PACE: Predictive Adaptive Context Extraction for Long-Horizon LLM Agents

Lei Wei[1,2]*, **Xiao Peng**[1,✉], **TT**[1], **Guannan Zhang**[1], **Chenhao Jiang**[1], **Hongyu Li**[1], **Lanbo Lin**[1], **Yuanwu Xu**[1], **Jiayao Liu**[1], **Kesu Wang**[1], **Bin Wang**[2]
[1]Alibaba International Digital Commerce Group
[2]Peking University

Abstract

Large Language Model (LLM) agents struggle with ultra-long-horizon tasks requiring hundreds or thousands of interaction steps. Traditional context management approaches face a fundamental dilemma: preserving complete histories rapidly exhausts context windows and forces crude truncation, while aggressive summarization discards critical information prematurely. We propose Predictive Adaptive Context Extraction (PACE), a novel framework that reconceptualizes context management as a Next Step Prediction problem. Inspired by neural attention, PACE dynamically constructs context by adjusting historical memory granularity based on its predicted relevance for the next action. Comprehensive evaluation across diverse benchmarks and models demonstrates that PACE consistently improves task success rates, with larger gains on complex tasks and robust cross-lingual performance. Crucially, PACE enables agents to sustain effective reasoning for 4,897 interaction steps in ultra-long-horizon scenarios, achieving a 66.2× improvement over the full-context ReAct baseline and 5.1× over advanced folding baselines. This fundamentally advances the capability of LLM-based agents in previously intractable long-horizon scenarios.

1 Introduction

The deployment of Large Language Models (LLMs) as autonomous agents has revolutionized how AI systems tackle complex, multi-step tasks across diverse domains (Wei et al., 2022; Yao et al., 2022; Schick et al., 2023; Chowdhery et al., 2023; Wu et al., 2023; Xi et al., 2023). These agent systems demonstrate remarkable capabilities in reasoning, tool use, and decision-making by maintaining context across extended interactions (Shinn et al., 2023; Sun et al., 2025; Ye et al., 2025; Li et al., 2023; Xie et al., 2023; Yang et al., 2025).

However, as task complexity and interaction length increase, a fundamental challenge emerges: the effective management of growing conversational context within the constraints of finite context windows. This limitation becomes increasingly severe in long-horizon tasks that may span hundreds or even thousands of interaction steps, where even models with extended context windows struggle to maintain decision quality as the history accumulates (Gajjar et al., 2025; Zhang et al., 2024b). Traditional approaches to context management in agent systems face significant limitations. ReAct-style methods (Yao et al., 2022; Belcak et al., 2025), by preserving complete interaction histories, introduce high levels of redundancy that quickly exhaust the context window, forcing a crude truncation of older, potentially vital information. Step-wise summarization approaches (Jiang et al., 2023; Xiao et al., 2023) compress history at each step but often discard relevant information prematurely (Chhikara et al., 2025). While recent methods such as AgentFold have introduced specialized folding mechanisms, their utility is often tied to fine-tuning the agent to generate explicit ‘fold‘ instructions. This approach intertwines the primary task-solving process with a secondary, learned meta-cognitive skill of memory management, which can limit the model’s applicability and general-purpose reasoning capabilities (Li et al., 2025b). This dependency, alongside existing context management issues, becomes particularly problematic in ultra-long-horizon scenarios (Mei et al., 2025). In such cases, agents may lose track of critical information from earlier steps, fail to synthesize insights across distant interactions, or become overwhelmed by irrelevant details that dilute decision-making quality (Liu et al., 2024). Critically, this context management problem represents a fundamental bottleneck in scaling agent capabil-

*Work done during internship at Alibaba.
✉Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 27184–27199

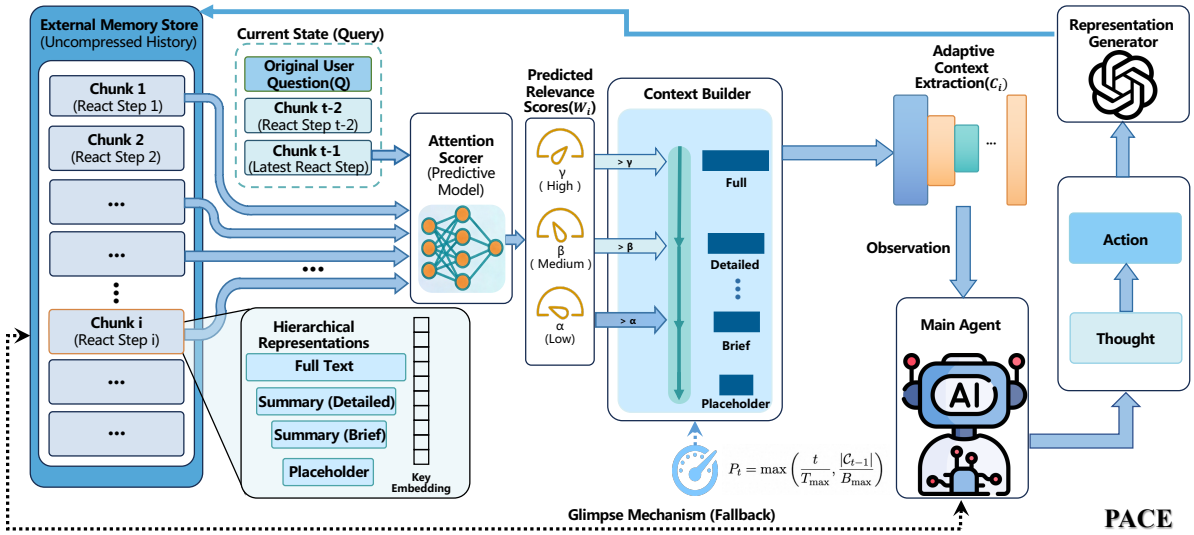

Figure 1: An overview of our PACE framework.

ities, as even with modern long-context models, the quadratic attention complexity and information density challenges limit practical deployment.

To address this critical challenge, we reconceptualize context management by drawing an analogy to the attention mechanisms in neural architectures (Vaswani et al., 2017). Inspired by how Transformer models dynamically weight input tokens to predict the next token, our framework reformulates historical context selection as a Next Step Prediction problem. We conceptualize historical interaction chunks as elements in an external memory store and compute relevance scores for each chunk with respect to the next-step decision (Besta et al., 2024; Chan et al., 2023). Unlike static summarization or fixed-window approaches, our method dynamically modulates the granularity of historical information—ranging from full detail to mere placeholders—based on its predicted utility for the upcoming action. This paradigm shift enables agents to construct context windows that are both compact and maximally informative, adapting fluidly as task requirements evolve. Crucially, this is achieved by implementing an adaptive pressure mechanism, which automatically adjusts compression intensity as the interaction history grows, thereby enabling the agent to maintain effective reasoning capabilities across extended horizons. Motivated by these considerations, we propose Predictive Adaptive Context Extraction (PACE), a novel framework for dynamic context management in LLM-based agent systems. PACE operationalizes the Next Step Prediction paradigm through a lightweight, vectorized attention scorer that predicts the relevance of each historical interaction chunk. This score, in turn, guides the dynamic extraction and assembly of a hierarchically structured context that balances detail preservation with computational efficiency. By leveraging multi-granularity representations of history and pressure-adaptive thresholds, PACE constructs contexts that are simultaneously concise and comprehensively informative, enabling effective scaling to thousands of interaction steps.

The contributions of this work are summarized as follows:

- We introduce PACE, a novel framework that reframes context management as a Next Step Prediction problem, dynamically constructing context through vectorized attention and pressure-adaptive compression based on predicted relevance for the immediate next action.
- We demonstrate comprehensive improvements across six diverse benchmarks and four models of varying scales, including state-of-the-art web agents. PACE achieves larger gains on complex tasks compared to simpler ones, while maintaining robust performance across languages and domains.
- We significantly extend agent operational longevity, achieving 4,897 interaction steps in ultra-long-horizon scenarios, representing a 66.2× improvement over the full-context ReAct baseline and 5.1× over advanced folding baselines.

2 Related Work

2.1 Web Agent Systems

LLM-based web agents represent a powerful paradigm for automating complex web interactions, spanning tasks from information retrieval to autonomous navigation (Hu et al., 2025b). Recent advancements have focused on enhancing agent autonomy and adaptability. This includes leveraging self-supervised reinforcement learning for dynamic web environments (Qi et al., 2025), developing co-evolving world models for iterative self-improvement (Fang et al., 2025), and specializing agents for high-stakes domains (Liu et al., 2025a).

Despite advancing core reasoning and tool-use capabilities, the efficacy of these agents in long-horizon scenarios is fundamentally constrained by the need for robust context management. Our work, Predictive Adaptive Context Extraction (PACE), directly addresses this foundational bottleneck by providing a dynamic mechanism to sustain agent performance across extended interactions.

2.2 Context Management in LLM Agents

Effective context management is pivotal for LLM agents, as decision quality degrades with increasing interaction history. While traditional fixed-window truncation is inadequate for ultra-long-horizon tasks (Xu et al., 2024), recent work has formalized context engineering as a critical discipline (Mei et al., 2025), categorized memory architectures (Zhang et al., 2025b), and developed benchmarks like MemBench (Tan et al., 2025). Additional approaches include long-context alignment (Bai et al., 2024), training-time data sculpting (Lu et al., 2024), and multi-agent collaboration (Zhang et al., 2024a).

Innovative compression mechanisms have emerged, including AgentFold (Ye et al., 2025), Context-Folding (Sun et al., 2025), hierarchical memory management (Hu et al., 2025a), attention biasing (Zhu et al., 2025; Kang et al., 2025), and treating memory as action (Zhang et al., 2025a; Liu et al., 2025b). These methods dynamically consolidate past interactions through procedural rules or learned policies. PACE distinguishes itself through a predictive attention mechanism that scores each historical chunk's relevance for the immediate next step, combined with pressure-adaptive compression that fluidly adjusts granularity. This enables efficient scaling to ultra-long-horizon tasks where policy-based methods may falter.

3 Methodology

In this section, we present the Predictive Adaptive Context Extraction framework for dynamic context management in long-horizon agent tasks, as shown in Figure 1. We formalize the problem setup, detail our system architecture with multi-granularity memory representation, explain the predictive attention mechanism with pressure-adaptive compression, and describe the implementation details.

3.1 Problem Formulation

Consider an agent executing a complex task requiring T interaction steps. At each step t, the agent observes state o_t and selects action a_t. The complete interaction history is:

$$\mathcal{H}_t = \{(a_1, o_1), (a_2, o_2), \ldots, (a_{t-1}, o_{t-1})\} \quad (1)$$

Traditional agents condition actions on the full history, i.e., $a_t = \pi(o_t, \mathcal{H}_t)$, but this becomes computationally prohibitive as t grows. Moreover, preserving complete history introduces information overload that can degrade decision quality, particularly as distant interactions dilute attention from critical recent context.

The goal is to construct a compressed context $\mathcal{C}_t$ from $\mathcal{H}_t$ that satisfies $|\mathcal{C}_t| \ll |\mathcal{H}_t|$ while enhancing decision quality by filtering noise and emphasizing relevant information:

$$\text{Performance}(\pi(o_t, \mathcal{C}_t)) \geq \text{Performance}(\pi(o_t, \mathcal{H}_t)) \quad (2)$$

PACE addresses this as a predictive attention problem, predicting which historical chunks are most relevant for the next action a_t and dynamically adjusting their granularity to construct contexts that are both compact and more informative than unfiltered complete history.

3.2 System Architecture and Multi-Granularity Memory

PACE's architecture comprises five coordinated components. The **Main Agent** is the core decision-maker, receiving the dynamically constructed context $\mathcal{C}_t$ to generate actions using a state-of-the-art LLM. The **External Memory Store** ($\mathcal{M}$) maintains the complete interaction history: $\mathcal{M}_t = \{\text{Chunk}_1, \ldots, \text{Chunk}_{t-1}\}$. The **Representation**

Generator utilizes a LLM (e.g., Gemini 2.5 Flash-Lite) to asynchronously produce multi-level summaries for each chunk, avoiding blockage of the main agent loop. The **Attention Scorer** employs a locally-hosted dense retrieval model (e.g., BGE-M3) to predict chunk relevance via semantic similarity. Finally, the **Context Builder** synthesizes relevance scores with adaptive compression policies to assemble $\mathcal{C}_t$ within token budget constraints.

Each interaction chunk maintains four representations at different granularity levels:

$$\begin{aligned}\text{Chunk}_i = \{&id_i,\, type_i,\, t_i,\, R^{(i)}_{\text{full}},\, R^{(i)}_{\text{detailed}},\\ &R^{(i)}_{\text{brief}},\, R^{(i)}_{\text{ph}},\, \mathbf{k}_i\}\end{aligned} \tag{3}$$

where $\mathbf{k}_i$ is the pre-computed key embedding. The representations range from complete content (R_{full}), a comprehensive summary (R_{detailed}), a concise 1-2 sentence summary (R_{brief}), to a minimal placeholder (R_{ph}). These multiple granularity levels enable flexible compression adapting to varying relevance scores.

3.3 Predictive Attention with Adaptive Compression

PACE's core innovation treats historical context selection as a next-step prediction problem. Let N denote the number of recent chunks preserved in full detail (we set $N = 2$). Given the user's query Q and the most recent N chunks, we construct their concatenation $\mathcal{R}_t = Q \oplus \bigoplus_{j=t-N}^{t-1} R^{(j)}_{\text{full}}$. We encode both the query and historical chunks using an embedding model, which produces dense vectors:

$$\begin{aligned}\mathbf{q}_t &= \text{Enc}(\text{Truncate}(\mathcal{R}_t, L_{\max}))\\ \mathbf{k}_i &= \text{Enc}(\text{Truncate}(R^{(i)}_{\text{full}}, L_{\max}))\end{aligned} \tag{4}$$

where $L_{\max}$ is the encoder's maximum input length, and $\mathbf{k}_i$ is the key embedding for each historical chunk Chunk_i where $i \le t - N - 1$. This symmetric truncation strategy ensures both queries and keys fully utilize the encoder's capacity while maintaining balanced semantic representation. Key embeddings are computed once at chunk creation time and cached, enabling efficient retrieval even before detailed summaries are generated asynchronously. The summaries (R_{detailed}, R_{brief}) are used solely for multi-granularity presentation in the final context.

Let $M = t - N - 1$ denote the number of chunks requiring scoring. Raw similarity scores are computed via cosine similarity: $s_i = \cos(\mathbf{q}_t, \mathbf{k}_i)$. We then apply softmax with low temperature $\tau = 0.3$ to sharpen the distribution, accentuating relevant chunks while suppressing irrelevant ones:

$$w_i = \frac{\exp(s_i/\tau)}{\sum_{j=1}^{M} \exp(s_j/\tau)} \tag{5}$$

To enable adaptive thresholding, we compute relative weights: $\tilde{w}_i = M \cdot w_i$. A relative weight $\tilde{w}_i > 1$ indicates above-average relevance. With low temperature τ, the softmax becomes highly peaked, concentrating weight on top-ranked chunks while driving most $\tilde{w}_i$ values well below 1, naturally facilitating intensified compression.

Pressure-Adaptive Thresholds. A key PACE feature is adapting compression intensity based on interaction state. We define "compression pressure" $P_t \in [0, 1]$ reflecting both task progression and context budget utilization:

$$P_t = \max\left(\frac{t}{T_{\max}}, \frac{|\mathcal{C}_{t-1}|}{B_{\max}}\right) \tag{6}$$

where $T_{\max}$ is the expected maximum task length, $B_{\max}$ is the token budget (set to 128K), and $|\mathcal{C}_{t-1}|$ denotes the previous context's token count. In practice, $T_{\max}$ is set to the 95th percentile of observed task lengths in each benchmark's validation set, providing a robust estimate without requiring precise horizon knowledge. We initialize $|\mathcal{C}_0|$ as the system prompt plus query length, ensuring well-defined initial conditions and avoiding circular dependency.

Base thresholds $(\alpha_0, \beta_0, \gamma_0) = (0.4, 0.8, 1.5)$ are adjusted dynamically: $\alpha_t = \alpha_0 \cdot (1 + \lambda P_t)$, and similarly for β_t and γ_t, where $\lambda = 0.5$ controls adaptation rate. As pressure P_t increases, thresholds rise proportionally, making it progressively harder for chunks to qualify for detailed representations. This pushes more chunks toward brief summaries or placeholders, intensifying compression automatically as context accumulates. The adjusted relative weight $\tilde{w}_i$ determines the representation level:

$$\text{Select}(\text{Chunk}_i) = \begin{cases} R^{(i)}_{\text{full}} & \text{if } \tilde{w}_i > \gamma_t \\ R^{(i)}_{\text{detailed}} & \text{if } \beta_t < \tilde{w}_i \le \gamma_t \\ R^{(i)}_{\text{brief}} & \text{if } \alpha_t < \tilde{w}_i \le \beta_t \\ R^{(i)}_{\text{ph}} & \text{if } \tilde{w}_i \le \alpha_t \end{cases} \tag{7}$$

The final context is assembled as:

$$\mathcal{C}_t = \text{Sys} \oplus Q \oplus \left(\bigoplus_{i=1}^{M} \text{Sel}_i\right) \oplus \left(\bigoplus_{j=t-N}^{t-1} R^{(j)}_{\text{full}}\right) \tag{8}$$

where Sys denotes the system prompt and $\text{Sel}_i = \text{Select}(\text{Chunk}_i)$.

3.4 Implementation Details

Glimpse Mechanism. To mitigate potential scorer errors, PACE includes a glimpse tool allowing the agent to explicitly request full details of any over-compressed chunk. The agent can invoke $\text{glimpse}(i) \to R_{\text{full}}^{(i)}$ when encountering insufficient information. To maintain bounded context growth, we limit glimpse requests to at most 3 per step, with retrieved content included in the next step's token budget. This provides a crucial fallback mechanism while preserving efficiency.

Embedding and Summary Generation. We deploy BGE-M3 locally for vectorized attention computation, with key embeddings computed once at chunk creation and cached, enabling $O(M)$ dot-product operations. BGE-M3 is a multilingual dense retrieval model that natively supports both English and Chinese, enabling cross-lingual robustness without additional fine-tuning. Summary generation runs in background threads; if unavailable when needed, the system falls back to either R_{full} or R_{ph} based on budget constraints. The asynchronous architecture ensures that the main agent loop is never blocked waiting for summary generation, maintaining system responsiveness even under heavy load.

Computational Complexity. At each step t, PACE performs $O(M)$ dot-product operations for attention scoring, where $M = t - N - 1$ is the number of historical chunks. The softmax computation and threshold-based selection are also $O(M)$. Context assembly involves simple concatenation operations. The overall per-step complexity is linear in the history length, making PACE highly scalable compared to quadratic attention mechanisms in transformers.

The complete workflow is summarized in Algorithm 1.

Algorithm 1 Predictive Adaptive Context Extraction (Single Step)

Require: Memory $\mathcal{M}_{t-1}$, observation o_t, query Q, recent count N
Ensure: Action a_t, updated memory $\mathcal{M}_t$

$\mathbf{q}_t \leftarrow \text{Enc}(Q \oplus R_{\text{full}}^{(t-N:t-1)})$
$M \leftarrow t - N - 1$
Retrieve cached $\{\mathbf{k}_i\}_{i=1}^{M}$ from $\mathcal{M}_{t-1}$
$s_i \leftarrow \cos(\mathbf{q}_t, \mathbf{k}_i)$ for all $i \in \{1, \ldots, M\}$
$w_i \leftarrow \text{softmax}(s_i/\tau)$
$\tilde{w}_i \leftarrow M \cdot w_i$
$P_t \leftarrow \max(t/T_{\max}, |\mathcal{C}_{t-1}|/B_{\max})$
$(\alpha_t, \beta_t, \gamma_t) \leftarrow \text{AdaptThresh}(P_t)$
$R_i^* \leftarrow \text{Select}(\text{Chunk}_i, \tilde{w}_i, \alpha_t, \beta_t, \gamma_t)$ for all i
$\mathcal{C}_t \leftarrow \text{Assemble}(Q, \{R_i^*\}, R_{\text{full}}^{(t-N:t-1)})$
$a_t \leftarrow \pi(o_t, \mathcal{C}_t)$ {Agent may use glimpse tool}
$o_{t+1} \leftarrow \text{Env}(a_t)$
Create Chunk_t with $\mathbf{k}_t = \text{Enc}(\text{Trunc}(R_{\text{full}}^{(t)}))$
Enqueue async summary generation for Chunk_t
$\mathcal{M}_t \leftarrow \mathcal{M}_{t-1} \cup \{\text{Chunk}_t\}$
return $a_t, \mathcal{M}_t$

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate PACE on six diverse benchmarks: BrowseComp (Wei et al., 2025) and its Chinese variant BrowseComp-ZH (Zhou et al., 2025) for compositional web browsing tasks; WideSearch (Wong et al., 2025) for broad-scope web exploration; GAIA (Mialon et al., 2023) for general AI assistant tasks requiring real-world reasoning; xbench-DeepResearch (Chen et al., 2025) for deep research capabilities; and WebWalkerQA (Wu et al., 2025) for web navigation question answering. These benchmarks span varying interaction lengths and task complexities, making them ideal for evaluating long-horizon context management. Detailed evaluation protocols for each benchmark are provided in Appendix E.

Models. We evaluate four LLMs across different scales. At the medium scale (30-32B parameters), we select WebSailor-32B (Li et al., 2025a), an open-source agentic model specialized in complex web navigation, and tongyi-deepresearch-30B (Team et al., 2025), the current sota open-source model on web agent tasks with particular strengths in deep research and agent reasoning. For large-scale models, we select DeepSeek-V3.1-671B (DeepSeek-AI et al., 2025) and Claude-4-Sonnet (Team, 2025), which represent leading large-scale LLMs. Deployment details are described in Appendix A.

Baselines. We compare against three context management strategies: (1) **ReAct Agent** (Yao et al., 2022), which preserves complete interaction histories; (2) **Summary Agent** (Wu et al., 2026), which generates cumulative summaries at each step; and (3) **Folding Agent** (Ye et al., 2025; Sun et al., 2025), which employs adaptive folding

Method	BrowseComp	BrowseComp-ZH	WideSearch	GAIA	xbench-DR	WebWalkerQA
WebSailor-32B						
ReAct Agent	7.3	20.7	43.2	46.7	58.0	52.3
Summary Agent	10.5	25.5	47.6	53.5	63.0	57.8
Folding Agent	11.3	26.8	49.5	55.1	65.0	60.2
PACE (Ours)	**13.2**	**29.3**	**52.8**	**59.1**	**68.0**	**63.5**
DeepSeek-V3.1-671B						
ReAct Agent	25.8	44.3	54.8	58.3	66.0	56.4
Summary Agent	30.0	49.2	59.3	63.0	71.0	61.2
Folding Agent	31.6	50.9	61.2	65.4	73.0	62.8
PACE (Ours)	**35.1**	**54.8**	**65.7**	**69.3**	**74.0**	**66.5**
tongyi-deepresearch-30B						
ReAct Agent	38.2	41.6	52.7	64.6	69.0	66.8
Summary Agent	43.4	46.7	57.8	70.1	75.0	72.2
Folding Agent	44.8	47.2	60.1	72.4	77.0	74.6
PACE (Ours)	**47.6**	**51.2**	**64.2**	**74.0**	**81.0**	**78.1**
Claude-4-Sonnet						
ReAct Agent	11.4	27.2	58.4	65.2	62.0	57.9
Summary Agent	12.2	29.1	62.0	68.5	65.0	61.7
Folding Agent	14.5	28.3	64.0	70.1	69.0	60.9
PACE (Ours)	**17.8**	**32.6**	**67.0**	**76.4**	**72.0**	**65.8**

Table 1: Performance comparison of context management methods across six benchmarks. xbench-DR denotes xbench-DeepResearch. The best scores are **bolded**.

mechanisms for selective compression. To ensure fair comparison, all methods use the same summarization model (Gemini 2.5 Flash-Lite) and token budgets across experiments.

4.2 Main Results

Table 1 summarizes our findings, which highlight three key advantages of the PACE framework.

PACE Redefines the State-of-the-Art in Context Management. PACE decisively outperforms the full spectrum of baselines, from naive full-history methods to sophisticated folding agents. This superiority is exemplified on the challenging xbench-DeepResearch benchmark, where tongyi-deepresearch-30B equipped with PACE achieves an 81.0% success rate—a significant +4.0 percentage point gain over the strong Folding Agent baseline. This establishes our predictive attention mechanism as a more effective and generalizable solution.

PACE Unlocks the Latent Potential of Powerful LLMs. The framework creates a powerful synergy with capable models by removing the context bottleneck that often throttles their performance. This is starkly illustrated with Claude-4-Sonnet on the GAIA benchmark, where its success rate leaps to 76.4%—a remarkable +6.3 percentage point increase over the Folding Agent. This demonstrates that PACE enables models to operate closer to their true reasoning potential.

PACE Demonstrates Exceptional Cross-Lingual and Cross-Domain Robustness. The framework's generalizability is confirmed by its consistent high performance across diverse settings. It delivers comparable gains on both English (BrowseComp) and Chinese (BrowseComp-ZH) benchmarks and proves effective across the full spectrum of tested domains, from web navigation to deep research. This establishes PACE as a fundamental architectural improvement rather than a narrow, domain-specific solution.

4.3 Ablation Study

To understand the contribution of individual components within PACE, we conduct ablation experiments on tongyi-deepresearch-30B by selectively removing key mechanisms: (1) multi-granularity representations, retaining only full content and placeholders; (2) pressure-adaptive thresholds, using fixed values throughout execution; (3) low-temperature softmax, reverting to standard temperature $\tau = 1.0$; and (4) the glimpse mechanism, disabling on-demand detail retrieval.

As shown in Table 2, removing multi-granularity representations causes the most severe performance

Configuration	BrowseComp	BrowseComp-ZH	WideSearch	GAIA
PACE (Full)	**47.6**	**51.2**	**64.2**	**74.0**
w/o Multi-gran.	42.3	45.5	55.9	68.5
w/o Pressure	43.9	46.4	58.4	70.9
w/o Softmax	43.1	46.0	56.7	69.3
w/o Glimpse	44.2	47.5	59.3	71.7
ReAct Agent	38.2	41.6	52.7	64.6
Summary Agent	43.4	46.7	57.8	70.1
Folding Agent	44.8	47.2	60.1	72.4

Table 2: Ablation study showing the contribution of each PACE component. ReAct Agent, Summary Agent, and Folding Agent baselines are included for reference.

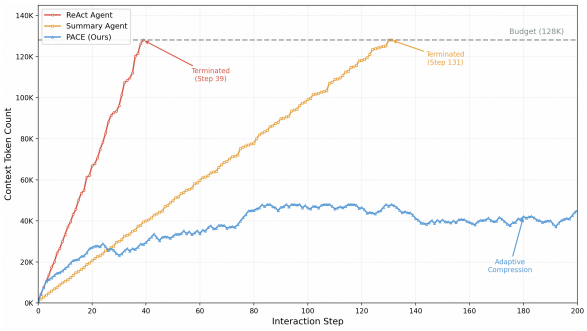

Figure 2: Context token usage over interaction steps on GAIA Level 3 tasks (128K budget).

degradation (5.3–8.3% across datasets), falling below both Folding Agent and Summary Agent baselines and approaching the performance level of ReAct Agent. This confirms that flexible detail levels are the cornerstone of PACE's effectiveness. Removing the low-temperature softmax also results in substantial degradation (4.7–7.5%), falling below both Summary Agent and Folding Agent on most benchmarks, which demonstrates that sharpening the attention distribution is critical for accurately identifying relevant historical chunks. The pressure-adaptive mechanism yields moderate performance drops, with the ablated variant performing comparably to Summary Agent but below Folding Agent. Notably, *w/o Pressure* falls below Summary Agent on BrowseComp-ZH (46.4% vs 46.7%), suggesting that adaptive compression becomes particularly important for certain task characteristics. The glimpse mechanism provides the smallest but consistent contribution (2.3–4.9%), with the ablated variant still outperforming Summary Agent but falling short of Folding Agent on most benchmarks. The full PACE system consistently outperforms all ablated variants and all baselines, demonstrating that these components work synergistically to achieve optimal performance that surpasses even the sophisticated Folding Agent approach.

4.4 Context Growth Analysis

To quantify PACE's efficiency in managing context accumulation, we analyze token usage dynamics across interaction steps on GAIA Level 3 tasks, which frequently require extended execution horizons. Figure 2 compares the context growth patterns of ReAct, Summary Agent, and PACE under a 128K token budget.

ReAct exhibits near-linear growth as it preserves complete interaction history, rapidly exhausting the 128K budget and forcing termination at step 39. Summary Agent mitigates this through cumulative summarization, extending viable execution to step 131—a 3.4× improvement over ReAct. However, its compression strategy remains fundamentally linear, inevitably reaching the budget ceiling.

In contrast, PACE demonstrates fundamentally different dynamics. During the first 20 steps, context grows approximately linearly, reaching 27.1K tokens. Beyond this point, the pressure-adaptive mechanism effectively bounds growth, with context size exhibiting fluctuations while maintaining a gradual upward trend as the system dynamically balances information retention against budget constraints.

Notably, PACE's context usage at step 200 (45.1K) remains comparable to step 20 (27.1K), representing only a 1.7× increase despite a 10× growth in interaction length. At step 39 (ReAct's termination point), PACE uses 28.5K tokens compared to ReAct's 128K—a 78% reduction. At step 131 (Summary Agent's termination point), PACE maintains 48.0K tokens versus Summary Agent's 128K—a 63% reduction. This bounded growth directly enables the performance gains observed in Table 1: while baselines reach their context limits and are forced to perform crude truncation that permanently discards information from distant interactions, leading to degraded performance, PACE sustains effective reasoning across the full task horizon, translating context efficiency into higher task completion rates.

4.5 Performance vs. Task Complexity

To assess robustness under increasing task difficulty, we analyze GAIA's three-level difficulty hierarchy and compare success rates across methods. Level 1 tasks are simple problems requiring minimal reasoning and tool use, Level 2 tasks demand moderate multi-step reasoning with increased complexity, and Level 3 tasks involve highly com-

Model	Method	Level 1	Level 2	Level 3
tongyi-deepresearch-30B	ReAct Agent	83.3	57.6	42.1
	Summary Agent	85.7	65.2	47.4
	Folding Agent	85.7	68.2	52.6
	PACE (Ours)	**88.1**	**68.2**	**57.9**
Claude-4-Sonnet	ReAct Agent	83.3	59.1	42.1
	Summary Agent	83.3	63.6	47.4
	Folding Agent	85.7	63.6	52.6
	PACE (Ours)	**88.1**	**71.2**	**63.2**

Table 3: Success Rate (%) across GAIA difficulty levels. PACE's advantage grows substantially with task complexity.

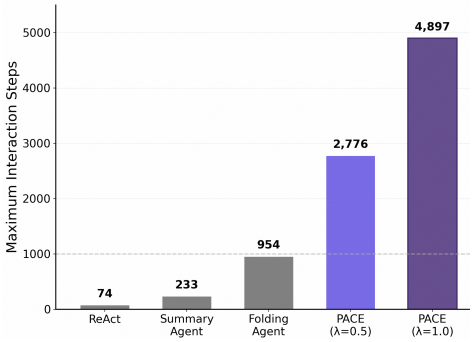

Figure 3: Operational longevity comparison in the ultra-long-horizon stress test (256K context budget).

plex long-horizon problems requiring sophisticated planning and multi-tool coordination.

As shown in Table 3, all methods perform comparably on simpler Level 1 tasks where context management is less critical, with differences within 5 percentage points. However, the performance gap widens dramatically with increasing complexity. On Level 2 tasks, PACE with Claude-4-Sonnet achieves 71.2%, a +12.1 point gain over ReAct (59.1%) and +7.6 points over Folding Agent (63.6%). The most striking results emerge on challenging Level 3 tasks: PACE enables Claude-4-Sonnet to reach 63.2%, representing a +21.1 point improvement over ReAct (42.1%) and +10.6 points over Folding Agent (52.6%). This escalating advantage reflects PACE's core strength: as task horizons extend, traditional approaches suffer from either context overflow (ReAct) or information loss (Summary and Folding), while PACE's pressure-adaptive mechanism dynamically balances detail preservation with compression efficiency.

4.6 Ultra-Long Horizon Stress Test

To rigorously probe the scaling limits of our framework under realistic constraints, we designed an ultra-long-horizon stress test. Standard benchmarks do not naturally generate thousands of interaction steps, so we constructed a single "meta-task" by concatenating all questions from GAIA's Level 2 and Level 3 test sets. The agent, powered by Gemini 2.5 Pro, was instructed to solve every question sequentially within one continuous session. We configured the agent with a 256K token context window, defining failure as the point where this limit was exceeded. This design forces the agent to manage a vast and complex history spanning dozens of distinct sub-tasks.

The results, depicted in Figure 3, reveal a stark divergence in operational longevity. The standard ReAct agent reached its context limit in a mere 74 steps, overwhelmed by the history of just a few sub-tasks. The Summary Agent extended this to 233 steps but lost track of the overall task structure due to non-selective compression. The more advanced Folding Agent fared significantly better, reaching 954 steps before its context became unmanageable.

In stark contrast, PACE with its standard adaptive pressure ($\lambda = 0.5$) sustained operation for 2,776 steps—a $2.9\times$ improvement in operational longevity over the Folding Agent and a remarkable $37.5\times$ improvement over the full-context ReAct baseline. Even more remarkably, by increasing the adaptation rate for aggressive compression ($\lambda = 1.0$), PACE extended its operational horizon to 4,897 steps, achieving a $5.1\times$ improvement over Folding Agent and an exceptional $66.2\times$ improvement over ReAct. This result powerfully demonstrates that PACE's predictive, pressure-adaptive mechanism is critical for navigating ultra-long-horizon tasks, enabling agents to maintain coherent reasoning over thousands of interaction steps within a finite, albeit large, context window.

5 Conclusion

We presented PACE, a framework that addresses context management in long-horizon agent tasks by reframing it as a Next Step Prediction problem. PACE dynamically constructs compact, informative contexts using lightweight attention scoring and pressure-adaptive compression. Extensive evaluations demonstrate substantial performance gains across diverse domains and model scales, while extending operational longevity to 4,897 steps—a $66.2\times$ improvement over the full-context ReAct baseline. This work fundamentally extends the capability of LLM-based agents in ultra-long-horizon scenarios that were previously intractable.

Limitations

While PACE demonstrates strong performance, several limitations warrant consideration. First, our attention scorer relies on semantic similarity as a proxy for relevance, which may not fully capture task-specific dependencies or causal relationships between historical steps. Second, the multi-granularity representations are generated by a fixed summarization model; learning task-adaptive compression strategies could further improve information retention.

Ethical considerations

Our research focuses on improving context management for LLM-based agents and does not involve the collection of personal data or human subjects. All experiments were conducted using publicly available benchmarks (BrowseComp, BrowseComp-ZH, WideSearch, GAIA, xbench-DeepResearch, and WebWalkerQA) in accordance with their respective licenses and intended usage guidelines. Artifacts restricting modification or commercial use were excluded to comply with intellectual property rights. We explicitly defined the intended use for the artifacts we created and verified compatibility with original access conditions. The agent systems evaluated interact only with public web content and do not access private or sensitive information.

References

Yushi Bai, Xin Lv, Jiajie Zhang, Yuze He, Ji Qi, Lei Hou, Jie Tang, Yuxiao Dong, and Juanzi Li. 2024. Longalign: A recipe for long context alignment of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1376–1395.

Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. 2025. Small language models are the future of agentic ai. *Preprint*, arXiv:2506.02153.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2023. Chateval: Towards better llm-based evaluators through multi-agent debate. *arXiv preprint arXiv:2308.07201*.

Kaiyuan Chen, Yixin Ren, Yang Liu, Xiaobo Hu, Haotong Tian, Tianbao Xie, Fangfu Liu, Haoye Zhang, Hongzhang Liu, Yuan Gong, Chen Sun, Han Hou, Hui Yang, James Pan, Jianan Lou, Jiayi Mao, Jizheng Liu, Jinpeng Li, Kangyi Liu, and 14 others. 2025. xbench: Tracking agents productivity scaling with profession-aligned real-world evaluations. *Preprint*, arXiv:2506.13651.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready ai agents with scalable long-term memory. *Preprint*, arXiv:2504.19413.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and 1 others. 2023. Palm: Scaling language modeling with pathways. *Journal of machine learning research*, 24(240):1–113.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2025. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Tianqing Fang, Hongming Zhang, Zhisong Zhang, Kaixin Ma, Wenhao Yu, Haitao Mi, and Dong Yu. 2025. Webevolver: Enhancing web agent self-improvement with co-evolving world model. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 8970–8986.

Pranshav Gajjar, Cong Shen, and Vijay K Shah. 2025. Tele-llm-hub: Building context-aware multi-agent llm systems for telecom networks. *Preprint*, arXiv:2511.09087.

Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. 2025a. HiAgent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32779–32798, Vienna, Austria. Association for Computational Linguistics.

Xueyu Hu, Tao Xiong, Biao Yi, Zishu Wei, Ruixuan Xiao, Yurun Chen, Jiasheng Ye, Meiling Tao, Xiangxin Zhou, Ziyu Zhao, Yuhuai Li, Shengze Xu, Shenzhi Wang, Xinchen Xu, Shuofei Qiao, Zhaokai Wang, Kun Kuang, Tieyong Zeng, Liang Wang, and 10 others. 2025b. OS agents: A survey on MLLM-based agents for computer, phone and browser use. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7436–7465, Vienna, Austria. Association for Computational Linguistics.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. Llmlingua: Compressing prompts for accelerated inference of large language models. *Preprint*, arXiv:2310.05736.

Minki Kang, Wei-Ning Chen, Dongge Han, Huseyin A. Inan, Lukas Wutschitz, Yanzhi Chen, Robert Sim, and Saravan Rajmohan. 2025. Acon: Optimizing context compression for long-horizon llm agents. *Preprint*, arXiv:2510.00615.

Woosuk Kwon. 2025. *vLLM: An Efficient Inference Engine for Large Language Models*. Ph.D. thesis, UC Berkeley.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in neural information processing systems*, 36:51991–52008.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, Weizhou Shen, Junkai Zhang, Dingchu Zhang, Xixi Wu, Yong Jiang, Ming Yan, Pengjun Xie, Fei Huang, and Jingren Zhou. 2025a. Websailor: Navigating super-human reasoning for web agent. *Preprint*, arXiv:2507.02592.

Mo Li, L. H. Xu, Qitai Tan, Long Ma, Ting Cao, and Yunxin Liu. 2025b. Sculptor: Empowering llms with cognitive agency via active context management. *Preprint*, arXiv:2508.04664.

Bin Liu, Yanjie Zhao, Guoai Xu, and Haoyu Wang. 2025a. Llm agents for automated web vulnerability reproduction: Are we there yet? *arXiv preprint arXiv:2510.14700*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173.

Shukai Liu, Jian Yang, Bo Jiang, Yizhi Li, Jinyang Guo, Xianglong Liu, and Bryan Dai. 2025b. Context as a tool: Context management for long-horizon swe-agents. *Preprint*, arXiv:2512.22087.

Keer Lu, Xiaonan Nie, Zheng Liang, Da Pan, Shusen Zhang, Keshi Zhao, Weipeng Chen, Zenan Zhou, Guosheng Dong, Bin Cui, and Wentao Zhang. 2024. Datasculpt: Crafting data landscapes for long-context llms through multi-objective partitioning. *Preprint*, arXiv:2409.00997.

Lingrui Mei, Jiayu Yao, Yuyao Ge, Yiwei Wang, Baolong Bi, Yujun Cai, Jiazhi Liu, Mingyu Li, Zhong-Zhi Li, Duzhen Zhang, Chenlin Zhou, Jiayi Mao, Tianze Xia, Jiafeng Guo, and Shenghua Liu. 2025. A survey of context engineering for large language models. *Preprint*, arXiv:2507.13334.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2023. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Jiadai Sun, Xinyue Yang, Yu Yang, Shuntian Yao, Wei Xu, Jie Tang, and Yuxiao Dong. 2025. WebRL: Training LLM web agents via self-evolving online curriculum reinforcement learning. In *The Thirteenth International Conference on Learning Representations*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652.

Weiwei Sun, Miao Lu, Zhan Ling, Kang Liu, Xuesong Yao, Yiming Yang, and Jiecao Chen. 2025. Scaling long-horizon llm agent via context-folding. *arXiv preprint arXiv:2510.11967*.

Haoran Tan, Zeyu Zhang, Chen Ma, Xu Chen, Quanyu Dai, and Zhenhua Dong. 2025. Membench: Towards more comprehensive evaluation on the memory of llm-based agents. *Preprint*, arXiv:2506.21605.

Anthropic Team. 2025. Claude 4: Advancing multimodal reasoning and agent capabilities. *Technical Report*.

Tongyi DeepResearch Team, Baixuan Li, Bo Zhang, Dingchu Zhang, Fei Huang, Guangyu Li, Guoxin Chen, Huifeng Yin, Jialong Wu, Jingren Zhou, Kuan Li, Liangcai Su, Litu Ou, Liwen Zhang, Pengjun Xie, Rui Ye, Wenbiao Yin, Xinmiao Yu, Xinyu Wang, and 38 others. 2025. Tongyi deepresearch technical report. *Preprint*, arXiv:2510.24701.

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

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. 2025. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances*

in neural information processing systems, 35:24824–24837.

Ryan Wong, Jiawei Wang, Junjie Zhao, Li Chen, Yan Gao, Long Zhang, Xuan Zhou, Zuo Wang, Kai Xiang, Ge Zhang, Wenhao Huang, Yang Wang, and Ke Wang. 2025. Widesearch: Benchmarking agentic broad info-seeking. *Preprint*, arXiv:2508.07999.

Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, and Fei Huang. 2025. Webwalker: Benchmarking llms in web traversal. *Preprint*, arXiv:2501.07572.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation. *Preprint*, arXiv:2308.08155.

Xixi Wu, Kuan Li, Yida Zhao, Liwen Zhang, Litu Ou, Huifeng Yin, Zhongwang Zhang, Xinmiao Yu, Dingchu Zhang, Yong Jiang, Pengjun Xie, Fei Huang, Minhao Cheng, Shuai Wang, Hong Cheng, and Jingren Zhou. 2026. Resum: Unlocking long-horizon search intelligence via context summarization. *Preprint*, arXiv:2509.13313.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, and 10 others. 2023. The rise and potential of large language model based agents: A survey. *Preprint*, arXiv:2309.07864.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*.

Tianbao Xie, Fan Zhou, Zhoujun Cheng, Peng Shi, Luoxuan Weng, Yitao Liu, Toh Jing Hua, Junning Zhao, Qian Liu, Che Liu, Leo Z. Liu, Yiheng Xu, Hongjin Su, Dongchan Shin, Caiming Xiong, and Tao Yu. 2023. Openagents: An open platform for language agents in the wild. *Preprint*, arXiv:2310.10634.

Heng-Da Xu, Xian-Ling Mao, Puhai Yang, Fanshu Sun, and Heyan Huang. 2024. Rethinking task-oriented dialogue systems: From complex modularity to zero-shot autonomous agent. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2748–2763, Bangkok, Thailand. Association for Computational Linguistics.

Bufang Yang, Lilin Xu, Liekang Zeng, Kaiwei Liu, Siyang Jiang, Wenrui Lu, Hongkai Chen, Xiaofan Jiang, Guoliang Xing, and Zhenyu Yan. 2025. Contextagent: Context-aware proactive llm agents with open-world sensory perceptions. *Preprint*, arXiv:2505.14668.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Rui Ye, Zhongwang Zhang, Kuan Li, Huifeng Yin, Zhengwei Tao, Yida Zhao, Liangcai Su, Liwen Zhang, Zile Qiao, Xinyu Wang, Pengjun Xie, Fei Huang, Siheng Chen, Jingren Zhou, and Yong Jiang. 2025. Agentfold: Long-horizon web agents with proactive context management. *Preprint*, arXiv:2510.24699.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Ö Arık. 2024a. Chain of agents: Large language models collaborating on long-context tasks. *Advances in Neural Information Processing Systems*, 37:132208–132237.

Yuxiang Zhang, Jiangming Shu, Ye Ma, Xueyuan Lin, Shangxi Wu, and Jitao Sang. 2025a. Memory as action: Autonomous context curation for long-horizon agentic tasks. *arXiv preprint arXiv:2510.12635*.

Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2024b. A survey on the memory mechanism of large language model based agents. *Preprint*, arXiv:2404.13501.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2025b. A survey on the memory mechanism of large language model-based agents. *ACM Transactions on Information Systems*, 43(6):1–47.

Peilin Zhou, Bruce Leon, Xiang Ying, Can Zhang, Yifan Shao, Qichen Ye, Dading Chong, Zhiling Jin, Chenxuan Xie, Meng Cao, Yuxin Gu, Sixin Hong, Jing Ren, Jian Chen, Chao Liu, and Yining Hua. 2025. Browsecomp-zh: Benchmarking web browsing ability of large language models in chinese. *Preprint*, arXiv:2504.19314.

Youxiang Zhu, Ruochen Li, Danqing Wang, Daniel Haehn, and Xiaohui Liang. 2025. Focus directions make your language models pay more attention to relevant contexts. *arXiv preprint arXiv:2503.23306*.

Appendix

A Computational Environment

All open-source LLMs (WebSailor-32B and tongyi-deepresearch-30B) were deployed on a cluster featuring 4 NVIDIA A100 GPUs (80GB), utilizing the VLLM framework (Kwon, 2025) for efficient inference. DeepSeek-V3.1-671B and Claude-4-Sonnet were accessed through their official APIs provided by DeepSeek AI and Anthropic, respectively.

B Latency Analysis

To evaluate the computational efficiency of PACE, we analyzed per-step latency across 1,000 trajectories sampled from the BrowseComp benchmark. Table 4 reports the average time required to process each interaction step, measured from receiving the environment observation to generating the next action.

Method	Avg. Latency (s/step)
ReAct Agent	47.63
Summary Agent	56.71
PACE (Ours)	48.08

Table 4: Per-step latency comparison on BrowseComp.

The ReAct Agent serves as the baseline with 47.63 seconds per step. The Summary Agent incurs substantial additional overhead (56.71 seconds per step, a 19.1% increase) due to synchronous summarization at each step, where the agent must wait for the summary model to process the complete history before proceeding.

PACE achieves competitive efficiency at 48.08 seconds per step—only 0.9% slower than ReAct while providing significantly better context management. Although PACE introduces a minimal overhead for (1) encoding the current query and recent context into a query vector using BGE-M3, and (2) performing $t-1$ cached dot-product operations to compute relevance scores, this cost is negligible compared to the main agent inference time. Crucially, PACE's asynchronous architecture ensures that summary generation for historical chunks occurs in background threads and does not block the main agent loop. This design allows PACE to maintain near-baseline latency while constructing dynamically optimized contexts, in stark contrast to Summary Agent's synchronous approach which adds nearly 10 seconds per step. The modest 0.45-second overhead represents an excellent trade-off given PACE's substantial performance gains reported in Table 1.

C Hyperparameter Analysis

PACE introduces four key hyperparameters that govern the adaptive context extraction process: the base thresholds α_0, β_0, and γ_0 that determine representation granularity levels, and the adaptation rate λ that controls how aggressively compression intensifies under pressure. In this section, we systematically analyze the impact of these hyperparameters on performance across all six benchmarks to validate our default settings and provide insights into the framework's robustness.

C.1 Base Threshold Analysis

The base thresholds $(\alpha_0, \beta_0, \gamma_0)$ define the boundaries for selecting representation levels when compression pressure is zero. Recall from Section 3 that these thresholds are applied to the relative attention weights $\tilde{w}_i = M \cdot w_i$ to determine whether each historical chunk receives full detail ($\tilde{w}_i > \gamma_0$), detailed summary ($\beta_0 < \tilde{w}_i \leq \gamma_0$), brief summary ($\alpha_0 < \tilde{w}_i \leq \beta_0$), or placeholder ($\tilde{w}_i \leq \alpha_0$).

We conduct a grid search over threshold configurations while fixing $\lambda = 0.5$ and evaluate performance on tongyi-deepresearch-30B. Table 5 presents results for representative configurations that maintain the ordering $\alpha_0 < \beta_0 < \gamma_0$.

Several observations emerge from this analysis. First, the default configuration $(0.4, 0.8, 1.5)$ achieves the best or near-best performance across all benchmarks, validating its selection. Second, thresholds that are too conservative (low values like $0.3, 0.6, 1.2$) fail to compress sufficiently, leading to context overflow in longer tasks. Conversely, overly aggressive thresholds (high values like $0.6, 1.0, 1.9$) discard too much information prematurely, particularly hurting performance on complex benchmarks like GAIA and xbench-DeepResearch where detailed historical context is crucial. Third, the relative spacing between thresholds matters: configurations that compress the range (e.g., $0.4, 0.7, 1.5$) perform worse than those with balanced separation, suggesting that maintaining distinct granularity levels is important for flexible compression.

The performance degradation from suboptimal thresholds is moderate (typically 2-4 percentage points), indicating reasonable robustness. However,

α_0	β_0	γ_0	BrowseComp	BrowseComp-ZH	WideSearch	GAIA	xbench-DR	WebWalkerQA
0.3	0.6	1.2	42.1	45.8	58.3	70.1	77.0	73.2
0.3	0.7	1.3	44.9	48.1	61.5	72.4	78.0	75.0
0.4	0.8	1.5	**47.6**	**51.2**	**64.2**	**74.0**	**81.0**	**78.1**
0.5	0.9	1.7	45.3	48.6	61.8	72.4	79.0	75.4
0.6	1.0	1.9	43.5	46.3	59.1	70.9	77.0	73.8
0.4	0.7	1.5	44.2	47.5	60.2	71.7	77.0	74.5
0.4	0.9	1.5	45.8	48.9	62.1	72.4	79.0	75.9
0.4	0.8	1.3	44.6	47.8	60.8	71.7	78.0	74.8
0.4	0.8	1.7	46.1	49.3	62.4	73.2	79.0	76.2

Table 5: Performance across benchmarks for different base threshold configurations using tongyi-deepresearch-30B with $\lambda = 0.5$. The default configuration $(\alpha_0, \beta_0, \gamma_0) = (0.4, 0.8, 1.5)$ is highlighted in bold.

the consistent advantage of $(0.4, 0.8, 1.5)$ across diverse benchmarks demonstrates that this configuration achieves an effective balance between aggressive early compression and preserving critical details.

C.2 Adaptation Rate Analysis

The adaptation rate λ controls how rapidly thresholds increase with compression pressure P_t, where adjusted thresholds follow $\alpha_t = \alpha_0 \cdot (1 + \lambda P_t)$. Higher λ values lead to more aggressive compression as tasks progress or context budgets fill, while lower values maintain more consistent compression throughout execution.

Table 6 presents performance across different λ values using the default base thresholds $(0.4, 0.8, 1.5)$ on tongyi-deepresearch-30B.

The results reveal a clear inverted-U relationship between λ and performance. When $\lambda = 0$ (no adaptation), performance degrades significantly across all benchmarks, particularly on longer-horizon tasks like GAIA and xbench-DeepResearch. This confirms that static thresholds are insufficient—as context accumulates, fixed compression leads to either premature information loss or context overflow. The optimal range appears to be $\lambda \in [0.3, 0.7]$, with $\lambda = 0.5$ achieving the best overall performance.

Excessively high adaptation rates ($\lambda \geq 1.0$) hurt performance by compressing too aggressively in later task stages. This is particularly evident on complex benchmarks: GAIA performance drops from 74.0% at $\lambda = 0.5$ to 71.7% at $\lambda = 1.0$. This suggests that even under high pressure, maintaining some level of detailed historical information remains important for effective reasoning.

Interestingly, the sensitivity to λ varies across benchmarks. Simpler tasks like BrowseComp show relatively stable performance across $\lambda \in [0.3, 0.7]$ (variation within 2.1 percentage points), while complex multi-step tasks like xbench-DeepResearch exhibit larger variation (2.0 percentage points), indicating that adaptive pressure management becomes increasingly critical as task complexity grows.

C.3 Cross-Model Validation

To validate that our hyperparameter choices generalize across different model architectures and scales, we evaluate the default configuration $(\alpha_0, \beta_0, \gamma_0, \lambda) = (0.4, 0.8, 1.5, 0.5)$ against alternative settings on all four models used in our main experiments. Table 7 compares the default configuration with two representative alternatives: conservative thresholds with low adaptation $(0.3, 0.6, 1.2, 0.3)$ and aggressive thresholds with high adaptation $(0.5, 0.9, 1.7, 0.7)$.

The default configuration consistently outperforms both alternatives across all four models and all six benchmarks. This consistency demonstrates that our hyperparameter choices are not tuned to a specific model architecture but rather capture fundamental properties of the context management problem.

Notably, the relative performance gap between configurations remains fairly consistent across models. For instance, the advantage of the default over conservative settings on GAIA ranges from 4.3 to 6.3 percentage points across models, suggesting that the hyperparameters' effectiveness scales with model capability. The smaller performance differences on simpler benchmarks like BrowseComp (2.4-3.6 points) versus complex ones like xbench-DeepResearch (2.0-5.2 points) further reinforce that adaptive pressure management becomes more critical as task difficulty increases.

λ	BrowseComp	BrowseComp-ZH	WideSearch	GAIA	xbench-DR	WebWalkerQA
0.0	42.8	46.1	58.9	70.9	76.0	73.5
0.3	45.5	49.0	62.1	72.4	79.0	76.1
0.5	**47.6**	**51.2**	**64.2**	**74.0**	**81.0**	**78.1**
0.7	46.2	49.5	62.6	73.2	79.0	76.6
1.0	44.3	47.6	60.5	71.7	78.0	74.8

Table 6: Performance across benchmarks for different adaptation rates λ using tongyi-deepresearch-30B with base thresholds $(0.4, 0.8, 1.5)$. The default $\lambda = 0.5$ is highlighted in bold.

Model	Config	BrowseComp	BrowseComp-ZH	WideSearch	GAIA	xbench-DR	WebWalkerQA
WebSailor-32B	Conservative	10.8	26.5	49.2	53.5	65.0	59.5
	Default	**13.2**	**29.3**	**52.8**	**59.1**	**68.0**	**63.5**
	Aggressive	11.5	27.2	50.1	55.1	66.0	60.8
DeepSeek-V3.1-671B	Conservative	31.2	50.5	61.8	63.0	71.0	62.1
	Default	**35.1**	**54.8**	**65.7**	**69.3**	**74.0**	**66.5**
	Aggressive	32.5	51.8	63.1	65.4	72.0	63.6
tongyi-deepresearch-30B	Conservative	44.2	47.6	59.5	70.9	78.0	74.3
	Default	**47.6**	**51.2**	**64.2**	**74.0**	**81.0**	**78.1**
	Aggressive	45.1	48.5	60.8	72.4	79.0	75.2
Claude-4-Sonnet	Conservative	14.2	29.8	63.5	70.1	69.0	62.1
	Default	**17.8**	**32.6**	**67.0**	**76.4**	**72.0**	**65.8**
	Aggressive	15.3	30.5	64.8	72.4	70.0	63.2

Table 7: Performance comparison across models and hyperparameter configurations. Conservative: $(0.3, 0.6, 1.2, 0.3)$; Default: $(0.4, 0.8, 1.5, 0.5)$; Aggressive: $(0.5, 0.9, 1.7, 0.7)$.

C.4 Summary

Our comprehensive hyperparameter analysis leads to the following conclusions: (1) PACE demonstrates reasonable robustness to hyperparameter choices, with performance degrading gradually rather than catastrophically when deviating from optimal settings, with typical performance drops of 2-5 percentage points for moderate deviations. (2) The default configuration $(\alpha_0, \beta_0, \gamma_0, \lambda) = (0.4, 0.8, 1.5, 0.5)$ consistently performs best across diverse benchmarks, models, and task complexities, validating our design choices. (3) The pressure-adaptive mechanism ($\lambda > 0$) is essential for strong performance, particularly on long-horizon tasks, with static thresholds ($\lambda = 0$) resulting in 2-5 percentage point drops on complex benchmarks. (4) While the default configuration works well generally, practitioners working on extremely long-horizon scenarios (>2000 steps) may benefit from slightly higher λ (e.g., 0.7-1.0) to prevent context overflow. (5) The optimal hyperparameters generalize well across different model architectures and scales, from 30B to 671B parameters, suggesting that the framework's effectiveness is rooted in algorithmic design rather than model-specific tuning.

D Tool Descriptions

The agent systems evaluated in this work have access to six primary tools for interacting with external environments and information sources. These tools enable the agents to perform web search, extract targeted information from web pages, execute computational tasks, retrieve academic literature, and process multimodal files. Our tool configuration generally follows the setup of Tongyi DeepResearch (Team et al., 2025), while several implementation details, APIs, and tool behaviors are modified for our system, and additional task-specific adaptations are introduced where necessary.

Search. This tool provides web search capabilities powered by Google's search engine. It accepts one or more search queries as input and executes them concurrently. For each query, the tool returns the top-10 search results, where each result contains a title, a brief descriptive snippet, and the corresponding URL. The tool automatically detects the language of input queries and adjusts the search region accordingly (e.g., Chinese queries trigger searches within the China region).

Visit. The Visit tool enables goal-directed information extraction from web pages. It takes as input a set of URLs, each paired with a specific

information-seeking objective. The tool first retrieves the full content of each web page and converts it to Markdown format using the You.com Contents API. Subsequently, a summary model processes the content to extract only the information relevant to the specified objective for that page. The output is structured into three components: a rationale explaining the extraction strategy, evidence snippets supporting the extracted information, and a concise summary addressing the goal.

PythonInterpreter. This tool executes Python code within a secure sandboxed environment (SandboxFusion). The input is a string containing Python code, which must be enclosed within <code> tags. The tool runs the provided code with a 50-second timeout and captures the standard output. Supported libraries include NumPy, Pandas, and Matplotlib, enabling data manipulation, numerical computation, and visualization. The tool implements automatic retry logic with up to 8 attempts across multiple endpoints to ensure robustness.

GoogleScholar. The GoogleScholar tool retrieves academic publications through the Google Scholar search engine via the Serper.dev API. It accepts a list of search queries and returns scholarly literature for each query. Results include article titles, publication years, journal or conference information, citation counts, and PDF links when available. This tool is particularly useful for tasks requiring authoritative references or literature reviews.

ParseFile. This tool processes multimodal files from local storage or URLs, supporting formats including PDF, DOCX, PPTX, TXT, CSV, XLSX, MP4, and MP3. For text-based documents, the tool uses Alibaba Cloud's Dashscope document intelligence service to extract plain text content. For audio and video files, it performs automatic transcription to convert speech into text. Once all inputs are converted to a unified textual representation, a summary model analyzes the content to generate answers to user-specified questions. The tool implements automatic content compression when the extracted text exceeds length limits.

Glimpse. The Glimpse tool enables on-demand retrieval of full details from historical interaction chunks that have been compressed in the current context. It accepts one or more chunk identifiers as input, with a maximum of 3 chunks per invocation to maintain bounded context growth. For each requested chunk, the tool retrieves the complete original content including the full observation and action from that interaction step, bypassing any summarization or compression that may have been applied. The retrieved content is returned in its original format and is automatically incorporated into the context for the next reasoning step. This tool serves as a fallback mechanism when the agent encounters insufficient information in compressed summaries, allowing it to access detailed historical information without maintaining all past interactions in full fidelity throughout the entire task execution.

All tools are integrated into the agent's action space and can be invoked through structured function calls during task execution. Tool selection and parameter specification are determined by the agent's reasoning process at each interaction step.

E Evaluation Details

We employ benchmark-specific evaluation protocols to ensure consistency with prior work and enable fair comparison across methods. Each benchmark uses tailored metrics and judging procedures as detailed below.

BrowseComp. Following the original evaluation protocol (Wei et al., 2025), we measure task success rate by comparing agent-generated answers against ground-truth targets. We deploy GPT-4o-2024-08-06 as the automatic judge to assess whether the agent's final answer correctly addresses the compositional web browsing query. The judging prompt is adapted from the original benchmark specification, which evaluates semantic equivalence between the agent's answer and the reference answer. Each task is evaluated as a binary outcome (success/failure), and the overall success rate is computed as the percentage of successfully completed tasks.

BrowseComp-ZH. This benchmark evaluates web browsing capabilities in Chinese using a similar protocol to BrowseComp (Zhou et al., 2025). We employ GPT-4o-2024-08-06 as the judge with prompts adapted for Chinese language evaluation. The judge verifies correctness through semantic matching with ground truth, accounting for acceptable variations in phrasing. Success rate is calculated as the percentage of correctly answered tasks.

WideSearch. This benchmark evaluates broad-scope web exploration using Item-F1 as the primary reported metric (Wong et al., 2025). For each query, the agent must retrieve a comprehensive set of relevant items (e.g., entities, facts, or documents). We

employ GPT-5 as the judge for semantic matching between retrieved items and reference items. The Item-F1 score measures the harmonic mean of precision and recall between the retrieved item set and the reference item set. Specifically, precision quantifies the fraction of retrieved items that are relevant, while recall measures the fraction of reference items successfully retrieved. We report the macro-averaged Item-F1 across all test queries.

GAIA. We evaluate GAIA by comparing the agent's final answer against gold-standard annotations. Rather than using the benchmark's canonical exact-match style scorer, we employ GPT-4o-2024-08-06 as the judging model to assess whether the agent's response semantically matches the ground truth, accounting for acceptable variations in phrasing while ensuring factual correctness. Success rate is calculated as the proportion of tasks where the agent's answer is deemed correct by the judge.

xbench-DeepResearch. This benchmark assesses deep research capabilities through complex multi-step information synthesis tasks (Chen et al., 2025). We utilize GPT-5 as the automatic evaluator. The judging process evaluates both the correctness and completeness of the agent's research report against reference solutions. The evaluation prompt follows the benchmark specification used in our implementation, which instructs the judge to assess whether key findings are accurately identified and properly supported by evidence. We report success rate as the percentage of tasks receiving a "correct" judgment.

WebWalkerQA. This benchmark evaluates web navigation and question-answering capabilities through realistic browsing scenarios (Wu et al., 2025). We employ GPT-4o-2024-08-06 as the judging model. The judge compares the agent's extracted answer against the ground-truth answer, determining whether the response correctly addresses the question based on information from the target web page. The evaluation prompt follows the original benchmark specification to ensure reproducibility. We report answer accuracy as the percentage of correctly answered tasks.

Evaluation Consistency. All judge-based evaluations use deterministic decoding to ensure reproducibility. For benchmarks employing LLM judges, we standardize on GPT-4o-2024-08-06 for most tasks and GPT-5 for complex evaluations requiring nuanced semantic understanding (WideSearch semantic matching, xbench-DeepResearch open-ended assessment). This standardization ensures consistent evaluation quality across benchmarks while preserving the integrity of each benchmark's evaluation philosophy.

Multiple Runs and Aggregation. All reported results, except for xbench-DeepResearch, represent averages over 3 independent runs with different random seeds to account for stochastic variation in agent behavior and tool execution. For each run, we compute the benchmark-specific metric (success rate or Item-F1), and the final reported score is the mean across the three runs. For xbench-DeepResearch, we report results from a single run.