OSET: ONLINE EVIDENCE TREES UNDER SUBLINEAR RETAINED-STATE BUDGETS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models can process increasingly long inputs, yet retaining every past token, passage, or index entry causes memory to grow linearly with the input stream. Many compression and retrieval approaches reduce only the context presented to the reader or the key–value cache resident on the GPU, while preserving a linear external history. We study online long-context reasoning under a closed sublinear retained-state constraint. After reading n input tokens, the memory is bounded by $B(n) = B_0 + \lceil cn^\alpha \rceil$, where $0 < \alpha < 1$, and discarded source content cannot be recovered from an uncounted archive. We introduce OSET, an online evidence tree that incrementally incorporates incoming text and selectively compacts and prunes its contents to satisfy this state envelope. OSET separates short semantic access cues from source-bearing evidence, allowing a frozen language-model reader to recover verifiable information for questions that are unknown during memory construction. We evaluate OSET with Llama-3.1-8B-Instruct on HotpotQA, MuSiQue, 2WikiMultiHopQA, RULER from 8K to 128K tokens, and LongBench v2. Across the three multi-hop QA datasets, OSET obtains a macro F1 of 45.0, compared with 46.0 for full-context inference. On LongBench v2, it outperforms full-context inference, while retaining textual state equal to 10.97% of the source length. Across the complete RULER evaluation, OSET reduces the average reader context by 91.3% and logged peak GPU memory by 73.6%. On the evidence-oriented RULER task families, it achieves an average of 98.9 on needle-in-a-haystack retrieval and 97.1 on variable tracking, although tasks requiring corpus-wide aggregation remain challenging. These results show that online sublinear retention provides a distinct low-state operating regime rather than a lossless replacement for full context.

1 INTRODUCTION

Large language models (LLMs) are increasingly expected to reason over inputs far longer than a conventional prompt, including document collections, interaction histories, code repositories, and continuously arriving text streams. Simply extending the nominal context window does not fully solve this problem. Standard self-attention incurs quadratic prefill cost with sequence length, while the KV cache grows linearly with the number of retained tokens. Moreover, access to a long context does not guarantee effective use of it: relevant evidence may be underutilized when buried deep in the prompt, and controlled evaluations such as RULER show that effective context length can be substantially shorter than the advertised window. These limitations motivate long-context systems that avoid retaining and repeatedly exposing the complete history to the reader.

Existing approaches reduce this cost from several directions. KV-cache compression and sparse attention reduce the token-level state accessed by the model, while retrieval and prompt compression reduce the context eventually presented to the reader. Explicit memory systems, instead, summarize, organize, or selectively retain information outside the reader. However, reducing active GPU state or reader context does not necessarily bound the *complete persistent state*: an append-only cache, vector store, or hierarchical memory can still grow linearly with the amount of observed input. At the other extreme, a fixed-capacity memory satisfies a strict bound but must increasingly overwrite earlier information as the stream grows.

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.

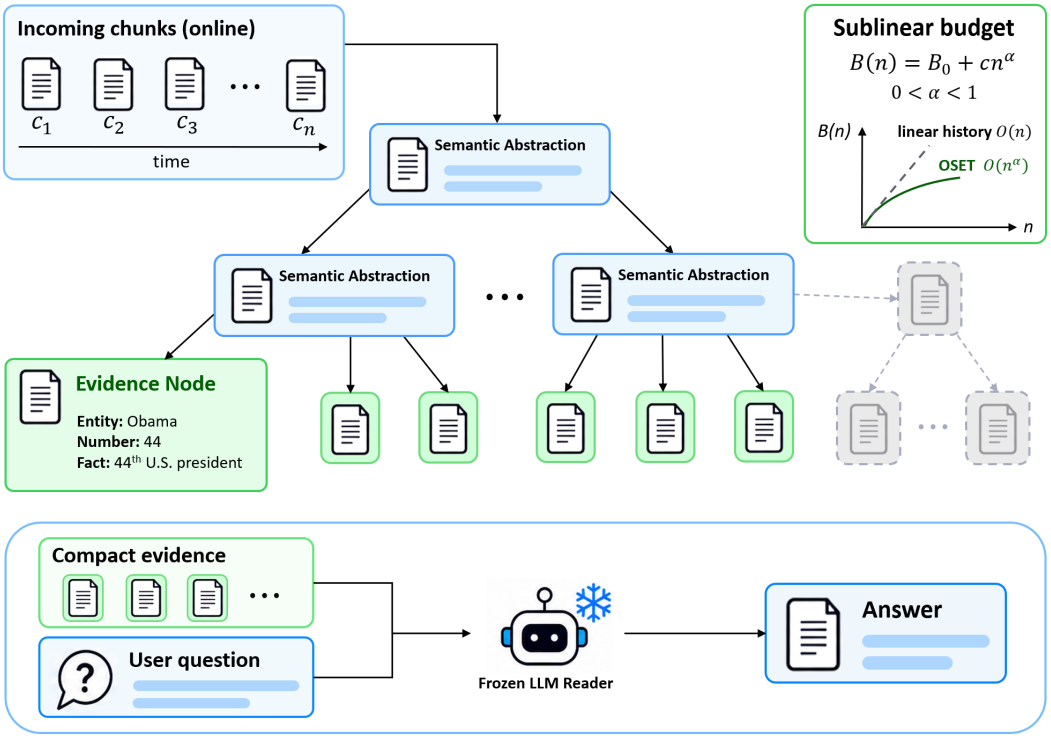

Figure 1: OSET for online long-context reasoning under a sublinear retained-state budget.

We therefore study an intermediate regime: *online long-context reasoning under a closed sublinear retained-state budget*. After processing n input tokens, the complete retained textual state is constrained by

$$B(n) = B_0 + \lceil cn^\alpha \rceil, \qquad 0 < \alpha < 1,$$

and discarded source content cannot be recovered from an uncounted external archive. This formulation separates two resources that are often conflated: the *retained-state budget*, which limits how much persistent information the system may keep, and the *reader-context budget*, which limits how much evidence is finally presented to the language model. The setting is particularly challenging online, because future questions are unknown when memory is constructed: aggressive compression may remove exact evidence needed later, whereas preserving all source content violates the sublinear state constraint.

To address this problem, we introduce **OSET**, an **O**nline **S**ublinear **E**vidence **T**ree. As illustrated in Fig. 1, OSET incrementally converts incoming text into evidence-preserving memory nodes that separate compact semantic access cues from literal source-grounded evidence. These nodes are organized into an online hierarchy and maintained under the sublinear state envelope through evidence-aware compaction and pruning. At query time, OSET retrieves a small set of complementary evidence using lexical, exact, semantic, and structural signals and renders it into a compact context for a frozen LLM reader. The reader itself is neither modified nor fine-tuned.

Experiments with Llama-3.1-8B-Instruct on multi-hop QA, RULER, and LongBench v2 show that OSET preserves strong performance on evidence-oriented long-context tasks while substantially reducing persistent state and reader context. Its main limitation appears on tasks requiring aggregation over a large fraction of the original stream, highlighting the trade-off imposed by sublinear retention rather than treating the method as a lossless replacement for unrestricted full-context inference.

Our contributions are fourfold:

- We formulate online long-context reasoning under a *closed sublinear retained-state constraint*, requiring the complete retained state to grow as $O(n^\alpha)$, $0 < \alpha < 1$, while accounting for both persistent memory and reader-context cost.

- We introduce OSET, an online evidence tree that combines semantic abstractions with source-grounded evidence, enabling efficient access to long histories without relying on summaries alone to preserve factual information for future queries.
- We develop progressive evidence-aware compaction and pruning to maintain OSET under the sublinear state budget. This maintains the sublinear retained-state constraint while preferentially preserving evidence needed for sparse retrieval and multi-hop reasoning as the input grows.
- We evaluate long-context memory at both intermediate and end-task levels by tracing whether supporting evidence is retained, retrieved, and used by the reader, thereby separating memory and retrieval failures from downstream reasoning errors.

2 Related Work

2.1 Efficient Long-Context Inference

A large body of work improves long-context inference by reducing the token-level state or computation required by Transformer attention. KV-cache eviction and compression methods retain only selected historical states, using attention importance, observation-window selection, or non-uniform cache allocation across layers and heads; representative examples include H_2O (Zhang et al., 2023), StreamingLLM (Xiao et al., 2024b), SnapKV (Jiang et al., 2024), and PyramidKV (Cai et al., 2024). Other work reduces attention computation through sparse or approximate access, such as SparQ (Ribar et al., 2024) and Quest (Tang et al., 2024). A further line retrieves or transfers historical KV states on demand, including InfLLM (Xiao et al., 2024a), ShadowKV (Sun et al., 2024), and InfiniGen (Lee et al., 2024).

These approaches substantially reduce active attention cost, GPU-resident KV state, or memory bandwidth, but generally optimize how a historical state is *accessed* rather than how much persistent information is retained after processing the stream. A method may therefore sparsely access or offload a history whose total size still grows linearly with sequence length. OSET studies a different constraint: the complete persistent memory itself must remain within a sublinear state envelope.

2.2 Hierarchical and Structured Memory

Hierarchical memory provides a natural mechanism for organizing long inputs at multiple levels of abstraction. RAPTOR recursively clusters and summarizes text for multi-level retrieval (Sarthi et al., 2024), while MemWalker organizes long documents into a tree that an LLM can navigate (Chen et al., 2023). GraphRAG instead introduces relational structure over entities and communities (Edge et al., 2024). More recent systems, including MemTree (Rezazadeh et al., 2025), H-MEM (Sun et al., 2026), and xMemory (Hu et al., 2026), extend structured memory to online or long-term settings using semantic, temporal, or event-level organization.

Such structures improve coarse-to-fine access and expose relations that are difficult to capture in a flat store. However, hierarchy alone does not imply bounded memory growth. Retaining fine-grained leaves together with recursively generated summaries still produces at least a linear persistent state unless deletion or compaction is explicitly incorporated into memory maintenance. In OSET, the hierarchy is therefore not only an access structure: leaf evidence, internal abstractions, and structural relations all compete within the same retained-state budget and may be compacted or removed as the stream grows.

2.3 Bounded Online Memory and Evidence Retrieval

Recent memory systems increasingly treat online construction and maintenance as first-class operations. MemGPT moves information between active context and archival storage (Packer et al., 2023), while Mem0 extracts persistent factual memories from interaction histories (Chhikara et al., 2025). These systems reduce the amount of information directly exposed to the reader, but their archival or persistent stores are not necessarily bounded as experience accumulates.

Closer to our setting, ProStream maintains a fixed-capacity hierarchy through selective merging and removal (Wang et al., 2026), MemAgent learns to overwrite a fixed-size textual memory while

processing long inputs (Yu et al., 2026), and MemForest reduces online update cost through hierarchical temporal indexing while retaining canonical factual records (Chen et al., 2026). These approaches demonstrate complementary strategies for bounded memory and efficient online construction. OSET differs in allowing the retained state to grow with the stream, but only sublinearly, avoiding both append-only linear growth and a strictly fixed-capacity bottleneck.

Retrieval from compressed memory introduces an additional challenge: semantic similarity alone may fail to recover exact entities, numbers, or complementary evidence required by multi-hop questions. Iterative and graph-based retrieval systems such as IRCoT, HippoRAG, and NeuroPath show the value of combining semantic relevance with lexical or relational structure (Trivedi et al., 2023; Gutiérrez et al., 2024; Li et al., 2025). OSET follows this general principle but applies it to a memory that has already undergone budget-constrained online maintenance, combining semantic, lexical, exact-match, and structural signals to recover complementary surviving evidence.

Positioning of OSET. Unlike methods that only reduce active KV state or retrieved reader context, OSET explicitly bounds the complete retained memory. Its capacity grows sublinearly rather than remaining fixed, while semantic access cues and literal evidence share the same retained-state budget. This places OSET between fixed-capacity online memory and linearly growing external memory.

3 METHOD: OSET

3.1 OVERVIEW

OSET maintains a structured textual memory over an incoming stream while constraining the complete retained state to grow sublinearly with the observed input. Its design addresses a central tension of query-unknown memory construction: aggressive compression improves efficiency but may remove exact evidence required by future questions, whereas retaining all source content leads back to linear memory growth.

As illustrated in Fig. 2, OSET combines four components: evidence-preserving node construction, online hierarchical organization, budgeted memory maintenance, and evidence retrieval for a frozen reader. The following sections describe these components in detail.

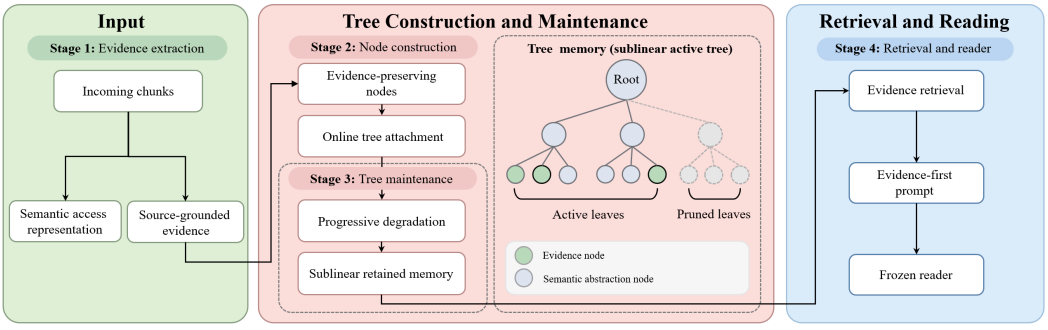

Figure 2: Core mechanisms of OSET: evidence-preserving node construction, online tree organization, sublinear budgeted maintenance, and query-time evidence retrieval.

3.2 MEMORY REPRESENTATION

Because future queries are unknown during construction, OSET must support semantic access without replacing exact answer-bearing evidence by generated summaries. Each chunk c_i produces a leaf node

$$m_i = (a_i, e_i, \pi_i, R_i), \tag{1}$$

where a_i is a compact semantic access representation, e_i stores source-grounded evidence, π_i records provenance, and R_i contains relations to other active nodes. We write $e_i = (E_i, N_i, F_i, W_i)$ for retained entities, exact values, factual spans, and short source windows. We further use m to

denote a generic active memory node, including both chunk-derived leaves and internal abstraction nodes.

The semantic field a_i is generated by a frozen LLM for organization and retrieval, whereas e_i is extracted from the source and remains attributable through π_i. Leaf nodes retain chunk-level evidence, while internal nodes store bounded semantic abstractions over related descendants. All retained textual fields are charged to the state budget in Sec. 3.4.

3.3 Online Tree Construction

OSET processes chunks in source order and updates only the currently active memory. Let $\mathcal{M}_{i-1}$ denote the active tree before inserting m_i. Rather than comparing the new node with every node in $\mathcal{M}_{i-1}$, OSET retrieves a bounded candidate-parent set from semantic, entity, lexical, and recency indexes. For each candidate parent $p \in \mathcal{C}_i$, OSET computes

$$s_{\text{attach}}(m_i, p) = \beta_t s_{\text{topic}}(m_i, p) + \beta_e s_{\text{entity}}(m_i, p) + \beta_l s_{\text{lex}}(m_i, p) + \beta_r s_{\text{recent}}(m_i, p), \qquad (2)$$

where the four component scores measure semantic compatibility, entity overlap, lexical overlap, and recency, respectively. All are normalized to $[0, 1]$, and $\beta_t, \beta_e, \beta_l, \beta_r$ are fixed non-negative fusion weights.

The highest-scoring candidate $p_i^\star = \arg\max_{p \in \mathcal{C}_i} s_{\text{attach}}(m_i, p)$ is selected and attached only if $s_{\text{attach}}(m_i, p_i^\star) \geq \tau_{\text{attach}}$, where τ_{attach} is the minimum compatibility threshold; otherwise, m_i starts a new branch. Thus, the term "compatible parent" refers explicitly to a candidate whose attachment score exceeds τ_{attach}.

The hierarchy groups related evidence for later compaction and provides parent–child or cross-node connections useful for multi-hop retrieval. Insertion updates only the selected branch, affected ancestors, and retrieval indexes, avoiding repeated global clustering. For efficiency, multiple adjacent chunks are summarized in one frozen-LLM call, while each chunk receives an independent bounded record and is inserted in source order.

3.4 Sublinear Budgeted Maintenance

A hierarchical memory remains linear if every incoming leaf and every generated abstraction are retained indefinitely. OSET therefore places the complete active textual memory under an explicit state envelope. After processing n source tokens,

$$S(\mathcal{M}_n) \leq B(n), \qquad B(n) = B_0 + \lceil cn^\alpha \rceil, \qquad 0 < \alpha < 1, \qquad (3)$$

where $\mathcal{M}_n$ denotes the active memory after observing n source tokens, $S(\mathcal{M}_n)$ is the charged retained-state size, $B_0 \geq 0$ is the initial budget, $c > 0$ controls its growth, and $0 < \alpha < 1$ ensures sublinear scaling.

For a node $m_i = (a_i, e_i, \pi_i, R_i)$, the charged textual state is defined as

$$S(\mathcal{M}) = \sum_{m \in \mathcal{M}} \big(|a(m)| + |e(m)| + |\pi(m)| + |R^{\text{text}}(m)| \big),$$

where $|\cdot|$ denotes the number of retained textual tokens under the reader tokenizer. R^{text} denotes any retained textual relation metadata; integer node identifiers and structural pointers are excluded from the textual-state accounting.

When $S(\mathcal{M}) > B(n)$, OSET restores the budget through *progressive degradation*, proceeding from the least destructive operation to node removal.

Redundancy removal. For overlapping source-backed spans x and y, OSET measures source overlap as

$$\rho(x, y) = \frac{\ell(\pi(x) \cap \pi(y))}{\min\{\ell(\pi(x)), \ell(\pi(y))\}},$$

where $\ell(\cdot)$ denotes span length in source-token positions, and $\pi(x)$ denotes the source interval associated with retained span x. If $\rho(x, y) \geq \tau_{\text{dup}}$ and all protected anchors are already covered, the longer or less informative span is removed while preserving its factual anchors.

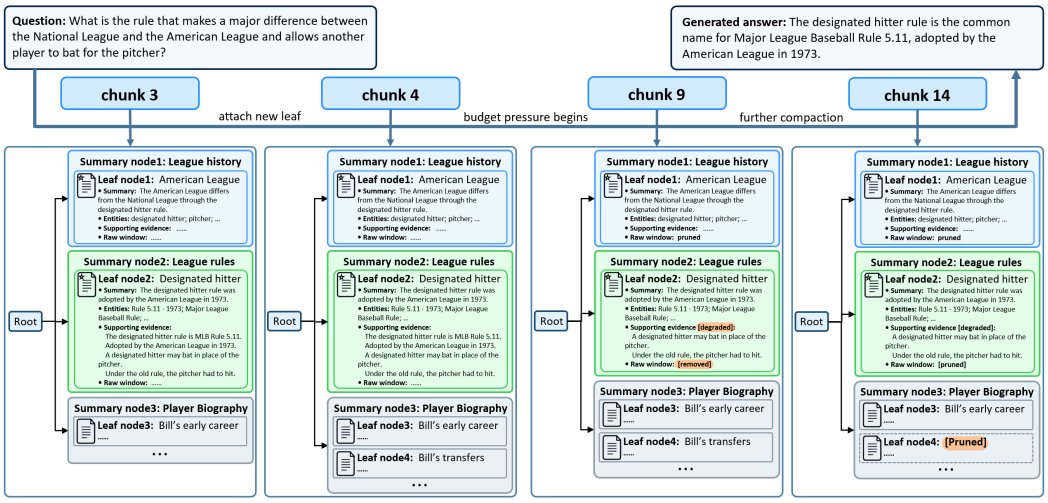

Figure 3: Example of online OSET memory evolution. Stars mark gold evidence leaves; orange labels indicate degradation or removal, and dashed nodes indicate pruning.

Field-level degradation. If redundancy removal is insufficient, OSET shortens semantic abstractions and local source windows while protecting entities, exact values, and factual spans until node-level pruning becomes necessary.

Node utility and pruning. Because future queries are unavailable during construction, OSET cannot use query-conditioned relevance to decide what to retain. Instead, it assigns each active node a query-independent retention utility:

$$u(m) = \lambda_f u_{\text{fact}}(m) + \lambda_u u_{\text{uniq}}(m) + \lambda_r u_{\text{rec}}(m) + \lambda_s u_{\text{struct}}(m), \tag{4}$$

where higher $u(m)$ indicates greater retention priority. Here u_{fact} measures the amount of protected factual evidence stored in the node, u_{uniq} rewards evidence not already covered by other active nodes, u_{rec} provides a weak recency prior, and u_{struct} protects nodes that organize surviving descendants or connect otherwise separated evidence. The coefficients $\lambda_f, \lambda_u, \lambda_r, \lambda_s$ control the contribution of the four query-independent signals.

OSET repeatedly removes the lowest-utility removable node, reattaches surviving children to the nearest retained ancestor, and updates affected indexes until $S(\mathcal{M}_n) \leq B(n)$.

Figure 3 illustrates this process on a concrete example, showing how OSET preserves answer-bearing evidence while progressively degrading and pruning lower-value content as the stream grows.

3.5 Evidence Retrieval and Reading

OSET uses a retrieve–expand–rerank pipeline because exact, lexical, semantic, and structural signals capture complementary evidence. Given a query q, it first forms a bounded seed set from exact-match, lexical, and semantic retrieval, then expands the seeds through bounded parent, child, sibling, and maintained cross-node relations. This allows evidence connected to a strong seed to enter the candidate pool even when it has weak direct lexical overlap with the query.

For semantic relevance, each candidate node m is rendered as text x_m, and a lightweight neural retriever computes $s_{\text{neu}}(q, m) = \text{sim}(f_q(q), f_m(x_m))$, where f_q and f_m are trainable projections over frozen Llama hidden representations and sim is cosine similarity. Final ranking combines complementary evidence signals:

$$s_{\text{ret}}(q, m) = w_n s_{\text{neu}}(q, m) + w_l s_{\text{lex}}(q, m) + w_e s_{\text{exact}}(q, m) + w_s s_{\text{struct}}(q, m). \tag{5}$$

Here s_{neu} captures semantic similarity, s_{lex} captures lexical relevance, s_{exact} rewards matches on protected literal anchors such as entities and numeric values, and s_{struct} measures query-time support induced by tree expansion from high-confidence seeds. The fusion weights w_n, w_l, w_e, w_s control their relative contributions. Unlike the query-independent u_{struct} in Eq. 4, s_{struct} is computed only at retrieval time and depends on the query-conditioned candidate expansion.

The neural component is trained using contrastive, hard-negative, multi-hop coverage, and candidate-ranking objectives; training details are provided in Appendix A.6. After reranking, OSET removes near-duplicate candidates and selects the highest-scoring surviving nodes under the reader-context budget. Retrieved literal evidence is rendered before generated abstractions in the final prompt. Because the number of retrieved nodes is bounded independently of $B(n)$, persistent-memory and reader-context budgets remain separate.

4 Experiments

We evaluate whether sublinear retained state preserves task utility, how quality trades off against retained and reader state, and how performance changes with context length. We use HotpotQA, MuSiQue, and 2WikiMultiHopQA for multi-hop reasoning, RULER from 8K to 128K tokens, and LongBench v2 for heterogeneous long-context evaluation. All generative experiments use the same frozen Llama-3.1-8B-Instruct reader whenever applicable.

4.1 Compared Methods

We compare against KV-cache compression and eviction methods, including StreamingLLM (Xiao et al., 2024b), H_2O (Zhang et al., 2023), SnapKV (Li et al., 2024), PyramidKV (Cai et al., 2024), Ada-SnapKV (Feng et al., 2024), K-VEC (Roy et al., 2026), SparQ (Ribar et al., 2024), and CSAttention (Song et al., 2026); query-dependent sparse or retrieval-based methods, including Quest, Loki, InfLLM, PQCache, and MagicPIG (Tang et al., 2024; Singhania et al., 2024; Xiao et al., 2024a; Zhang et al., 2025; Chen et al., 2025); and offloading-based methods including ShadowKV and InfiniGen (Sun et al., 2024; Lee et al., 2024). These methods primarily constrain active token-level state rather than complete retained memory.

We additionally evaluate **H_2O-Sublinear**, whose KV budget follows the same $B(n)$ schedule as OSET, isolating sublinear token eviction from explicit evidence construction.

4.2 Experimental Configuration

All generative evaluations use frozen Llama-3.1-8B-Instruct. Unless otherwise stated, OSET uses Eq. 3 with $B_0 = 1024$, $c = 8$, and $\alpha = 0.6$, and retrieves at most eight memory nodes per query. We use each benchmark's official metric; chunking, prompting, decoding, maintenance frequency, and implementation details are provided in Appendix B.

4.3 Multi-hop Reasoning

We evaluate HotpotQA, MuSiQue, and 2WikiMultiHopQA using token-level F1 on the same held-out examples across methods. These benchmarks test whether the retained state preserves evidence distributed across multiple passages.

As shown in Table 1, OSET remains competitive with full-context and KV-compression baselines under the sublinear state constraint. Its strongest result is on HotpotQA, while larger degradation on MuSiQue and 2WikiMultiHopQA indicates greater sensitivity to missing intermediate evidence and cross-passage dependencies. H_2O-Sublinear degrades substantially more under the same growth schedule, suggesting that explicit evidence preservation is more effective than token eviction alone under tight retained-state budgets.

4.4 RULER: Quality Across Context Lengths

RULER evaluates controlled long-context capabilities from 8K to 128K tokens, including sparse retrieval, multi-query retrieval, variable tracking, question answering, and global aggregation.

Method	HotpotQA	MuSiQue	2Wiki
Full context	58.00	36.00	45.00
OSET	61.00	31.00	43.00
StreamingLLM	46.57	25.24	39.14
PyramidKV	55.06	31.30	44.22
SnapKV	54.24	31.52	44.34
Ada-SnapKV	54.80	31.40	44.01
K-VEC	58.20	31.39	48.09
InfLLM	58.06	32.04	42.63
H_2O-Sublinear	50.48	21.63	28.83

Table 1: F1 scores on multi-hop benchmarks.

Method	Avg.
Full attention	82.12
OSET	78.37
Loki	35.52
InfiniGen	59.27
Quest	76.29
ShadowKV	83.57

Table 2: Average RULER score at 128K.

At 128K, OSET reaches 78.37 average score versus 82.12 for full attention (Table 2), remaining close to Quest and substantially above Loki and InfiniGen. Task-level results in Table 3 show a clear capability boundary: OSET remains strong on sparse needle retrieval and variable tracking, but degrades on multi-value, multi-query, QA, and especially word-frequency aggregation. This is consistent with its design trade-off between bounded evidence retention and exhaustive corpus coverage.

Metric	niah_s1	niah_s2	niah_m1	niah_m2	niah_mv	niah_mq	vt	fwe	qa_1	qa_2	10-task avg
Native Llama	98.80	98.20	**98.80**	96.80	**97.60**	**99.20**	13.64	**83.60**	**84.60**	**50.00**	**82.12**
OSET	**100.00**	**99.80**	98.60	**99.00**	78.45	76.20	**97.56**	30.07	61.60	42.40	**78.37**
OSET – Native	+1.20	+1.60	-0.20	+1.20	-19.15	-23.00	+83.92	-53.53	-23.00	-7.60	**-3.75**

Table 3: Task-level RULER results at 128K. S1/S2 denote single-needle retrieval, M1/M2 multi-needle retrieval, MV multi-value retrieval, MQ multi-query retrieval, VT variable tracking, and FWE frequent-word extraction. QA1/QA2 denote the two QA tasks.

4.5 LongBench v2

LongBench v2 evaluates heterogeneous long-context reasoning over document QA, multi-document QA, in-context learning, dialogue, code, and structured data. We evaluate all 503 multiple-choice examples using classification accuracy.

OSET achieves 31.6 overall accuracy versus 31.0 for full attention (Table 4), with no systematic degradation across context-length subsets. Its performance on heterogeneous inputs further indicates that the learned retrieval component is not limited to the multi-hop QA training distribution.

Method	Overall Acc	Easy	Hard	Short	Medium	Long
Full attention	31.0	35.4	28.3	37.2	26.0	30.6
OSET	31.6	37.0	28.3	37.2	28.8	27.8
H_2O	29.9	32.9	28.0	33.8	27.9	31.5
PQCache	29.8	33.3	27.7	37.8	22.3	31.5
SparQ	26.2	27.6	25.4	30.0	22.3	27.8
MagicPig	29.2	29.5	29.0	31.8	26.9	29.4
CSAttention	31.2	34.4	29.3	37.8	25.1	32.4

Table 4: LongBench v2 classification accuracy.

5 Analysis and Ablations

We next isolate the contributions of retrieval, hierarchical organization, evidence representation, and pruning. Unless otherwise stated, the underlying memory budget and reader configuration are held fixed.

5.1 Retrieval Ablation

We fix the constructed OSET memories and vary only the ranking function. All variants rank the same 100 candidates and retrieve the top eight. Table 5 reports MRR, nDCG@10, evidence Recall@8, all-group Recall@8, and Answer Recall@8; metric definitions are provided in Appendix B.4.

The results show complementary contributions from lexical, semantic, exact, and structural signals. BM25 substantially improves over random ranking, while the full retriever achieves the highest overall ranking quality and Answer Recall@8. These results indicate that semantic similarity alone is insufficient for recovering exact and complementary evidence.

Ranker on identical OSET memories	MRR	nDCG@10	R@8	All-group R@8	Answer R@8
Random	0.359	0.314	0.271	0.071	0.526
Node BM25	0.427	0.390	0.295	0.102	0.658
Trainable neural	0.522	0.471	0.309	0.129	0.690
Frozen cosine	0.622	0.563	0.317	0.133	0.780
Rule only	0.645	0.583	**0.320**	0.109	0.797
Full retrieval	**0.650**	**0.615**	0.319	**0.133**	**0.812**

Table 5: Retrieval ablation on identical OSET memories.

5.2 Memory Design Ablations

We isolate hierarchical organization, evidence representation, and pruning while holding the reader and retained-state budget fixed.

Table 6 shows that literal evidence preservation and evidence-aware pruning provide the largest gains. Removing anchors reduces answer survival, while FIFO pruning causes a sharp collapse in coverage. Semantic hierarchy and structural retrieval provide smaller but consistent additional improvements.

Component	**Variant**	**Group survival**	**Answer survival**	**All-group R@8**	**Answer R@8**	**State/Input**
Structure	Flat memory	54.6	–	53.6	67.6	–
	Temporal tree	67.2	–	60.8	60.6	–
	Semantic content only	**69.0**	–	64.0	67.8	–
	OSET full	**69.0**	–	**64.4**	**68.4**	–
Evidence	Summary only	–	63.6	62.0	58.4	0.95%
	Anchors only	–	70.8	63.2	69.2	6.99%
	Summary + anchors	–	**71.0**	**64.8**	**69.8**	7.91%
Pruning	FIFO	13.6	–	13.2	23.2	–
	Evidence only	**69.0**	–	64.0	68.6	–
	OSET	**69.0**	–	**64.8**	**69.2**	–

Table 6: Ablations of OSET memory design.

6 Conclusion

We introduced OSET, an online evidence-tree memory system that constrains the complete retained textual state to grow sublinearly with the observed stream. By separating semantic access cues from literal source-grounded evidence and combining hierarchical organization with evidence-aware maintenance, OSET preserves strong performance on retrieval-oriented long-context tasks while substantially reducing retained state and reader context. Experiments across multi-hop benchmarks, RULER, and LongBench v2 also reveal a clear limitation on corpus-wide aggregation, highlighting the trade-off imposed by strict sublinear retention. These results establish sublinear evidence memory as an intermediate regime between fixed-capacity memory and linearly growing history.

AI USE STATEMENT

In this work, we used generative AI tools for drafting portions of the manuscript and for language optimization, including improving grammar, clarity, style, and readability, and suggesting alternative phrasing. We also generated figures/images with the assistance of AI. We have not used generative AI tools for generating research ideas, conducting literature reviews, generating code, generating data, or producing experimental results. Additionally, we did not use generative AI tools for any tasks with recommended disclosure beyond those described above. We have reviewed all AI-assisted work. Specifically, all AI-assisted text was carefully reviewed, edited, and verified by the authors for accuracy, originality, and technical correctness; citations, technical claims, and terminology were checked against primary sources. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, et al. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024.

Han Chen, Zining Zhang, Wenqi Pei, Bingsheng He, Ming Wu, Jason Zeng, Michael Heinrich, Wei Wu, and Hongbao Zhang. Memforest: An efficient agent memory system with hierarchical temporal indexing. arXiv preprint arXiv:2605.23986, 2026. URL https://arxiv.org/abs/2605.23986.

Howard Chen, Ramakanth Pasunuru, Jason Weston, and Asli Celikyilmaz. Walking down the memory maze: Beyond context limit through interactive reading. *arXiv preprint arXiv:2310.05029*, 2023.

Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, and Beidi Chen. Magicpig: Lsh sampling for efficient llm generation. In *International Conference on Learning Representations*, 2025.

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.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. arXiv technical report arXiv:2404.16130, 2024. URL https://arxiv.org/abs/2404.16130.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. Optimizing kv cache eviction in llms: Adaptive allocation for enhanced budget utilization. *arXiv preprint arXiv:2407.11550*, 2024.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. In *NeurIPS*, 2024. URL https://openreview.net/forum?id=hkujvAPVsg.

Zhanghao Hu, Qinglin Zhu, Hanqi Yan, Yulan He, and Lin Gui. Beyond rag for agent memory: Retrieval by decoupling and aggregation. arXiv preprint arXiv:2602.02007, 2026. URL https://arxiv.org/abs/2602.02007.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. In *ACL*, 2024. URL https://aclanthology.org/2024.acl-long.91/.

Wonbeom Lee, Jungi Lee, Junghwan Seo, and Jaewoong Sim. Infinigen: Efficient generative inference of large language models with dynamic kv cache management. *arXiv preprint arXiv:2406.19707*, 2024.

Junchen Li, Rongzheng Wang, Yihong Huang, Qizhi Chen, Jiasheng Zhang, and Shuang Liang. Neuropath: Neurobiology-inspired path tracking and reflection for semantically coherent retrieval. In *NeurIPS*, 2025. URL https://arxiv.org/abs/2511.14096.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM Knows What You Are Looking for Before Generation. In *Neural Information Processing Systems*, 2024. doi: 10.52202/079017-0722. URL https://mlanthology.org/neurips/2024/li2024neurips-snapkv/.

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.

Alireza Rezazadeh, Zichao Li, Wei Wei, and Yujia Bao. From isolated conversations to hierarchical schemas: Dynamic tree memory representation for llms. In *ICLR*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/0382cb76309820f71c6eacd47b36ce71-Paper-Conference.pdf.

Luka Ribar, Ivan Chelombiev, Luke Hudlass-Galley, Charlie Blake, Carlo Luschi, and Douglas Orr. Sparq attention: Bandwidth-efficient llm inference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *PMLR*, pp. 42558–42583, 2024.

Shuvendu Roy, Mengyao Zhai, Hossein Hajimirsadeghi, and Golnoosh Samei. Coverage-driven kv cache eviction for efficient and improved inference of llm. *arXiv preprint arXiv:2606.29563*, 2026.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher Manning. Raptor: Recursive abstractive processing for tree-organized retrieval. In *ICLR*, 2024. URL https://openreview.net/forum?id=GN921JHCRw.

Prajwal Singhania, Siddharth Singh, Shwai He, Soheil Feizi, and Abhinav Bhatele. Loki: Low-rank keys for efficient sparse attention. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Chuxu Song, Zhencan Peng, Jiuqi Wei, and Chuanhui Yang. Csattention: Centroid-scoring attention for accelerating llm inference. *arXiv preprint arXiv:2604.08584*, 2026.

Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie Chi, and Beidi Chen. Shadowkv: Kv cache in shadows for high-throughput long-context llm inference. *arXiv preprint arXiv:2410.21465*, 2024.

Haoran Sun, Shaoning Zeng, and Bob Zhang. H-mem: Hierarchical memory for high-efficiency long-term reasoning in llm agents. In *EACL*, 2026. URL https://aclanthology.org/2026.eacl-long.15/.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context llm inference. In *International Conference on Machine Learning*, 2024.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *ACL*, 2023. URL https://aclanthology.org/2023.acl-long.557/.

Bingbing Wang, Jing Li, and Ruifeng Xu. Proactive memory for ad-hoc recall over streaming dialogues. arXiv preprint arXiv:2603.04885, 2026. URL https://arxiv.org/abs/2603.04885.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. Infllm: Training-free long-context extrapolation for llms with an efficient context memory. In *Advances in Neural Information Processing Systems*, volume 37, 2024a.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *ICLR*, 2024b. URL https://openreview.net/forum?id=NG7sS51zVF.

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 *ICLR*, 2026. URL https://arxiv.org/abs/2507.02259. Oral.

Hailin Zhang, Xiaodong Ji, Yilin Chen, Fangcheng Fu, Xupeng Miao, Xiaonan Nie, Weipeng Chen, and Bin Cui. Pqcache: Product quantization-based kvcache for long context llm inference. *Proceedings of the ACM on Management of Data*, 3(3):1–30, 2025.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In *NeurIPS*, 2023. URL https://openreview.net/forum?id=RkRrPp7GKO.

A Detailed OSET Formulation

A.1 Online Processing Procedure

Algorithm 1 summarizes the complete online workflow. OSET processes incoming chunks in source order and maintains only the currently active memory. Each chunk is converted into an evidence-preserving leaf, attached to the active hierarchy, and incorporated into the retrieval indexes. Whenever the charged retained state exceeds the current budget, budgeted maintenance is applied until the sublinear constraint in Eq. 3 is restored.

Algorithm 1 Online OSET construction

Require: Chunk stream $\{c_1, \ldots, c_T\}$, budget function $B(\cdot)$
Initialize active memory $\mathcal{M} \leftarrow \emptyset$
Initialize retrieval indexes $\mathcal{I}$
for $i = 1, \ldots, T$ **do**
    Construct leaf m_i from c_i
    Retrieve bounded candidate-parent set $\mathcal{C}_i$
    Attach m_i using Eq. 2
    Update affected ancestors and indexes $\mathcal{I}$
    if $S(\mathcal{M}) > B(n)$ **then**
        $\mathcal{M} \leftarrow \textsc{BudgetMaintain}(\mathcal{M}, B(n))$
    end if
end for
return $\mathcal{M}, \mathcal{I}$

A.2 Memory-Node Fields and State Accounting

As defined in Eq. 1, each chunk-derived leaf is represented as $m_i = (a_i, e_i, \pi_i, R_i)$. We use m to denote a generic active node, including both leaves and internal abstraction nodes.

Semantic access field. The field a_i is a compact semantic description produced by the frozen language model. It is used for organization and semantic retrieval, but is not treated as a replacement for literal source evidence.

Literal evidence field. We write

$$e_i = (E_i, N_i, F_i, W_i),$$

where E_i contains retained entities, N_i exact numeric or symbolic values, F_i selected factual spans, and W_i short source windows. These fields preserve source-grounded information that may be difficult to recover from generated abstractions alone.

Provenance. The provenance field π_i records the source location associated with retained evidence. It is used both for attribution and for detecting overlapping retained spans during maintenance.

Structural state. R_i records the active parent–child and maintained cross-node relations used by organization and retrieval. Structural identifiers and pointers are implementation metadata rather than textual memory.

State accounting. The charged textual state is

$$S(\mathcal{M}) = \sum_{m \in \mathcal{M}} \left(|a(m)| + |e(m)| + |\pi(m)| + |R^{\text{text}}(m)| \right),$$

where $|\cdot|$ denotes the number of retained textual tokens under the reader tokenizer. Generated abstractions, literal evidence, and textual provenance are therefore all charged to the same retained-state budget.

A.3 Candidate Generation and Online Attachment

For a new leaf m_i, OSET retrieves candidate parents from four query-independent sources: semantic similarity, entity overlap, lexical similarity, and recently active nodes. Let these source-specific candidate sets be $\mathcal{C}_i^{\text{sem}}$, $\mathcal{C}_i^{\text{ent}}$, $\mathcal{C}_i^{\text{lex}}$, and $\mathcal{C}_i^{\text{rec}}$. Their deduplicated union forms the bounded candidate set $\mathcal{C}_i$.

Each candidate $p \in \mathcal{C}_i$ is scored using Eq. 2. The semantic term measures topic compatibility between the incoming node and the candidate parent; the entity term measures overlap between retained entities; the lexical term measures literal similarity; and the recency term favors recently active branches. The component scores are normalized to a common range before fusion, and the combination weights are fixed throughout evaluation.

The selected parent is

$$p_i^\star = \arg\max_{p \in \mathcal{C}_i} s_{\text{attach}}(m_i, p).$$

The node is attached only if $s_{\text{attach}}(m_i, p_i^\star) \geq \tau_{\text{attach}}$; otherwise, a new branch is created.

Candidate-set sizes, fusion weights, and the attachment threshold are held fixed for all evaluated datasets rather than adapted per query.

A.4 Budgeted Maintenance

Maintenance is applied whenever the retained textual state must be restored to the budget in Eq. 3. OSET proceeds in increasing order of information loss.

Redundancy removal. For retained source-backed spans x and y, let $\pi(x)$ and $\pi(y)$ denote their provenance intervals. Their overlap is

$$\rho(x, y) = \frac{\ell(\pi(x) \cap \pi(y))}{\min\{\ell(\pi(x)), \ell(\pi(y))\}},$$

where $\ell(\cdot)$ denotes source-span length. When the overlap exceeds a fixed redundancy threshold and the protected literal anchors of one span are already covered elsewhere, redundant source context is removed while the anchors are retained.

Field-level degradation. If redundancy removal is insufficient, OSET shortens degradable fields before removing complete nodes. Semantic abstractions and local source windows are reduced first, while entities, exact values, and factual spans are protected whenever possible.

Query-independent retention utility. When node removal becomes necessary, OSET uses the retention utility in Eq. 4. The four components are normalized before fusion. u_{fact} rewards nodes containing protected factual evidence; u_{uniq} rewards evidence that is weakly represented elsewhere in the active memory; u_{rec} provides a weak recency prior; and u_{struct} protects nodes that organize surviving descendants or connect otherwise separated evidence.

Higher $u(m)$ therefore indicates greater retention priority. The fusion weights are fixed during evaluation and are not conditioned on the future query.

OSET repeatedly removes the lowest-utility removable node, repairs the local tree structure when necessary, and updates affected indexes until the retained state again satisfies the current budget.

A.5 Evidence Retrieval Architecture

At query time, OSET first retrieves exact-match, lexical, and semantic seed candidates and then performs bounded structural expansion through parent, child, sibling, and maintained cross-node relations.

For neural scoring, let

$$h_q = \mathrm{Llama}(q)_{\mathrm{last}}, \qquad h_m = \mathrm{Llama}(x_m)_{\mathrm{last}},$$

where the frozen Llama-3.1-8B-Instruct backbone provides the hidden state of the final non-padding token for the query and rendered node text, respectively. Lightweight trainable projections produce

$$z_q = \mathrm{norm}_2(W_q h_q + b_q), \qquad z_m = \mathrm{norm}_2(W_m h_m + b_m).$$

The neural relevance score is then

$$s_{\mathrm{neu}}(q, m) = z_q^\top z_m,$$

which is cosine similarity because both vectors are normalized.

The final score follows Eq. 5. Here s_{lex} measures query–node lexical relevance, s_{exact} rewards matches on protected literal anchors such as entities and numeric values, and s_{struct} measures query-time support induced by structural expansion from high-confidence seeds.

Importantly, s_{struct} is distinct from the query-independent structural utility u_{struct} in Eq. 4: the former is computed only after a query is available, whereas the latter is used during online memory maintenance.

A.6 Retriever Training Objectives

The retriever is trained with evidence supervision from the training splits of HotpotQA, MuSiQue, and 2WikiMultiHopQA. The training objective contains four complementary groups of supervision.

Contrastive discrimination. For a query q, multiple nodes may provide complementary valid evidence. We therefore treat all evidence-bearing nodes associated with the query as positives and contrast them against irrelevant nodes, including in-batch negatives. This objective establishes the basic separation between relevant and irrelevant memory content.

Hard-negative ranking. Random negatives are often substantially easier than nodes that mention the same entities, numbers, or topics as the gold evidence. We therefore include semantically or lexically related non-evidence nodes as hard negatives and explicitly encourage positive evidence to rank above these distractors.

Multi-hop coverage. Multi-hop questions may require several distinct evidence groups. Optimizing only the highest-scoring positive can repeatedly favor evidence from an easy hop while omitting another required reasoning step. We therefore supervise evidence groups separately and encourage the retrieved set to contain at least one useful node from each required group.

Candidate-level ranking. Training on small local negative sets can differ from ranking within the larger candidate pools generated by the deployed system. We therefore include ranking supervision over deployment-like candidate sets, incorporating evidence relevance, answer coverage, and evidence-group coverage.

The complete retriever objective is written as

$$\mathcal{L}_{\mathrm{ret}} = \lambda_{\mathrm{con}}\mathcal{L}_{\mathrm{con}} + \lambda_{\mathrm{hard}}\mathcal{L}_{\mathrm{hard}} + \lambda_{\mathrm{hop}}\mathcal{L}_{\mathrm{hop}} + \lambda_{\mathrm{rank}}\mathcal{L}_{\mathrm{rank}}.$$

B Experimental Details

B.1 Common Evaluation Setup

All generative experiments use Llama-3.1-8B-Instruct in bfloat16 with deterministic decoding. The reader remains frozen in all OSET experiments.

Unless otherwise stated, we use Eq. 3 with $B_0 = 1024$, $c = 8$, and $\alpha = 0.6$. HotpotQA, MuSiQue, and 2WikiMultiHopQA use 224-token chunks with 48-token overlap; RULER and LongBench v2 use 512-token chunks with 64-token overlap. The query-time retriever returns at most eight nodes.

B.2 Benchmark-Specific Setup

Multi-hop QA. We evaluate HotpotQA, MuSiQue, and 2WikiMultiHopQA on the same held-out examples across the compared methods and use token-level F1 as the primary answer metric. Our pipeline directly generates answer text and does not produce benchmark-specific supporting-fact identifiers; supporting-fact and joint metrics are therefore not used for the main comparison.

RULER. We evaluate ten RULER tasks at context lengths from 8K to 128K tokens. Table 7 lists the full task names corresponding to the abbreviations used in the main text. The selected tasks cover sparse single- and multi-target retrieval, multi-value or multi-query retrieval, variable tracking, question answering, and word-frequency aggregation. We report the official RULER task scores.

Table 7: Full names of the RULER tasks used in our evaluation.

Abbrev.	Full task name
S1	Single Needle-in-a-Haystack: Passkey Retrieval
S2	Single Needle-in-a-Haystack: Vanilla NIAH
M1	Multi-keys Needle-in-a-Haystack: Multi-keys
M2	Multi-keys Needle-in-a-Haystack: Full Haystack
MV	Multi-values Needle-in-a-Haystack
MQ	Multi-queries Needle-in-a-Haystack
VT	Variable Tracking
FWE	Frequent Words Extraction
QA1	Question Answering with SQuAD
QA2	Question Answering with HotpotQA

LongBench v2. We evaluate all 503 multiple-choice examples and report classification accuracy overall and across the benchmark's difficulty and context-length subsets.

B.3 Baseline and Budget Matching

For H_2O-Sublinear, the retained KV-token count follows the same sublinear schedule $B(n)$ used by OSET. OSET interprets this budget in retained textual tokens, whereas H_2O-Sublinear applies it to retained token positions in the KV cache. Thus, this comparison matches the state-growth law rather than exact byte-level memory footprint.

All other reader and benchmark settings are held fixed whenever the baseline implementation permits.

B.4 Evaluation Metrics

Let $G(q)$ denote the gold evidence nodes for query q, and $R_k(q)$ the top-k retrieved nodes.

Evidence Recall@8.

$$\mathrm{Recall@8}(q) = \frac{|G(q) \cap R_8(q)|}{|G(q)|}.$$

All-group Recall@8. Let $\mathcal{G}(q)$ be the set of required evidence groups. A query is counted as successful if every group contributes at least one retrieved node:

$$\mathbf{1}\left[\forall g \in \mathcal{G}(q),\ g \cap R_8(q) \neq \emptyset\right].$$

Answer Recall@8. Answer Recall@8 records whether the retrieved evidence contains the reference answer after the same answer normalization used by the evaluation pipeline.

Group survival. Group survival is measured before query-time ranking and indicates whether every required evidence group remains represented in the retained memory after construction and maintenance.

Answer survival. Answer survival indicates whether retained source-grounded content still contains the normalized reference answer.