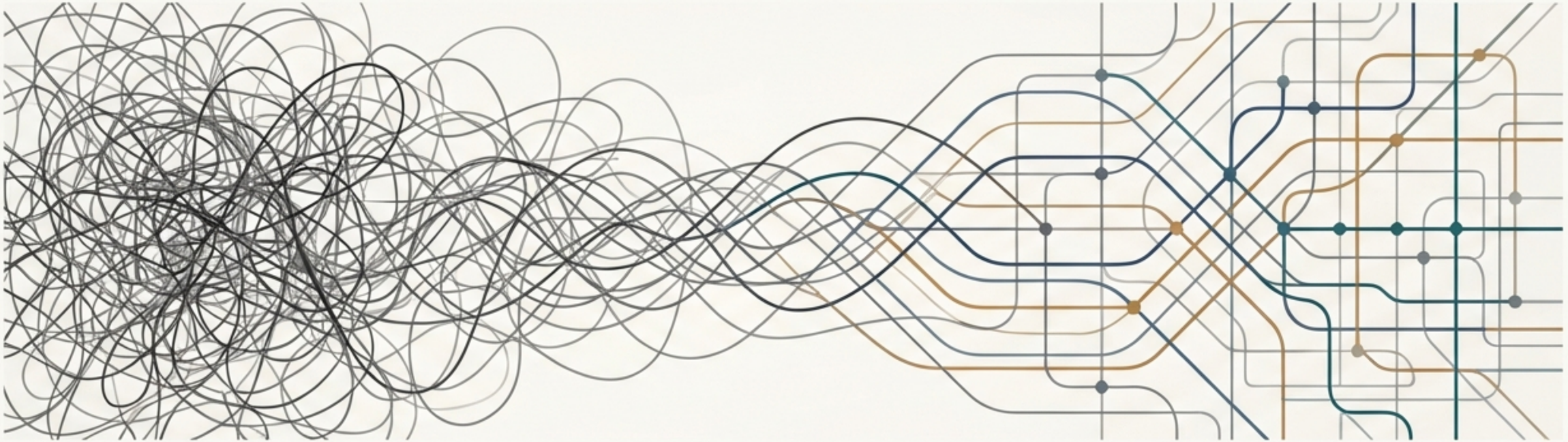


From Chaos to Clarity: A Unified Map for AI Agent Memory

A visual summary of the survey ‘Memory in the Age of AI Agents: Forms, Functions and Dynamics’

A summary of research from: National University of Singapore, Renmin University of China, Fudan University, Peking University, Nanyang Technological University, Oxford University, and others.

The Agent Memory Landscape is Expanding Rapidly—and Becoming Fragmented

As research on agent memory rapidly expands, the field has become increasingly fragmented. Works under the "agent memory" umbrella differ substantially in motivations, implementations, and assumptions.

The proliferation of loosely defined terminologies has obscured conceptual clarity, and traditional taxonomies like long/short-term memory are insufficient to capture the diversity of modern systems.

This highlights the urgent need for a coherent taxonomy that can unify emerging concepts.

To Bring Clarity, We Address Five Key Questions:

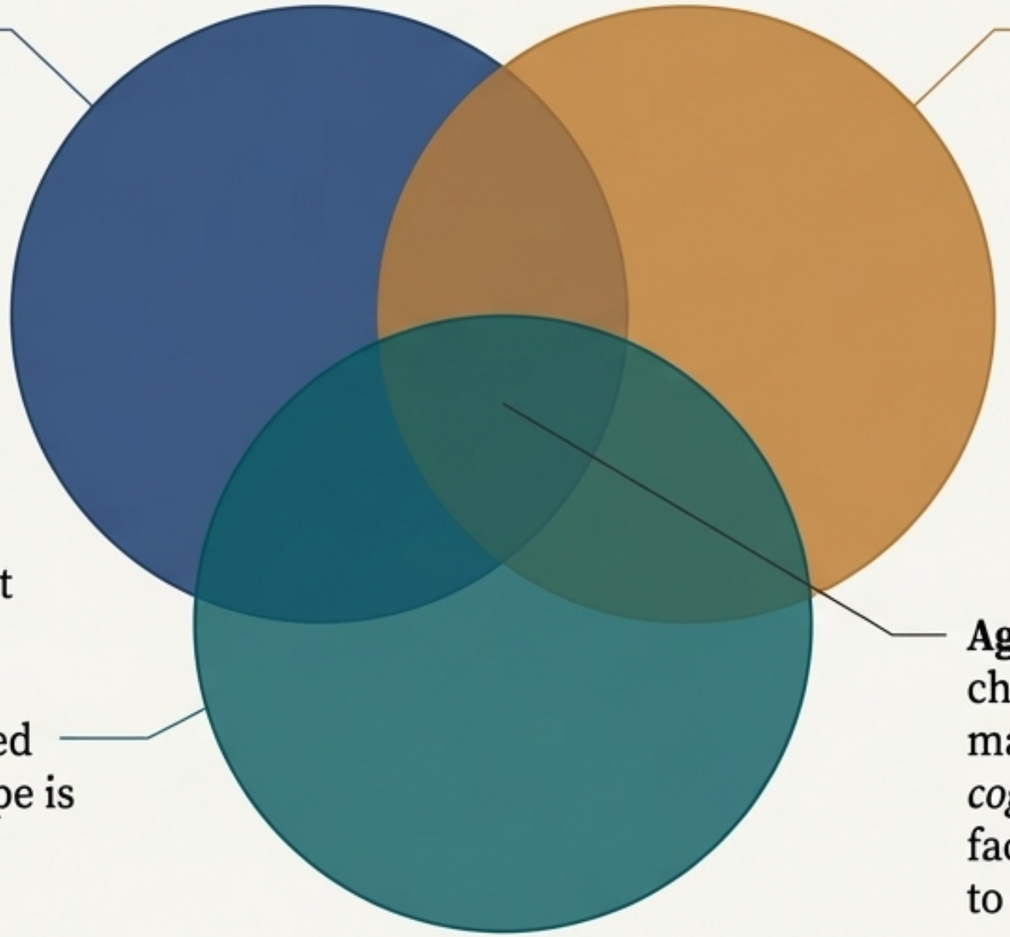
1. **Definition**: How is agent memory defined and distinguished from related concepts like RAG and LLM memory?
2. **Forms**: What architectural or representational forms can agent memory take?
3. **Functions**: Why is agent memory needed, and what roles does it serve?
4. **Dynamics**: How does agent memory operate, adapt, and evolve over time?
5. **Frontiers**: What are the most promising frontiers for advancing research?

First, Let's Define Our Territory

Agent Memory is distinct from, but related to, LLM Memory, RAG, and Context Engineering.

LLM Memory: Primarily concerns *internal model dynamics*, like KV cache management and long-context architectures. It focuses on the model's representational capacity, not an agent's evolving external state.

Context Engineering: A design methodology that treats the context window as a *constrained computational resource*. Agent memory is one component managed by context engineering, but its scope is broader, encompassing the agent's entire cognitive state.

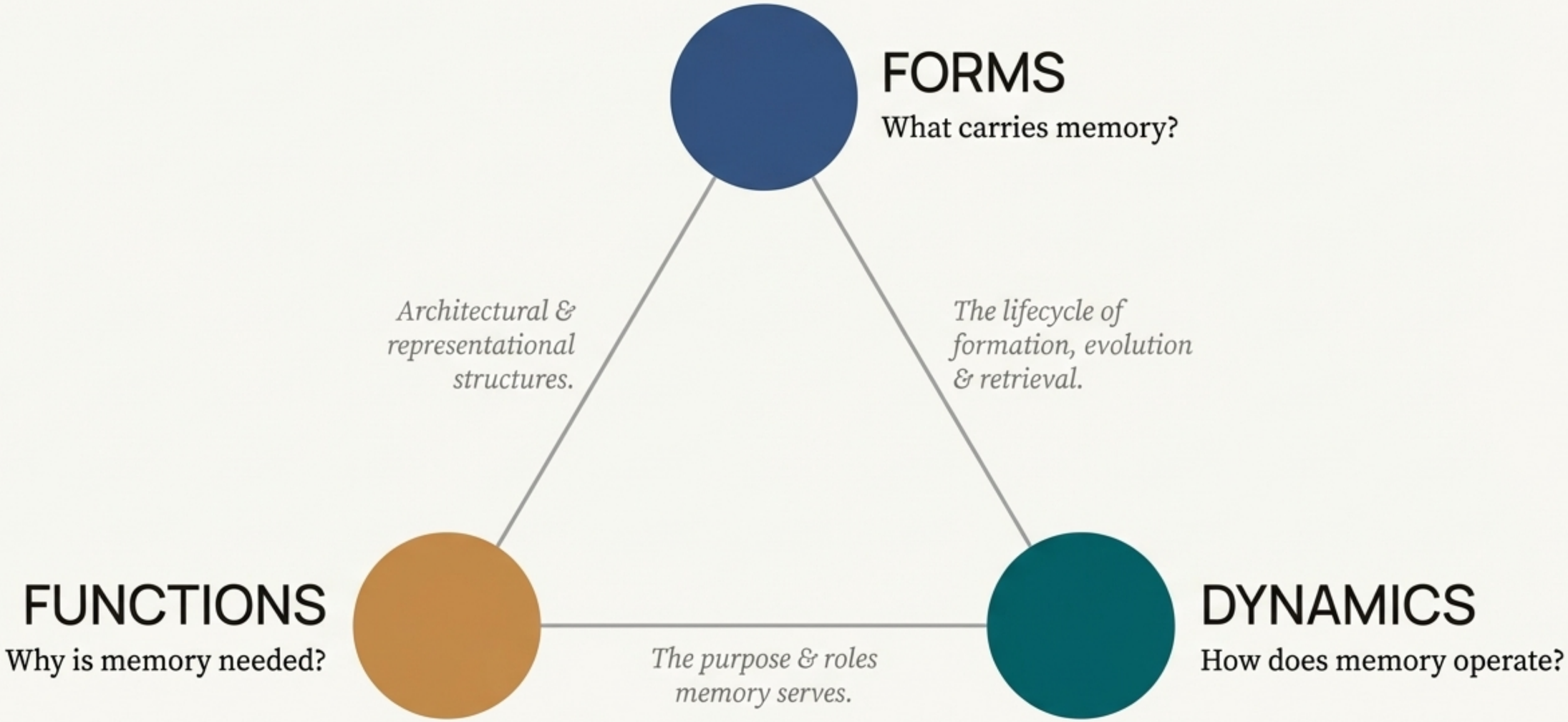


RAG (Retrieval-Augmented Generation): Historically focused on augmenting LLMs with *static, external knowledge sources* for a single task. The boundary is blurring, but agent memory is defined by its focus on a *persistent, self-evolving memory base* updated through interaction.

Agent Memory (The Core): Uniquely characterized by its focus on maintaining a *persistent and self-evolving cognitive state* that integrates both factual knowledge and lived experience to enable adaptation and autonomy.

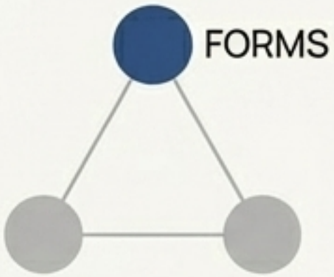
A Unified Framework: The Forms, Functions, and Dynamics of Memory

We can organize the complex landscape of agent memory research along three fundamental axes. This framework serves as our map for understanding not just what memory is made of, but why it's needed and how it operates.



Part I: The FORMS of Memory

What Carries Memory? Three Dominant Architectural Forms



Token-level Memory

Memory organized as explicit, discrete units (text, images, etc.) that are externally accessible and inspectable.

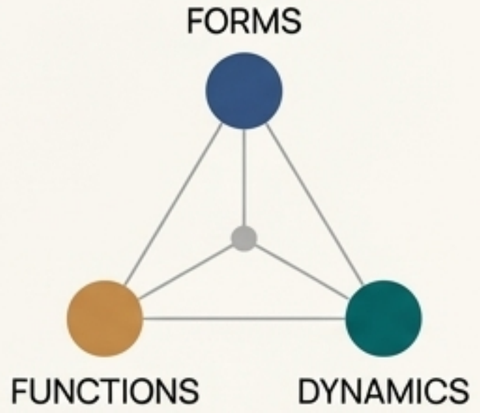
Parametric Memory

Memory stored implicitly within the model's parameters (weights and biases), accessed during forward computation.

Latent Memory

Memory represented in the model's internal hidden states or continuous representations (e.g., KV cache, embeddings).

Form 1: Token-level Memory is Organized by Structural Complexity

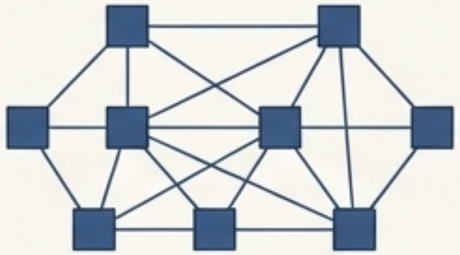


Token-level memory stores information as persistent, discrete units. The key differentiator is how these units are organized, which determines how efficiently an agent can search, update, or reason over them.

Flat Memory (1D)

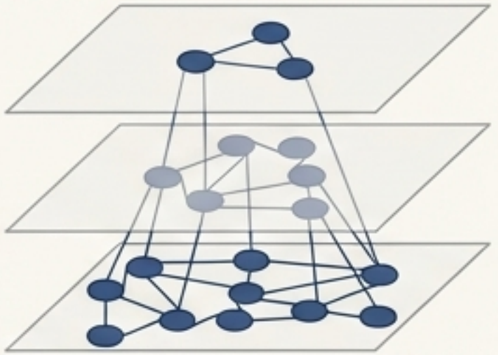
No explicit topology. Memories are stored as sequences or bags of units (e.g., dialogue logs, experience pools). Simple and scalable, but can accumulate noise.

Planar Memory (2D)

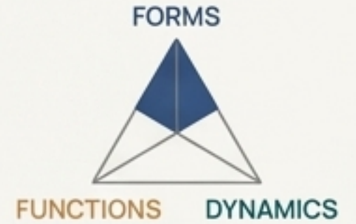


A single-layer structure like a graph or tree. Relationships are explicit but not layered. Encodes relationships, but can become a monolithic module.

Hierarchical Memory (3D)



Structured across multiple layers with inter-layer links. Supports different levels of abstraction. Enables deep abstraction, but complexity is a challenge.



Form 2: Parametric Memory Embeds Knowledge into the Model's Weights

Instead of external storage, parametric memory internalizes information by adjusting the statistical patterns of the model's parameter space.

Internal Parametric Memory

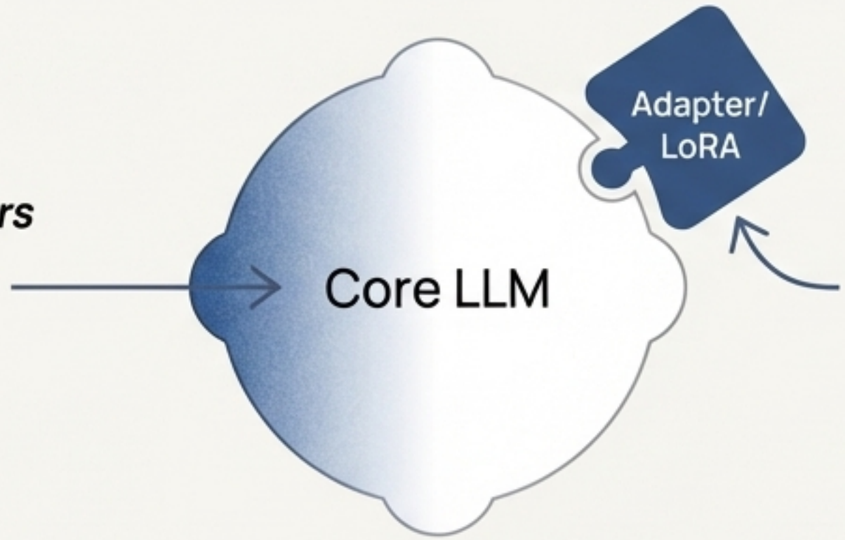
Encoded within the ***original parameters*** of the base model.

Pros

- No extra inference overhead.

Cons

- Difficult to update; requires costly retraining and is prone to catastrophic forgetting. Best for stable, stable, domain-level knowledge.



External Parametric Memory

Stored in ***auxiliary parameter sets*** like adapters or LoRA modules, leaving the base model frozen.

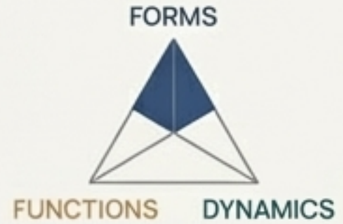
Pros

- Modular, reversible updates that avoid catastrophic forgetting.

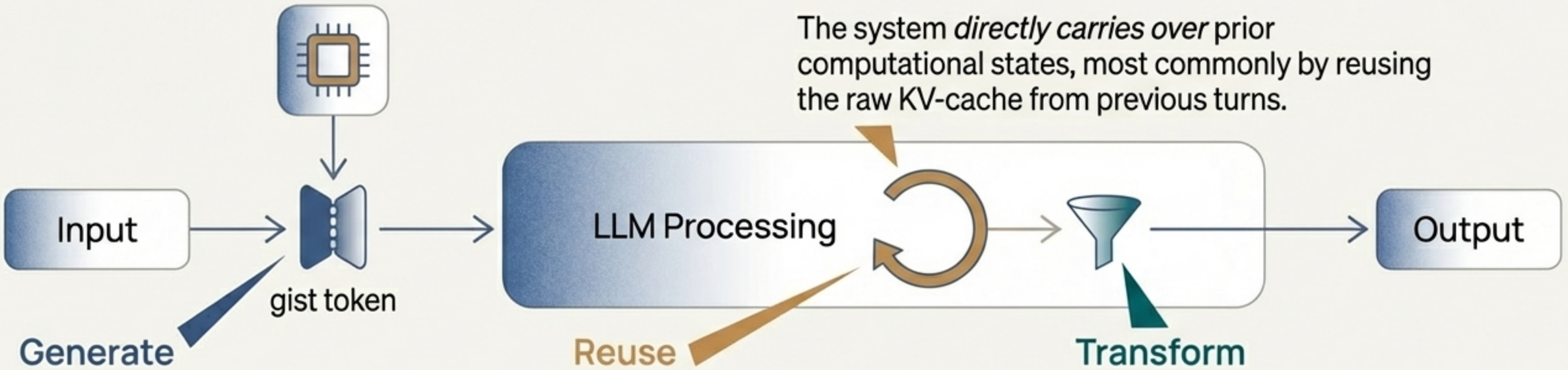
Cons

- Can add inference latency; effective-effectiveness depends on how well modules interface with the core model.

Form 3: Latent Memory is Carried in the Model's Internal Representations



Latent memory is not stored as readable tokens or dedicated parameters, but implicitly within the model's internal states (e.g., KV cache, hidden embeddings). This is machine-native and token-efficient.



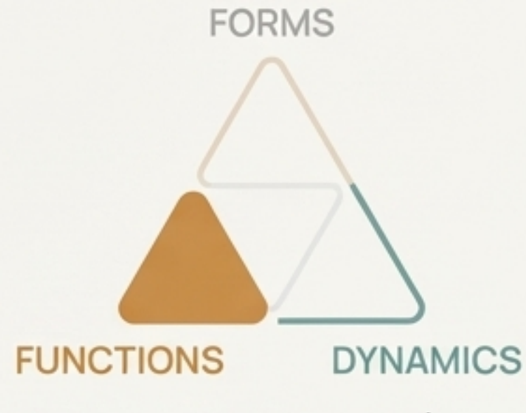
An auxiliary model or module *creates new* compact latent embeddings (e.g., "gist tokens") to summarize long contexts.

The system *directly carries over* prior computational states, most commonly by reusing the raw KV-cache from previous turns.

Existing latent states (like the KV cache) are *modified* through pruning, aggregation, or compression to create a more efficient memory representation.

Part II: The FUNCTIONS of Memory

Why Do Agents Need Memory? Three Functional Pillars



Memory is not a monolithic component but a set of distinct functional capabilities. We can classify these capabilities based on the core requirements of a persistent, intelligent agent.

Factual Memory

The agent's declarative knowledge base.

Answers: "What does the agent know?"

Experiential Memory

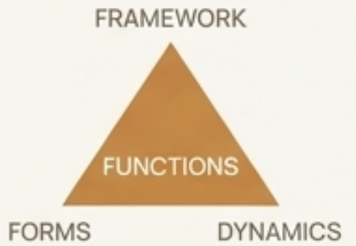
Procedural knowledge for continual learning.

Answers: "How does the agent improve?"

Working Memory

The active, in-session scratchpad.

Answers: "What is the agent thinking about now?"



Function 1: Factual Memory Provides a Declarative Knowledge Base

Factual memory allows an agent to store and retrieve explicit facts about past events, users, and the environment. This is the cornerstone for context-aware, personalized, and consistent behavior.

Key Roles of Factual Memory

Coherence: Recalls interaction history to maintain topical continuity in conversations.

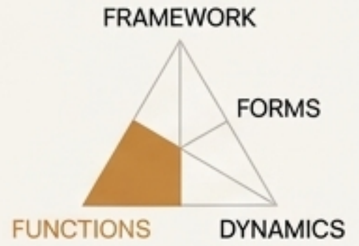
Consistency: Maintains a stable internal state to avoid contradictions in behavior and persona.

Adaptability: Personalizes responses based on stored user profiles and historical feedback.

Primary Sub-types

- **User Factual Memory:** Sustains consistency in human-agent interactions (e.g., user preferences, dialogue history).
- **Environment Factual Memory:** Sustains consistency with the external world (e.g., document states, tool capabilities, shared multi-agent knowledge).

Function 2: Experiential Memory Enables Continual Learning and Self-Evolution



This memory system allows agents to accumulate and transfer procedural knowledge across different episodes, enabling them to improve over time by learning from successes and failures. It is the foundation for a non-parametric path to adaptation.

Case-based Memory

Stores minimally processed records of past episodes (trajectories, solutions). Provides high-fidelity examples for imitation.

Strategy-based Memory

Distills transferable reasoning patterns, high-level insights, and workflows from raw cases. Guides planning and improves generalization.

Skill-based Memory

Encapsulates executable capabilities like code snippets, functions, or APIs. Operationalizes abstract strategies into verifiable actions.

Function 3: Working Memory is an Active, Controllable Workspace

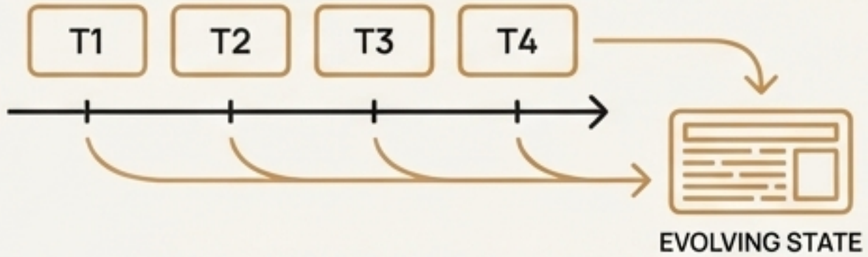
An LLM's context window is a passive buffer. Working Memory refers to the mechanisms for *active management and manipulation of context within a single episode*. It transforms the context window into a controllable scratchpad to increase information density and suppress noise.

Single-turn Working Memory

Manages massive *immediate* inputs (e.g., a long document).

Goal: Condense and abstract the input to maximize the information payload for a single forward pass.

Techniques: Input Condensation (token pruning) & Observation Abstraction (structuring raw data).



Multi-turn Working Memory

Manages *accumulating* history over a sequence of interactions.

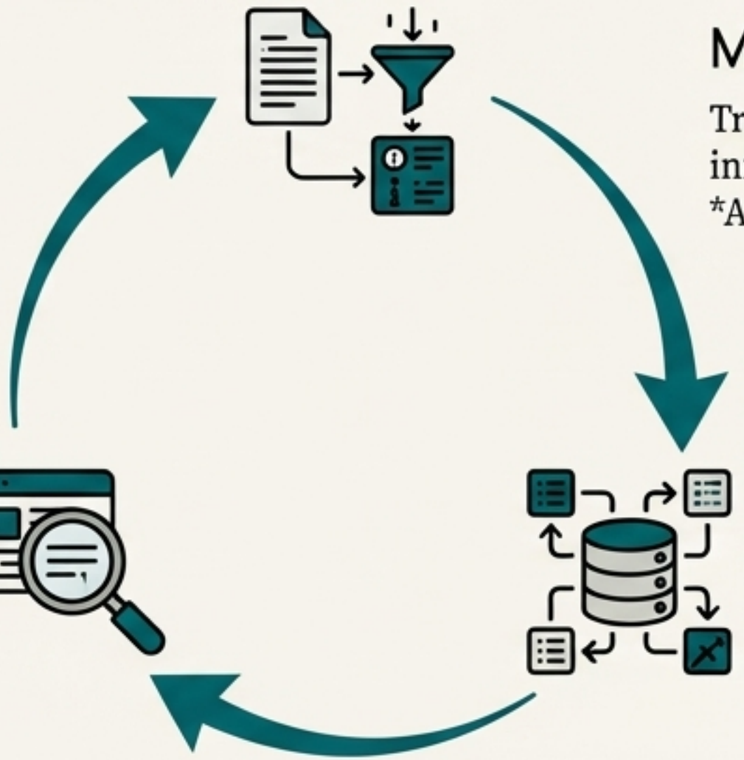
Goal: Maintain task state and goals without being overwhelmed by history.

Techniques: State Consolidation (rolling summaries) & Hierarchical Folding (compressing completed sub-tasks).

Part III: The DYNAMICS of Memory

How Memory Operates and Evolves: A Three-Stage Lifecycle

Agentic memory is not static; it is a dynamic system that continuously adapts. This lifecycle shows how agents transform raw experience into refined knowledge and use it to inform action.



Memory Formation

Transforms raw experience into compact, information-dense memory units.
*Answers: 'How to extract memory?'

Memory Evolution

Integrates new memories into the existing base through consolidation, updating, and forgetting.
*Answers: 'How to refine memory?'

Memory Retrieval

Constructs context-aware queries to access the right information at the right time.
*Answers: 'How to utilize memory?'

The Horizon: Emerging Frontiers in Agent Memory Research

The ‘Forms-Functions-Dynamics’ framework provides a map of current research, but also points to uncharted territories. Key emerging frontiers include:

- **Automation-Oriented Memory Design:** Developing memory systems that can autonomously structure, maintain, and optimize themselves with minimal human intervention.

- **Deep Integration of Reinforcement Learning:** Moving beyond using RL for simple retrieval policies to deeply integrating it with memory formation and evolution, allowing agents to learn *what* to remember.

- **Multimodal Memory:** Building unified memory architectures that can seamlessly store, relate, and reason over information from text, images, audio, and other modalities.

- **Shared Memory for Multi-Agent Systems:** Designing robust protocols and structures for teams of agents to build and leverage a collective memory, enabling more complex collaboration.

- **Trustworthiness & Safety:** Addressing critical issues like memory poisoning, privacy leaks in stored information, and ensuring the reliability and verifiability of an agent's knowledge base.

Resources for Your Expedition

To support empirical research and practical development, we have compiled a comprehensive collection of resources.

Benchmarks

Representative benchmarks for evaluating memory systems:

- Long-context Dialogue (e.g., LoCoMo, LongMemEval)
- Complex Problem Solving (e.g., GAIA, XBench)
- Code-centric Tasks (e.g., SWE-bench Verified)
- Lifelong Learning (e.g., StreamBench)

Open-Source Frameworks

Key frameworks for building agents with memory:

- Memary
- MemOS
- Mem0
- Zep

Explore the full paper and a curated list of resources:

Github: https://github.com/Shichun-Liu/Agent-Memory-Paper-List