requests a boxed answer. Pages split recursively to fit the input budget, preserving complete entries.

Decision slots. At each `<gate_slot>` end marker, fixed vocabulary output rows w_R, w_S give the logit $\ell_i = (w_R - w_S)^\top h_i + b_\phi$ and score $a_i = \sigma(\ell_i)$. Scores are extracted directly in one forward pass per page. No autoregressive decision tokens are generated.

Inference selection. Given threshold τ, let $S_\tau = \{i : a_i \geq \tau\}$ and let $\mathcal{F}$ contain chunks whose previews reach the output cap. Inference includes these chunks and falls back to the highest-scoring chunk if the resulting set is empty:

$$\widetilde{S} = S_\tau \cup \mathcal{F}, \qquad S = \begin{cases} \widetilde{S}, & \widetilde{S} \neq \varnothing, \\ \{\arg\max_i a_i\}, & \widetilde{S} = \varnothing. \end{cases} \tag{17}$$

The highest-score fallback applies at inference. Training can sample an empty set when no chunk is forced.

Memory-update runtime. In Equation 4, a standalone `<KEEP_MEMORY>` response preserves memory after whitespace stripping and case normalization. Other responses replace it, including those reaching the output cap. Memory carries across preview pages.

B Training settings

Trainable parameters. During selective SFT and selective RL, we update the selector's attention LoRA adapters (Hu et al., 2022) and scalar bias while keeping the reader fixed. Objectives are defined in Section 3.3.

Evidence alignment. For supporting fact e_j, let $\mathcal{A}_j$ contain its alternative source occurrences. Each occurrence is a set of chunk indices containing the full fact, including every chunk needed by a boundary-crossing sentence. Alignment uses question/answer identity, document title, and sentence offsets. The labels in Equation 5 and coverage indicators in Equation 7 are

$$y_i^{\mathrm{ev}} = \mathbf{1}\left[i \in \bigcup_j \bigcup_{A \in \mathcal{A}_j} A \right], \qquad g_j(S) = \mathbf{1}[\exists A \in \mathcal{A}_j : A \subseteq S]. \tag{18}$$

Objective coefficients. We use $w_+ = 3$ in Equation 5; $(\alpha, \beta, \gamma, \delta, \eta, \zeta) = (2, 2, 0.1, 0.2, 0.5, 0.25)$ in Equation 8; and $(\lambda_{\mathrm{KL}}, \lambda_{\mathrm{ev}}, \lambda_{\mathrm{ent}}) = (0.05, 0.02, 0.002)$ in Equation 10.

Sampling and reading bonus. For each nonforced chunk, $p_i = 0.04 + 0.92 a_i$. Chunks in $\mathcal{F}$ from Appendix A are always selected, so $\mathcal{I} = \{1, \ldots, N\} \setminus \mathcal{F}$ in Equation 6. The reading-bonus schedule in Equation 8 is $\lambda_u = 0.25 \min(1, u/32)$ for training updates $u = 1, 2, \ldots$. All component scores use a 0–1 scale.

Loss aggregation. In Equation 10, the KL term compares the transformed Bernoulli probabilities with those of the supervised reference. KL, entropy, and evidence regularization include forced chunks and average over chunks. Advantages are held fixed when differentiating the policy loss. The final loss averages over questions, with one selector update per sampled batch.

C PAIRED TRAJECTORY CASE STUDY

Question and runs. The motivating example in Figure 2 is HQA source `5ae63dad55429929b0807afe` at 112K tokens. It asks: *When was the British author who wrote the novel on which "Here We Go Round the Mulberry Bush" was based born?* The correct answer is 7 January 1936. The paired runs use the same source document; the three key chunks (11, 16, and 22) were checked to be identical despite differences in the full-document chunking implementations. The MemAgent trajectory uses the public MemAgent-7B weights and original single-pass prompts in an independent diagnostic run. The PGM trajectory uses the supervised selector before reinforcement learning (selector step 2944 and reader `stage2/global_step_5`). The MemAgent trajectory follows the sampled evaluation protocol, whereas PGM uses greedy decoding. These case-study runs do not replace the benchmark results.

Visible evidence and memory. Chunk 11 states that Edward Hunter Davies was born on 7 January 1936. Chunk 16 contains the same-title nursery rhyme, and chunk 22 identifies the film as an adaptation of Davies's novel. PGM selects chunks 4, 8, 9, 11, 13, 16, and 22, covering 31.38% of raw-source tokens in the second pass. Between chunks 11 and 22, MemAgent performs ten intervening updates; PGM reads only chunks 13 and 16. Its page at chunk 11 covers chunks 1–12; the page at chunk 16 covers 13–23 and lists the later film title without the author relation or birth date. PGM retains the date at chunk 16 and adds the author relation at chunk 22. Both facts occur in its final memory. Its answer is 7 January 1936; the original MemAgent answer is February 24, 1936. The diagnostic records preserve the source passages, actual preview pages, selection decisions, and intermediate memories.

Local rewrite control. Replaying PGM reproduces all seven memory updates and the final answer. We then require an explicit rewrite at chunk 16, fixing all other inputs, selected chunks, reader weights, preceding updates, budgets, and greedy decoding. An instruction-only attempt still returns `<KEEP_MEMORY>`. The effective intervention combines the rewrite instruction with a ban on the first-token < entry used by KEEP; it also excludes other outputs starting with that token, without constraining subsequent tokens or inserting evidence. Chunk 22 and final answering use their original prompts with the resulting memory. The model rewrites the memory, retains the birthday, and answers correctly. Evidence retention in this case therefore does not depend on taking the explicit KEEP branch.