ADAMEM: ADAPTIVE MEMORY ORGANIZATION FOR SCALABLE LONG-CONTEXT REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) face fundamental scalability challenges when reasoning over extremely long contexts, where processing the full sequence incurs prohibitive computational and memory costs. Existing context compression methods alleviate this burden by transforming long contexts into compact memory representations, but they typically rely on fixed segmentation and uniform compression strategies, leading to semantic fragmentation and inefficient information preservation. To address these limitations, we propose AdaMem, an adaptive memory organization framework that enables scalable long-context reasoning by dynamically organizing, allocating, and preserving contextual information. AdaMem first identifies semantically coherent units through boundary-aware chunking, avoiding information fragmentation caused by rigid partitioning. It then estimates the information novelty of each chunk and adaptively allocates memory budgets, allowing critical evidence to retain richer representations while compressing redundant content aggressively. Finally, AdaMem accumulates chunk-level memories and introduces selective replay to recover original contexts only when compressed memories are insufficient. Extensive experiments on long-context question answering benchmarks up to 1M tokens demonstrate that AdaMem consistently outperforms state-of-the-art long-context models, achieving superior reasoning accuracy with substantially improved efficiency.

1 INTRODUCTION

Large language models (LLMs) have demonstrated remarkable capabilities across a wide range of knowledge-intensive tasks (Khattab et al., 2022), motivating their deployment in increasingly complex real-world scenarios, such as long-horizon workflows and large-scale cross-document reasoning (Wang et al., 2023; Zhao et al., 2026). These applications require models to continuously process and leverage massive amounts of contextual information accumulated over extended interactions (Xiong et al., 2026). However, the standard self-attention mechanism in Transformers (Vaswani et al., 2017) incurs quadratic computational and memory complexity with respect to sequence length, resulting in substantial inference latency and resource consumption for long-context scenarios (Ge et al., 2024a; Behrouz et al., 2026). This fundamental scalability bottleneck limits the further expansion of LLMs, particularly in environments with constrained computational and memory resources.

To improve long-context efficiency, existing studies have explored methods that optimize attention computation patterns (Wang et al., 2026), such as sparse attention (Kitaev et al., 2020; Tay et al., 2022) and linear attention (Wang et al., 2020; Sun et al., 2023), enabling models to process longer sequences with reduced computational costs. Although these methods alleviate the computational overhead of attention, they still rely on manually designed patterns or predefined structures that are insensitive to input-dependent variations, making it difficult to adaptively identify and preserve critical semantic information in complex long-context scenarios (Xiong et al., 2026). In contrast to methods that extend the model's access to the original context, context compression provides an alternative paradigm for long-context modeling by reducing the amount of information that needs to be retained (Chevalier et al., 2023). Inspired by the abstraction and summarization processes in human memory, context compression methods (Xu et al., 2026) address the limited context capacity of LLMs by partitioning long sequences into manageable chunks and transforming them into com-

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.

pact memory slots or latent representations, allowing models to reason over compressed memories instead of repeatedly processing the full original contexts.

Despite their promising potential, most existing context compression methods (Jiang et al., 2024; Yu et al., 2025; Ge et al., 2024b; Li et al., 2026) suffer from the following limitations. ❶ *Semantic fragmentation*. Constrained by limited context windows, current approaches heavily rely on fixed-size sliding windows or predefined segmentation rules to partition long contexts. Such deterministic splitting disregards the intrinsic semantic boundaries of natural language, frequently fragmenting coherent contextual dependencies and causing irreversible information loss during compression. ❷ *Inefficient capacity allocation.* Most methods apply a fixed compression ratio or representation capacity across diverse context segments, despite substantial variations in their information density and semantic importance. This non-adaptive strategy inevitably over-compresses high-value content while allocating excessive capacity to low-information regions, thereby degrading the overall representation efficiency. ❸ *Long-term memory forgetting.* Compressed representations are typically aggregated into a single evolving memory state in an incremental manner. As contexts continue to expand, early but critical evidence can be progressively diluted or overwritten by subsequent inputs, impairing the model's ability to perform reliable long-range retrieval and multi-hop reasoning. Therefore, how to harness standard attention mechanisms and design a flexible and general compression memory mechanism for structured organization and deep understanding long sequences remains challenges.

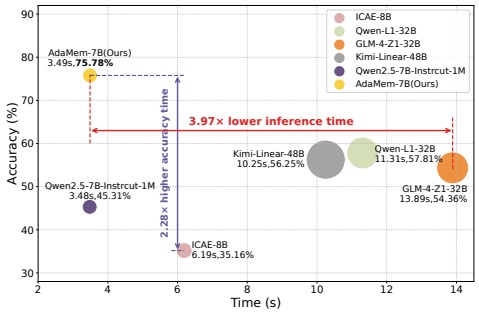


Figure 1: Accuracy and per-query inference time of long-context Compress methods on HotpotQA under a 32K context. Marker sizes indicate model scales.

To address these challenges, we propose **AdaMem**, an adaptive memory organization framework for scalable long-context reasoning. AdaMem dynamically organizes and preserves information from long contexts into compact memory representations without modifying the underlying attention mechanism. Specifically, AdaMem addresses the limitations of existing compression approaches through three cascaded components: (1) a *Boundary-Aware Adaptive Semantic Chunking* module that leverages multi-view transition signals to identify coherent semantic units beyond heuristic fixed-length segmentation, preserving contextual coherence and reducing semantic fragmentation during compression; (2) a *Surprise-Aware Adaptive Memory Budgeting and Compression* module that estimates the information novelty of each chunk and dynamically allocates memory capacity, enabling important contexts to retain richer representations while redundant regions are compressed aggressively; and (3) an *Accumulated Memory Reasoning with Selective Replay* mechanism that preserves accumulated chunk-level memories over long sequences and selectively recovers original contexts when compressed memories are insufficient for reliable reasoning to mitigate historical forgetting. As illustrated in Figure 1, AdaMem achieves a superior trade-off between reasoning accuracy and efficiency on HotpotQA (32K context), outperforming long-context baseline methods while requiring significantly lower inference latency. Our contributions are summarized as follows:

• ***Perspective.*** We establish an adaptive memory perspective for long-context reasoning, showing that scalable understanding can be achieved via selective information organization and preservation rather than extending context length.

• ***Algorithm.*** We propose AdaMem, a cognitively inspired memory organization framework integrating boundary-aware chunking, surprise-aware adaptive budgeting, and selective replay for efficient memory construction under standard attention architectures.

• ***Experiments.*** We conduct extensive experiments on long-context question answering benchmarks up to 1M tokens, demonstrating that AdaMem consistently outperforms existing long-context models in accuracy and scalability.

2 Related Work

Attention Mechanism Optimization. To bypass the quadratic complexity of self-attention (Dao et al., 2022), attention optimization approaches alter computation patterns via architectural designs, kernel approximations, or sparse patterns (Wang et al., 2026). Linear attention methods, such as RetNet (Sun et al., 2023) and Linformer (Wang et al., 2020), achieve linear complexity using kernel functions or low-rank KV projections, albeit at the expense of fine-grained retrieval fidelity (Katharopoulos et al., 2020). Sparse attention variants like Longformer (Beltagy et al., 2020) and Native Sparse Attention (Ding et al., 2025) activate selective token connections, though introducing non-trivial scheduling and implementation overheads. To balance efficiency and expressiveness, recent hybrid architectures like Kimi Linear (Team et al., 2025),GLM-4-Z1 (Glm Team et al., 2024) and DeepSeek-V4 (Xu et al., 2026) combine linear-dense mixing or sparse-compressed attention. Overall, optimized attention paradigms lower long-context inference costs but necessitate trade-offs among training overhead, retrieval fidelity, and deployment compatibility.

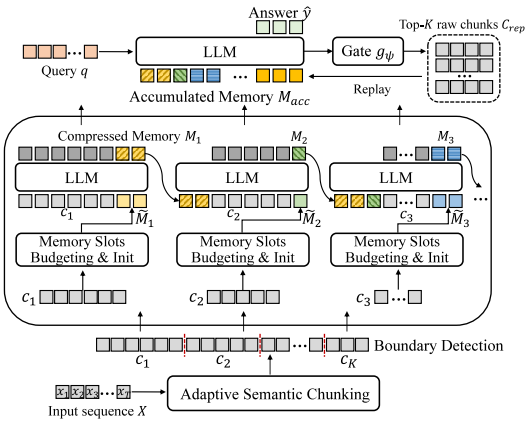


Figure 2: Overview of AdaMem.

Context Representation Compression Strategies. Distinct from attention modifications, context compression reduces computational overhead by compacting input text or KV cache (Wang et al., 2024). Hard compression paradigms, such as LLMLingua (Jiang et al., 2023), LongLLMLingua (Jiang et al., 2024), and MemAgent (Yu et al., 2025), prune redundant tokens or produce compressed text via discrete operations; however, their heuristic segmentation often induces semantic fragmentation and irreversible information loss. Conversely, soft compression methods like AutoCompressor (Chevalier et al., 2023), ICAE (Ge et al., 2024b), and LCLM (Li et al., 2026) map long contexts into continuous latent representations or soft memory slots. Despite preserving global context, they typically rely on uniform compression ratios and recurrent state updates, leading to suboptimal memory allocation and progressive information dilution over extended sequences. Overall, existing strategies lack an adaptive, holistic mechanism to dynamically organize, allocate, and preserve critical information for long-context reasoning.

3 METHOD

3.1 PROBLEM FORMULATION

Let $X = [x_1, x_2, \ldots, x_T]$ denote a long-context input sequence of T tokens, where $x_t \in \mathcal{V}$ represents a token in vocabulary $\mathcal{V}$. In standard long-context reasoning, an LLM processes X alongside a query Q to generate an answer Y. Computing full self-attention over the context introduces quadratic time and space complexity with respect to the sequence length, i.e., $\mathcal{O}(T^2)$. To bypass this bottleneck under attention architectures without modifying model backbones, context compression aims to map X into a compact memory representation $\mathcal{M}$ (e.g., memory slots or compressed memory vectors) that satisfies $|\mathcal{M}| \ll T$. Downstream reasoning is subsequently performed via $\mathrm{P}_{\mathrm{LLM}}(Y \mid Q, \mathcal{M})$, drastically reducing processing costs by enabling the model to retrieve and reason over critical contextual evidence directly from the compressed memory.

3.2 BOUNDARY-AWARE ADAPTIVE SEMANTIC CHUNKING

A key challenge in long-context memory compression is to organize input contexts into meaningful semantic units before memory construction. Existing compression methods (Li et al., 2026; Yu et al., 2025) typically rely on rigid fixed-length windowing to segment long sequences. This ignores the inherent semantic boundaries of natural language and splits closely related information across chunks, leading to fragmented memory representations. To address this issue, we introduce a *boundary-aware adaptive semantic chunking* module that dynamically identifies context transition points and partitions the input context into semantically coherent chunks for memory compression.

Specifically, we formulate semantic chunking as a boundary detection problem to identify transitions between adjacent context regions. Given an input sequence $X = [x_1, \ldots, x_T]$, we estimate the boundary likelihood r_t at each position t by aggregating multiple transition cues:

$$r_t = \sigma(\mathbf{w}^\top \mathbf{e}_t + b), \quad \mathbf{e}_t = [e_t^{\text{tok}}, e_t^{\text{sta}}, e_t^{\text{sem}}]^\top, \tag{1}$$

where $\mathbf{e}_t$ represents the multi-view transition cues extracted at position t, and $\mathbf{w}$ and b are learnable parameters of a lightweight prediction head over frozen backbone representations.

To capture complementary transition patterns across different levels of context understanding, we incorporate three cues into the boundary representation, including token-level predictability shifts, state-level latent dynamics, and semantic-level representation divergence.

Token-Level Predictability Shift. The token-level predictability cue captures local variations in language modeling difficulty across neighboring context regions. Specifically, we evaluate the predictability shift at boundary candidate t by measuring the disparity in mean conditional log-likelihood across adjacent context windows of size w:

$$e_t^{\text{tok}} = \left| \frac{1}{w} \sum_{k \in \{-1,1\}} k \sum_{i=1}^{w} \log p(x_{t+k\cdot i} \mid x_{<t+k\cdot i}) \right|. \tag{2}$$

A larger shift value e_t^{tok} signifies an abrupt change in local context predictability, serving as an effective token-level indicator for semantic boundaries.

State-Level Latent Transition. This cue characterizes underlying transitions in contextual states reflected by the evolution of internal LLM representations. We project the last-layer hidden vector $\mathbf{h}_t$ into a compact latent space via $\mathbf{z}_t = \pi(\mathbf{h}_t)$, where $\pi(\cdot)$ denotes a lightweight learnable projection function. The transition score is defined as $e_t^{\text{sta}} = \|\mathbf{z}_t - \mathbf{z}_{t-1}\|_2 / \sqrt{d_z}$, where d_z is the latent dimension for normalization. A larger e_t^{sta} indicates a sharp latent transition, revealing implicit contextual shifts uncaptured by surface predictability or semantic divergence.

Semantic-Level Representation Divergence. This cue quantifies high-level topic shifts by measuring the representation divergence between adjacent context windows, defined as $e_t^{\text{sem}} = 1 - \cos(\mathbf{g}_t^{\text{left}}, \mathbf{g}_t^{\text{right}})$, where $\mathbf{g}_t^{\text{left}}$ and $\mathbf{g}_t^{\text{right}}$ denote the mean-pooled hidden representations of the left and right context windows (each of size w), respectively.

Given the boundary likelihoods $\{r_t\}_{t=1}^T$, we select candidate boundaries whose likelihoods exceed a threshold δ. To reduce redundant boundaries and avoid over-segmentation, we retain local peaks via non-maximum suppression and enforce a minimum chunk length constraint. The remaining positions define semantic boundaries that partition the input sequence into coherent chunks $\mathcal{C} = \{c_1, c_2, \ldots, c_K\}$ for subsequent adaptive memory compression.

3.3 Surprise-Aware Adaptive Memory Budgeting

Standard long-context compression approaches typically assign a static memory budget across all context regions. However, different chunks may contain substantially different amounts of valuable information. Uniform compression can waste memory capacity on redundant contexts while excessively compressing unexpected but informative content. To address this issue, we propose an information-aware compression strategy that dynamically allocates memory capacity according to the intrinsic novelty of each chunk. The key idea is to estimate the new information a chunk introduces beyond previously accumulated knowledge. Specifically, a chunk well predicted from historical memory is considered redundant and can be compressed aggressively, whereas a chunk that deviates from existing memory requires additional capacity for preservation. For each chunk c_i, we maintain an auxiliary associative memory $\mathcal{M}_{<i}$, initialized as $\mathcal{M}_0 = \mathbf{0}$, which stores key-value correlations from preceding chunks. Given the historical memory, the chunk-level surprise is measured by the discrepancy between the recalled and actual value representations:

$$S_i = \frac{1}{L_i} \sum_{t=1}^{L_i} \log \left(1 + \|\mathbf{k}_{it}\mathcal{M}_{<i} - \mathbf{v}_{it}\|_2^2\right), \tag{3}$$

where L_i denotes the token count of chunk c_i, and $\mathbf{k}_{it} = \mathbf{h}_{it}\mathbf{W}_K$ and $\mathbf{v}_{it} = \mathbf{h}_{it}\mathbf{W}_V$ are the key and value projections of the token representation $\mathbf{h}_{it}$. The term $\mathbf{k}_{it}\mathcal{M}_{<i}$ is the predicted value from

historical associations. Therefore, a larger discrepancy indicates that the current chunk contains information underrepresented in previous memory, resulting in a higher surprise score.

After estimating the chunk c_i's surprise via $\mathcal{M}_{<i}$, we update the auxiliary associative memory to incorporate new information while selectively forgetting outdated associations:

$$\alpha_i = \sigma(\mathbf{W}_f \overline{\mathbf{H}}_i + b_f), \qquad \mathcal{M}_{<i+1} = (1-\alpha_i)\mathcal{M}_{<i} + \alpha_i \frac{\mathbf{K}_i^\top \mathbf{V}_i}{L_i d}, \tag{4}$$

where $\overline{\mathbf{H}}_i$ denotes the mean-pooled hidden representation of chunk c_i, and $\mathbf{W}_f$ and b_f are learnable parameters for computing the forget coefficient α_i. The key and value matrices of chunk c_i are denoted as $\mathbf{K}_i = [\mathbf{k}_{i1}, \ldots, \mathbf{k}_{iL_i}]$ and $\mathbf{V}_i = [\mathbf{v}_{i1}, \ldots, \mathbf{v}_{iL_i}]$, respectively, and d is the hidden dimension. In the memory update equation, $(1-\alpha_i)\mathcal{M}_{<i}$ represents the retained historical memory after forgetting, while $\alpha_i \frac{\mathbf{K}_i^\top \mathbf{V}_i}{L_i d}$ corresponds to the newly written key-value associations from the current chunk. This gated update enables the associative memory to adaptively balance memory retention and new information integration.

Based on the surprise scores S_i, we adaptively allocate memory budgets to different chunks. Since the scale of surprise varies across inputs, we normalize the scores within each sample and map them into the available budget range:

$$B_i = B_{\min} + \Delta B \cdot \sigma\left(\frac{S_i - \mu_S}{\sigma_S + \epsilon}\right), \tag{5}$$

where μ_S and σ_S denote the mean and standard deviation of chunk-level surprise scores per sample, ϵ is a small constant for numerical stability, and $\Delta B = B_{\max} - B_{\min}$ is the memory budget range. $B_{\min}$ and $B_{\max}$ define the lower and upper bounds of memory tokens allocated per chunk, where $B_{\max} \ll L_i$. A larger surprise score yields a larger memory budget, allowing informative chunks to preserve richer representations while aggressively compressing predictable content.

3.4 Information-Aware Memory Compression

After determining the dynamic memory budget B_i for each chunk c_i, we compress the original context into a compact set of latent memory tokens. Existing latent context compression methods initialize memory slots with random embeddings and optimize them to extract information through attention interactions Chevalier et al. (2023). However, such input-independent initialization ignores the intrinsic token distribution of the original context, forcing memory slots to discover meaningful representations from scratch, which increases optimization difficulty and risks redundant capacity.

To address this limitation, we propose a budget-aware memory initialization strategy that derives memory slots from the input embeddings of the original chunk. Given a chunk $c_i = \{x_{i1}, \ldots, x_{iL_i}\}$, we obtain its input embedding matrix from the LLM embedding table $\mathbf{U}_i = [\mathbf{u}_{i1}, \ldots, \mathbf{u}_{iL_i}]^\top \in \mathbb{R}^{L_i \times d}$. According to the allocated budget B_i, we evenly partition $\mathbf{U}_i$ into B_i consecutive embedding segments $\{\mathbf{U}_{ij}\}_{j=1}^{B_i}$ following the original token order. Since neighboring tokens in natural language are typically semantically coherent, each segment provides an informative local prior for its corresponding memory slot. The initialized memory slots are computed as:

$$\tilde{\mathbf{m}}_{ij} = \mathrm{Pool}(\mathbf{U}_{ij}), \quad j = 1, \ldots, B_i, \tag{6}$$

where $\mathbf{U}_{ij}$ denotes the j-th consecutive segment of $\mathbf{U}_i$, and $\tilde{\mathbf{m}}_{ij}$ is the initialized representation of the corresponding memory slot. We denote the resulting memory slots as $\tilde{\mathbf{M}}_i = [\tilde{\mathbf{m}}_{i1}, \ldots, \tilde{\mathbf{m}}_{iB_i}]^\top \in \mathbb{R}^{B_i \times d}$. Compared with random initialization, this strategy preserves the local semantic structure of the original context and provides each memory slot with a context-aware semantic anchor before compression.

Memory slots $\tilde{\mathbf{M}}_i$ are appended to input embeddings $\mathbf{U}_i$ and jointly processed by the backbone:

$$\mathbf{H}_i = \mathrm{Backbone}([\mathbf{U}_i; \tilde{\mathbf{M}}_i]), \tag{7}$$

where $\mathbf{H}_i$ denotes the hidden states of the augmented sequence. Through self-attention interactions, the appended memory slots attend to the entire chunk and progressively absorb informative context. We retain the hidden states corresponding to the appended slots as the final compressed memory representation: $\mathbf{M}_i = \mathbf{H}_i[L_i+1 : L_i+B_i]$, where $\mathbf{M}_i \in \mathbb{R}^{B_i \times d}$ contains the output representations of the B_i memory slots. Therefore, a chunk containing L_i original tokens is compressed into a latent memory representation with only B_i memory tokens.

3.5 ACCUMULATED MEMORY REASONING WITH REPLAY

After compressing each semantic chunk into an adaptive number of latent memory tokens, the remaining challenge is to perform reliable reasoning over the complete long context. Recurrently updating a single memory state, as commonly adopted in RNN-style compression methods (Bulatov et al., 2022; Kim et al., 2024), may progressively overwrite or dilute information preserved from earlier chunks. To avoid this issue, we preserve each chunk-level memory independently and accumulate them into a unified memory sequence for downstream reasoning.

Specifically, given the compressed memories $\{\mathbf{M}_i\}_{i=1}^{K}$, where $\mathbf{M}_i \in \mathbb{R}^{B_i \times d}$ denotes the memory representation of chunk c_i, we construct the accumulated memory as $\mathbf{M}_{\text{acc}} = [\mathbf{M}_1; \mathbf{M}_2; \cdots; \mathbf{M}_K]$, where $[\cdot;\cdot]$ denotes concatenation along the sequence dimension. Unlike recurrent memory updates that repeatedly rewrite a shared state, this append-only accumulation preserves a direct information pathway from every semantic chunk to downstream reasoning.

Given a query q, its input embeddings $\mathbf{U}_q$ are appended to the accumulated memory, and the backbone model performs coarse reasoning based solely on the compressed memory:

$$\hat{y}^{\,\text{mem}} = f_\theta([\mathbf{M}_{\text{acc}}; \mathbf{U}_q]), \quad p_{\text{rep}} = g_\psi(\mathcal{H}(\hat{y}^{\,\text{mem}})), \tag{8}$$

where f_θ is the LLM backbone, and $\mathcal{H}(\hat{y}^{\,\text{mem}})$ evaluate the predictive entropy reflecting memory uncertainty. The lightweight gate $g_\psi(\cdot)$ estimates the replay probability p_{rep} based on this uncertainty signal. If p_{rep} falls below a predefined threshold τ, the coarse prediction $\hat{y}^{\,\text{mem}}$ is directly adopted as the final output. Otherwise, selective replay is activated to retrieve relevant raw context chunks. Specifically, we mean-pool each chunk memory and select the top-R most similar chunks to the query. The final prediction is then refined as:

$$\hat{y}^{\,\text{final}} = f_\theta([\mathbf{M}_{\text{acc}}; \mathbf{C}_{\text{rep}}; \mathbf{U}_q]), \quad \mathbf{C}_{\text{rep}} = [c_i]_{i \in \mathcal{I}_q}, \tag{9}$$

where $\mathcal{I}_q$ indexes the top-R chunks ranked by similarity, with $|\mathcal{I}_q| = R$. By executing raw context replay only when uncertainty is high, this coarse-to-fine mechanism achieves an optimal balance between reasoning accuracy and inference efficiency.

3.6 TRAINING OBJECTIVE

We optimize AdaMem via a two-stage training scheme: (1) *compression-aware pre-training* to learn semantic-aware chunking and predictive memory encoding, and (2) *task-oriented fine-tuning* to adapt compressed memories for downstream reasoning while jointly optimizing selective replay.

Compression-aware Pre-training. The pre-training phase consists of two sequential steps: First, we warm up the semantic chunking adapter to identify reliable transitions between adjacent semantic units. Given binary boundary labels $\{y_t\}_{t=1}^{T}$ synthesized by a teacher LLM, the boundary predictor is trained via a weighted binary cross-entropy loss:

$$\mathcal{L}_{\text{bd}} = -\frac{1}{T}\sum_{t=1}^{T}\left[\alpha y_t \log p_t + (1-y_t)\log(1-p_t)\right], \tag{10}$$

where p_t denotes the predicted boundary probability and α balances class distribution. Detailed teacher annotation protocols are deferred to the Appendix B.

Second, after boundary warm-up, we optimize memory compression quality through a memory-conditioned next-token prediction objective. Specifically, the model learns to predict subsequent tokens based on the accumulated compressed memory:

$$\mathcal{L}_{\text{mem}} = -\frac{1}{N}\sum_{i=1}^{K}\sum_{t=1}^{|c_i|}\log p_\theta(x_{i,t} \mid x_{i,<t}, \mathbf{M}_{<i}), \tag{11}$$

where $\mathbf{M}_{<i}$ denotes the accumulated memory before processing chunk c_i, and N is the total number of tokens. By predicting future tokens from compressed memories, this objective encourages the compressor to retain information that is most valuable for representing long-range context.

Task-oriented Fine-tuning. After pre-training, we adapt the model to downstream reasoning tasks. Given a query q, compressed memory $\mathbf{M}_{\text{acc}}$, and target response y, we optimize the standard causal

language modeling objective:

$$\mathcal{L}_{\text{sft}} = -\frac{1}{|y|}\sum_{t=1}^{|y|}\log p_\theta(y_t \mid y_{<t}, q, \mathbf{M}_{\text{acc}}). \tag{12}$$

Concurrently, we train the lightweight gate $g_\psi(\cdot)$ to calibrate the need for selective replay. For each training sample, an oracle replay signal $a^* \in \{0,1\}$ is constructed by evaluating the task performance disparity before and after raw context replay:

$$a^* = \mathbb{I}\left(\mathcal{R}_{\text{rep}} > \mathcal{R}_{\text{mem}}\right), \quad \mathcal{L}_{\text{rep}} = \text{BCE}(p_{\text{rep}}, a^*), \tag{13}$$

where $\mathcal{R}_{\text{mem}}, \mathcal{R}_{\text{rep}} \in \{0,1\}$ denote binary performance rewards obtained using compressed memory alone and with replayed context, respectively. This loss encourages context replay only to fix incorrect predictions and depresses its likelihood when memory is accurate, enabling selective replay that balances performance and efficiency.

Finally, the overall objective jointly optimizes task fine-tuning and replay calibration.

4 Experiments

4.1 Experiment Settings

Datasets. For long-context question answering, we use RULER (Hsieh et al., 2024) to construct datasets with varying context lengths from HotpotQA (Yang et al., 2018), 2WikiMQA (Ho et al., 2020), and TriviaQA (Joshi et al., 2017). AdaMem is fine-tuned only on HotpotQA and evaluated zero-shot on 2WikiMQA and TriviaQA. The Appendix B provides the data-cleaning rules, sample-synthesis pipeline, and training hyperparameters.

Baselines. We compare AdaMem with representative models, including the hard-token compression method LLMLingua-2 (Pan et al., 2024), soft-compression methods ICAE (Ge et al., 2024b) and LCLM (Li et al., 2026), and attention-optimized long-context models Qwen2.5-7B-Instruct-1M (Yang et al., 2025), QwenLong-L1-32B (Wan et al., 2025), GLM-4-Z1 (Glm Team et al., 2024), and Kimi-Linear-48B (Team et al., 2025).

4.2 Main Results

We evaluate AdaMem on three datasets and compare it with the benchmark models over a shared context-length range of 8K to 1M tokens. Table 1 presents the accuracy results.

Table 1 shows differences among the three types of long-context modeling approaches. At 1M tokens, hard-compression approaches score below 24% on all three datasets. Soft-compression methods remain competitive at shorter lengths but decline at extreme context lengths; the strongest baseline in this group achieves 30.47%, 25.00%, and 38.28% on the three datasets, respectively. Native long-context and attention-optimized models reach 40.63%, 38.28%, and 47.66% at 1M tokens, respectively. AdaMem obtains 69.53%, 63.28%, and 66.41% at the same context length.

Across the evaluated context lengths, AdaMem obtains the best results on all three datasets, with average accuracies of 75.00%, 69.04%, and 75.68% on HotpotQA, 2WikiMultihopQA, and TriviaQA, respectively. At 1M tokens, it scores 69.53%, 63.28%, and 66.41% on the three benchmarks, respectively. Additional case studies of adaptive chunking and budget allocation are provided in Appendix A, where AdaMem is compared with fixed strategies.

4.3 Memory Budget Analysis for Ground Chunks.

Overall, Figure 3 shows that gold chunks receive substantially more memory than non-gold chunks: their mean budgets are 26 and 10 summary vectors, respectively, corresponding to a 2.6× difference. This separation indicates that AdaMem assigns additional capacity to chunks containing query-supporting evidence, allowing relevant entities and relations to retain richer representations during compression. The distributions nevertheless overlap, suggesting that the allocation is graded rather than determined by a hard gold/non-gold rule. This behavior follows the surprise-aware mapping

Table 1: Performance comparison between AdaMem and representative context-compression and long-context modeling baselines across three datasets and varying context lengths. Scores are reported as EM accuracy. Bold numbers denote the best result per column, and underlined values denote the second-best result. AdaMem is fine-tuned only on HotpotQA.

Models	*Dataset: HotpotQA*								
	8K	**16K**	**32K**	**64K**	**128K**	**256K**	**512K**	**1M**	**Avg**
ICAE-8B	50.78	42.19	35.16	25.78	20.31	12.50	–	–	31.12
LCLM-5B	78.13	75.00	70.31	65.63	60.94	55.47	48.44	30.47	60.55
LLMLingua-2-7B	33.59	25.78	24.22	27.34	22.66	27.34	20.31	5.47	23.34
GLM-4-Z1-32B	60.16	51.56	48.44	46.88	42.19	39.84	35.16	31.25	44.44
Kimi-Linear-48B	56.25	54.69	55.47	51.56	51.56	60.16	46.09	40.63	52.05
QwenLong-L1-32B	64.06	62.50	57.81	45.31	18.75	14.84	10.94	5.47	34.96
Qwen2.5-7B-Instruct-1M	55.47	50.00	45.31	41.41	34.38	30.47	8.59	0.00	33.20
AdaMem-7B (Ours)	**78.91**	**80.47**	**75.78**	**76.56**	**73.44**	**73.44**	**71.88**	**69.53**	**75.00**
Models	*Dataset: 2WikiMQA*								
	8K	**16K**	**32K**	**64K**	**128K**	**256K**	**512K**	**1M**	**Avg**
ICAE-8B	33.59	31.25	32.81	20.31	21.09	20.31	–	–	26.56
LCLM-5B	67.19	60.16	57.81	51.56	42.97	39.06	30.47	25.00	46.78
LLMLingua-2-7B	30.47	29.69	26.56	23.44	25.78	22.66	16.41	7.81	22.85
GLM-4-Z1-32B	54.69	42.97	40.63	35.16	31.25	34.62	29.69	27.34	37.04
Kimi-Linear-48B	56.25	57.03	51.56	54.69	50.78	47.66	45.31	38.28	50.20
QwenLong-L1-32B	61.72	60.16	53.13	31.25	26.56	20.31	11.72	7.81	34.08
Qwen2.5-7B-Instruct-1M	48.44	50.78	47.66	45.31	33.59	30.47	14.06	3.13	34.18
AdaMem-7B (Ours)	**76.56**	**73.44**	**70.31**	**68.75**	**65.63**	**67.97**	**66.41**	**63.28**	**69.04**
Models	*Dataset: TriviaQA*								
	8K	**16K**	**32K**	**64K**	**128K**	**256K**	**512K**	**1M**	**Avg**
ICAE-8B	57.03	54.69	51.56	58.59	50.00	46.09	–	–	52.99
LCLM-5B	75.00	71.88	71.09	64.84	58.59	50.00	47.66	38.28	59.67
LLMLingua-2-7B	44.53	47.66	41.41	46.09	40.63	31.25	31.25	23.44	38.28
GLM-4-Z1-32B	75.00	70.31	71.09	64.84	60.16	58.59	50.78	45.31	62.01
Kimi-Linear-48B	76.56	74.22	67.97	67.19	66.41	68.75	60.16	47.66	66.12
QwenLong-L1-32B	65.63	62.50	60.94	55.47	61.72	45.31	28.91	20.31	50.10
Qwen2.5-7B-Instruct-1M	60.16	59.38	57.81	57.03	54.69	40.63	18.75	15.63	45.51
AdaMem-7B (Ours)	**84.38**	**80.47**	**75.00**	**73.44**	**74.22**	**74.22**	**73.44**	**66.41**	**75.20**

in Eq. (5), where a higher chunk surprise score S_i produces a larger budget B_i after within-sample normalization. Consequently, informative non-gold chunks can still receive additional capacity, while predictable gold chunks may be compressed more aggressively when their content is already represented by accumulated memory.

4.4 Ablation Study

Table 2 presents ablation results for AdaMem across three long-document question-answering benchmarks and three context lengths. We evaluate three variants, each of which removes or replaces one core component: fixed semantic chunking (AdaMem (w/o SC)), fixed memory allocation (AdaMem (w/o AB)), and selective replay removal (AdaMem (w/o SR)).

Semantic Chunking. Removing semantic-aware chunking and replacing it with rigid fixed-size segmentation consistently degrades overall performance across all benchmarks. For example, AdaMem (w/o SC) drops from 75.78% to 67.97% on HotpotQA at 32K and from 71.88% to 67.19% at 512K. This indicates that semantic chunking better preserves discourse coherence, enabling more effective memory construction for long-range reasoning.

Adaptive Budgeting. Replacing adaptive budgeting with a fixed memory allocation strategy leads to substantial performance degradation, especially under longer contexts. Compared with the full model, AdaMem (w/o AB) decreases by 15.62 and 13.29 percentage points on HotpotQA at 32K and 512K, respectively. These results demonstrate that uniform allocation fails to capture the uneven distribution of informative content, while surprise-aware budgeting effectively prioritizes reasoning-critical information.

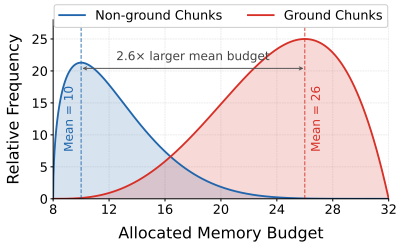


Figure 3: Comparison of budget allocation between Non-Ground chunks and Ground chunks.

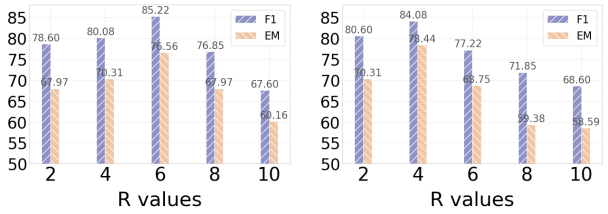


Figure 4: Hyperparameter sensitivity of AdaMem under 64K contexts. Evaluated on HotpotQA and TriviaQA across different retrieval top-R settings.

Table 2: Ablation results on HotpotQA, 2WikiMQA, and TriviaQA. We compare AdaMem with variants that remove semantic chunking (w/o SC), adaptive budgeting (w/o AB), or selective replay (w/o SR).

Models	HotpotQA			2WikiMQA			TriviaQA		
	32K	128K	512K	32K	128K	512K	32K	128K	512K
AdaMem (Ours)	**75.78**	**73.44**	**71.88**	**70.31**	**65.63**	**66.41**	**75.00**	**74.22**	**73.44**
AdaMem(w/o SC)	67.97	65.63	67.19	62.50	59.38	60.16	64.84	61.72	62.50
AdaMem(w/o AB)	60.16	64.84	58.59	57.81	57.03	53.91	63.28	60.16	58.59
AdaMem(w/o SR)	66.12	60.94	54.69	53.13	50.00	51.56	60.16	62.01	57.81

Selective Replay. Removing selective replay causes substantial degradation, especially at longer context lengths. At 512K, performance drops by 17.19, 14.85, and 15.63 percentage points on HotpotQA, 2WikiMQA, and TriviaQA, respectively. At 32K, the corresponding drops are 9.66, 17.18, and 14.84 points, while at 128K they are 12.50, 15.63, and 12.21 points. Across all nine dataset-length comparisons, the average degradation is 14.41 percentage points. These results show that replay helps recover fine-grained evidence after memory compression, with particularly strong benefits in longer-context settings.

4.5 Parameter Study

We analyze AdaMem's sensitivity to the retrieval top-R used in selective replay. As shown in Figure 4, moderate R values yield the highest observed performance among the settings evaluated. On HotpotQA, performance peaks at $R = 6$, reaching 85.22% F1 and 76.56% EM, and declines to 67.60% and 60.16% at $R = 10$. On TriviaQA, performance peaks at $R = 4$, reaching 84.08% F1 and 78.44% EM, before declining at larger retrieval sizes. This behavior is consistent with dataset-specific differences in evidence distribution across both evaluation metrics. A small R may omit critical evidence, whereas a large R may introduce redundant or irrelevant context. These results illustrate the role of replay scope in long-context reasoning and its trade-off between evidence coverage and contextual noise.

5 Conclusion

We introduced AdaMem, a framework for scalable long-context reasoning. Unlike the evaluated compression approaches that rely on fixed-size chunking and uniform memory budgets, AdaMem organizes and preserves long-context information through boundary-aware semantic chunking, surprise-aware memory allocation, and accumulated memory reasoning with selective replay. This design is intended to reduce semantic fragmentation, prioritize information-rich chunks, and mitigate historical forgetting without modifying standard attention mechanisms. Experiments on three long-context question-answering benchmarks, with context lengths from 8K to 1M tokens, show that AdaMem outperforms the evaluated baselines in reasoning accuracy. The framework combines adaptive information preservation with compression for reasoning over long contexts.

AI USE STATEMENT

We used generative AI tools for LaTeX formatting and language editing. We have not used generative AI tools to generate experimental results, fabricate data, or alter reported measurements. We reviewed all AI-assisted work and take responsibility for the final content, including any text or artifacts produced with such assistance.

REPRODUCIBILITY STATEMENT

The paper describes the model components, training objectives, datasets, baselines, and key hyperparameters. The appendix provides the data construction procedure and implementation details needed to reproduce the experiments; the figures and bibliography are included with the submission.

REFERENCES

Ali Behrouz, Zeman Li, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. Memory caching: Rnns with growing memory. *arXiv preprint arXiv:2602.24281*, 2026.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory transformer. *Advances in Neural Information Processing Systems*, 35:11079–11091, 2022.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3829–3846, 2023.

Tri Dao, Daniel Y Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *NeurIPS*, 2022.

Yuxuan Ding, Xin Lv, Wenxuan Zhou, Zhe Zhang, Zhijie Lin, and Qingfu Liu. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *arXiv preprint arXiv:2502.11089*, 2025.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. In *Proceedings of the Twelfth International Conference on Learning Representations*, Vienna, Austria, 2024a.

Tao Ge, Yifan Zhang, Xiaozhi Wang, and Furu Wei. In-context autoencoder for context compression in large language models. In *ICLR*, 2024b.

Glm Team, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, et al. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*, 2024.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6621, 2020.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

Huiqiang Jiang, Qianqian Zhang, Ziyang Chen, and Yutong Li. Llmlingua: Compressing long prompts for faster inference. *arXiv preprint arXiv:2310.05736*, 2023.

Huiqiang Jiang, Qianqian Zhang, and Ziyang Chen. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. In *ICLR*, 2024.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1604–1614, 2017.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp. *arXiv preprint arXiv:2212.14024*, 2022.

Jang-Hyun Kim, Junyoung Yeom, Sangdoo Yun, and Hyun Oh Song. Compressed context memory for online language model interaction. In *International Conference on Learning Representations*, volume 2024, pp. 35316–35334, 2024.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Ming Li, Shaoshen Chen, and Haitao Zheng. Latent context language models: Learning compact latent representations for unlimited long context. *arXiv preprint arXiv:2601.04782*, 2026.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, et al. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 963–981, 2024.

Yutian Sun, Li Dong, Shaohan Huang, Shuming Ma, Furu Wei, and Houfeng Wang. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Computing Surveys*, 55(6):1–28, 2022.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, et al. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Zihan Wang, Yang Liu, and Jinze Bai. A survey of context compression methods for large language models. *arXiv preprint arXiv:2411.03148*, 2024.

Zihan Wang, Yang Liu, and Jinze Bai. Efficient attention mechanisms for large language models: A comprehensive survey. *arXiv preprint arXiv:2507.19595*, 2026.

Siheng Xiong, Joe Zou, Faramarz Fekri, and Yae Jee Cho. Long-context modeling with dynamic hierarchical sparse attention for memory-constrained llm inference. In *Forty-third International Conference on Machine Learning*, 2026.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

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*, 2025.

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, 2018.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, et al. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent. *arXiv preprint arXiv:2507.02259*, 2025.

Runsong Zhao, Shilei Liu, Jiwei Tang, Langming Liu, Haibin Chen, Weidong Zhang, Yujin Yuan, Tong Xiao, Jingbo Zhu, Wenbo Su, and Bo Zheng. Comet: Collaborative memory transformer for efficient long context modeling. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 2026.

A Case Study

We present a case study to illustrate how AdaMem combines semantic chunking and dynamic budget allocation for long-context reasoning. As shown in Figure 5, the example concerns the Athens War Museum and asks which museum its first director previously held a position at. AdaMem identifies semantic transitions and partitions the context into units describing the Athens War Museum, the director's biography, and unrelated background content. This boundary-aware segmentation is intended to reduce fragmentation across arbitrary windows and provide semantic units for compression. AdaMem then allocates memory capacity according to each chunk's information density. The information-rich chunks C_{i-1} and C_i receive budgets of $b_{i-1} = 28$ and $b_i = 15$, respectively, while the redundant chunk C_{i+1} is compressed into $b_{i+1} = 4$ summary vectors. The baseline uses fixed-window segmentation and assigns $b = 10$ to each chunk, regardless of its semantic content. In this example, relevant information may therefore receive less representational capacity, while irrelevant fragments use memory. The comparison illustrates the complementary roles of the two components: semantic chunking preserves contextual coherence, while adaptive budgeting assigns more capacity to informative content and less to redundant context.

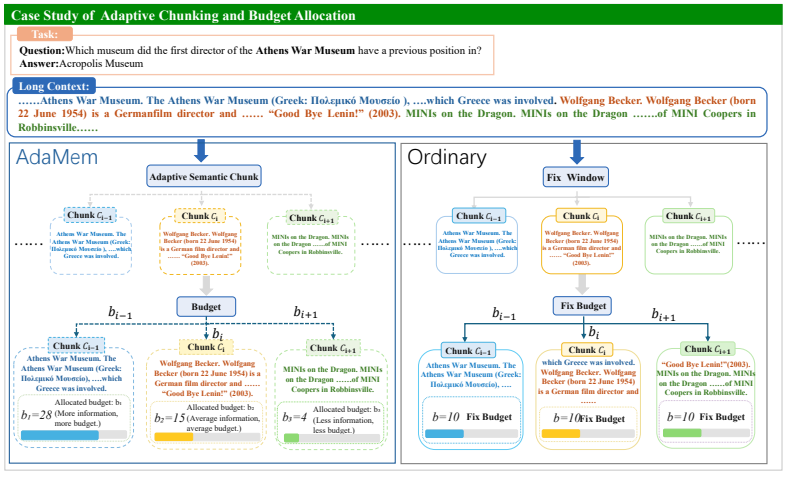


Figure 5: Effectiveness of adaptive semantic chunking and memory budgeting in AdaMem. AdaMem preserves coherent semantic units and assigns larger budgets to information-rich chunks while aggressively compressing redundant content, outperforming fixed-window segmentation with uniform allocation.

B Dataset Construction Details

RULER-Based Long-Context Construction. We construct long-context question-answering instances from HotpotQA (Yang et al., 2018), 2WikiMQA (Ho et al., 2020), and TriviaQA (Joshi et al., 2017). HotpotQA and 2WikiMQA require multi-hop reasoning across supporting documents, whereas TriviaQA evaluates evidence retrieval from diverse textual sources. Following the RULER (Hsieh et al., 2024) protocol, we retain reference documents containing question-relevant evidence and place them among distractor documents sampled from the corresponding source dataset. The query specifies the retrieval task, and the reference documents provide the evidence needed to derive the answer. We randomly arrange supporting and distractor documents to avoid fixed evidence positions. By varying the number of distractor documents while keeping the question, answer, and supporting evidence fixed, we construct contexts ranging from 8K to 1M tokens. This RULER-style Needle-in-a-Haystack setting evaluates evidence localization and multi-document reasoning across context lengths.

Training and Evaluation Set Construction. For training, we process an initial 30,000 samples from the HotpotQA training set and use Qwen2.5-7B-Instruct to identify questions that can be answered without contextual evidence. We remove samples for which the model answers correctly in 100% of cases, as these may rely on factual knowledge acquired during pre-training rather than context-dependent reasoning. This procedure filters approximately 40% of the original samples. We then retain the top 10,000 valid samples and apply the RULER-based construction procedure to generate the training set. For evaluation, we use the same synthesis and cleaning strategy to construct 128 test samples from each of the HotpotQA validation set, 2WikiMQA, and TriviaQA. Within each dataset, the same questions and supporting evidence are retained across context-length settings; only the number of distractor documents varies. This design controls for question and evidence content when evaluating different context lengths.

Warm-up Dataset Construction. We construct the semantic chunking warm-up set from 10,000 RULER-style HotpotQA training instances with 16K-token contexts. Each context consists of question-relevant passages interleaved with distractor passages. We remove explicit passage markers and treat transitions between adjacent passages as candidate boundaries. For each candidate, DeepSeek-V4-Pro (Xu et al., 2026) examines neighboring text spans and labels whether the transition reflects a substantial change in topic, entity, event, or discourse structure. Annotation uses a temperature of 0.0. Accepted boundaries receive positive labels, and the remaining token positions serve as negative examples.

We align character-level annotations with the Qwen2.5 tokenizer using offset mappings to produce token-level boundary labels $\{y_t\}_{t=1}^{T}$. We exclude the final document endpoint because it does not represent an internal semantic transition. We randomly divide the resulting data into 9,000 training instances and 1,000 validation instances. To extract boundary features, we process each context in overlapping 2,048-token windows with a stride of 1,024 tokens and retain only each window's central region to reduce boundary artifacts. The frozen backbone extracts token-level predictability, latent-state transition, and semantic-divergence signals for adapter warm-up. Because semantic boundaries are sparser than non-boundary positions, we use class-weighted supervision when optimizing $\mathcal{L}_{\mathrm{bd}}$.

C Implementation Detail

We implement AdaMem with Qwen2.5-7B-Instruct and optimize it using AdamW. We first warm up the semantic boundary predictor for 50 epochs with a learning rate of 1×10^{-3}, then conduct end-to-end training for 100 epochs with a learning rate of 5×10^{-6} and a batch size of 1. We apply LoRA to the query, key, value, and output projections, with rank 16, scaling factor 32, and dropout 0.05. The adaptive memory budget ranges from 8 to 32 summary vectors per chunk. During selective replay, we retrieve the top 3 most relevant chunks and use a gate threshold of 0.5. We train models in bfloat16 with gradient checkpointing and SDPA attention. Experiments use 8 NVIDIA GeForce RTX 5090 GPUs.