MiniContext: Structured Hybrid Memory for Long-Horizon LLM Agents

Asanshay Gupta Charlotte Ka Yee Yan

Stanford University Department of Computer Science

The Problem: Context Pressure Breaks Agents

LLM agents in multi-step tasks must manage information that exceeds their effective context windows. As sequences grow, agents:

- **Contradict earlier decisions** when prior commitments fall out of context, leading to inconsistent behavior that undermines task completion
- **Repeat solved subtasks**, wasting compute by re-deriving conclusions the agent has already reached in earlier steps
- **Get "lost in the middle"**, forgetting critical facts needed for downstream reasoning

Scaling context length provides only partial relief. Longer windows delay but do not eliminate capacity limits. Models struggle to utilize extended context effectively, as attention degrades for information in the middle of long sequences [Liu et al., 2024], and compute costs scale poorly with length. The core issue is not capacity but *policy*: which information should persist, and in what form?

Why Existing Memory Systems Don't Solve This

Current memory architectures extend effective context through sophisticated storage, retrieval, and summarization mechanisms. However, they choose what to retain using proxy signals rather than reasoning about task requirements:

- **Similarity-based retrieval** surfaces information that matches the current query embedding, but misses facts that are relevant yet semantically distant
- **Recency-based retention** prioritizes recent context, discarding older commitments and constraints the agent must still honor
- **Task-agnostic summarization** compresses context without foresight, losing details that turn out to be critical dependencies for future steps

Related Work

Long-context limitations. Scaling context length yields diminishing returns: models exhibit degraded recall for information in the middle of long sequences [Liu et al., 2024] and struggle with retrieval as context grows.

Replay and error propagation. Errors in agentic tasks compound across steps [Xiong et al., 2025]. Retention failures early in a trajectory cascade into downstream errors, making the cost of poor policies visible.

Cognitive architectures for agents. Work on LLM agents draws on cognitive science models of memory, particularly the distinction between working memory, episodic memory, and procedural knowledge [Sumers et al., 2024].

Memory-augmented LLMs. Mem0 [Chhikara et al., 2025] extracts and consolidates facts with ADD/UPDATE/DELETE operations keyed to current salience. A-MEM [Xu et al., 2025] links memories as Zettelkasten-style notes; H-MEM [Sun & Zeng, 2025] adds hierarchical abstraction layers; Zep [Rasmussen et al., 2025] introduces bi-temporal knowledge graphs for temporal reasoning.

MiniContext

To effectively evaluate this memory system in a realistic environment, we place a model in an artificially context-limited environment. For example, although GPT-4o-mini has a 128k context by default, we test using a 8k context limit to effectively test the use of the memory system.

We implement 3 main strategies to make the best use of this context:

- **Agentic Context Engineering**: Before each step, the agent has the chance to edit the elements in its context, similar to Mem0. It can label each one KEEP, DROP, or SUMMARIZE, which allows it to keep relevant information.
- **Writing to memory**: We then run a second LLM call over the system to extract essential memories and store them in the appropriate buckets. All episodes are stored to episodic memory as is, and the agent selects procedures and facts for procedural and semantic memory.
- **Retrieval of relevant memories**: Finally, at each step we retrieve the top-k most relevant memories from each bucket. This allows us to augment the agent with only step-specific memories instead of generic context.

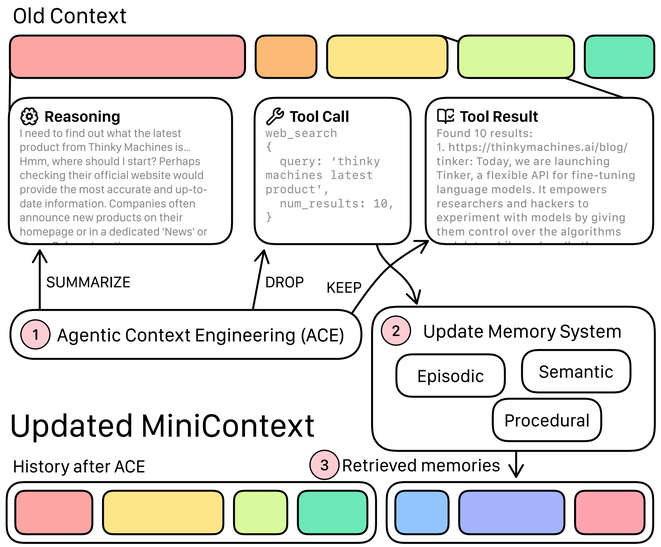

Figure 1. The MiniContext context management and memory system transforms an old bloated context into a seamless hybrid of history and memory.

Evaluation

We evaluate on **LongBench v2**, which consists of 503 multiple-choice questions over documents from 8K to 2M words, spanning dialogue tracking to code-repository analysis.

Why this benchmark? The extreme context lengths and high difficulty force models to identify and retain only what matters. Performance directly reflects how well an agent prunes, compresses, and retrieves under a strict token budget.

Results

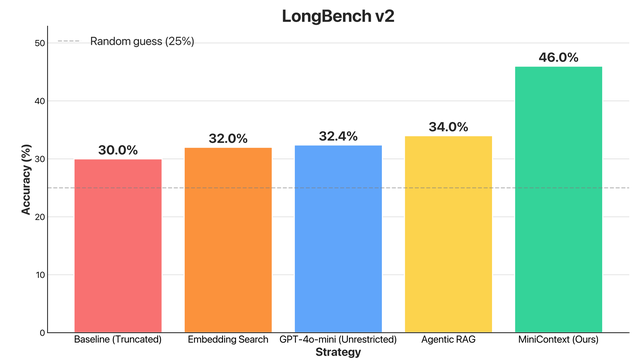

Figure 2. LongBench v2 Accuracy

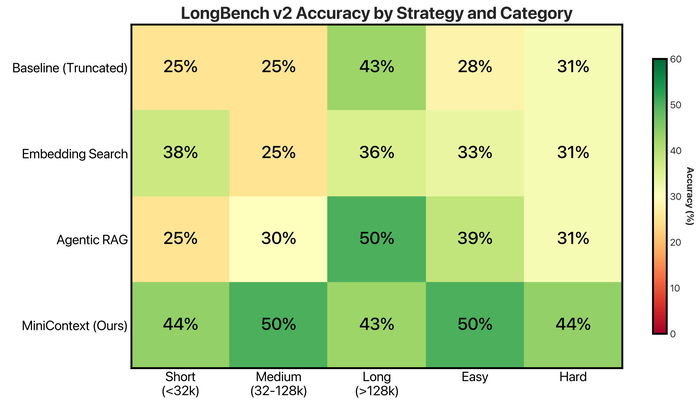

Figure 3. LongBench v2 performance across categories

Analysis and Conclusions

As can be seen in Figure 2, the MiniContext agent performs significantly better than SOTA methods like embedding search and Agentic RAG when given the same context limit. In fact, the hybrid memory system in MiniContext outperforms the base model when run with the full 128k context.

Performance gains for MiniContext are especially visible in hard tasks, where MiniContext achieves a significantly higher accuracy than other tasks due to the relevant retrieved memories. In extremely long context environments, Agentic RAG outperforms MiniContext, which seems to be because at the limited context, MiniContext fails to leave enough space for new information when summarizing.

MiniContext demonstrates that intelligent, agent-driven control over memory content offers a more scalable and robust alternative to simply enlarging context windows. By structuring what to preserve and how to retrieve it, LLM agents maintain coherence, reduce repeated deliberation, and improve downstream decision-making in long-horizon tasks.