ChatGPT

Always-On LLM Chat with Human-Like Memory and Forgetting

Inspiration from Human Memory Processes

Designing an "always-on" conversational AI involves emulating key behaviors of human memory. In human brains, **short-term experiences are selectively consolidated into long-term memory** during periods of rest (especially sleep) [1]. The hippocampus repeatedly "replays" recent events in dialogue with the cortex at night, stabilizing important memories for long-term storage [1]. Meanwhile, unimportant details are pruned through processes like **synaptic downscaling**, which globally weaken seldom-used neural connections during sleep [2] [3]. This **"use it or lose it"** principle ensures that frequently reactivated memories ("strong" synapses) are retained while noisy or infrequent ones are weakened [2] [3]. The net effect is improved signal-to-noise ratio in memory, extraction of the "gist" of experiences, and integration of new memories into the knowledge network [3]. Human memory also tags certain experiences as *salient* or *emotional*, leading to stronger recall for those events. For example, encountering high-value or emotionally charged information triggers dopamine-reward circuits (nucleus accumbens, VTA) that **up-regulate hippocampal encoding**, yielding better retention of those important items [4]. In practice, people naturally **prioritize remembering valuable or meaningful events** and gradually forget trivial ones [4]. These neuroscience insights – nightly consolidation, pruning of unused memories, and salience-based retention – inspire the memory architecture for an always-on LLM assistant. The goal is for the AI to **"sleep" and consolidate** like a human: periodically review recent conversations, distill the important pieces into long-term memory, and let irrelevant details fade. Memories that are revisited often (or associated with strong emotion) should become more permanent, whereas idle or low-value memories should decay in influence (but never be *completely* irretrievable, akin to how a specific cue can revive a "forgotten" human memory [5]).

Continuous Conversation Without Resets

In a seamless always-on chat, the user should not feel conversation "threads" restarting. The system must **maintain context continuously** – potentially over days or months – without exhausting the model's fixed input window. Modern LLMs have finite context (e.g. 8K, 128K tokens), so a naive approach of piling the entire history will eventually fail. One straightforward solution is larger context windows (indeed, models like GPT-4 Turbo offer up to 128K or more tokens [6] [7], and new systems boast up to millions of tokens). **Large contexts improve coherence** by allowing the model to directly attend to long conversation histories [8]. However, massive context windows carry **performance and cost trade-offs**: processing longer prompts requires more memory and computation, increasing latency and expense [9]. Even with 100K+ tokens, an "always-on" chat spanning weeks might exceed limits, so complementary strategies are needed. The user experience demands **no visible lag or interruptions** during memory management, and the conversation flow should remain natural and immersive. This means any "forgetting" or summarizing should happen incrementally or asynchronously. For example, the assistant might perform background consolidation during idle moments (analogous to nightly sleep) to avoid pausing mid-chat. The AI's responses should reflect human-like continuity: recalling earlier topics when relevant, but *not* awkwardly

dredging up every past detail. This balance – remembering the important, letting go of the irrelevant – underpins a realistic and emotionally engaging conversation. In practical terms, the system must decide **what to remember, how much to remember, and when to forget** [10]. Key considerations include: How to keep the most relevant context at hand? How to avoid context dilution or factual drift over long dialogues? And how to ensure the model doesn't "forget" critical facts (e.g. the user's name or a recent emotional event) that a human friend would remember. We next explore memory management strategies to achieve these goals.

Memory Management Strategies for Long Conversations

To maintain an indefinite conversation, the AI needs mechanisms to **prune, compress, or externalize** memory. Several approaches (often combined) have emerged:

- **Sliding Context Window (FIFO Memory):** The simplest strategy is to always feed the model only the *N* most recent messages, dropping older ones as new messages come in. This *first-in-first-out* approach ensures the prompt stays within token limits and focuses on recent context. Its advantage is conceptual simplicity and relevance to the latest turn (recent messages are likely most pertinent). However, important older details can vanish entirely from the model's awareness. A purely sliding window can cause the AI to **"forget" earlier plot points or user preferences** once they scroll out of the window [11]. This is akin to human short-term memory with no long-term storage – efficient for immediate context but poor at global continuity.

- **Episodic Summarization (Rolling Summaries):** To avoid complete loss of older context, many systems employ **incremental summarization**. As the conversation grows, the AI periodically generates a concise summary of earlier dialogue and uses that summary in place of raw transcripts. For example, after every 50 messages, it might compress them into a few paragraphs that capture key facts and emotional highlights, and only keep that summary plus recent messages going forward. This dramatically reduces tokens while **preserving the gist** of past interactions. In fact, one evaluation found that using incremental summaries allowed a 4K-token model to match the performance of a long-context model on extended conversations [12]. The **pro** is that the assistant retains long-term continuity (it "knows" what happened in previous sessions, in broad strokes). This mirrors how humans remember narratives: we retain core story arcs and significant moments rather than every utterance. But the **cons** include potential *loss of nuance or errors* introduced by the model's summary. Over multiple re-summarizations, there is a risk of the "telephone game" effect where details become distorted. There is also a trade-off with *emotionally rich content* – a terse summary might omit subtle emotional cues that were present originally. Careful prompt engineering can mitigate this (e.g. instruct the model to carry over the user's expressed feelings or important decisions into the summary).

- **Retrieval-Augmented Memory (Vector Databases):** A powerful approach is to store conversation history in an **external memory store** (e.g. a vector database or knowledge base) and fetch relevant pieces on the fly. Instead of pushing the entire history into the prompt, the system **indexes past dialogues as embeddings** and **retrieves only the most relevant past snippets** given the current conversation turn [13] [14]. This is analogous to human associative recall, where a cue or topic will remind you of a past experience. Technically, it works by chunking the dialogue into segments (e.g. each user query/assistant answer pair, or each topic episode) and computing an embedding for each. When a new user message arrives, the system finds which stored embeddings are semantically

closest or contextually relevant to that message, and injects those memory snippets into the prompt (often prefaced as "Recall:" or similar). For instance, if months ago the user mentioned their dog's name and now they ask about buying dog food, the retrieval system should pull up the snippet where they said "I have a dog named Fido." This **selective memory** approach means the AI can tap into *arbitrarily long histories* without increasing the prompt size, since irrelevant content stays out of the prompt [14] [15]. The **benefit** is scalability and pertinence: the AI "remembers" everything (in storage) but only **thinks about what's relevant** at any moment [16]. It also offloads memory to an external system, freeing the LLM to focus on reasoning in the local context [17]. However, there are several **trade-offs**: the quality of this approach hinges on the embedding retrieval accuracy. If the similarity search fails to retrieve a crucial fact (perhaps because it was phrased very differently), the model may act as if it forgot. Also, maintaining a large vector database has engineering overhead – one must manage updating it with new interactions and possibly *pruning it as it grows*. Another consideration is *latency*: vector search is usually fast (milliseconds) but in a highly interactive setting it adds an extra step. Fortunately, optimizations and caches can keep this overhead minimal, enabling no noticeable lag to the user.

- **Hybrid & Hierarchical Memory Systems:** In practice, the best solutions combine the above methods to balance their strengths. For example, the **Kindroid AI companion** uses a multi-tier memory: a *persistent memory* (always-present facts like backstory or profile), a *cascaded short-term memory* for recent conversation, and a *retrievable long-term memory* for older interactions [18] [19]. The cascaded memory is essentially an intelligent sliding window – it **extends the recent context window by hierarchically organizing messages based on significance**, so that a kindroid can recall "hundreds or thousands of messages prior" if needed [20] [21]. The core principle *"mirrors human memory patterns"*: the AI remembers **recent and emotionally exceptional events more vividly** than mundane or distant ones [22]. Meanwhile, truly long-term memory is handled by consolidation: the system periodically distills older chats into durable "journal entries" or knowledge records [23] [24]. These long-term memories are stored indefinitely (analogous to a person's lifetime memories), but are only *retrieved when relevant* (often via embedding relevance or key-phrase triggers) [25] [26]. Crucially, the long-term store is infinite in capacity, but the recall from it is *probabilistic* or "less reliable" than short-term memory – an intentional design reflecting that if something hasn't been brought up in a while, the AI might need a strong cue to recall it [27] [28]. This creates a **gradual forgetting curve** rather than a hard cut-off. Academic proposals echo this multi-scale approach. For instance, researchers have suggested explicitly separating **episodic memory (specific recent events)** and **semantic memory (general world knowledge or stable facts)** in LLM-based agents [29] [30]. In one study, agents with a composite episodic+semantic memory outperformed those with only one type, especially by using pre-trained semantic memory for stable knowledge and episodic memory for recent context [30] [31]. The combination also requires a strategy to decide **which memories to forget when memory is full and which to retrieve when responding**, as described by Kim et al. (2022) [30] [31]. Their finding was that an agent emulating both human working memory and long-term memory is more robust, provided it can intelligently prune and recall. Techniques like **MemoryBank** (Wang et al., 2023) explicitly implement a *memory strength* parameter: each conversation turn is encoded into a vector and stored, and each time a memory is actually recalled (used by the agent) its strength score increases [32]. Less-used memories naturally fade in importance, whereas frequently recalled ones persist (simulating the **spacing effect** in human memory where recall solidifies memory) [32]. We see a common theme: **recency, frequency of use, and significance** are key metrics for an AI to manage its memories.

- **Extending Model Architecture:** A more experimental avenue is modifying the model architecture itself to handle long-term memory. For example, researchers have proposed adding dedicated **memory modules or memory layers** to LLMs that explicitly store information outside of the normal context window [33] [34]. These can act as a differentiable cache or an appended RNN-like state that carries information forward indefinitely. Others have explored transformer variants that handle very long sequences (e.g. Transformer-XL, Reformer, or recent architectures like **LongNet** capable of billion-token contexts [35]). Some of these approaches effectively give the model an **"internal" long-term memory** beyond the usual self-attention window. While promising, such techniques can be complex to implement and are still emerging – they might be more relevant in future always-on chat systems. For now, most practical systems rely on the external strategies above (summarization, retrieval, etc.), since they can be bolted on to existing LLMs without retraining.

Pros & Cons Summary: Each strategy has upsides and downsides. A *sliding window* ensures immediacy and low overhead but loses global context. *Summarization* keeps the dialogue history compact and maintains narrative continuity [36], but risks omitting details or emotional subtleties. *Embedding retrieval* scales to arbitrarily long histories with high relevance [14], yet depends on embedding quality and adds system complexity. A *hybrid system* (sliding + summaries + retrieval) tends to offer the best of both: for example, one can keep a short recent window for conversational coherence, a summary for each past "episode" to maintain general context, and a vector store for precise facts or callbacks. This mirrors how humans have a working memory, a narrative memory of past events, and a precise recall for salient facts when cued.

Human-Like Forgetting and Salience Tagging

A distinguishing feature of the desired system is **automatic forgetting with human realism** – not simply dropping data arbitrarily, but mimicking the patterns of human forgetting. In the human brain, forgetting is *selective*: we retain highlights and emotionally charged memories far longer than trivial ones. The AI can imitate this by assigning an **"importance" or salience score** to memories. In Stanford's *Generative Agents* framework, every memory (an event in the agent's life) gets an importance rating from 1 to 10, where e.g. "brushing teeth" might be 1 (mundane) and "a close friend's breakup" might be 9 or 10 (poignant) [37]. They implemented this by prompting an LLM to judge the poignancy of each new event, and any event above a certain threshold was considered significant [38] [39]. These importance scores directly influence retention: highly important memories are both **more likely to be retrieved** and also prompt the agent to reflect on them (thus consolidating them). Less important ones may be quickly forgotten unless repeatedly brought up. The Generative Agents system combined **recency, importance, and relevance** in a weighted retrieval function [40]. At query time, it scored all past memories by a mix of how recent they were, how important the agent deemed them, and how semantically relevant they were to the current situation, then retrieved the top few [40]. This ensured that, for instance, an agent mostly remembers recent happenings *unless* an older memory has very high importance or direct relevance. Such a mechanism maps well to a companion chatbot: your AI friend might not recall every casual remark from weeks ago, but it will recall *meaningful events* (e.g. your graduation, a personal story you shared) especially if they relate to the current conversation. This creates an **illusion of "emotional" memory** – the AI remembers what *matters*. It aligns with neuroscience: as discussed, humans exhibit better recall for high-value or emotional items due to reward circuit engagement [4]. By tagging chat messages with sentiment or intensity (using sentiment analysis or explicit user feedback), the system can boost the longevity of those messages in memory. For example, if the user had an emotional rant one day, the model's nightly consolidation could mark that summary with a high emotional weight so that weeks later, if the user touches on that subject, the AI can respond with appropriate recollection and empathy.

Nighttime Consolidation Analogy: To implement *nighttime-like memory sorting*, the system could perform a **daily maintenance cycle**. Suppose the user is mostly active during the day; at night, the AI could autonomously summarize that day's conversation, extract key facts or unresolved concerns, and store them into the long-term memory store. It could also run a "cleanup" routine where it goes through recent transient memory and **decays the importance of interactions that neither recurred nor were marked important**. This concept is hinted at in research – Hou et al. (2024) model a **recall probability that decays over time**, but never hits zero [41] [5]. In their model, if a memory is not recalled for a long time, its recall probability drops (simulating forgetting), but if a suitable cue appears, the memory can resurface [5]. They also increased the decay rate for memories that were rarely recalled, versus slower forgetting for frequently recalled ones [41] [42]. In practice, an AI could implement this by lowering the retrieval score of unused memories as time passes, unless periodically reinforced. Additionally, they propose a **sigmoid "memory consolidation" function that increases with each recall but plateaus** – meaning recalling something many times yields diminishing returns in strengthening it [42] [43] (again, similar to human memory which improves a lot with the first few reviews of information and then levels off).

Synaptic Pruning Analogy: The AI's vector database or memory log could also be pruned akin to synaptic pruning. If certain old conversation chunks never get retrieved in months, the system might archive or delete them to save space (or perhaps compress them further). The *Synaptic Homeostasis Hypothesis* suggests that during sleep, the brain uniformly downscales synaptic strengths that grew during wake, which preferentially **removes the weakest, least-used connections** [2] [3]. Translated to AI, one could periodically downscale all memory importance scores, which naturally causes low-scoring items to fall below a threshold and be dropped, while items that have been "strengthened" through repetition remain above threshold. Simulations of this process in neural networks have shown it **improves signal-to-noise and encourages gist extraction** just as in biological brains [3].

Emotional Continuity: A challenging aspect is capturing *emotional context* over long spans. A human conversational partner implicitly remembers the other person's recent moods and concerns (even if the exact words are forgotten). An always-on chatbot can maintain an **"affective memory"** by storing metadata about each session (e.g. "user was sad about work on Jan 5" or sentiment scores over time). Then, if a related topic arises, the AI can recall **the user's emotional state**. This can be implemented via long-term memory notes (e.g. a vector that encodes not just the facts discussed but the emotional tone). Some systems allow attaching sentiment tags to memory entries to bias retrieval – e.g. if the user is currently sounding distressed, the AI might more readily retrieve past moments of distress to avoid repeating the same consolations or to follow up on them. While academic literature on **affective LLM memory** is still nascent, it aligns with the idea of **"memory consolidation with emotional significance"** suggested by Hou et al. – they note that incorporating emotional significance into the consolidation criteria could enhance the system's memory model [44]. In short, the AI should not only remember *facts*, but also *feelings*, mimicking human friendship.

Feasibility and Implementation Pathways

Building an always-on LLM chat with these properties is **feasible today** by combining existing techniques. Recent research prototypes (like Generative Agents [45] [46], MemoryBank, and others) and even commercial AI companions [19] [47] have demonstrated key pieces: long-term vector memory, memory

consolidation algorithms, and realistic forgetting curves. The architecture would roughly consist of the following components working in unison:

1. **Persistent Storage of Dialogue:** All user and AI utterances (or their important portions) are stored in a database with timestamps and metadata (e.g. embedding, sentiment, importance score). This acts as the AI's "experience repository" – analogous to a hippocampus recording experiences in detail.

2. **Short-Term Context Buffer:** For each new user query, the system includes a window of the most recent conversation turns in the LLM prompt (e.g. the last few exchanges that fit in, say, 1-2K tokens). This ensures immediate context and conversational continuity (the AI responds with awareness of what was just said). The short-term buffer corresponds to working memory.

3. **Dynamic Retrieval of Long-Term Memories:** In parallel, the system performs a **memory retrieval query** against the stored dialogue repository. This could be a semantic search using the embedding of the current user query to find relevant past mentions [14]. The top-k relevant memories (which might be older conversations, facts the user shared, or summary notes) are fetched. These are then appended to the prompt in a special section (e.g. "Here are things you remember:"). Crucially, a **relevance threshold or probability** can be used so that recall only triggers when clearly useful [48] [49] – avoiding injecting random off-topic old info. Hou et al. implement a recall trigger based on a probability that increases with relevance and decreases with time since last recall [48] [49]. In practice, this means the AI might not always tap long-term memory every single turn (to save tokens and avoid confusion), but when the conversation hits a cue strongly linked to past content, the gate opens.

4. **Memory Consolidation Jobs:** At defined intervals, the system runs a consolidation process. One trigger could be time (e.g. daily at 2 AM the user's time, or after X hours of inactivity). Another trigger could be the conversation length – if the short-term buffer is getting full or the vector DB has grown by N new entries, initiate consolidation. Consolidation involves:

5. Summarizing recent interactions into summary records. For example, produce a summary of "Today's chat on 2025-06-16," highlighting any new facts learned about the user or decisions made. This summary is saved as a long-term memory entry (with its own embedding). It might also replace the full log of today's chat in the vector index (to keep the index size bounded).
6. **Updating importance scores:** Go through new memory entries and assign importance (via an LLM prompt or heuristics). If any exceed a high threshold, ensure they are flagged to always retrieve (or even consider storing them in a fixed profile). If some are very low importance and also old, consider them for pruning.
7. **Pruning/Archiving:** Delete or archive raw memory entries that have been superseded by summaries or deemed unimportant. For instance, if the AI has already summarized last week's chatter, the detailed logs of each trivial exchange from last week can be removed (perhaps retaining only the summary and any high-importance items from that period).
8. **Re-indexing**: Refresh the vector index if needed – e.g. remove vectors for pruned items, add vectors for new summaries. This ensures search remains efficient and focused on consolidated memories.

This consolidation step is analogous to the brain's nighttime activity – it's effectively the AI "sleeping" on its experiences, discarding clutter, and organizing significant memories for future retrieval.

1. **Long-Term Profiles/Knowledge Base:** In addition to episodic chat memories, an always-on assistant can maintain a structured profile of the user. This might include factual data (name, birthday, family members, favorite things) and longer-term patterns (e.g. "User often stressed on Mondays" or "Has been learning guitar since March"). Some of these can be *derived* from the chat history via reflection. In Generative Agents, the authors introduced a **reflection phase** where, after accumulating enough observations, the agent asks itself questions to infer higher-level conclusions [46] [50]. For example, seeing multiple mentions of going to the library, the agent might reflect "I enjoy reading and spend a lot of time at the library." These reflections then become part of the agent's memory, allowing more abstract understanding of the user or itself [50]. Similarly, our assistant might reflect: "The user has mentioned being anxious about work on several occasions – this is a recurring theme." Storing this as a long-term semantic memory means the AI can respond with deeper empathy ("I recall you often feel anxious about work – how are things going there lately?"). The technical pathway for this is using the LLM itself (in a different mode) to analyze logs and generate such insights. This could be done during consolidation (perhaps for events with high cumulative importance). The insight is then saved (likely with an importance tag, as it's a distilled truth about the user).

2. **Real-time Operation and Optimization:** Throughout the chat, careful engineering is needed to keep things running smoothly. The system should minimize any delay from memory retrieval or updates. In practice, retrieving a few top memories via a vector search is very fast (especially with optimized libraries or approximate nearest neighbor search). Summarization is more expensive since it requires an LLM call, but doing it during idle times or after a session can hide the cost from the user. If the user is continuously chatting, one could even use a second, low-priority background LLM worker to summarize older content while the main model handles live prompts – the summary result can then be injected on the next opportunity. This way, **there are no visible lags or pauses** in conversation flow, satisfying the seamless UX goal.

Performance Considerations: The multi-tier memory architecture does introduce overhead in terms of storage and computations, but these are manageable. Vector databases can handle millions of embeddings; keeping a long chat log indexed is not an issue (a year of dialogue might be on the order of tens of thousands of messages, which is trivial for modern indexes). Periodic summarization actually keeps the long-term store efficient, since it reduces many messages into one chunk. From a latency perspective, experiments combining sliding windows with summarization have shown it **maintains continuity without exceeding capacity** [16] – essentially by keeping relevant recent content and a summary for older content, an LLM can generate coherent responses while staying within token limits [16]. The memory retrieval mechanism can even *improve* response quality in long dialogues: by surfacing *only* the relevant bits, it prevents the model from being distracted or overwhelmed by irrelevant older context [51]. One user-observed issue with vanilla ChatGPT is *context dilution* – the model "forgets" key points in a long conversation while remembering some inconsequential detail, likely due to how the compression or attention works in a long prompt [52]. An intelligent retrieval approach can mitigate this by explicitly ensuring the key points are present when needed (because they are tagged as important and retrieved).

Comparative Trade-offs: It's worth noting that there's a continuum between doing everything within the model versus leveraging external systems. Expanding context windows (or training the model with a recurrent memory) tries to solve the problem internally, but at high compute cost. The external memory +

retrieval approach solves it by systems engineering, which is often more feasible to deploy (you don't need to train a new giant model; you can use an existing model and add a memory module around it). The trade-off is that the external approach is less "end-to-end optimal" – the model doesn't natively learn what to forget; we enforce it. But this trade-off is acceptable for now, given the substantial benefits in keeping long-term context. In fact, many believe that **LLM applications need an architecture around the core model** to handle extended interactions [53] [51]. The memory system we describe is a prime example of such architecture.

Conclusion and Future Outlook

In summary, an always-on LLM chat system with continuous context and human-like forgetting is well within reach by combining known techniques: **sliding context windows for recent dialogue, automatic summarization for older context, and embedding-based long-term memory retrieval** for important facts and events. These components echo human memory divisions (working memory vs. long-term memory) and processes (consolidation, selective recall). By weighting memories with factors like recency and importance [40], the system can **prioritize salient, emotionally charged interactions** – giving the user an experience of talking to a partner that truly "remembers" what matters to them. At the same time, the built-in forgetting ensures the conversation doesn't become unnaturally cluttered by bygone minutiae, which also contributes to *realism*: humans forget minor details, and an AI that clings to every word can seem robotic. Research prototypes have validated the effectiveness of these ideas. For example, *Generative Agents* showed that an agent with a **memory retrieval model (relevance + recency + importance)** and periodic reflection produces far more coherent long-term behavior than one without [45] [46]. Another study integrated a **"remember to remember" cueing strategy**, enabling an LLM agent to autonomously recall needed past information when contextually appropriate [54] [55]. This led to more contextually relevant and coherent dialogues in user tests [55] [56]. Such results underscore that adding a cognitive-inspired memory layer can significantly enhance an LLM's conversational abilities.

Moving forward, we may see further **neuroscience-inspired improvements**. For instance, one could incorporate a **spaced repetition schedule** for the AI's own memory: having it *rehearse* important user facts at intervals (perhaps during its "offline" time) to reinforce them, just as humans benefit from revisiting information over time. There are also interesting possibilities in **metacognitive control** – e.g. the AI deciding *not* to bring up an old memory unless it's confident it's relevant (preventing non sequiturs, which humans also avoid through intuition). Additionally, **ethical and privacy considerations** will guide how forgetting is implemented – in some cases, users might *want* the AI to forget certain things (the digital equivalent of letting something go). Designing the forgetting mechanism with a degree of user control or transparency might be important for trust. On the technical side, as LLM context windows inevitably grow and new architectures arise, the balance may shift: we might not need as much summarization if we can directly handle very long transcripts. But the fundamental benefit of **structured memory (with tagging, salience, and purposeful recall)** will remain. After all, humans don't remember by scrolling through an exact log of their life; we remember through stories, important moments, and learned knowledge – and that is exactly what our always-on chat system aims to emulate.

Sources: The concepts discussed are supported by recent research and implementations. Park et al. (2023) demonstrate the memory/retrieval/reflection architecture in *Generative Agents* [46] [40]. Hou et al. (2024) provide a mathematical model of memory decay and consolidation for LLM agents [41] [42]. Kindroid's documentation of their memory system illustrates a real-world application of multi-tier memory with human-like patterns [22] [23]. Neurological grounding for these techniques can be found in studies of sleep-

based memory consolidation [1] and synaptic homeostasis [3], as well as cognitive experiments on value-based memory retention [4]. All together, they paint a clear picture: by marrying LLM technology with inspiration from the human brain, we can create continuous chat experiences that are not only technically feasible but feel natural, caring, and contextually aware over the long term.

---

[1] Hippocampo-cortical coupling mediates memory consolidation during sleep - PubMed
https://pubmed.ncbi.nlm.nih.gov/27182818/

[2] [3] Evidence Supporting SHY (A) Experiments in rats and mice show that the... | Download Scientific Diagram
https://www.researchgate.net/figure/Evidence-Supporting-SHY-A-Experiments-in-rats-and-mice-show-that-the-number-and_fig1_259697580

[4] Memory Recall for High Reward Value Items Correlates With Individual Differences in White Matter Pathways Associated With Reward Processing and Fronto-Temporal Communication - PMC
https://pmc.ncbi.nlm.nih.gov/articles/PMC6020774/

[5] [29] [30] [31] [32] [41] [42] [43] [44] [48] [49] [54] [55] [56] ”My agent understands me better”: Integrating Dynamic Human-like Memory Recall and Consolidation in LLM-Based Agents
https://arxiv.org/html/2404.00573v1

[6] [7] [8] [9] [13] [14] [15] LLM Context Windows: Why They Matter and 5 Solutions for Context Limits - Kolena
https://www.kolena.com/guides/llm-context-windows-why-they-matter-and-5-solutions-for-context-limits/

[10] [11] [16] Memory and State in LLM Applications - Arize AI
https://arize.com/blog/memory-and-state-in-llm-applications/

[12] [PDF] Evaluating Very Long-Term Conversational Memory of LLM Agents
https://aclanthology.org/2024.acl-long.747.pdf

[17] [51] [53] Frontiers | Enhancing memory retrieval in generative agents through LLM-trained cross attention networks
https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2025.1591618/full

[18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [47] Memory - Kindroid Knowledge Base
https://docs.kindroid.ai/memory

[33] [34] ROBOMETRICS® MACHINES - Human Memory & LLM Efficiency: Optimized Learning through Temporal Memory
https://www.robometricsagi.com/blog/ai-policy/human-memory-llm-efficiency-optimized-learning-through-temporal-memory

[35] The Rise of Generative Agents in Interactive Simulations | by Kourosh Sharifi | Medium
https://medium.com/@kourosh.sharifi/the-rise-of-generative-agents-in-interactive-simulations-cc5eded2736d

[36] Context Window Management Strategies - ApX Machine Learning
https://apxml.com/courses/langchain-production-llm/chapter-3-advanced-memory-management/context-window-management

[37] [38] [39] [40] [45] [46] [50] Generative Agents: Interactive Simulacra of Human Behavior
https://3dvar.com/Park2023Generative.pdf

[52] Persistent Memory & Context Issues with ChatGPT-4 Despite ...
https://community.openai.com/t/persistent-memory-context-issues-with-chatgpt-4-despite-extensive-prompting/1049995