EFFICIENT LONG-TERM MEMORY COMPRESSION FOR LLMS WITH SUBMODULAR OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) increasingly rely on external memory to support long-term interactions, but continuously growing memory introduces substantial storage, token, and computation overhead. Existing memory-management approaches often rely on heuristic retrieval or repeated LLM-based summarization and rewriting, which either provide limited control over redundancy under a fixed budget or incur additional model inference cost. We study long-term memory compression under two practically distinct access regimes. In the *full-access* setting, the complete candidate memory is available when compression is performed, as in periodically consolidated memory stores and query-time context construction. In the more restrictive *streaming* setting, memory chunks arrive continuously and must be processed in one pass without access to future interactions or future queries, as in long-running and resource-constrained agents. We propose SOMC, a submodular optimization framework that formulates memory selection as monotone submodular maximization under a single knapsack constraint. SOMC operates directly on existing memory chunks and therefore avoids repeatedly invoking an auxiliary generative LLM for memory summarization or rewriting. For full-access compression, we develop STATIC-SMK, a granularity-adaptive algorithm with provable approximation guarantees and linear oracle complexity. For continual memory maintenance, we develop STREAMING-SMK, which maintains a bounded, query-independent memory state in one pass and subsequently applies STATIC-SMK for query-aware context construction. Its approximation guarantee improves as individual memory chunks become smaller relative to the available budget, while its oracle complexity is linear in the stream length for fixed accuracy. Experiments on widely used widely used long-term memory benchmarks show that SOMC consistently preserves or improves downstream task performance under substantial memory compression, while reducing the number of processed tokens and end-to-end computational cost compared with model-based compression methods and conventional memory-management baselines. Results demonstrate that principled submodular optimization provides an efficient and model-agnostic alternative for scalable LLM memory management.

1 INTRODUCTION

Large language models (LLMs) are increasingly deployed as long-running assistants and agents that interact with users, tools, and environments over extended periods. Supporting such systems requires access to information that may have appeared many sessions earlier, including user preferences, past decisions, task experience, and factual updates. Consequently, modern LLM systems increasingly augment the model with an external long-term memory that stores and retrieves information beyond the active context window (Wang et al., 2023; Park et al., 2023; Packer et al., 2023; Zhong et al., 2024; Chhikara et al., 2025). As interaction histories grow, however, retaining and repeatedly processing all historical information becomes increasingly expensive in storage, token consumption, retrieval, and inference.

Long-term memory compression arises under two fundamentally different access regimes. In a *full-access* setting, the current candidate memory is already stored and can be revisited when compression is performed. The system can therefore globally select a compact subset from the complete memory collection. This setting naturally arises in server-side memory stores, periodic offline con-

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.

solidation, and query-time context construction, where the current memory pool remains accessible. In contrast, a long-running agent may receive memory continuously while being unable or unwilling to retain the complete raw interaction history. In this *streaming* setting, each memory chunk must be processed when it arrives; future chunks and future queries are unknown, and discarded chunks cannot later be recovered. Such a setting is particularly relevant to always-on personal assistants, continuously operating agents, and resource-constrained deployments.

Existing memory systems address memory growth through retrieval, truncation, summarization, distillation, or learned memory controllers (Wang et al., 2025; Chhikara et al., 2025; Xu et al., 2026; Fang et al., 2026; Ma et al., 2026). LLM-based summarization and rewriting can integrate information flexibly, but repeated autoregressive generation introduces additional latency, token cost, and dependence on the underlying model. Lightweight retrieval and filtering avoid much of this generation cost, but memory selection is often driven by heuristic scores or local decisions, offering limited formal guidance for jointly balancing information coverage, redundancy, and heterogeneous token costs under a hard memory budget. These observations motivate two research questions:

- $\mathcal{Q}1$: *Under full access to the current memory collection, can memory compression be formulated as a principled and computationally efficient budgeted selection problem with provable approximation guarantees?*
- $\mathcal{Q}2$: *When memory arrives continuously, can we maintain a bounded long-term memory state in one pass, without storing or revisiting the complete interaction history, while retaining provable utility guarantees and low update overhead?*

Submodular optimization provides a natural answer to $\mathcal{Q}1$. Its diminishing-returns property assigns decreasing marginal value to information that is already represented by the selected memory, while favoring chunks that add new coverage. Together with a knapsack constraint for heterogeneous token costs, this yields a direct formulation of memory compression as monotone submodular maximization under a single knapsack constraint. Based on this formulation, we develop STATIC-SMK for the full-access setting. Unlike generative compression, STATIC-SMK selects existing memory chunks without repeatedly invoking an auxiliary LLM to summarize or rewrite them. At the same time, it provides a principled approximation guarantee for the resulting memory utility rather than relying solely on heuristic ranking or truncation. A key observation is that memory compression has additional structure that generic knapsack instances do not necessarily possess: an individual memory chunk is typically small relative to the total memory budget. We capture this structure by a granularity parameter β, defined formally in Section 4.2, which measures the largest feasible chunk cost relative to the current budget. For an accuracy parameter $\epsilon > 0$, STATIC-SMK preserves the generic $1/2 - \epsilon$ guarantee while obtaining increasingly stronger guarantees for finer-grained memories, approaching the classical $1 - 1/e$ benchmark as both β and ϵ become small. In the fine-grained regime, it additionally avoids the more expensive general-case post-processing and uses only $O(n/\epsilon)$ value-oracle evaluations.

Full-access compression, however, does not solve the entire long-term memory problem. It assumes that the complete candidate history remains available until compression is performed. For a continuously operating agent, repeatedly storing an ever-growing raw history and rerunning global compression whenever new interactions arrive can itself become a memory-management bottleneck. This motivates the streaming setting considered in $\mathcal{Q}2$. To address $\mathcal{Q}2$, we develop STREAMING-SMK, which incrementally maintains a bounded, query-independent memory state in one pass. Each memory chunk is processed upon arrival, and discarded chunks never need to be revisited. STREAMING-SMK combines online value calibration with density-based selection, allowing the maintained memory to adapt to a continuously growing stream without rerunning optimization over the complete interaction history. The algorithm provides a granularity-dependent approximation guarantee that becomes stronger as individual memory chunks become smaller relative to the memory budget, while its value-oracle complexity is linear in the stream length for fixed accuracy. Once a user query arrives, STATIC-SMK is applied only to the retained memory state to construct the final query-aware context.

Together, STATIC-SMK and STREAMING-SMK form SOMC, a unified optimization framework for long-term memory compression under two complementary access regimes. The two settings share the same budgeted submodular view but differ fundamentally in the information available at decision time: full-access compression can globally optimize over the complete current memory collection,

whereas streaming compression must make irrevocable decisions using only the observed prefix. Importantly, the same granularity parameter β strengthens the approximation guarantees in both settings, establishing a direct connection between a practical system design choice—memory chunk size—and the quality achievable by budget-constrained memory selection.

Our main contributions are as follows:

- **A unified formulation of two memory-access regimes.** We distinguish full-access compression, where the complete current memory collection can be revisited, from streaming compression, where memory arrives sequentially and discarded chunks cannot be recovered. We formulate both settings as monotone submodular maximization under a single knapsack constraint, with memory utilities that capture semantic coverage, diversity, and, when available, query relevance.
- **Efficient and theoretically grounded full-access compression.** We develop STATIC-SMK, a non-generative memory-selection algorithm that avoids repeated LLM-based summarization or rewriting. It achieves a granularity-dependent approximation guarantee that retains the generic $1/2 - \epsilon$ bound and approaches the classical $1 - 1/e$ benchmark as the chunk granularity and accuracy parameter become small. In the fine-grained regime, the algorithm requires only $O(n/\epsilon)$ value-oracle evaluations.
- **Provable one-pass continual memory maintenance.** We develop STREAMING-SMK for long-running agents whose complete interaction history cannot be repeatedly retained and revisited. The algorithm maintains a bounded, query-independent memory state in one pass and provides a granularity-dependent approximation guarantee: it has a $1/3 - \epsilon$ general-case floor and approaches $1/2 - \epsilon$ as memory chunks become fine-grained. Its value-oracle complexity is $O((n/\epsilon)\log(1/\epsilon))$, which is linear in the stream length for fixed accuracy.
- **Extensive evaluation on long-term memory benchmarks.** We evaluate SOMC under both full-access and streaming compression on long-term memory benchmarks and across local and online LLMs. Results show that SOMC substantially reduces retained tokens and memory-management overhead while preserving competitive downstream answer quality.

2 RELATED WORK

2.1 SUBMODULAR OPTIMIZATION

Submodular optimization is a natural framework for representative subset selection, as diminishing returns favor informative but non-redundant elements. Classical work established strong guarantees for monotone submodular maximization under cardinality constraints (Nemhauser et al., 1978), while (Sviridenko, 2004) extended this setting to non-uniform costs with the optimal $1 - 1/e$ approximation under a knapsack constraint. Submodular objectives have also been widely used for data and document summarization (Lin & Bilmes, 2011). Recent work has developed increasingly efficient algorithms for monotone submodular maximization under knapsack constraints. Han et al. (2021) study approximation algorithms for submodular data summarization, while Li et al. (2022) give a deterministic clean-time algorithm with a $1/2$ approximation for a single knapsack constraint. For sequentially arriving data, streaming methods process elements without repeatedly scanning the entire ground set (Badanidiyuru et al., 2014; Kazemi et al., 2019; Huang et al., 2020). More recently, Cui et al. (2025) develop linear-time algorithms for representative subset selection from streams. Our work uses this optimization toolkit for LLM memory compression, where token costs are heterogeneous and the ratio between the largest memory chunk and the available budget provides additional structure. This structure is exploited by both STATIC-SMK and STREAMING-SMK.

2.2 LONG-TERM MEMORY COMPRESSION FOR LARGE LANGUAGE MODELS

Long-term memory for LLMs was initially explored through external memory, hierarchical context management, and summarization. LongMem augments language models with a dedicated long-term memory retriever (Wang et al., 2023), while MemGPT manages multiple memory tiers to extend effective context beyond the model window (Packer et al., 2023). Recursive summarization instead repeatedly integrates new dialogue into compact summaries (Wang et al., 2025). MemoryBank

further introduces persistent storage, retrieval, and memory updating for long-term personalized interaction (Zhong et al., 2024). Meanwhile, benchmarks such as LoCoMo and LongMemEval (Maharana et al., 2024; Wu et al., 2024) expose the difficulty of maintaining factual, temporal, and cross-session information over extended conversations.

Recent work increasingly treats memory as an actively constructed and evolving resource. Mem0 extracts, consolidates, and retrieves salient information, with a graph-based variant for relational memory (Chhikara et al., 2025), whereas A-MEM dynamically links and updates interconnected memory notes (Xu et al., 2026). LightMem emphasizes efficient memory management through lightweight filtering, topic-aware consolidation, and offline long-term updates (Fang et al., 2026), while Nemori studies what information deserves retention through adaptive memory distillation (Ma et al., 2026). The preprint work (Ghulyani et al., 2026) formulates query-time context assembly as budget-constrained submodular selection over an available context pool. Meanwhile, the preprint work (Alqithami, 2025) studies memory-budgeted retention and principled forgetting under privacy and efficiency constraints. In contrast, SOMC jointly supports offline memory compression and causal one-pass memory maintenance, together with query-aware selection after the query arrives.

3 PRELIMINARY

3.1 MEMORY COMPRESSION SETTINGS

Let the complete interaction history be $\mathcal{H} = (d_1, \ldots, d_T)$, where d_t denotes the t-th User–LLM interaction turn. A preprocessing function partitions consecutive interactions into n memory chunks $\mathcal{X} = (x_1, \ldots, x_n)$. Each chunk x_i has token cost $c_i = c(x_i) < B_c$, and for any $S \subseteq \mathcal{X}$ we write $c(S) = \sum_{x_i \in S} c_i$. We consider the same two access regimes introduced in Section 1.

- **Full-access compression.** The current candidate collection is completely available and can be revisited during optimization. Static-SMK may therefore be used to construct a query-independent memory bank $\mathcal{M}$ under storage budget B_m, and later, after query q arrives, to select a query-aware context S_q under prompt budget B_q. This regime covers server-side memory stores, periodic offline consolidation, and query-time context construction.
- **Streaming compression.** Memory chunks arrive sequentially as $\mathcal{X}_t = (x_1, \ldots, x_t)$. At time t, future chunks and queries are unknown, and a discarded chunk cannot be revisited. Streaming-SMK incrementally maintains a bounded memory state $\mathcal{M}_t$ under budget B_s. Once a query q arrives, Static-SMK is applied only to $\mathcal{M}_t$ to obtain the final query-aware context S_q under budget B_q. This regime models long-running assistants and agents for which retaining and repeatedly recompressing the complete raw history is undesirable.

The distinction is therefore an information-access constraint rather than a difference in the downstream task: full-access compression can optimize over the complete current candidate set, whereas streaming compression must make irrevocable decisions from the observed prefix. In both cases, the selected memory is combined with the user query and passed to the LLM to generate the response.

3.2 PROBLEM FORMULATION

Given memory chunks $\mathcal{X}$, we seek a representative subset $S \subseteq \mathcal{X}$ that maximizes retained memory utility under a token budget. Let $f : 2^{\mathcal{X}} \to \mathbb{R}_{\geq 0}$ be a normalized set function, i.e., $f(\emptyset) = 0$, where $f(S)$ measures the utility of S. For $x \in \mathcal{X} \setminus S$, its marginal gain is $f(x \mid S) = f(S \cup \{x\}) - f(S)$. We assume that f is monotone and submodular: for any $A \subseteq D \subseteq \mathcal{X}$ and $x \notin D$, $f(A) \leq f(D)$ and $f(x \mid A) \geq f(x \mid D)$. Throughout the theoretical analysis, $\epsilon \in (0, 1/3)$ denotes the accuracy parameter. Given a budget $B > 0$, each compression stage is formulated as monotone **S**ubmodular **M**aximization under a single **K**napsack constraint (abbreviated as **SMK**):

$$\mathrm{OPT} \in \arg\max_{S \subseteq \mathcal{X}} f(S) \quad \text{s.t.} \quad c(S) \leq B, \tag{1}$$

where $f(\mathrm{OPT})$ denotes the optimal utility. The same optimization template is instantiated with different information and budgets: $B = B_m$ for full-access memory-bank construction, $B = B_q$ for query-aware context selection, and $B = B_s$ for one-pass streaming maintenance. For a streaming prefix $\mathcal{X}_t = (x_1, \ldots, x_t)$, we analogously write OPT_t for an optimal feasible subset of $\mathcal{X}_t$.

4 METHODOLOGY

4.1 OVERVIEW OF SOMC

In this work, we propose a submodular optimization framework for efficient long-term memory compression (SOMC) in LLMs. Figure 1 summarizes the two pipelines of our framework. For full access, we propose STATIC-SMK, which globally selects a representative memory bank from the current history and can be invoked again after a query arrives to construct the final query-aware context. For streaming access, we propose STREAMING-SMK, which replaces the first global compression stage: it processes each arriving chunk once and continuously maintains a bounded query-independent memory state. The query-time stage is then identical in both regimes, with STATIC-SMK selecting the final context from the retained memory.

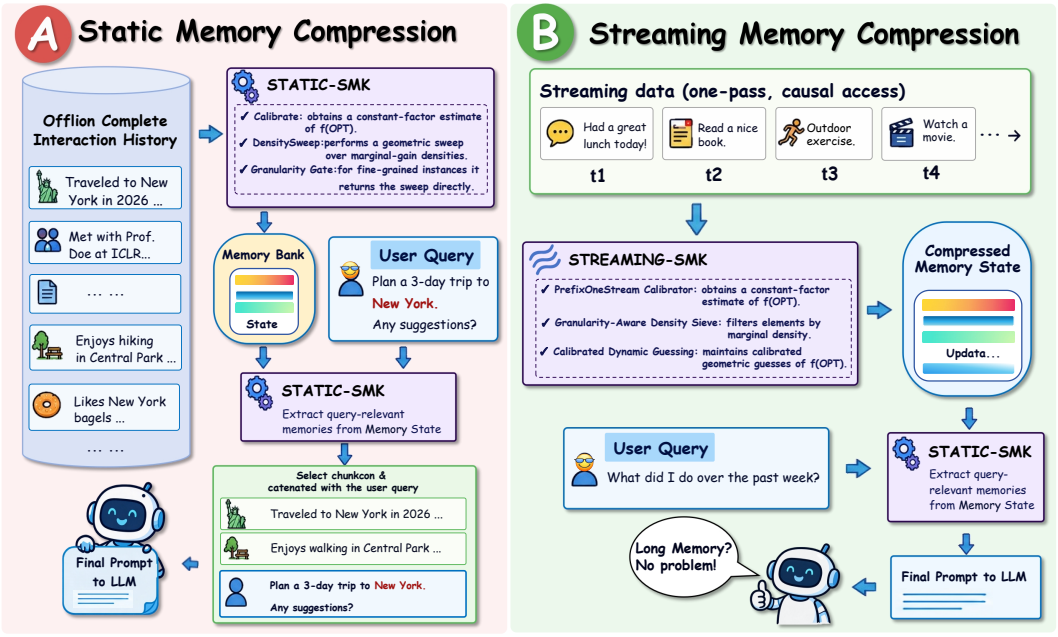

Figure 1: Submodular optimization framework for efficient long-term memory compression

4.2 SUBMODULAR MEMORY UTILITY AND GRANULARITY

Let $\mathcal{X} = (x_1, \ldots, x_n)$ denote the candidate memory chunks of a compression stage, where x_i has token cost $c_i = c(x_i) > 0$ and normalized embedding $\mathbf{e}_i \in \mathbb{R}^d$. We model memory utility through three complementary factors: semantic coverage, diversity, and query relevance when q is available.

Memory utility. Given a reference set $\mathcal{R}$, semantic coverage is measured by $f_{\text{cov}}(S;\mathcal{R}) = \frac{1}{|\mathcal{R}|}\sum_{r\in\mathcal{R}} \max_{x_j\in S}[\mathbf{e}_r^\top \mathbf{e}_j]_+$, where $[z]_+ = \max\{z, 0\}$ and the maximum is defined as 0 for $S = \emptyset$. This term rewards subsets that collectively represent the reference information. To encourage semantic diversity, define $K_{ij} = \mathbf{e}_i^\top \mathbf{e}_j$, $f_{\text{div}}(S) = \log\det(I_{|S|} + K_S)$ and $f_{\text{div}}(\emptyset) = 0$, where K_S is the principal submatrix indexed by S.

When a query q becomes available, each memory chunk x_i is assigned a non-negative hybrid relevance score $r(x_i, q) \in [0, 1]$, combining dense cosine similarity with a BM25-based lexical score augmented by entity and temporal matching. The resulting relevance utility is $f_{\text{rel}}(S; q) = \sum_{x_i\in S} r(x_i, q)$. Since $r(x_i, q)$ is fixed once q is observed, this term is non-negative and modular.

Objectives under the two access regimes. Under full access, all current candidates are known before selection. The candidate collection can therefore serve as the coverage reference set, giving

$$f_{\text{stat}}(S; q) = \lambda_{\text{cov}} f_{\text{cov}}(S; \mathcal{X}) + \lambda_{\text{div}} f_{\text{div}}(S) + \lambda_{\text{rel}} f_{\text{rel}}(S; q). \tag{2}$$

During query-independent memory-bank construction, the future query is unavailable, so $\lambda_{\text{rel}} = 0$ and $B = B_m$. At query time, q is known and $B = B_q$. Streaming compression cannot use the

complete future memory as a reference set and cannot evaluate query relevance before the query arrives. We therefore use a fixed feature-saturation coverage. Each embedding $\mathbf{e}_i$ is split into its positive and negative components and then l_1-normalized to obtain the non-negative feature vector $\mathbf{z}_i$, and $f_{\mathrm{sat}}(S) = \frac{1}{\tau_{\mathrm{sat}}} \sum_j \log\big(1 + \tau_{\mathrm{sat}} \sum_{x_i \in S} z_{ij}\big)$ where $\tau_{\mathrm{sat}} > 0$. The streaming objective is

$$f_{\mathrm{str}}(S) = \lambda_{\mathrm{cov}} f_{\mathrm{sat}}(S) + \lambda_{\mathrm{div}} f_{\mathrm{div}}(S). \tag{3}$$

Both objectives are normalized, non-negative, monotone, and submodular for non-negative coefficients; Appendix A gives the proof.

Memory granularity. For a compression stage with budget B, after discarding chunks with $c_i > B$, define $\beta = \max_{x_i : c_i \le B} \frac{c_i}{B} \in (0, 1]$. Thus, β measures the largest indivisible memory chunk relative to the available budget. Under full access it is computed from the current candidates; under streaming access the chunking policy supplies an ex-ante cap β_s satisfying $c_i \le \beta_s B_s$. This structural parameter is independent of memory objective and is used only in approximation analysis.

4.3 Full-Access Compression with Static-SMK

In the full-access setting, all $n = |\mathcal{X}|$ candidate memory chunks are available when compression is performed. Static-SMK therefore optimizes the selected subset globally under the token budget. The algorithm follows a density-based selection strategy: it first estimates the unknown optimal utility $f(\mathrm{OPT})$ and then progressively admits memory chunks according to their marginal utility per token. The corresponding auxiliary procedures, Calibrate and DensitySweep, are deferred to Appendix B. For an ordered acceptance list A, let $\mathrm{Pref}(A, b)$ denote its longest prefix with total cost at most b, and let $\mathrm{set}(A)$ denote the set of elements in A.

The main difficulty arises from coarse-grained memory chunks. When every chunk is small relative to the budget, i.e., the granularity $\beta = \max_i c_i/B$ is small, the density sweep alone already yields a stronger approximation guarantee. When β is large, however, an individually valuable chunk may become infeasible after several smaller chunks have been selected. To handle this case, Static-SMK augments the density-sweep solution with a small family of prefix-based one-element candidates. The algorithm switches between the two branches at $\beta_\epsilon^\star = 1 - \frac{\ln(1/(1/2+\epsilon))}{1-\epsilon}$, which is exactly the granularity at which the granularity-dependent guarantee $1 - e^{-(1-\epsilon)(1-\beta)}$ matches the generic guarantee $1/2 - \epsilon$. Thus, when $\beta \le \beta_\epsilon^\star$, the simpler density-sweep branch is already sufficient; otherwise, the robust branch is used.

Algorithm 1 Static-SMK

Require: $\mathcal{X}$, utility f, costs c, budget B, and $\epsilon \in (0, 1/3)$
Ensure: feasible subset S_{off}

Discard x_i with $c_i > B$; $\beta \leftarrow \max_i c_i/B$; $\Gamma \leftarrow$ Calibrate$(\mathcal{X}, f, c, B)$
if $\beta \le \beta_\epsilon^\star$ **then**
  $A \leftarrow$ DensitySweep$(\mathcal{X}, f, c, B, \Gamma, \epsilon, 1)$
  return $\mathrm{set}(A)$
end if
$A \leftarrow$ DensitySweep$(\mathcal{X}, f, c, B, \Gamma, \epsilon, 1/\epsilon)$; $\mathcal{C} \leftarrow \{\mathrm{set}(A)\}$
$J \leftarrow \lfloor \log_{1+\epsilon}(1/\epsilon) \rfloor$
for $j = -1, 0, \ldots, J$ **do**
  $P \leftarrow \emptyset$ if $j = -1$; otherwise $P \leftarrow \mathrm{Pref}(A, \epsilon B(1+\epsilon)^j)$
  $x^\star \leftarrow \arg\max_{x_i \notin P:\, c_i \le B - c(P)} f(x_i \mid P)$, if one exists
  Add $P \cup \{x^\star\}$ to $\mathcal{C}$ if $x^\star$ exists; otherwise add P
end for
return $\arg\max_{S \in \mathcal{C}} f(S)$

The following theorem summarizes the approximation and oracle-complexity guarantees of Static-SMK; the complete analysis is given in Appendix B.

Theorem 1 (Static-SMK Guarantee). *For the accuracy parameter $\epsilon \in (0, 1/3)$ defined in Section 3,* Static-SMK *returns a feasible set S_{off} satisfying*

$$f(S_{\mathrm{off}}) \ge \max\left\{\frac{1}{2} - \epsilon,\ 1 - e^{-(1-\epsilon)(1-\beta)}\right\} f(\mathrm{OPT}), \tag{4}$$

where OPT *is optimal solution for SMK problem. When* $\beta \le \beta_\epsilon^\star$*, the algorithm uses* $O(n/\epsilon)$ *value-oracle evaluations. Otherwise, the algorithm uses* $O(n\epsilon^{-1}\log(1/\epsilon))$ *value-oracle evaluations.*

4.4 ONE-PASS COMPRESSION WITH STREAMING-SMK

STREAMING-SMK replaces the first full-access compression stage when the complete history cannot be retained and revisited. It optimizes the query-independent objective f_{str} under budget B_s and the ex-ante granularity cap $c_i \le \beta_s B_s$. The main difficulty is that the optimal utility scale changes over time and cannot be computed from the unseen suffix of the stream.

Online value estimation. For each prefix $\mathcal{X}_t = (x_1, \ldots, x_t)$, let OPT_t denote an optimal feasible solution to the streaming SMK instance induced by $\mathcal{X}_t$. We run the OneStream algorithm of Cui et al. (2025) with $\hbar = 2$ on the same pass. After each arrival, a non-mutating prefix-output wrapper applies OneStream's original terminal extraction rule to a copy of the current state and returns a feasible set R_t. Letting $H_t = \max_{s \le t} f(R_s)$, the fixed-accuracy guarantee yields $f(\mathrm{OPT}_t) \le C_0 H_t$ with $C_0 = 16$. This estimate is used only to restrict the range of plausible optimal values; the memory candidates themselves are maintained by the density instances described below. Appendix C.1 states the exact OneStream routine and the prefix wrapper.

Algorithm 2 STREAMING-SMK

Require: one-pass stream $x_1, x_2, \ldots$, utility f, costs c, budget B_s, cap β_s, and $\epsilon \in (0, 1/3)$
Ensure: feasible retained memory Q

$\kappa \leftarrow \max\{1/2, 1-\beta_s\}$; $\theta \leftarrow (1-\epsilon)/(1+\kappa)$; $\gamma \leftarrow \theta\kappa$
Initialize the exact ONESTREAM state with $\hbar = 2$; $H, LB \leftarrow 0$; $Q, R_H \leftarrow \emptyset$; active set $\mathcal{I} \leftarrow \emptyset$
for each arriving x_i with $c_i \le B_s$ **do**
  Update the ONESTREAM state with x_i; $R \leftarrow$ PREFIXOUTPUT(state)
  if $f(R) > H$ **then**
    $H \leftarrow f(R)$; $R_H \leftarrow R$
  end if
  if $H > LB$ **then**
    $Q \leftarrow R_H$; $LB \leftarrow H$
  end if
  if $H = 0$ **then**
    continue
  end if
  $L \leftarrow LB/\gamma$; $U \leftarrow C_0 H/\epsilon$
  Delete active guesses $v < L$; activate missing $v \in \mathcal{G}_\epsilon \cap [L, U]$ with $S_v \leftarrow \emptyset$
  for each active $v \in \mathcal{I}$ **do**
    $\Delta \leftarrow f(x_i \mid S_v)$
    if $c(S_v) + c_i \le B_s$ and $\Delta \ge (\theta v/B_s)c_i$ **then**
      $S_v \leftarrow S_v \cup \{x_i\}$; update (Q, LB) if $f(S_v) > LB$
    end if
  end for
  Delete active guesses $v < LB/\gamma$
end for
return Q

Density-based selection. Set $\kappa_{\beta_s} = \max\left\{\frac{1}{2}, 1-\beta_s\right\}$, $\theta = \frac{1-\epsilon}{1+\kappa_{\beta_s}}$, and $\gamma = \theta\kappa_{\beta_s}$. The algorithm maintains a geometric grid $\mathcal{G}_\epsilon = \{(1+\epsilon)^j : j \in \mathbb{Z}\}$ of plausible optimal values. If LB_t is the best maintained solution value, only guesses in $[LB_t/\gamma,\, C_0 H_t/\epsilon]$ remain active. Each active guess v maintains a feasible set S_v and accepts arriving chunk x_i when $c(S_v)+c_i \le B_s$, $f(x_i \mid S_v) \ge \frac{\theta v}{B_s} c_i$. The complete one-pass procedure is summarized in Algorithm 2.

At each arrival, the online estimate is updated before the same chunk is evaluated by all currently relevant density instances. Consequently, Streaming-SMK adapts to the growing stream without rerunning optimization over previously observed chunks. Define

$$\rho_{\mathrm{str}}(\beta_s) = \frac{\kappa_{\beta_s}}{1+\kappa_{\beta_s}} = \begin{cases} \dfrac{1-\beta_s}{2-\beta_s}, & \beta_s < 1/2, \\ \dfrac{1}{3}, & \beta_s \ge 1/2. \end{cases} \tag{5}$$

Theorem 2 (Streaming-SMK Guarantee). *For every finite prefix $\mathcal{X}_t$,* STREAMING-SMK *returns a feasible Q_t satisfying*

$$f(Q_t) \geq (\rho_{\mathrm{str}}(\beta_s) - \epsilon) f(\mathrm{OPT}_t). \tag{6}$$

For n arrivals, it uses $O((n/\epsilon)\log(1/\epsilon))$ value-oracle evaluations. If token costs satisfy $c_i \geq 1$, the maintained density instances store $O(B_s/\epsilon)$ elements.

The approximation and complexity analyses are provided in Appendix C. The guarantee has a $1/3-\epsilon$ floor for arbitrary granularity and approaches $1/2 - \epsilon$ as $\beta_s \to 0$. Accordingly, the streaming pipeline first uses STREAMING-SMK to maintain a bounded query-independent memory state $\mathcal{M}_t$, and then applies STATIC-SMK to $\mathcal{M}_t$ once the query arrives to construct the final query-aware context S_q. Pseudocode for the static and streaming memory compression procedures is provided in Appendix D.

5 EXPERIMENTS

We evaluate SOMC against state-of-the-art baselines on widely used datasets of varying scales, considering both memory quality under static and streaming compression and runtime efficiency. We further assess its generalizability across locally deployed and API-based LLMs. Additional experimental settings, parameter configurations, ablation studies, theoretical bounds and complexity analyses, and prompt templates are provided in the appendix. The code is available at the following anonymous repository: `https://anonymous.4open.science/r/soma-CCD3`.

5.1 EXPERIMENTAL SETTINGS

Datasets: We conduct experiments on two widely used long-term memory benchmarks. $LoCoMo_{10}$ (Maharana et al., 2024), officially released as part of LoCoMo, contains 10 conversations with an average length of about 24,000 tokens and 1,540 questions spanning four reasoning categories. $LongMemEval_S$ (Wu et al., 2024) contains 500 evaluation instances, each associated with a substantially longer multi-session interaction history of approximately 105K tokens on average under the same tokenizer. $LongMemEval_S$ provides substantially longer contexts, allowing us to evaluate SOMC scalability under demanding long-context conditions.

Baselines and LLMs: We compare SOMC against seven baselines: Full Context, which uses the entire dialogue history; Naive RAG, based on dense retrieval; MemoryOS (Kang et al., 2025); PACMS (Ghulyani et al., 2026), reimplemented following its original procedure; SeCom (Pan et al., 2025); LightMem (Fang et al., 2026); and NEMORI (Ma et al., 2026). To assess generalizability across deployment settings, we evaluate all methods using both a locally deployed LLM (Qwen-3.5 9B) and an online LLM (GPT-5).

Evaluation metrics and parameter settings: Following (Ma et al., 2026), we adopt an **LLM-Judge score**, using GPT-5 as the judge. We additionally report **F1** (Lin, 2004) and **BLEU** (Papineni et al., 2002) scores. To evaluate computational efficiency, we report the average token cost incurred by **Token-Used**, the average number of **Prompt Tokens**, and the average **running time**. For all baseline, we use their original parameter settings. All experiments are repeated five times. For SOMC, we set $B_q = 4096$ tokens. Before the query arrives, the objective weights are set to $f_{\mathrm{cov}} = 0.2$ and $f_{\mathrm{div}} = 0.8$. Once the query becomes available, we set $f_{\mathrm{cov}} = f_{\mathrm{div}} = 0.1$ and $f_{\mathrm{rel}} = 0.8$. Further details regarding parameter selection and configuration are shown in Appendix E.

5.2 MAIN RESULTS

We compare SOMC with existing baselines on Qwen-3.5 9B and GPT-5 using $LoCoMo_{10}$ and $LongMemEval_S$. SOMC_a denotes global static compression, while SOMC_s denotes streaming compression. The results are summarized in Table 1. SOMC_a consistently achieves performance comparable to, and in several cases better than, state-of-the-art LLM-based methods such as NEMORI and LightMem. One possible explanation is that SOMC explicitly prioritizes preserving useful evidence rather than generating higher-quality summaries. SOMC_a also consistently outperforms the closely related submodular memory compression method PACMS, which we attribute to our approximation algorithm with stronger theoretical guarantees, in contrast to the CELF(Leskovec et al., 2007) procedure adopted by PACMS. As expected, SOMC_s exhibits some performance degradation relative to its static counterpart. This reflects the inherent cost of causality in streaming setting: a streaming algorithm must decide which memories to retain before observing

future interactions or queries, whereas static and LLM-based methods can revise or construct their memory representations using more complete information. Category performance comparisons between SOMC and SOTA methods on the $LoCoMo_{10}$ and $LongMemEval_S$ datasets are provided in Appendix F.1.

Table 1: Performance comparison across two datasets using different LLMs. "–" indicates an out-of-memory (OOM) error. The best results are shown in **bold**.

Method		$LoCoMo_{10}$			$LongMemEval_S$		
		↑LLM-Judge	↑F1	↑BLEU	↑LLM-Judge	↑F1	↑BLEU
Qwen-3.5 9B	Full Context	71.5	52.2	20.9	-	-	-
	Naive RAG	32.6	27.9	12.7	21.8	19.1	7.5
	PACMS	52.9	30.8	14.6	39.2	28.7	8.8
	MemoryOS	59.3	38.1	14.9	47.8	32.2	9.3
	SeCom	70.1	49.6	15.7	57.0	44.6	13.8
	LightMem	72.8	50.5	15.9	58.0	43.5	13.1
	NEMORI	73.5	50.7	16.2	60.6	44.9	13.6
	SOMC_s	73.1	51.8	18.1	56.2	41.3	11.9
	SOMC_a	**74.6**	**52.3**	**18.7**	**61.2**	**46.4**	**14.3**
GPT-5	Full Context	74.4	52.5	21.2	70.0	51.5	16.3
	Naive RAG	38.2	25.4	16.1	24.4	18.0	10.2
	PACMS	53.5	31.5	14.7	42.6	30.4	11.9
	MemoryOS	59.4	37.3	15.5	58.0	36.1	13.4
	SeCom	72.3	51.8	18.6	67.8	51.0	14.9
	LightMem	77.0	54.3	19.1	70.1	52.4	15.6
	NEMORI	77.2	55.1	18.9	**71.2**	**53.7**	**16.3**
	SOMC_s	73.7	52.9	19.9	68.0	46.9	14.0
	SOMC_a	**77.5**	**56.3**	**21.8**	70.6	51.8	16.1

5.3 Efficiency Analysis

We consider three efficiency metrics. **Token-Used** measures the average number of tokens consumed by additional external LLM calls required during memory compression, while **Prompt Tokens** measures the average number of tokens output by the compression algorithm and ultimately passed to the target LLM. We also report the average runtime of the memory compression algorithm **Running Time**. The results are summarized in Table 2. SOMC completely eliminates the need for additional LLM calls, leading to substantial savings in external LLM invocation costs compared with existing approaches. Compared with PACMS, which also does not require additional LLM calls, SOMC achieves better compression performance while maintaining the same model-free property. Although SOMC_s has higher per-run latency than SOMC_a, this comparison does not fully reflect its advantage in streaming scenarios: SOMC_a requires recomputation when new interactions arrive, whereas SOMC_s continuously maintains the memory state through online updates. Therefore, SOMC achieves an effective trade-off between performance and computational efficiency, providing efficient long-term memory management under both static and streaming settings.

Table 2: Efficiency comparison on two datasets.

Method	$LoCoMo_{10}$			$LongMemEval_S$		
	Token-Used	Prompt Tokens	Running Time (s)	Token-Used	Prompt Tokens	Running Time (s)
PACMS	0	3827.8	7.8	0	3984.0	11.3
LightMem	106217.4	4236.9	946.9	95356.5	4051.2	755.7
NEMORI	278617.2	3258.1	1009.5	640364.9	3786.5	1431.6
SOMC_s	0	3669.1	29.6	0	3807.6	63.4
SOMC_a	0	3512.5	25.8	0	3176.9	44.9

6 Conclusion

We presented SOMC, a submodular optimization framework for efficient long-term memory compression in LLM agents under both full-access and streaming settings. By formulating memory selection as monotone submodular maximization under a token budget, SOMC avoids repeated generative summarization while providing principled approximation guarantees through Static-SMK and Streaming-SMK. Experiments on long-term memory benchmarks show that SOMC can preserve or improve downstream performance under substantial compression while reducing token processing and computational cost. These results suggest that submodular optimization offers an efficient, model-agnostic foundation for scalable long-term memory management.

AI USE STATEMENT

Generative AI tools were used to polish the writing of the manuscript and to provide limited assistance with software implementation. All AI-assisted text and code were reviewed, edited, and verified by the authors.

REFERENCES

Saad Alqithami. Forgetful but faithful: A cognitive memory architecture and benchmark for privacy-aware generative agents. *arXiv preprint arXiv:2512.12856*, 2025.

Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: Massive data summarization on the fly. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, pp. 671–680, 2014.

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.

Shuang Cui, Kai Han, and Jing Tang. Linear-time algorithms for representative subset selection from data streams. In *Proceedings of the ACM on Web Conference 2025 (WWW)*, pp. 4710–4721, 2025.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, et al. Lightmem: Lightweight and efficient memory-augmented generation. In *International Conference on Learning Representations (ICLR)*, volume 2026, pp. 98706–98729, 2026.

Manu Ghulyani, Arunabh Singh, Karan Bharadwaj, Ankit Nath, and Suranjan Goswami. Pacms: Submodular context selection as a pluggable engine for llm agents. *arXiv preprint arXiv:2606.20047*, 2026.

Kai Han, Shuang Cui, Tianshuai Zhu, Enpei Zhang, Benwei Wu, Zhizhuo Yin, Tong Xu, Shaojie Tang, and He Huang. Approximation algorithms for submodular data summarization with a knapsack constraint. *Proceedings of the ACM on Measurement and Analysis of Computing Systems (POMACS)*, 5(1):1–31, 2021.

Chien-Chung Huang, Naonori Kakimura, and Yuichi Yoshida. Streaming algorithms for maximizing monotone submodular functions under a knapsack constraint. *Algorithmica*, 82(4):1006–1032, 2020.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory os of ai agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 25972–25981, 2025.

Ehsan Kazemi, Marko Mitrovic, Morteza Zadimoghaddam, Silvio Lattanzi, and Amin Karbasi. Submodular streaming in all its glory: Tight approximation, minimum memory and low adaptive complexity. In *International Conference on Machine Learning (ICML)*, pp. 3311–3320, 2019.

Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, pp. 420–429, 2007.

Wenxin Li, Moran Feldman, Ehsan Kazemi, and Amin Karbasi. Submodular maximization in clean linear time. *Advances in neural information processing systems (NeurIPS)*, 35:17473–17487, 2022.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies (ACL)*, pp. 510–520, 2011.

Wenquan Ma, Jiayan Nan, and Wenlong Wu. What deserves memory: Adaptive memory distillation for llm agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (ACL)*, pp. 34789–34812, 2026.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (ACL)*, pp. 13851–13870, 2024.

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

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.

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.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Xufang Luo, Hao Cheng, Dongsheng Li, Yuqing Yang, Chin-Yew Lin, H Vicky Zhao, Lili Qiu, et al. Secom: On memory construction and retrieval for personalized conversational agents. In *International Conference on Learning Representations (ICLR)*, volume 2025, pp. 91851–91885, 2025.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics (ACL)*, pp. 311–318, 2002.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology (UIST)*, pp. 1–22, 2023.

Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

Qingyue Wang, Yanhe Fu, Yanan Cao, Shuai Wang, Zhiliang Tian, and Liang Ding. Recursively summarizing enables long-term dialogue memory in large language models. *Neurocomputing*, 639:130193, 2025.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. *Advances in neural information processing systems (NeurIPS)*, 36:74530–74543, 2023.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*, 2024.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems (NeurIPS)*, 38: 17577–17604, 2026.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence (AAAI)*, volume 38, pp. 19724–19731, 2024.

APPENDIX

A SUBMODULARITY OF THE MEMORY OBJECTIVES

This appendix verifies the structural assumptions used by both optimization algorithms. All embeddings, feature vectors, reference sets, relevance scores, and non-negative coefficients are fixed within a single optimization instance.

Proposition 1. *Both $f_{\mathrm{stat}}(\cdot; q)$ in Eq. equation 2 and f_{str} in Eq. equation 3 are normalized, non-negative, monotone, and submodular.*

Proof. For facility-location coverage, write $a_{rx} = [\mathbf{e}_r^\top \mathbf{e}_x]_+ \geq 0$ and $m_r(S) = \max_{x \in S} a_{rx}$, with $m_r(\emptyset) = 0$. The marginal contribution of adding u to the term indexed by r is $\max\{0, a_{ru} - m_r(S)\}$, which is non-negative and cannot increase as S grows. Hence f_{cov} is normalized, monotone, and submodular.

For the diversity term, let E_S be the matrix whose columns are the embeddings in S. Since $K_S = E_S^\top E_S$, Sylvester's determinant identity gives

$$f_{\mathrm{div}}(S) = \log\det\left(I_d + \sum_{x \in S} \mathbf{e}_x \mathbf{e}_x^\top\right).$$

Let $D_S = I_d + \sum_{x \in S} \mathbf{e}_x \mathbf{e}_x^\top$. The matrix determinant lemma yields

$$f_{\mathrm{div}}(u \mid S) = \log\bigl(1 + \mathbf{e}_u^\top D_S^{-1} \mathbf{e}_u\bigr) \geq 0.$$

If $S \subseteq T$, then $D_T \succeq D_S$ and $D_T^{-1} \preceq D_S^{-1}$, so the marginal gain cannot increase. Thus f_{div} is normalized, monotone, and submodular. The relevance term $f_{\mathrm{rel}}(S; q) = \sum_{x_i \in S} r(x_i, q)$ is non-negative modular.

For feature saturation, fix a coordinate j and set $w_i = z_{ij} \geq 0$. The function

$$h_j(S) = \frac{1}{\tau_{\mathrm{sat}}} \log\left(1 + \tau_{\mathrm{sat}} \sum_{x_i \in S} w_i\right)$$

is a non-decreasing concave function applied to a non-negative modular sum. Its marginal gain

$$h_j(u \mid S) = \frac{1}{\tau_{\mathrm{sat}}} \log\left(1 + \frac{\tau_{\mathrm{sat}} w_u}{1 + \tau_{\mathrm{sat}} \sum_{x_i \in S} w_i}\right)$$

is non-negative and decreases as S grows. Therefore f_{sat} is normalized, monotone, and submodular. Non-negative linear combinations preserve these properties. □

B ANALYSIS OF STATIC-SMK

We keep the notation of the main text. After discarding individually infeasible chunks with $c(x_i) > B$, let $\mathcal{X} = \{x_1, \ldots, x_n\}$ denote the remaining candidate set, let $c(S) = \sum_{x \in S} c(x)$, and let OPT denote an optimal feasible solution of Eq. equation 1.

B.1 AUXILIARY PROCEDURES

For an ordered list $A = (u_1, \ldots, u_\ell)$, $\mathrm{Pref}(A, b)$ denotes the longest prefix whose total cost is at most b.

Lemma 1. CALIBRATE *uses $O(n)$ value-oracle evaluations and returns Γ satisfying $\Gamma \leq f(\mathrm{OPT}) \leq 8\Gamma$.*

Proof. Let R be the final set. For every $u \in \mathrm{OPT} \setminus R$, let R_u be the set present when u was examined. Since u was rejected,

$$f(u \mid R_u) < \frac{c(u)}{B} f(R_u) \leq \frac{c(u)}{B} f(R).$$

Algorithm 3 Calibrate

Require: $\mathcal{X}, f, c, B$
$R \leftarrow \emptyset$
for each $u \in \mathcal{X}$ **do**
  if $f(u \mid R) \geq c(u)f(R)/B$ **then**
    $R \leftarrow R \cup \{u\}$
  end if
end for
return $f(R)/4$

Algorithm 4 DensitySweep

Require: $\mathcal{X}, f, c, B$, estimate $\Gamma > 0$, $\epsilon \in (0, 1/3)$, scale $\alpha \geq 1$
$S \leftarrow \emptyset$; $A \leftarrow [\,]$; $\tau \leftarrow 8\alpha\Gamma/B$
while $\tau > (1-\epsilon)\Gamma/(eB)$ **do**
  for each $u \in \mathcal{X} \setminus S$ **do**
    if $c(S) + c(u) \leq B$ and $f(u \mid S) \geq \tau c(u)$ **then**
      $S \leftarrow S \cup \{u\}$; append u to A
    end if
  end for
  $\tau \leftarrow (1-\epsilon)\tau$
end while
return A

Submodularity gives $f(u \mid R) \leq f(u \mid R_u)$, and therefore

$$f(\mathrm{OPT}) \leq f(\mathrm{OPT} \cup R) \leq f(R) + \sum_{u \in \mathrm{OPT} \setminus R} f(u \mid R) < f(R) + \frac{f(R)}{B} c(\mathrm{OPT}) \leq 2f(R) = 8\Gamma.$$

For the reverse inequality, if $c(R) < B$, then R is feasible and $\Gamma = f(R)/4 \leq f(\mathrm{OPT})$. Otherwise, order the accepted elements by insertion time and let Q be the shortest suffix whose total cost is at least B; write $P = R \setminus Q$. Every accepted $u \in Q$ was inserted when the current set contained P, so its marginal gain is at least $c(u)f(P)/B$. Summing over Q gives $f(Q \mid P) \geq f(P)$ and hence $f(R) \geq 2f(P)$. Subadditivity yields $f(Q) \geq f(R) - f(P) \geq f(R)/2$. Let q be the first element of Q. By minimality of the suffix, both $\{q\}$ and $Q \setminus \{q\}$ are feasible, and submodularity gives $f(\{q\}) + f(Q \setminus \{q\}) \geq f(Q)$. Thus one of these feasible sets has value at least $f(R)/4 = \Gamma$, proving $f(\mathrm{OPT}) \geq \Gamma$. □

Let $A = (u_1, \ldots, u_\ell)$ be the acceptance order of DENSITYSWEEP and $S_h = \{u_1, \ldots, u_h\}$.

Lemma 2. *For every nonempty $T \subseteq \mathcal{X}$ and every $0 \leq h < \ell$, if $\max_{u \in T} c(u) \leq B - c(S_h)$, then*

$$\frac{f(u_{h+1} \mid S_h)}{c(u_{h+1})} \geq \min\left\{(1-\epsilon)\frac{f(T \mid S_h)}{c(T)}, \frac{\alpha f(\mathrm{OPT})}{B}\right\}.$$

Proof. Let τ be the threshold at which u_{h+1} is accepted. At the first threshold, the claim follows from $8\Gamma \geq f(\mathrm{OPT})$. Otherwise, the previous sweep used $\tau/(1-\epsilon)$. Every $v \in T \setminus S_h$ was feasible in that previous sweep and was not accepted. If S' denotes the set at that earlier scan, then $f(v \mid S')/c(v) < \tau/(1-\epsilon)$. Since $S' \subseteq S_h$, submodularity gives $f(v \mid S_h) \leq f(v \mid S')$. Hence

$$f(T \mid S_h) \leq \sum_{v \in T \setminus S_h} f(v \mid S_h) < \frac{\tau}{1-\epsilon} c(T),$$

so $\tau \geq (1-\epsilon)f(T \mid S_h)/c(T)$. The accepted element has density at least τ. □

Lemma 3. *Let S_ℓ be the final sweep solution. For every $T \subseteq \mathcal{X}$ satisfying $\max_{u \in T} c(u) \leq B - c(S_\ell)$,*

$$f(S_\ell) \geq f(T) - \frac{c(T)}{eB} f(\mathrm{OPT}).$$

In particular, if every element of OPT *still fits individually, then $f(S_\ell) \geq (1 - 1/e)f(\mathrm{OPT})$.*

Proof. Let τ_{last} be the last threshold scanned. The stopping rule gives $\tau_{\text{last}} \leq \Gamma/(eB) \leq f(\text{OPT})/(eB)$. Every $u \in T \setminus S_\ell$ remained feasible during its last scan but was not accepted, so $f(u \mid S_\ell) \leq \tau_{\text{last}} c(u)$. Summing these marginals gives $f(T \mid S_\ell) \leq c(T) f(\text{OPT})/(eB)$. Monotonicity implies $f(T) \leq f(S_\ell) + f(T \mid S_\ell)$, proving the claim. □

B.2 Approximation Analysis

Lemma 4. *Assume $c(u) \leq \beta B$ for every $u \in \mathcal{X}$ and $\alpha \geq 1$. The final sweep solution satisfies*

$$f(S_\ell) \geq \left(1 - e^{-(1-\epsilon)(1-\beta)}\right) f(\text{OPT}).$$

Proof. If $c(S_\ell) \leq (1-\beta)B$, every element of OPT still fits individually, so Lemma 3 gives $(1 - 1/e) f(\text{OPT})$, which is at least the displayed factor. Otherwise, let k be the first index with $c(S_k) > (1-\beta)B$. For every $h < k$, all elements of OPT fit into S_h, so Lemma 2 with $T = \text{OPT}$ gives

$$f(u_{h+1} \mid S_h) \geq \frac{(1-\epsilon) c(u_{h+1})}{B} \big(f(\text{OPT}) - f(S_h)\big).$$

Writing $R_h = f(\text{OPT}) - f(S_h)$ and using $1 - x \leq e^{-x}$ yields

$$R_k \leq \exp\left(-\frac{(1-\epsilon) c(S_k)}{B}\right) f(\text{OPT}) < e^{-(1-\epsilon)(1-\beta)} f(\text{OPT}).$$

Monotonicity gives the same bound for S_ℓ. □

Lemma 5. *For every $\epsilon \in (0, 1/3)$, $\exp(-(1-\epsilon)^2/\epsilon) \leq 2\epsilon$.*

Proof. Let $x = 1/(2\epsilon) > 1$. Then $(1-\epsilon)^2/\epsilon = 1/\epsilon - 2 + \epsilon \geq 2(x-1) \geq \ln x$. Exponentiating the negative of both sides gives the result. □

Lemma 6. *The robust branch of* Static-SMK *returns a set of value at least $(1/2 - \epsilon) f(\text{OPT})$ for arbitrary item sizes $0 < c(u) \leq B$.*

Proof. Let $A = (u_1, \ldots, u_\ell)$ be the acceptance order of the sweep and $S_h = \{u_1, \ldots, u_h\}$. If any checkpoint candidate has value at least $f(\text{OPT})/2$, the claim is immediate. Hence assume every checkpoint candidate has value below $f(\text{OPT})/2$. Let $r \in \text{OPT}$ be an element of maximum cost. If $\text{OPT} = \{r\}$, the empty checkpoint returns the best feasible singleton and is optimal. Otherwise set $T = \text{OPT} \setminus \{r\}$.

First suppose $c(r) \geq (1-\epsilon)B$. The empty checkpoint implies $f(\{r\}) < f(\text{OPT})/2$, so subadditivity gives $f(T) > f(\text{OPT})/2$ and $c(T) \leq \epsilon B$. If $c(S_\ell) \leq (1-\epsilon)B$, every element of T fits individually and Lemma 3 gives $f(S_\ell) > (1/2 - \epsilon) f(\text{OPT})$. Otherwise, let q be the first index with $c(S_q) \geq (1-\epsilon)B$. For every $h < q$, Lemma 2 with T and $\alpha = 1/\epsilon$ yields

$$f(u_{h+1} \mid S_h) \geq \frac{(1-\epsilon) c(u_{h+1})}{\epsilon B} \big(f(T) - f(S_h)\big).$$

If the multiplicative coefficient in the resulting gap recurrence becomes negative, then the value has already crossed $f(T)$ and the claim follows. Otherwise, multiplying the recurrence until q and using Lemma 5 gives $f(S_\ell) \geq (1 - 2\epsilon) f(T) > (1/2 - \epsilon) f(\text{OPT})$.

Now suppose $c(r) < (1-\epsilon)B$. Let $b_j = \epsilon B (1+\epsilon)^j$, and let i_r be the largest nonnegative integer with $b_{i_r} \leq B - c(r)$. Set $P = \text{Pref}(A, b_{i_r}) = S_{h_r}$. The element r is feasible as a one-element augmentation of P. Let z be the best feasible augmentation at this checkpoint. Under the standing assumption, $f(P \cup \{r\}) \leq f(P \cup \{z\}) < f(\text{OPT})/2$. If $h_r = \ell$, then every optimal element fits individually into S_ℓ and Lemma 3 gives more than $(1/2 - \epsilon) f(\text{OPT})$. Otherwise, for every $h \leq h_r$, all elements of T fit individually into S_h. Lemma 2 and submodularity imply

$$f(u_{h+1} \mid S_h) \geq \eta\, c(u_{h+1}), \qquad \eta := \frac{1-\epsilon}{B - c(r)} \left(f(\text{OPT}) - f(P \cup \{r\})\right).$$

Summing to $h_r + 1$ gives $f(S_{h_r+1}) \geq \eta c(S_{h_r+1})$. By maximality of i_r, $c(S_{h_r+1}) > b_{i_r} > (B - c(r))/(1+\epsilon)$. Therefore

$$f(S_\ell) > \frac{1-\epsilon}{1+\epsilon}\left(f(\mathrm{OPT}) - f(P \cup \{r\})\right) > \frac{1-\epsilon}{2(1+\epsilon)} f(\mathrm{OPT}) \geq (1/2 - \epsilon) f(\mathrm{OPT}).$$

This completes both cases. □

Proof of Theorem 1. If $\beta \leq \beta^\star_\epsilon$, STATIC-SMK uses the fast branch with $\alpha = 1$. Lemma 4 gives the granularity-dependent factor, and the definition of $\beta^\star_\epsilon$ makes it at least $1/2 - \epsilon$. If $\beta > \beta^\star_\epsilon$, the robust branch is used. Lemma 6 gives the $1/2 - \epsilon$ floor, while the full sweep solution remains among the candidates and Lemma 4 gives the granularity-dependent factor. Returning the best candidate yields their maximum.

CALIBRATE is linear. DENSITYSWEEP performs $O(\epsilon^{-1}(1 + \log \alpha))$ scans of the ground set. Thus the fine-grained branch with $\alpha = 1$ uses $O(n/\epsilon)$ oracle evaluations, whereas the robust sweep with $\alpha = 1/\epsilon$ uses $O(n\epsilon^{-1}\log(1/\epsilon))$. The logarithmic family of one-element augmentations has the same asymptotic cost. □

C ANALYSIS OF STREAMING-SMK

We keep the notation of the main text. For every prefix $\mathcal{X}_t = (x_1, \ldots, x_t)$, let OPT_t denote an optimal feasible subset under budget B_s. The chunking policy supplies an ex-ante cap $c(x_i) \leq \beta_s B_s$.

C.1 ONLINE VALUE ESTIMATION

Streaming-SMK uses the exact OneStream procedure of Cui et al. (2025) as a fixed-accuracy online value estimator. Algorithm 5 reproduces their Algorithm 1, with notation changed only to match our budget B_s and stream elements. No update rule is modified.

Algorithm 5 OneStream($\hbar$) (Cui et al., 2025)

Require: integer $\hbar \geq 1$
Initialize $i \leftarrow 1$, $j \leftarrow 1$, $S_i \leftarrow \emptyset$, and $e^\star \leftarrow \mathrm{null}$
Take one pass over the stream
while there is an incoming feasible element x **do**
  $W \leftarrow \bigcup_{r=j}^{i} S_r$
  if $f(x \mid W)/c(x) \geq f(W)/B_s$ **then**
    $S_i \leftarrow S_i \cup \{x\}$
    if $c(S_i) \geq B_s$ **then**
      if $i - j + 1 = 2\hbar$ **then**
        Delete $S_j, S_{j+1}, \ldots, S_{j+\hbar-1}$; $j \leftarrow j + \hbar$
      end if
      $i \leftarrow i + 1$; $S_i \leftarrow \emptyset$
    end if
  end if
  $e^\star \leftarrow \arg\max_{u \in \{e^\star, x\}} f(\{u\})$
end while
$i_n \leftarrow i$; $j_n \leftarrow j$; $W_n \leftarrow \bigcup_{r=j_n}^{i_n} S_r$
if $c(W_n) \leq B_s$ **then**
  $Q^\star \leftarrow W_n$
else
  Let $S(z)$ be the set of the last z elements added to W_n; find z such that $c(S(z)) \leq B_s$ and $c(S(z+1)) > B_s$
  $Q^\star \leftarrow \arg\max_{Q \in \{S(z), \{e^\star\}\}} f(Q)$
end if
return $Q^\star, W_n$

To obtain an estimate at every prefix without altering OneStream, we use the following wrapper. After processing x_t, it applies the terminal extraction rule to the current OneStream state and additionally compares the resulting feasible candidate with the best singleton $e^\star$ seen so far. The persistent OneStream state is left unchanged.

Algorithm 6 PrefixOutput for OneStream

Require: Current OneStream state $(i, j, S_j, \ldots, S_i, e^\star)$
$i_t \leftarrow i$; $j_t \leftarrow j$; $W_t \leftarrow \bigcup_{r=j_t}^{i_t} S_r$
if $c(W_t) \le B_s$ **then**
  $T_t \leftarrow W_t$
else
  Let $S_t(z)$ be the set of the last z elements added to W_t; find z such that $c(S_t(z)) \le B_s$ and $c(S_t(z+1)) > B_s$
  $T_t \leftarrow S_t(z)$
end if
$R_t \leftarrow \arg\max_{Q \in \{T_t, \{e^\star\}\}} f(Q)$
return R_t

Let $\bar{R}_t$ denote the solution returned by the original terminal extraction rule of OneStream after processing prefix $\mathcal{X}_t$. Since Algorithm 6 does not modify the persistent OneStream state, this state is exactly the state obtained by running the original OneStream algorithm on $\mathcal{X}_t$ and stopping there. Moreover, by construction, $f(R_t) \ge f(\bar{R}_t)$. Theorem 4.6 of Cui et al. (2025) states that OneStream with $\hbar = \log_2(1/(8\zeta)) + 1$ returns a $(1/8 - \zeta)$-approximation in one pass with linear query complexity. Setting $\zeta = 1/16$ gives $\hbar = 2$ and approximation factor $1/16$. Therefore, for every prefix,

$$f(R_t) \ge f(\bar{R}_t) \ge \frac{1}{16} f(\mathrm{OPT}_t). \tag{7}$$

Define $H_t = \max_{s \le t} f(R_s)$ and set $C_0 = 16$. Then H_t is nondecreasing and

$$f(\mathrm{OPT}_t) \le C_0 H_t \qquad \text{for every } t. \tag{8}$$

The original OneStream state uses $O(n)$ oracle queries over n arrivals. Evaluating the prefix output adds at most constant oracle work per arrival, so the online estimation remains $O(n)$.

C.2 Approximation Analysis

Fix an active value guess v. Every element inserted into S_v has marginal density at least $\theta v / B_s$, so at every time

$$f(S_v) \ge \frac{\theta v}{B_s} c(S_v). \tag{9}$$

Lemma 7. *If an arriving element passes the density test for an active v but cannot be inserted because it would violate the budget, then the best feasible solution retained by the algorithm has value at least $\theta \kappa_{\beta_s} v$, where $\kappa_{\beta_s} = \max\{1/2, 1 - \beta_s\}$.*

Proof. Let S^- be the state immediately before such an element u. Since $c(S^-) + c(u) > B_s$ and $c(u) \le \beta_s B_s$, Eq. equation 9 gives $f(S^-) \ge \theta(1 - \beta_s)v$. Independently, the density test gives

$$f(S^-) + f(u \mid S^-) \ge \frac{\theta v}{B_s} \big(c(S^-) + c(u)\big) > \theta v.$$

Submodularity gives $f(\{u\}) \ge f(u \mid S^-)$, so the better of S^- and $\{u\}$ has value at least $\theta v/2$. Taking the better of this bound and the granularity bound gives $\theta \kappa_{\beta_s} v$. □

Fix a finite stopping prefix $\mathcal{X}_t$ and let $v^\star$ be the largest point of the $(1+\epsilon)$ grid not exceeding $f(\mathrm{OPT}_t)$. Then

$$v^\star \le f(\mathrm{OPT}_t) < (1+\epsilon) v^\star. \tag{10}$$

Lemma 8. *Either the maintained output already has value at least $\gamma v^\star$, or the $v^\star$ instance is activated and is not subsequently pruned before time t. If its first activation occurs at time t_0, then $f(\mathrm{OPT}_{t_0-1}) < \epsilon v^\star$.*

Proof. If $v^\star < LB_s/\gamma$ at any time $s \le t$, then the current output already has value greater than $\gamma v^\star$. Otherwise the lower cutoff never removes $v^\star$. Equation equation 8 gives $C_0 H_s \ge f(\mathrm{OPT}_s)$, so the upper endpoint $C_0 H_s/\epsilon$ eventually reaches $v^\star$. Immediately before the first activation, $v^\star > C_0 H_{t_0-1}/\epsilon$, hence $f(\mathrm{OPT}_{t_0-1}) < \epsilon v^\star$. Newly activated instances process the current element, so only the strict prefix is missed. □

Lemma 9. *Assume the $v^\star$ instance is active and not pruned, and that every later optimal element that passes the density test can be inserted. Its final set S satisfies $f(S) \ge (1-\theta-\epsilon)v^\star$.*

Proof. Partition $\mathrm{OPT}_t \setminus S$ into O_{pre}, arriving before activation, and O_{post}, arriving afterwards. Since O_{pre} is feasible, $f(O_{\mathrm{pre}} \mid S) \le f(O_{\mathrm{pre}}) \le f(\mathrm{OPT}_{t_0-1}) < \epsilon v^\star$. Every $u \in O_{\mathrm{post}} \setminus S$ must have failed the density test at arrival; submodularity can only decrease its later marginal. Thus $f(u \mid S) < (\theta v^\star/B_s)c(u)$. Therefore

$$f(\mathrm{OPT}_t) \le f(\mathrm{OPT}_t \cup S) \le f(S) + \epsilon v^\star + \frac{\theta v^\star}{B_s} c(O_{\mathrm{post}}) \le f(S) + (\epsilon+\theta)v^\star.$$

Since $f(\mathrm{OPT}_t) \ge v^\star$, rearranging proves the claim. □

Proof of Theorem 2. If lower-bound pruning has already produced value at least $\gamma v^\star$, the approximation follows. Otherwise Lemma 8 activates and preserves the relevant grid value. If a density-qualified element cannot be inserted because of capacity, Lemma 7 gives value at least $\theta\kappa_{\beta_s} v^\star$. If no such event occurs, Lemma 9 gives $(1-\theta-\epsilon)v^\star$. With $\theta = (1-\epsilon)/(1+\kappa_{\beta_s})$,

$$\theta\kappa_{\beta_s} = 1-\theta-\epsilon = \frac{(1-\epsilon)\kappa_{\beta_s}}{1+\kappa_{\beta_s}}.$$

Combining this with Eq. equation 10 yields

$$f(Q_t) \ge \frac{1-\epsilon}{1+\epsilon}\frac{\kappa_{\beta_s}}{1+\kappa_{\beta_s}} f(\mathrm{OPT}_t).$$

Since $\kappa_{\beta_s}/(1+\kappa_{\beta_s}) \le 1/2$, the multiplicative loss is at most an additive ϵ, giving Eq. equation 6. □

C.3 Complexity

Lemma 10. *At every time with $H_t > 0$, the number of active value guesses is $O(\epsilon^{-1}\log(1/\epsilon))$.*

Proof. Because the best prefix OneStream output is also considered by the algorithm, $LB_t \ge H_t$. Hence, the ratio of the upper and lower endpoints of the active interval is at most

$$\frac{C_0 H_t/\epsilon}{LB_t/\gamma} \le \frac{C_0\gamma}{\epsilon} \le \frac{16}{\epsilon}.$$

A multiplicative $(1+\epsilon)$ grid therefore contains $O(\epsilon^{-1}\log(1/\epsilon))$ points in the interval. □

OneStream together with the prefix wrapper contributes $O(n)$ queries. Each arrival is then evaluated once against every active density instance, so Lemma 10 gives $O((n/\epsilon)\log(1/\epsilon))$ total value-oracle evaluations. For storage, Eq. equation 9 and $f(S_v) \le LB_t$ imply $c(S_v) \le B_s LB_t/(\theta v)$. After pruning, active guesses satisfy $v \ge LB_t/\gamma$ and increase geometrically. Therefore

$$\sum_v c(S_v) \le \frac{B_s\gamma}{\theta}\frac{1+\epsilon}{\epsilon} = O(B_s/\epsilon).$$

If token costs satisfy $c(u) \ge 1$, the same expression bounds the number of stored elements. With $\hbar = 2$, the fixed-accuracy OneStream state stores $O(k)$ elements, where k is the maximum cardinality of a feasible set; under $c(u) \ge 1$, $k \le B_s$, so this additional storage is dominated by $O(B_s/\epsilon)$.

D THE PSEUDOCODE OF SOMC

In this section, we will respectively introduce the algorithm Pseudo code of SOMC in static and streaming scenarios. For the static setting, SOMC_a first partitions the complete history into bounded-size memory chunks and applies STATIC-SMK with the query-independent coverage–diversity objective to construct a compact memory bank $\mathcal{M}$. Once the query arrives, STATIC-SMK is applied again with the query-aware objective, which additionally incorporates relevance, to select the final context S_q under budget B_q. The selected context is then provided to the LLM for answer generation Algorithm 7.

Algorithm 7 SOMC_a

Require: History $\mathcal{H} = (d_1, \ldots, d_T)$; query q; memory budget B_m; query budget B_q; chunk-size cap B_c; $\epsilon \in (0, 1/3)$
Ensure: Answer $\mathcal{A}$
$\mathcal{X} \leftarrow \mathrm{Chunk}(\mathcal{H}, B_c)$, where $\mathcal{X} = (x_1, \ldots, x_n)$ and $c_i \le B_c$
$f_{\mathrm{bank}} \leftarrow \lambda_{\mathrm{cov}} f_{\mathrm{cov}} + \lambda_{\mathrm{div}} f_{\mathrm{div}}$
$\mathcal{M} \leftarrow$ STATIC-SMK $(\mathcal{X}, B_m, \epsilon, f_{\mathrm{bank}})$
$f_{\mathrm{stat}}(\cdot; q) \leftarrow \lambda_{\mathrm{cov}} f_{\mathrm{cov}} + \lambda_{\mathrm{div}} f_{\mathrm{div}} + \lambda_{\mathrm{rel}} f_{\mathrm{rel}}(\cdot; q)$
$S_q \leftarrow$ STATIC-SMK $(\mathcal{M}, B_q, \epsilon, f_{\mathrm{stat}}(\cdot; q))$
$\mathcal{A} \leftarrow \mathrm{LLM}(S_q, q)$
return $\mathcal{A}$

For the streaming setting, SOMC_s processes memory chunks in causal arrival order using STREAMING-SMK, maintaining a bounded query-independent memory state without revisiting previously discarded chunks. After the query becomes available, the retained memory $\mathcal{M}$ is passed to STATIC-SMK for query-aware compression, producing the final context S_q for downstream generation Algorithm 8. This two-stage design separates continual memory maintenance from query-specific context construction.

Algorithm 8 SOMC_s

Require: Memory stream $\mathcal{H} = (d_1, \ldots, d_T)$; streaming budget B_s; query budget B_q; chunk-size cap B_c; granularity cap β_s; $\epsilon \in (0, 1/3)$
Ensure: Answer $\mathcal{A}$
Initialize the state of STREAMING-SMK
for each arriving memory d_t **do**
  $\mathcal{X}_t \leftarrow \mathrm{Chunk}(d_t, B_c)$
  for each chunk $x_i \in \mathcal{X}_t$ **do**
    Update STREAMING-SMK with $(x_i, B_s, \beta_s, \epsilon, f_{\mathrm{str}})$
  end for
end for
$\mathcal{M} \leftarrow \mathrm{Extract}$(STREAMING-SMK state)
Upon receiving query q
$S_q \leftarrow$ STATIC-SMK$(\mathcal{M}, B_q, \epsilon, f_{\mathrm{stat}}(\cdot; q))$
$\mathcal{A} \leftarrow \mathrm{LLM}(S_q, q)$
return $\mathcal{A}$

E PARAMETER SETTING DETAILS

For all settings, we set the chunk budget to $B_c = 256$ tokens. All text embeddings are computed using e5-base-v2 Wang et al. (2022), and the local Qwen-3.5 9B model is used with its default inference parameters. For the static setting, $\beta$is determined automatically by the algorithm, and the static accuracy parameter is set to $\epsilon = 0.1$. For the streaming setting, we set the ex-ante granularity cap $\beta_s = 1/32$ and $\epsilon = 0.3$ for streaming memory compression, while using the static accuracy parameter $\epsilon = 0.1$ for prompt extraction. For $LoCoMo_{10}$, we set the streaming memory budget to $B_s = 20{,}000$ tokens, whereas for $LongMemEval_S$, we set $B_s = 80{,}000$ tokens. For static setting, we set $B_m = 20{,}000$ tokens and $B_m = 80{,}000$ tokens, respectively. Owing to the design of the

submodular approximation algorithm, the compressed memory state typically does not fully exhaust the prescribed budget. We further set $\tau_{\text{sat}} = 1$. The weights $\text{div}, \text{cov}, and\text{rel}$ are selected through grid search with a step size of 0.1. All experiments are conducted on a Linux server equipped with an Intel Xeon W-5317 CPU, 256 GB of system memory, and an NVIDIA A100 GPU.

F Additional Experimental Results and Analysis

F.1 Analysis of Category

Table 3: Category-level performance comparison between SOMC and state-of-the-art methods on the $LoCoMo_{10}$ dataset.

Method		Multi-hop			Temporal reasoning			Open-domain knowledge			Single-hop		
		LLM-Judge	F1	BLEU	LLM-Judge	F1	BLEU	LLM-Judge	F1	BLEU	LLM-Judge	F1	BLEU
Qwen-3.5 9B	Full Context	77.6	48.4	12.4	46.1	19.6	4.4	31.2	13.6	3.0	83.8	70.5	32.2
	NEMORI	76.2	44.8	11.1	55.4	51.2	13.1	41.7	22.3	4.8	83.1	55.7	20.4
	SOMC$_s$	71.2	42.3	10.7	71.3	47.6	12.4	42.7	19.5	4.2	77.9	60.4	24.4
	SOMC$_a$	67.4	40.5	9.9	61.1	47.2	12.6	38.5	18.2	4.1	86.3	62.1	25.7
Gpt5	Full Context	74.5	46.4	11.6	43.3	28.3	8.8	35.4	19.7	3.9	90.8	67.5	31.3
	NEMORI	79.1	47.4	12.1	62.9	55.1	14.7	44.8	25.4	5.6	85.7	61.1	24.4
	SOMC$_s$	69.8	40.6	9.5	60.7	58.2	19.8	35.4	23.8	5.4	84.3	58.3	25.2
	SOMC$_a$	70.6	41.9	10.6	70.1	57.1	19.5	46.8	23.0	4.6	86.2	64.7	28.4

We first analyze the category-level performance of SOMC and the LLM-based state-of-the-art method NEMORI on $LoCoMo_{10}$, as reported in Table 3. The results indicate that the main advantage of SOMC lies in its ability to preserve critical original evidence, rather than in generating higher-quality summaries. Temporal reasoning and Single-hop questions typically depend on explicit temporal cues or a small number of localized facts. In such cases, memory selection based on coverage, redundancy reduction, and relevance can preserve precise information more reliably. By contrast, Multi-hop and Open-domain knowledge questions rely more heavily on integrating information across multiple fragments, semantic abstraction, or the incorporation of external knowledge. LLM-based methods can more readily construct higher-level representations through summarization and semantic reorganization, which may be more directly usable for downstream reasoning; accordingly, the advantage of SOMC is more limited in these two categories.

The comparison between the static and streaming variants further suggests a clear interaction between the compression strategy and the capability of the downstream model. For Qwen3.5 9B, the static variant substantially outperforms the streaming variant on Single-hop questions, likely because access to both the complete history and the query enables more precise localization of answer-relevant evidence. In contrast, for categories such as Temporal, Multi-hop, and Open-domain, which often require broader contextual support, the query-independent memory maintained continuously by the streaming variant may preserve more diverse and better-covered contextual information, which can be particularly helpful for a less capable local model. GPT-5, by comparison, has stronger evidence utilization and cross-fragment reasoning capabilities and can therefore more fully exploit the global, query-aware compressed context produced by the static variant. Consequently, static SOMC outperforms its streaming counterpart across all categories on GPT-5. Moreover, because the streaming variant must make irreversible retention decisions without access to future interactions or queries, some information whose value becomes apparent only later may be discarded prematurely.

Table 4: Question-type-level performance comparison between SOMC and state-of-the-art methods on the $LongMemEval_S$ dataset (I).

Method		knowledge-update			multi-session			single-session-assistant		
		LLM-Judge	F1	BLEU	LLM-Judge	F1	BLEU	LLM-Judge	F1	BLEU
Qwen-3.5 9B	Full Context	-	-	-	-	-	-	-	-	-
	NEMORI	74.3	57.9	12.6	46.6	34.3	6.9	91.1	72.5	25.3
	SOMC$_s$	83.3	66.5	16.5	43.6	30.3	6.1	64.3	52.6	20.1
	SOMC$_a$	82.1	67.1	15.8	40.6	29.5	5.9	96.4	79.1	30.3
GPT-5	Full Context	85.9	64.0	17.1	68.4	46.4	10.0	94.6	79.3	31.5
	NEMORI	83.3	64.2	16.4	66.9	47.4	9.6	92.8	76.1	26.0
	SOMC$_s$	89.7	67.7	17.4	57.9	38.6	7.9	92.9	53.9	20.2
	SOMC$_a$	89.7	65.4	17.7	51.1	36.2	7.3	100	79.2	29.8

We further conduct a fine-grained analysis across the six question types in $LongMemEval_S$, with results reported in Tables 4 and 5. The overall trend is consistent with that observed on

Table 5: Question-type-level performance comparison between SOMC and state-of-the-art methods on the $LongMemEval_S$ dataset (II).

Method		single-session-preference			single-session-user			temporal-reasoning		
		LLM-Judge	F1	BLEU	LLM-Judge	F1	BLEU	LLM-Judge	F1	BLEU
Qwen-3.5 9B	Full Context	-	-	-	-	-	-	-	-	-
	NEMORI	36.7	4.6	0.2	90.0	67.4	21.8	43.6	33.7	15.0
	SOMC$_s$	23.3	3.4	0.1	91.4	74.1	25.2	38.4	24.2	7.6
	SOMC$_a$	23.3	3.6	0.1	92.9	74.3	25.0	46.2	32.6	12.9
GPT-5	Full Context	46.7	10.8	0.7	84.3	66.5	22.4	49.6	39.0	16.0
	NEMORI	46.6	8.0	0.3	94.3	74.3	24.6	57.1	43.9	18.2
	SOMC$_s$	50.0	7.5	0.3	90.0	75.3	25.2	47.4	34.2	12.7
	SOMC$_a$	36.7	6.6	0.2	95.7	78.5	27.3	60.9	44.1	16.0

$LoCoMo_{10}$: SOMC is particularly effective at preserving explicit, localized, and low-distortion evidence. Knowledge-update, single-session-assistant, single-session-user, and temporal-reasoning questions typically depend on concrete facts, updated information, or temporal cues. Directly retaining the original memory fragments can reduce information omission, rewriting, and loss of fine-grained details that may arise during LLM-based summarization. As a result, SOMC outperforms LLM-based methods on these question types for both Qwen3.5 9B and GPT-5. In contrast, multi-session questions require the integration of distributed evidence across multiple sessions, while single-session-preference questions often involve inferring implicit preferences and semantic tendencies. Such abstraction and information integration are more aligned with the strengths of LLM-based memory consolidation.

A similar distinction also emerges between the static and streaming variants. For single-session-assistant, single-session-user, and temporal-reasoning questions, the static method can perform global, query-aware selection over the complete candidate memory set after the query arrives, making it easier to identify the evidence most directly relevant to the answer. In contrast, for multi-session, single-session-preference, and knowledge-update questions, relevant information is often more distributed and may involve long-range dependencies, implicit semantics, or the evolution of factual states over time. In these cases, the query-independent memory state continuously maintained by the streaming method may preserve a broader view of the relevant background and temporal evolution. Static selection, by contrast, may become overly concentrated on locally relevant fragments that appear most directly related to the query, potentially overlooking cross-session context, implicit preferences, or chains of updates.

Taken together, the results across both datasets suggest that the primary advantage of SOMC over LLM-based memory compression lies in its precise and low-distortion preservation of original evidence, whereas its relative weakness is more apparent on tasks that require high-level semantic abstraction or cross-fragment information integration. At the same time, static SOMC is better suited to precise, query-specific evidence selection, while streaming SOMC may occasionally benefit tasks involving distributed evidence or weaker downstream models by maintaining a broader and more diverse query-independent memory state. This benefit, however, comes at the cost of irreversible information loss induced by causal constraints. These explanations should be interpreted as mechanism-level hypotheses consistent with the observed empirical patterns rather than as strict causal conclusions.

F.2 Ablation Study

Table 6: Ablation results for the static setting on both datasets using Qwen-3.5 9B.

	$LoCoMo_{10}$			$LongMemEval_S$		
	LLM-Judge	F1	BLEU	LLM-Judge	F1	BLEU
w/o STATIC-SMK Hybrid	72.1	50.2	17.5	59.0	45.5	13.2
w/o STATIC-SMK Rel	70.9	49.6	17.2	58.4	45.1	13.0
w/o STATIC-SMK Cov & Div	73.7	51.1	18.5	59.6	45.3	13.7
w/o STATIC-SMK + Greedy	41.5	32.1	13.7	39.0	29.8	11.1
SOMC$_a$	74.6	52.3	18.7	61.2	46.4	14.3

In this section, we conduct ablation studies of the proposed method on both datasets using Qwen-3.5 9B. For the static setting, we ablate individual components of STATIC-SMK on both datasets, including removing the entity- and temporal-matching enhancements from the BM25 lexical score, removing the query-relevance score, removing the coverage and diversity scores, and replacing STATIC-SMK with a marginal-density Greedy baseline that iteratively selects the feasible chunk with the largest marginal utility per token cost, $f(x|S)/c(x)$, while keeping the same objective function and budget constraint. The results are reported in Table 6. The complete SOMC_a achieves the best overall performance, indicating that the individual components play complementary roles in the final system. In particular, replacing STATIC-SMK with Greedy leads to a substantial performance degradation, suggesting that the gains of SOMC arise not only from the design of the utility function, but also from the proposed budget-aware optimization algorithm. By comparison, removing the coverage and diversity terms results in a smaller degradation than removing the relevance term. A plausible explanation is that, in the static setting, the query is already available, allowing relevance to directly identify evidence associated with the current question, whereas coverage and diversity mainly play auxiliary roles by reducing redundancy and preserving complementary information. Finally, ablating the entity- and temporal-matching enhancements in BM25 further indicates that generic lexical matching alone may be insufficient to reliably capture precise cues such as names, entities, and temporal expressions.

Table 7: Ablation results for the streaming setting on both datasets using Qwen-3.5 9B.

	$LoCoMo_{10}$			$LongMemEval_S$		
	LLM-Judge	F1	BLEU	LLM-Judge	F1	BLEU
w/o STREAMING-SMK Div	71.5	50.0	17.6	54.8	40.1	10.7
w/o STREAMING-SMK Cov	72.6	50.3	17.2	55.6	40.9	11.4
w/o STREAMING-SMK + Greedy	51.5	42.8	15.7	30.8	35.7	8.2
w/o STATIC-SMK Hybrid	70.3	48.7	16.5	53.0	39.5	11.0
w/o STATIC-SMK Rel	69.8	48.4	16.9	54.4	40.1	10.8
w/o STATIC-SMK Cov & Div	70.2	50.3	17.1	55.6	40.5	11.2
w/o STATIC-SMK + Greedy	39.7	30.6	11.8	24.0	28.5	11.4
SOMC_s	73.1	51.8	18.1	56.2	41.3	11.9

For the streaming setting, we further conduct ablation studies on the two-stage compression pipeline. Since STREAMING-SMK is responsible for compressing the memory before the query arrives, while STATIC-SMK subsequently performs query-aware context construction after the query becomes available, we separately examine the contribution of each stage. Specifically, during the memory compression stage, we remove the coverage and diversity scores from STREAMING-SMK and replace STREAMING-SMK with a Greedy algorithm. During the query-aware stage, following the same protocol as in the static setting, we separately remove the entity- and temporal-matching enhancements from the BM25 lexical score, the query-relevance score, and the coverage and diversity scores, and also replace STATIC-SMK with Greedy. The results are reported in Table 7. Overall, the ablation trends are consistent with those observed in the static setting. The complete SOMC_s achieves the best overall performance, indicating that the components involved in both streaming memory maintenance and subsequent query-aware compression play complementary roles. In particular, ablating STREAMING-SMK shows that maintaining a query-independent memory state with sufficient coverage and diversity before the query arrives is important for downstream answer quality. The performance degradation caused by replacing STREAMING-SMK with Greedy further suggests that the gains in the streaming stage arise not only from the objective function itself, but also from the proposed single-pass, budget-aware optimization procedure. Meanwhile, the ablation trends after the query arrives are broadly consistent with those in the static setting: relevance contributes more directly to precise evidence localization, whereas coverage and diversity primarily help reduce redundancy and preserve complementary information.

F.3 HYPERPARAMETER ANALYSIS OF β_s

In this section, we analyze the sensitivity of the streaming algorithm to the hyperparameter β_s. In the streaming setting, β_s specifies an ex-ante granularity cap on the cost of each memory chunk, i.e., $c_i \leq \beta_s B_s$, relative to the total streaming budget. As reflected in Eqs. equation 5 and equation 6, as well as Algorithm 2, a smaller β_s imposes a tighter granularity cap, whereas a larger β_s allows

an individual chunk to occupy a larger fraction of the streaming budget. The choice of β_s directly affects the approximation guarantee of STREAMING-SMK. It also changes the density threshold through κ_{β_s} and θ, and affects the lower cutoff of the active guess interval through γ, thereby regulating the acceptance behavior of memory chunks during streaming.

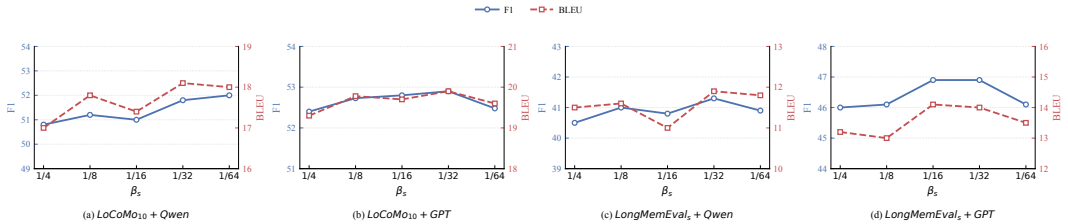

Figure 2: Hyperparameter analysis of β_s.

The results are shown in Figure 2. STREAMING-SMK achieves its best overall empirical performance when $\beta_s = 1/32$, while either increasing or decreasing β_s leads to a slight performance degradation. This observation suggests that β_s induces a practical trade-off in the filtering strength of the streaming selection process. Theoretically, a smaller β_s yields a stronger worst-case approximation guarantee; however, this does not imply that downstream performance should improve monotonically as β_s decreases. When β_s increases, κ_{β_s} decreases and the density parameter $\theta = (1-\epsilon)/(1+\kappa_{\beta_s})$ increases, resulting in a stricter acceptance threshold. Consequently, some memory chunks with moderate marginal utility but potential relevance to downstream queries may be filtered out. In contrast, when β_s becomes too small, the corresponding lower density threshold makes the streaming selector more permissive. Since the selection is performed in one pass without revisiting previously discarded or accepted chunks, candidate solutions may admit moderate-density chunks earlier and consume their budgets more rapidly, potentially preventing later, more useful chunks from being retained. Therefore, $\beta_s = 1/32$ appears to provide a favorable empirical balance between selective filtering and information preservation. This non-monotonic behavior does not contradict the theoretical guarantee, since the latter characterizes the worst-case approximation ratio with respect to the submodular objective rather than downstream task performance on a particular data distribution.

G RELATION BETWEEN SURROGATE UTILITY AND DOWNSTREAM EVALUATION METRICS

Our submodular objective does not directly optimize downstream metrics such as F1, BLEU, or LLM-based evaluation scores. Instead, it serves as a surrogate for the quality of evidence retained under a limited token budget. The coverage term preserves representative information, the diversity term discourages redundant use of the budget, and the query-aware relevance term favors answer-relevant evidence. Together, these components aim to provide the downstream LLM with sufficient, complementary, and less noisy information for generation.

Such a separation between the upstream optimization objective and downstream task metrics is also common in context and memory compression. (Pan et al., 2025) reduces redundancy in memory units to improve retrieval quality, which in turn benefits final response generation. Similarly, (Pan et al., 2024) optimizes a token-level compression objective for faithful information preservation and evaluates its effectiveness through diverse downstream tasks rather than optimizing their task-specific metrics directly. Our setting follows the same intuition: improving the quality and usability of the retained evidence can lead to better reference-overlap metrics such as F1 and BLEU as well as semantic LLM-based evaluation. The improvements across heterogeneous metrics therefore suggest that SOMC improves the underlying memory evidence rather than exploiting a particular evaluator. We emphasize, however, that our theoretical guarantees apply to the submodular utility, not directly to these downstream metrics.

H PROMPT OF LLM AS JUDGE

Following (Fang et al., 2026) and (Ma et al., 2026), LoCoMo uses a unified evaluation prompt for all problem categories. In contrast to LoCoMo's unified prompt, LongMemEvalS uses task-specific prompts for evaluation.

LoCoMo Prompt

Your task is to label an answer to a question as 'CORRECT' or 'WRONG'. You will be given the following data:
(1) a question (posed by one user to another user),
(2) a 'gold' (ground truth) answer,
(3) a generated answer
which you will score as CORRECT/WRONG.
The point of the question is to ask about something one user should know about the other user based on their prior conversations.
The gold answer will usually be a concise answer that includes the referenced topic, for example: Question: Do you remember what I got the last time I went to Hawaii? Gold answer: A shell necklace
The generated answer might be much longer, but you should be generous with your grading - as long as it touches on the same topic as the gold answer, it should be counted as CORRECT.
For time-related questions, the gold answer will be a specific date, month, year, etc. The generated answer might be much longer or use relative time references (like "last Tuesday" or "next month"), but you should be generous with your grading - as long as it refers to the same date or time period as the gold answer, it should be counted as CORRECT. Even if the format differs (e.g., "May 7th" vs "7 May"), consider it CORRECT if it's the same date.
Now it's time for the real question:
Question: {`question`}
Gold answer: {`gold answer`}
Generated answer: {`generated answer`}
First, provide a short (one-sentence) explanation of your reasoning, then finish with CORRECT or WRONG. Do NOT include both CORRECT and WRONG in your response, or it will break the evaluation script. Just return the label CORRECT or WRONG in a JSON format with the key as "label".

LongMemEval Temporal Reasoning Prompt

I will give you a question, a correct answer, and a response from a model. Please answer yes if the response contains the correct answer. Otherwise, answer no. If the response is equivalent to the correct answer or contains all the intermediate steps to get the correct answer, you should also answer yes. If the response only contains a subset of the information required by the answer, answer no. In addition, do not penalize off-by-one errors for the number of days. If the question asks for the number of days/weeks/months, etc., and the model makes off-by-one errors (e.g., predicting 19 days when the answer is 18), the model's response is still correct.
Question: {`question`}
Correct Answer: {`answer`}
Response: {`response`}

LongMemEval Knowledge Update Prompt

I will give you a question, a correct answer, and a response from a model. Please answer yes if the response contains the correct answer. Otherwise, answer no. If the response contains some previous information along with an updated answer, the response should be considered as correct as long as the updated answer is the required answer.
Question: {question}
Correct Answer: {answer}
Response: {response}

LongMemEval Single Session Preference Prompt

I will give you a question, a rubric for the desired personalized response, and a response from a model. Please answer yes if the response satisfies the desired response. Otherwise, answer no. The model does not need to reflect all the points in the rubric. The response is correct as long as it recalls and utilizes the user's personal information correctly.
Question: {question}
Correct Answer: {answer}
Response: {response}

LongMemEval Default Prompt

I will give you a question, a correct answer, and a response from a model. Please answer yes if the response contains the correct answer. Otherwise, answer no. If the response is equivalent to the correct answer or contains all the intermediate steps to get the correct answer, you should also answer yes. If the response only contains a subset of the information required by the answer, answer no.
Question: {question}
Correct Answer: {answer}
Response: {response}