

# MemTool: Optimizing Short-Term Memory Management for Dynamic Tool Calling in LLM Agent Multi-Turn Conversations

Elias Lumer, Anmol Gulati, Vamse Kumar Subbiah,  
Pradeep Honaganahalli Basavaraju and James A. Burke

*Commercial Technology and Innovation Office, PricewaterhouseCoopers, U.S.A*

## Abstract

Large Language Model (LLM) agents have shown significant autonomous capabilities in dynamically searching and incorporating relevant tools or Model Context Protocol (MCP) servers for individual queries. However, fixed context windows limit effectiveness in multi-turn interactions requiring repeated, independent tool usage. We introduce MemTool, a short-term memory framework enabling LLM agents to dynamically manage tools or MCP server contexts across multi-turn conversations. MemTool offers three agentic architectures: 1) Autonomous Agent Mode, granting full tool management autonomy, 2) Workflow Mode, providing deterministic control without autonomy, and 3) Hybrid Mode, combining autonomous and deterministic control. Evaluating each MemTool mode across 13+ LLMs on the ScaleMCP benchmark, we conducted experiments over 100 consecutive user interactions, measuring tool removal ratios (short-term memory efficiency) and task completion accuracy. In Autonomous Agent Mode, reasoning LLMs achieve high tool-removal efficiency (90–94% over a 3-window average), while medium-sized models exhibit significantly lower efficiency (0–60%). Workflow and Hybrid modes consistently manage tool removal effectively, whereas Autonomous and Hybrid modes excel at task completion. We present trade-offs and recommendations for each MemTool mode based on task accuracy, agency, and model capabilities.

## 1 Introduction

*The LLM is the CPU, and its context window is the RAM, representing the LLM’s working memory.<sup>1</sup>*

Recent breakthroughs in Large Language Model (LLM) agents have enabled powerful and scalable

agentic systems which autonomously search, discover, equip, and use a dynamic repository of tools or MCP servers (Model Context Protocol, 2025). Dynamic tool calling, also known as function calling, allows LLM agents to interact with external APIs or systems, while not being constrained to a fixed set of tools when the agent is initialized (Lumer et al., 2025b). This breakthrough enables LLM agents to go beyond a fixed number of tools, granting them the autonomy to explore, identify, and integrate new tools based off the user query, much like a human navigating a mobile app store. Moreover, as LLM agents become increasingly embedded in user session-based (chat, voice, video) applications, managing the limited context window of the model is necessary for multi-turn conversations. Known as context engineering and memory management, these systems allow multi-turn user interactions to persist not only across a single session (short-term memory) but also across multiple long-term sessions (long-term memory) (Schmid, 2025; LangChain, 2025).

Despite the advancements in dynamic tool-using agents and user-agent chat-based memory, there is still a significant gap in how LLM agents manage its short-term memory of tools in multi-turn dynamic tool retrieval settings. LLM agents can discover and add hundreds of new tools or MCP servers its context window, but need to reduce the number of available tools in its context upon solving the user question and no longer needing them. One primary concern is that the current literature on short-term memory management predominantly focuses on compressing the context between an assistant and a user through summarization and truncation of messages (Wu et al., 2025b; Breunig, 2025). However, these methods apply primarily to conversational tasks, where LLM agents perform a sequence of actions requested by the user, and do not address the dynamic retrieval and management of tools within an agent’s context window.

<sup>1</sup>Adapted from a talk by Andrej Karpathy, “Software in the Era of AI,” presented by Y Combinator, AI Startup School, 17 June 2025.(Karpathy, 2025).

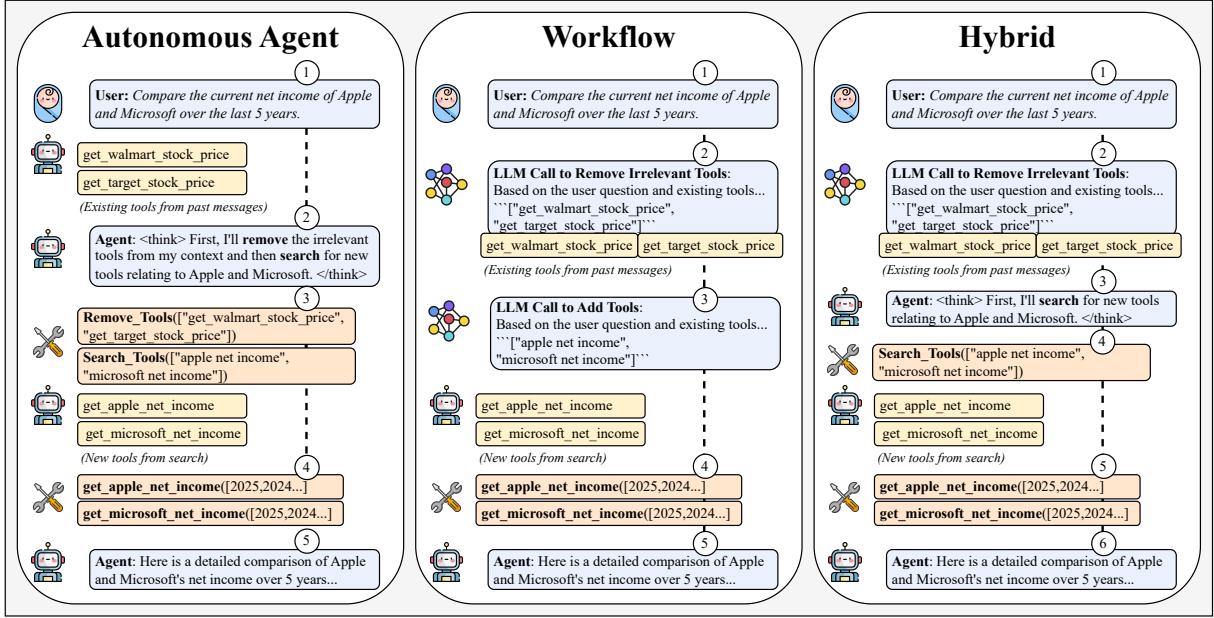


Figure 1: MemTool three modes architecture and end-to-end walkthrough of user-assistant interaction across multi-turn chats. In the example, the previous user session continues after the user asked about Walmart and Target’s stock price, given the existing tools still in the context window of the LLM agent. The LLM agent (robot icon) represents an LLM binded with a set of tools, where the LLM call (neural net icon) represents a fresh LLM chat completion message with no tools.

A second concern is the limited research on multi-turn scenarios among LLM agents with dynamic tool retrieval. Previous work has proven that LLM agents or agentic workflows can handle thousands of tools or MCPs without a significant drop in accuracy, due to only passing in relevant tools to the LLM agent by retrieval augmented generation (Lumer et al., 2025c; Chen et al., 2024). However, the existing state-of-the-art methods do not address the removal and management of tools (along with the addition of tools) within the LLM agent context window across a 100+ multi-turn session, while using native function calling (Fei et al., 2025; Lumer et al., 2025b). To effectively use dynamic tool retrieval agents in production environments, there needs to be thorough evaluation of how tools persist through multi-turn conversations in the same session. While some prior work has touched on long-term memory and personalization of tool-use (Zhang et al., 2025; Hao et al., 2025; Cheng et al., 2025), there remains a gap in how LLM agents manage their short-term tool memory through a multi-turn session.

In this paper, we introduce MemTool, a short-term memory framework for multi-turn dynamic tool use agents, enabling agents to manage their own context window of hundreds and thousands of searchable tools or MCP servers. MemTool is com-

prised of three modes—containing different levels of agency or autonomy for the LLM—for managing an LLM agent’s context window of tools across multi-turn conversations (See Figure 1). The first mode is the Autonomous Agent Mode, which gives full control to the LLM agent to add and remove tools from its context window (through function calling) while simultaneously answering the user question. The second mode is a predefined agentic workflow; first always removes irrelevant tools from its context, then selects additional tools through semantic or lexical search in a tool knowledge base, finally giving the LLM agent only the new tools to answer the user’s question. The third mode is a hybrid of autonomous agent and workflow, which first starts by always removing tools from its context, then giving the LLM agent autonomy to only add dynamic tools and use them to answer the user question.

After evaluating the three MemTool modes on the ScaleMCP benchmark of 5,000 MCP servers, we find that Autonomous Agent Mode achieves the highest tool-removal efficiency (90–94%) with reasoning LLMs, while Workflow and Hybrid Modes offer consistently effective tool removal across all models. In contrast, Autonomous Agent and Hybrid Modes demonstrate the highest task completion rates, leveraging agentic corrective loops for dynamic tool retrieval. Finally, we present trade-

offs and recommendations for each MemTool mode based on task accuracy, agency, and model capabilities.

## 2 Background

### 2.1 Memory for LLM agents

Memory for Large Language Model (LLM) agents enables the persistent accumulation of knowledge, sustained contextual awareness, and adaptive decision-making informed by historical interactions and prior experiences. Existing literature divides memory for AI systems into short-term and long-term memory (Wu et al., 2025b). For chat-based applications, short-term memory is often described for memories that exist within the session, and long-term memory enables memories beyond the session.

#### 2.1.1 Short-Term Memory

The majority of flagship model context windows fall between 100,000 to 1,000,000 tokens (Vel-lum.ai, 2025; Shang et al., 2025). Short-term memory targets all input tokens within the limited LLM context window. It is often broken down into sensory memory and working memory—where sensory memory encapsulates all multi-modal inputs and tool calls, and working memory consists of the system, assistant, and human messages or chain of thoughts that occur within the session (Wu et al., 2025b). Previous work on short-term memory is aimed at multi-turn conversations, which prevent the overflow of tokens within a model’s context window, as well as prune necessary context that is found to distract the model during a task (Laban et al., 2025; Hong et al., 2025). The context engineering strategies for short-term memory across multi-turn dialogues involve summarization or truncation of previous chat messages (Chirkova et al., 2025; Breunig, 2025; Schmid, 2025; LangChain, 2025; Alake, 2025). However, it is noted that summarizing long multi-turn dialogues is prone to over-summarization or information loss (Ravaut et al., 2024; Wang et al., 2025; Maharana et al., 2024; Laban et al., 2025; Pan et al., 2025; Wu et al., 2025a; Shan et al., 2025). In our paper, MemTool fits within short-term memory for LLM agents, that manage and optimize the available tool context of the model across multi-turn chats within a session. MemTool is the first study to address and evaluate short-term memory for dynamic tool-use across multi-turn conversations, while providing

three architectures (Figure 1) to optimize and manage available tools within the context window.

#### 2.1.2 Long-Term Memory

Long-term memory for LLMs involves storing information for extended periods that persist across sessions, enabling the model to retain knowledge and learned abilities over time (Wu et al., 2025b; Park et al., 2023; Zhong et al., 2023). Two types of long-term memory are explicit memory and implicit memory. Explicit memory consists of the recollection of key facts and events; implicit memory consists of learned skills, procedures, and long tasks. Explicit memories are further divided into episodic memory and semantic memory. Episodic memory refers to personal experiences and events, where semantic memory refers to facts and knowledge about the environment. Episodic memories are commonly used in chat applications that involve user personalization over time (e.g. knowing the user prefers a brand of clothing) (Alake, 2025).

Existing long-term memory frameworks include (Mem0, 2025), (Zep, 2025), and (Letta, 2025; Packer et al., 2024), which manage the storage and personalization of LLM agent long-term interactions with a user. In addition, large model providers that serve their model to users also store long-term memories across sessions for their users (OpenAI, 2025b; Google, 2025b; Perplexity, 2025).

Recent works have aimed at long-term memory for tool usage, that adapt the tool-use preferences of the user over time (Zhang et al., 2025; Hao et al., 2025; Cheng et al., 2025). MemTool is complementary to the aforementioned work, as long-term memory and short-term memory both serve as critical elements of context engineering for tools. Furthermore, some methods of updating or adding memories for tool personalization include an LLM agent calling a tool to update memories (Letta, 2025; Xu et al., 2025). In our work, we demonstrate that certain LLM models perform better than others in managing their short-term memory context while simultaneously solving the user query (Table 1). MemTool’s three modes (Figure 1) can be partially extrapolated to how LLM agents manage their long-term memory as well.

## 2.2 Tool Selection and Retrieval for LLM Agents

Large language models inherently face limitations regarding the number of tools or functions they can concurrently invoke or manage. Complex, multi-

step tool interactions place substantial demands on the LLM’s reasoning processes, complicating tool selection and sequencing. Additionally, leading model providers impose varying constraints on the number of tools a single LLM API request can have, within the range of 128-512 (OpenAI, 2024). To circumvent these limitations, recent studies (Lumer et al., 2025d; Chen et al., 2024) have introduced advanced retrieval-augmented generation (RAG) strategies. These approaches do not rely on model fine-tuning but instead store large toolsets externally within vector databases or structured knowledge graphs (Lumer et al., 2025a; Peng et al., 2024), dynamically selecting and equipping only the necessary tools during runtime. In Mem-Tool, we also rely on out-of-the-box embeddings for the 5,000 MCP servers stored in our tool knowledge base. However, to increase retrieval accuracy further, previous work have fine-tuned embedders for the tool knowledge base (Wu et al., 2024; Qin et al., 2023; Anantha et al., 2023; Yuan et al., 2024; Zheng et al., 2024).

An alternative methodology utilizes agentic RAG frameworks, which give LLM agents with specialized tool-discovery tools that autonomously identify and invoke appropriate tools as needed (Singh et al., 2025; Li et al., 2023; Du et al., 2024). This represents a departure from static, predefined retrieval setups (Lumer et al., 2024; Chen et al., 2024). However, previous research highlighted challenges, particularly with earlier GPT architectures that struggled to leverage these dynamic search functionalities effectively (Li et al., 2023). Recent contributions and stronger tool calling LLM models (Lumer et al., 2025b; Fei et al., 2025) significantly advance this domain by enabling an agent to dynamically query a comprehensive repository of over 5,000 MCP servers, facilitating efficient tool incorporation into the agent’s context window. However, these methods do not address the removal and management of dynamic tools for a 100+ multi-turn session. In our paper, we address this gap in short-term tool memory research, propose three novel modes or architectures to enable LLM agents to manage their tool context, and evaluate the three modes on over 10+ state-of-the-art LLM models. For our experiments, we use the ScaleMCP benchmark of 5,000 MCP servers (Lumer et al., 2025b) with out-of-the-box to store and retrieve tool names and descriptions.

---

**Algorithm 1:** Autonomous Agent Mode

---

**Input:** LLM, user query  $q$ , prior messages  $\text{msgs}_{\text{prev}}$ , prior tool set  $T_{\text{prev}}$ , vectorDB, top- $k=5$ , tool-limit  $L=128$ , history manager  $H \in \{\text{TRUNCATEHISTORY}, \text{SUMMARIZEHISTORY}\}$

**Output:** final answer  $a$

```

1  messages  $\leftarrow H(\text{msgs}_{\text{prev}}) + q;$ 
2   $T \leftarrow T_{\text{prev}} \cup \{$ 
3    Search_Tools(keywords: List[str]),
4    Remove_Tools(tool_names: List[str])
5  } while true do
6    reply  $\leftarrow LLM(\text{messages}, \text{tools} = T);$ 
7    if reply.type = tool_call then
8      switch reply.tool do
9        case Remove_Tools do
10        $R \leftarrow \text{reply.args};$ 
11        $T \leftarrow T \setminus (R \backslash \{ \text{Search\_Tools, Remove\_Tools} \});$ 
12
13      case Search_Tools do
14        $Q \leftarrow \text{reply.args};$ 
15        $S \leftarrow \text{VectorSearch}(Q, \mathcal{V}, k=5);$ 
16        $T \leftarrow T \cup S;$ 
17       if  $|T| > L$  then
18         Raise Error("Hit  $L$  limit, remove tools.");
19
20      otherwise do
21       result  $\leftarrow \text{ExecuteTool(reply)};$ 
22       append (reply.tool, result) to messages;
23
24    else
25      append reply to messages;
26       $a \leftarrow \text{reply.content};$ 
27      return  $a;$ 

```

---

### 2.3 Tool Calling or LLM Invocation

Beyond selecting and retrieving relevant tools, another critical aspect is the actual invocation of these tools by LLM agents. Several studies focus specifically on enhancing LLM capabilities for precise and effective tool invocation (Hao et al., 2024a; Qin et al., 2023; Patil et al., 2023). Recent advancements in fine-tuning methods have contributed significantly to improving LLMs’ proficiency in tool calling, including MOLoRA’s modular approach (Hao et al., 2024b), efficient tree-structured techniques (Zhu et al., 2025), and carefully constructed datasets generated collaboratively by multi-agent frameworks (Liu et al., 2024; Zhuang et al., 2025).

Despite the demonstrated benefits of fine-tuning strategies, our work deliberately adopts a different path, emphasizing plug-and-play methodologies compatible with out-of-the-box models and standard embedding solutions from providers such as OpenAI, Google, Anthropic, and Meta (OpenAI, 2025a; Google, 2025a; Anthropic, 2025; Meta Plat-

forms, 2025). Specifically, we investigate whether standard, commercially available LLMs can autonomously manage their tool context dynamically—removing tools when they lose relevance and incorporating new tools on demand. To this end, our framework equips the LLM with explicit operations for adding and removing tools from its active context, thus enabling greater autonomy in tool management. Additionally, acknowledging diverse operational requirements, we introduce two alternative execution modes designed to reduce autonomy in favor of more deterministic and controlled tool invocation behaviors.

### 3 Method

#### 3.1 MemTool

MemTool enables an LLM agent to manage its own context window of dynamic tools across multi-turn sessions. The broader implication of MemTool is that LLM agents can operate in production environments with a non-fixed set of tools at its disposal, searching, equipping, and removing tools or MCP servers, similarly to a human navigating a mobile app store. Specifically, we propose three modes or architectures (Figure 1) that grant varying degrees of autonomy to the LLM agent to optimize its short-term memory context window of tools (autonomous agent, workflow, and hybrid). We note trade-offs and recommendations for each mode, in each subsequent section.

##### 3.1.1 Autonomous Agent Mode

MemTool Autonomous Agent Mode grants full autonomy to the LLM agent to manage its context window of available tools while simultaneously answering the user task, across multi-turn conversations. The manner in which tools or MCP servers are removed or searched for is handled by two additional tools equipped to the LLM agent. The system prompt for the autonomous agent explains its task is to answer the user question by searching for available tools while also removing irrelevant tools it no longer needs. The full system prompt is in Appendix 5. Firstly, the agent is given a search tool to add tools to its context window. This tool is adapted from ScaleMCP (Lumer et al., 2025b). Secondly, the agent is given a remove tool, which is the critical element for managing over 100+ user questions sequentially. If the LLM agent calls the remove tool with a list of tools it has access to that are no longer relevant for the user question, we

---

#### Algorithm 2: Workflow Mode

---

```

Input: LLM, user query  $q$ , prior messages  $\text{msgs}_{\text{prev}}$ ,  

prior tool set  $T_{\text{prev}}$ , vectorDB  $\mathcal{V}$ , top- $k=5$ ,  

tool-limit  $L=128$ , history manager  $H \in$   

{TRUNCATEHISTORY, SUMMARIZEHISTORY}

Output: final answer  $a$ 

1 messages  $\leftarrow H(\text{msgs}_{\text{prev}}) + q$ ;
2  $T \leftarrow T_{\text{prev}}$ ;
3  $R \leftarrow LLM_{\text{prune}}(\text{messages}, T)$ ;
4  $T \leftarrow T \setminus R$ ;
5  $Q \leftarrow LLM_{\text{search}}(\text{messages}, q)$ ;
6 if  $Q \neq \emptyset$  then
7    $S \leftarrow \text{VectorSearch}(Q, \mathcal{V}, k=5)$ ;
8    $T \leftarrow T \cup S$ ;
9   while  $|T| > L$  do
10     $R \leftarrow LLM_{\text{prune}}(\text{messages}, T)$ ;
11    if  $R = \emptyset$  then
12      Raise Error(" $T > L$ ");
13     $T \leftarrow T \setminus R$ ;
14 while true do
15    $\text{reply} \leftarrow LLM(\text{messages}, \text{tools} = T)$ ;
16   if  $\text{reply.type} = \text{tool\_call}$  then
17      $\text{result} \leftarrow \text{ExecuteTool}(\text{reply})$ ;
18     append ( $\text{reply.tool}$ ,  $\text{result}$ ) to messages;
19   else
20     append  $\text{reply}$  to messages;
21      $a \leftarrow \text{reply.content}$ ;
22   return  $a$ ;

```

---

automatically remove them from the API call (for example, "tools=tools" in chat completions API) (OpenAI, 2024). We append any remove and/or search tool calls and tool call results to the context window, with helpful result messages such as "N tools added: tool names" or "N tools removed: tool names". Finally, the LLM agent will use its new tools to answer the user question, which the messages and existing are persisted to additional queries within the session.

In Algorithm 1, we first prune the previous messages either by truncation or summarization, in case the LLM agent used too many tokens in the previous query. This truncation does not affect the available tools, only dialogue messages. Then, the LLM agent enters a while loop, calling tools described above to manage its short-term memory context window of tools while answering the user question. Notably, when the SearchTool adds tools to the LLM agent's context, and the total exceeds the LLM API's tool limit, we return an error message and prompt the agent to remove tools from its context.

**Recommendations for Autonomous Agent Mode.**  
If opting to use Autonomous Agent Mode for MemTool's short-term tool memory:

- Certain LLM models fail to remove irrelevant tools from its context, causing the number of tools to increase until an upper limit.
- Prompting the system message of the LLM agent can provide detailed guidelines on when to remove tools and when to add them.
- Upper limit tool errors (e.g. 128) ‘encourage’ LLMs to remove tools.
- Including the current tool count (as a dynamic variable) in the system message helped the agent remove tools, rather than letting it infer from the tools equipped to the LLM from the API.

### 3.1.2 Workflow Mode

MemTool Workflow Mode limits autonomy to the LLM agent by abstracting the dynamic tool management to a fixed workflow that occurs after every user query. Instead, two fresh LLM calls occur in sequence to remove irrelevant tools (to the user query) and search for new tools to add. The irrelevant tools were once relevant in previous chat messages within the session, but no longer relevant. This Workflow Mode is commonly seen in state-of-the-art tool retrieval approaches (Lumer et al., 2025d; Chen et al., 2024). The novel differentiator that MemTool Workflow adds to previous work is enabling multi-turn conversations, adding a tool pruning LLM call, and by persisting the tools across the session.

As seen in Algorithm 2, the same messages and previous tools are initialized as the agent mode. Then, a fresh LLM call examines the current tool list along with the user query and decides to remove tools no longer relevant. Then, a fresh LLM examines the available tools and user query and decides to search for new tools if the existing ones are not sufficient. After removing and adding tools to the existing tool list, the LLM agent is finally initialized in the while loop, identical to the agent mode, less the ability to call SearchTool and RemoveTool.

**Recommendations for Workflow Mode.** If opting to use Workflow Mode for MemTool’s short-term tool memory:

- The simplicity of the workflow is a recommended first choice, however, once the LLM agent is initialized with the tool list, there is no mechanism to revert back to searching new tools.
- Limiting agency is seen to prevent any self-correctness. While previous works (Lumer

---

### Algorithm 3: Hybrid Mode

---

**Input:** LLM, user query  $q$ , prior messages  $\text{msgs}_{\text{prev}}$ , prior tool set  $T_{\text{prev}}$ , vectorDB  $\mathcal{V}$ , top- $k=5$ , tool-limit  $L=128$ , history manager  $H \in \{\text{TRUNCATEHISTORY}, \text{SUMMARIZEHISTORY}\}$

**Output:** final answer  $a$

```

1  messages  $\leftarrow H(\text{msgs}_{\text{prev}}) + q;$ 
2   $T \leftarrow T_{\text{prev}};$ 
3   $R \leftarrow LLM_{\text{prune}}(\text{messages}, T);$ 
4   $T \leftarrow T \setminus R;$ 
5   $T \leftarrow T_{\text{prev}} \cup \{$ 
6    Search_Tools(keywords: List[str])}
7  while true do
8    reply  $\leftarrow LLM(\text{messages}, \text{tools} = T);$ 
9    if reply.type = tool_call then
10      reply_tool case
11        otherwise Search_Tools case
12           $Q \leftarrow \text{reply.args};$ 
13           $S \leftarrow \text{VectorSearch}(Q, \mathcal{V}, k=5);$ 
14           $T \leftarrow T \cup S;$ 
15          while  $|T| > L$  do
16             $R \leftarrow LLM_{\text{prune}}(\text{messages}, T);$ 
17            if  $R = \emptyset$  then
18              Raise Error("T>L.");
19             $T \leftarrow T \setminus R;$ 
20          case
21            result  $\leftarrow \text{ExecuteTool(reply);}$ 
22            append (reply.tool, result) to
23            messages;
24        else
25          append reply to messages;
26           $a \leftarrow \text{reply.content};$ 
return  $a;$ 

```

---

et al., 2025d) have suggested to add a self-correctness step to the workflow, these patterns get overly complex and restrictive.

### 3.1.3 Hybrid Mode

MemTool Hybrid Mode grants the LLM agent partial autonomy while restricting others. After observing that LLMs tend to struggle with tool removal (the pruning aspect of short-term tool memory management) but perform well in searching and adding tools, Hybrid Mode combines both approaches to leverage their respective strengths. Hybrid Mode includes the fresh LLM call to remove no longer relevant tools from the context window, while granting the LLM agent autonomy to use the SearchTool to add and use a dynamic set of tools. This partial agency solves the limitation found in (Lumer et al., 2025d) of complex self-correcting workflows.

In Algorithm 3, the messages and existing tools are initialized as they were in Autonomous Agent and Workflow Mode. Then, the prune LLM call

Mode	LLM	Avg Removal Ratio 3T	Avg Residual 3T	Tool Correctness	Task Completion
Autonomous Agent	GPT-o3	<b>0.941</b>	7.44	0.75	<b>0.90</b>
	Gemini 2.5 Pro	0.924	6.51	0.81	0.80
	Gemini 2.5 Flash	0.905	<b>5.08</b>	0.74	0.65
	Claude Opus 4	0.878	13.85	<b>0.86</b>	0.84
	Claude Sonnet 4	0.840	24.44	0.80	0.83
	GPT-4.1	0.834	48.12	<b>0.86</b>	0.88
	GPT-4.1 Mini	0.733	58.93	0.78	0.80
	GPT-4o	0.713	37.48	0.68	0.76
	LLaMA 3 70B	0.244	123.33	0.42	0.72
	Claude 3.5 Sonnet	0.062	124.00	0.38	0.59
	GPT-4.1 Nano	0.000	0.00	0.13	0.60
Workflow	GPT-4o	<b>0.938</b>	7.19	0.71	0.70
	GPT-4.1	0.934	7.48	0.82	0.83
	LLaMA 3 70B	0.932	8.64	0.51	0.71
	Gemini 2.5 Pro	0.929	6.90	0.69	0.66
	Gemini 2.5 Flash	0.928	<b>6.60</b>	0.50	0.60
	GPT-o3	0.925	7.59	<b>0.88</b>	<b>0.84</b>
	GPT-4.1 Mini	0.922	7.72	0.72	0.81
	Claude 3.5 Sonnet	0.917	7.83	0.82	0.82
	Claude Opus 4	0.917	7.92	0.71	0.78
	Claude Sonnet 4	0.917	8.00	0.77	0.81
	GPT-4.1 Nano	0.904	8.96	0.64	0.66
Hybrid	GPT-4o	<b>0.943</b>	7.15	0.77	0.76
	GPT-4.1	0.941	7.29	<b>0.82</b>	0.80
	LLaMA 3 70B	0.938	7.52	0.60	0.76
	Gemini 2.5 Pro	0.938	6.52	0.74	0.75
	Claude 3.5 Sonnet	0.935	7.32	0.81	0.83
	GPT-o3	0.932	9.33	<b>0.82</b>	<b>0.87</b>
	Gemini 2.5 Flash	0.932	<b>6.15</b>	0.59	0.66
	GPT-4.1 Mini	0.929	7.26	0.81	0.79
	Claude Opus 4	0.920	10.46	<b>0.82</b>	0.81
	Claude Sonnet 4	0.912	10.27	<b>0.82</b>	0.81
	GPT-4.1 Nano	0.869	6.94	0.48	0.63

Table 1: LLM performance by mode, sorted within each mode by Avg Removal Ratio 3T. Avg Removal Ratio 3T measures the percentage of tools removed within a rolling 3-turn window after addition, indicating how well a model manages short-term memory over time. Avg Residual 3T captures the average number of tools remaining after three turns, reflecting tool accumulation behavior.

removes irrelevant tools based on the current message state, which exclude the SearchTools function. Finally, the LLM agent enters the while loop, dynamically querying the tool knowledge base for available tools or MCP servers, and executing them to solve the user query.

### Recommendations for Hybrid Mode

- The added SearchTool enables re-querying the knowledge base if the retrieved tools are not sufficient in answering the user query.
- The LLM agent is blinded or abstracted from any removal of tools. It could add too many tools and trigger a tool limit and cause complexity in the workflow.
- As LLM models or prompting techniques advance, Hybrid Mode can transition into Autonomous Agent mode.

### 3.2 Dataset

We use 5,000 tools or MCP servers from ScaleMCP and sample 100 instances for the 100 multi-turn tool-use dialogue. The sampling method is stratified sample, based on the number of tool calls per turn. ScaleMCP has an average of 5.0 tool calls per turn (Lumer et al., 2025b).

## 4 Evaluations

### 4.1 Experiment Settings

For all experiments, we evaluate 13 LLM models, ranging from closed and open source models. For the tool knowledge base, the embedding model used is Azure OpenAI’s text-embedding-ada-002, since ScaleMCP (Lumer et al., 2025b) already extensively tested 5 embedding models, 3 reranking models, and 5 retriever types-and found minimal difference between embedding models. We impose

a tool count API error limit and set it at 128 to standardize and account for all LLM models (OpenAI GPT series), even though models such as Gemini 2.5 Pro and Claude Opus 4 can use 256-512 tools in the function calling API request (OpenAI, 2024; Google, 2025a).

**Multi-Turn Conversation.** All experiments consist of 100 sequential user queries, with an average of 5 tool calls per turn. Each query can be related or unrelated to the immediate prior query.

$$\text{RemovalRatio} = \frac{\sum_{t=1}^T R_t}{\sum_{t=1}^T A_t} \leq 1, \quad (1)$$

$$\text{AvgRemovalRatio}_{3T} = \frac{1}{T-2} \sum_{t=3}^T \frac{\sum_{i=t-2}^t R_i}{\sum_{i=t-2}^t A_i} \quad (2)$$

$$\text{AvgResidual}_{+3} = \frac{1}{K} \sum_{k=1}^K \left( \frac{1}{3} \sum_{i=1}^3 T_{p_k+i} \right), \quad (3)$$

$$\text{TaskCompletion} = \text{Alignment}(\text{Task}, \text{Output}) \quad (4)$$

$$\text{Tool Correctness} = \frac{\text{Num. Correct Calls}}{\text{Total Calls}} \quad (5)$$

**Metrics.** Across 100 sequential multi-turn conversations, we primarily track the tool count at each turn to observe how the active tools fluctuate over turns. We define three key tool-calling context engineering metrics: the *Removal Ratio*, which measures the proportion of tools successfully removed after being added; the *Average Residual* after a 3-turn grace window, which captures how many tools remain shortly after each peak tool count; and the *Average Removal Ratio*  $3T$  over a 3-turn window (Equation 2), which smooths removal behavior by averaging removal ratios computed across successive 3-turn windows. For each turn, we use a *Task Completion Score* (Equation 4), computed using LLM-as-judge alignment scoring between the agent’s final response and the expected answer in the benchmark (Confident AI, 2025). We use OpenAI GPT-4o mini as the judge LLM model in all evaluations. Also, we compute the *Tool Correctness Score* (Equation 5), which compares each tool call to the expected tool call (Confident AI, 2025).

## 4.2 Autonomous Agent Mode Multi-Turn Experiments

In autonomous agent mode, we equip an LLM agent with two tools to manage its tool context and track how well it manages its context across 100+ sequential user queries, while answering the user with any new tools it retrieves.

### 4.2.1 Results Analysis

As seen in Table 1 and Figure 2, there is a wide range of performance across LLM models on how they manage their short-term tool memory across 100 sequential multi-turn conversations. The highest performing LLM models, denoted as ones that have high 3-window average removal ratios and task completion, consist of reasoning models such as Gemini 2.5 Pro/Flash, OpenAI GPT-o3, and Claude Opus 4. These models remove roughly 95% of their total tools across 100 conversations, and 87.8-100% of their tools on a rolling 3-window period. Visually, these models handle spikes in their tool count effectively, with any increase in tool count followed by a removal of unused tools for subsequent queries (Figure 2). This is shown by the high 3-window rolling average removal rates. Furthermore, these models are strong at simultaneously answering the user query, with Task Completion and Tool Correctness metrics averaging between 80-90%.

On the other hand, smaller and medium sized models, or models that were not post-trained with large reinforcement learning reasoning budgets, did not effectively manage their available tools while answering the user query. These models include OpenAI GPT-4o Mini, GPT-4.1 Mini, Anthropic Claude 3.5 Sonnet, and LLaMa 3 70B. Gemini 2.5 Flash is the only small model that had a rolling average 3-window removal ratio above 90%. However, the same model struggled to effectively answer the user query. Visually, these models struggle to remove unused tools, and are more focused on adding tools to their context window. As seen in Figure 2, LLaMa 3 70B and Claude 3.5 Sonnet climb to the 128 limit within turn 20, and the other models tend to add more tools than they remove over time, causing sporadic spikes and steady climbing spikes.

For the full results of all LLM models evaluated, see Appendix A, Table 2 and Appendix C, Figures 8 through 20.

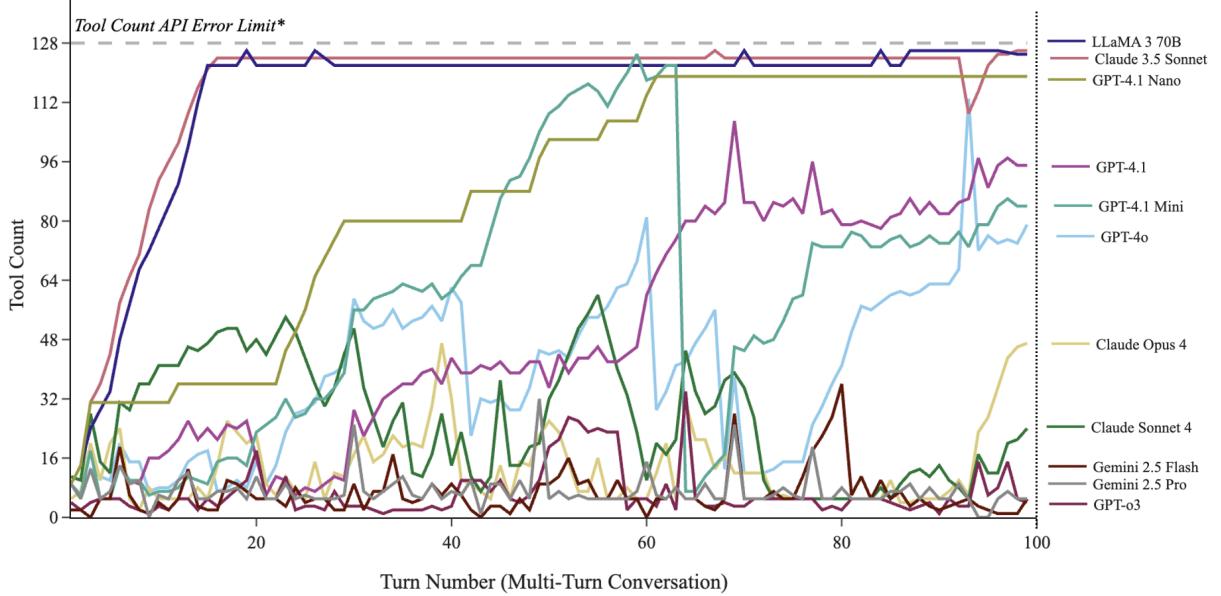


Figure 2: MemTool Autonomous Agent Mode: Tool Count across 100 multi-turn queries for various LLMs. While some models (e.g., GPT-o3, Gemini 2.5 Pro) maintain stable tool windows and consistently remove irrelevant tools added in previous turns, others (e.g., GPT-4.1 Nano, Claude 3.5 Sonnet, LLaMA 3 70B) fail to remove unused tools, exceeding the 128-tool API limit. This highlights large variation in agents’ ability to manage their short-term tool memory. Note: Tool Count API error Limit\* is set at 128 to standardize and account for all LLM models, even though models such as Gemini 2.5 Pro and Claude Opus 4 can use 256-512 tools in the function calling API request.

#### 4.2.2 Discussion

Medium and small LLM models in MemTool Autonomous Agent Mode struggle to efficiently manage its short-term tool memory while answering the user query. We recommend using reasoning models that have been post-trained with large reinforcement learning budgets, which impact how well models are able to manage its tool context window. The benefits of giving the LLM agent full autonomy over its tool context window consist of re-searching the tool knowledge base for more tools if it cannot find one the first time. While some models are able to remove irrelevant tools, others are too focused on answering the user query and forget to remove tools. Moreover, the searching and removing mechanism is fully dependent on the system prompt. We found very poor removal rates when the system prompt did not include the current number of tools in its context. To bypass this limitation, we pass a dynamic variable in the system prompt on every query, which is the length of the current tool count. Our results point to a clear distinction in the abilities of certain LLM models when handling short-term dynamic tool memory.

Common errors for Autonomous Agent Mode entail letting the tool count in the context window reach beyond the set 128 limit. Some models hover

around this limit and do not comprehend that it can remove 100 irrelevant tools at a time. Furthermore, we found that the system prompt, when it did not include the current tool count, caused the LLM agent to not remove tools. This indicates an inability for the LLM to understand the count of available tools it has, when interacting with the tools or function calling parameter in the API version of LLM models (OpenAI, 2024; Google, 2025b).

#### 4.3 Workflow Mode Multi-Turn Experiments

In Workflow Mode, LLMs operate under a deterministic short-term memory framework where tool context is centrally managed. Instead of granting the agent autonomy, the system prunes irrelevant tools and adds new ones through separate LLM calls, enforcing memory optimization before the agent answers each query.

##### 4.3.1 Results Analysis

As seen in Figure 3 and Table 1, Workflow Mode yielded consistently high performance across nearly all tested LLM models. The average 3-window removal ratio remained above 90% for all models, with GPT-4o, GPT-o3, and Gemini 2.5 Pro achieving top-tier performance in both removal and task metrics. Notably, all models avoided excessive tool accumulation across the 100-turn conversation

window, staying well below the API error threshold of 128 tools.

Tool Correctness and Task Completion varied more than removal efficiency. GPT-o3 achieved the highest Tool Correctness score (88%), followed closely by GPT-4.1 and Claude 3.5 Sonnet (both 82%). In Task Completion, GPT-o3 led at 84%, followed by GPT-4.1 and Claude 3.5 Sonnet again led with 83% and 82%, respectively. Lower-performing models in these categories included GPT-4.1 Nano and Gemini 2.5 Flash, which, while excellent at tool removal, struggled with reasoning over tools to answer the user question.

### 4.3.2 Discussion

The deterministic nature of Workflow Mode proves highly effective for short-term memory management. Since all LLM models are guided by the same two-step pipeline (prune-then-search), they consistently maintain a lean tool context and avoid tool accumulation issues seen in Autonomous Agent Mode. This removes the burden of memory optimization from the model, allowing even smaller models to perform well.

We recommend using Workflow Mode when cost efficiency and reliability are priorities. Given that nearly all models in this mode achieve comparable Removal Ratio and AvgRemovalRatio3T, the decision of which LLM to use can hinge on cost and performance in downstream tasks. For instance, if high Tool Correctness is critical, GPT-o3 or GPT-4.1 are preferable; if Task Completion is more important, Claude 3.5 Sonnet and GPT-4.1 stand out. In production settings, a low-cost model like GPT-4.1 Mini or Gemini 2.5 Flash can serve as the memory controller, while a more capable model handles the user-facing task.

## 4.4 Hybrid Mode Multi-Turn Experiments

Hybrid Mode blends agentic flexibility with structured control by allowing the LLM to add tools via `SearchTools`, while delegating tool removal to a separate pruning step. This balances dynamic retrieval with stable memory management across multi-turn conversations.

### 4.4.1 Results Analysis

As shown in Figure 4 and Table 1, Hybrid Mode demonstrates consistently strong performance across LLMs in short-term memory management. Nearly all models achieve average 3-window removal ratios above 90%, with OpenAI

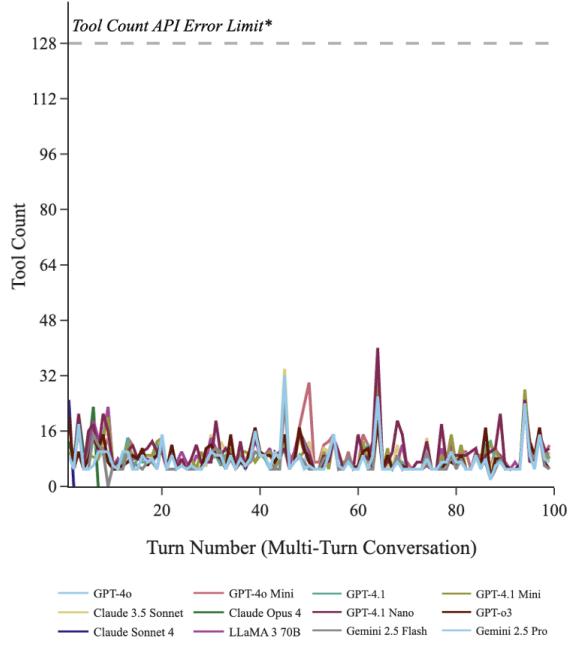


Figure 3: MemTool Workflow Mode: Tool Count across 100 multi-turn queries for various LLMs. All models maintain tool counts below the 128-tool API threshold.

o3 achieving a perfect 100% RemovalRatio and 93% AvgRemovalRatio3T. Task Completion and Tool Correctness scores are also generally high, led by Claude 3.7 Sonnet (88% and 83%), OpenAI o3 (87% and 82%), and GPT-4.1 (80% and 82%).

Claude Sonnet 4 and Claude Opus 4 perform well in correctness and task alignment, although they show slightly elevated average residuals after 3 turns (above 10). Gemini 2.5 Flash and GPT-4.1 Nano are the only models that fall below a 70% Task Completion threshold, consistent with their lower correctness scores (66% and 48% respectively). Notably, all models remain well below the 128-tool API limit, indicating successful memory regulation even in the absence of agent-led pruning.

### 4.4.2 Discussion

Hybrid Mode successfully balances structure and autonomy, enabling agents to correct and expand their toolset when needed, while avoiding tool bloat via deterministic pruning. This decoupled design helps stabilize tool count while maintaining agent adaptability.

OpenAI o3 emerges as one of the top-performing models overall, achieving perfect removal and the second-highest Task Completion (87%). Claude 3.7 Sonnet leads in task alignment (88%), followed closely by GPT-4.1 and Claude 3.5 Sonnet. While smaller models like Gemini 2.5 Flash and GPT-

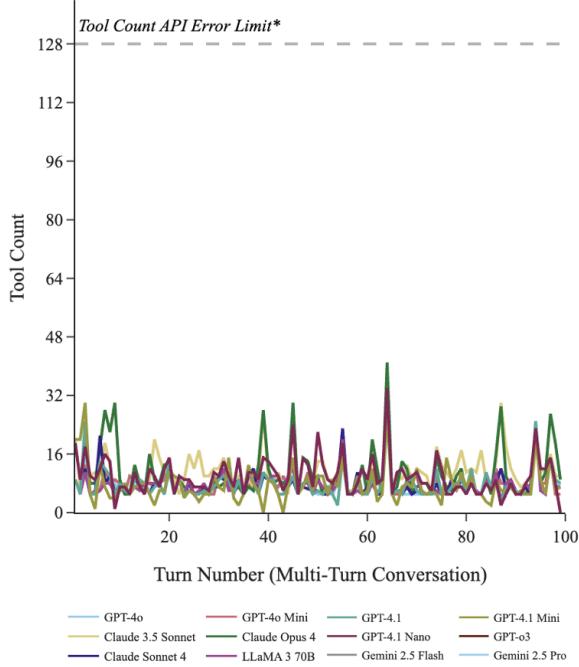


Figure 4: MemTool Hybrid Mode: Tool Count across 100 multi-turn queries for various LLMs. Most models stay stable and far below the 128-tool limit.

4.1 Nano exhibit lower downstream performance, they still maintain high removal ratios, suggesting Hybrid Mode allows efficient memory use even for less capable models.

We recommend selecting a model for Hybrid Mode based on task-level metrics such as Tool Correctness and Task Completion. When removal behavior is comparable across models (as seen in Table 1), performance in reasoning and retrieval becomes the differentiating factor.

## 5 Conclusion

As Large Language Model (LLM) agents demonstrate significant capabilities in dynamically searching and incorporating relevant tools or MCP servers, fixed context windows limit their effectiveness in multi-turn interactions requiring repeated, independent tool usage. We introduced MemTool, a short-term memory framework enabling LLM agents to dynamically manage their context window of tools or MCP servers across multi-turn conversations. MemTool provides three agentic architectures with varying autonomy levels: 1) Autonomous Agent Mode (full tool management autonomy), 2) Workflow Mode (deterministic control without autonomy), and 3) Hybrid Mode (combining autonomous and deterministic control). Evaluating each MemTool mode across 13+ LLM

models on the ScaleMCP benchmark (5,000 MCP servers), we found that reasoning LLMs in Autonomous Agent Mode achieved high tool-removal efficiency (90–94% over a 3-window average), whereas medium-sized models demonstrated significantly lower efficiency (0–60%). Workflow and Hybrid modes consistently managed tool removal effectively, while Autonomous and Hybrid modes excelled in adaptive tool addition and task completion. MemTool pushes the needle forward in short-term memory for tool-using LLM agents, providing evidence that LLM agents can effectively manage their context windows of dynamic tools. Looking forward, MemTool can be paired with long-term memory advancements in tool-using LLM agents.

## Limitations

Although we demonstrate MemTool as reliable for dynamic tool management within short-term memory, it does have several limitations. MemTool Autonomous Agent Mode struggles with reliably removing tools, especially when non-reasoning models are used; this limitation can be addressed by using the MemTool Workflow Mode, which is deterministic. However, MemTool Workflow Mode itself constrains the agent’s capacity to loop back and explore additional tools; thus, MemTool Hybrid Mode becomes beneficial, balancing deterministic structure with agent autonomy. Nevertheless, MemTool Hybrid Mode encounters limitations if the agent explores too many tools and breaches pre-defined limits; one feasible mitigation strategy involves invoking a dedicated pruning LLM, though this approach can inadvertently remove essential context, suggesting a potential need to temporarily revert to MemTool Autonomous Agent Mode. Additionally, both the Workflow and Hybrid modes require careful selection of the appropriate LLM. We address this limitation by providing Tool Correctness and Task Completion metrics for each LLM, allowing for an informed selection or mixing of LLMs based on their respective strengths, such as proficiency in tool removal and searching versus actual tool usage. Ultimately, selecting the appropriate MemTool mode aligned with the specific LLM model and task context can effectively mitigate these limitations.

## References

- Richmond Alake. 2025. Architecting agent memory: Principles, patterns, and best practices. Presented

- at AIEWF 2025 (AI Engineer World's Fair). San Francisco, CA. Hosted by MongoDB. Accessed: July 19, 2025.
- Raviteja Anantha, Bortik Bandyopadhyay, Anirudh Kashi, Sayantan Mahinder, Andrew W. Hill, and Srinivas Chappidi. 2023. *Protip: Progressive tool retrieval improves planning*. [Preprint](#), arXiv:2312.10332.
- Anthropic. 2025. Anthropic. <https://www.anthropic.com/>.
- Drew Breunig. 2025. How to fix your context. <https://www.dbreunig.com/2025/06/26/how-to-fix-your-context.html?ref=blog.langchain.com>.
- Yanfei Chen, Jinsung Yoon, Devendra Singh Sachan, Qingze Wang, Vincent Cohen-Addad, Mohammadhossein Bateni, Chen-Yu Lee, and Tomas Pfister. 2024. *Re-Invoke: Tool invocation rewriting for zero-shot tool retrieval*. [Preprint](#), arXiv:2408.01875.
- Zihao Cheng, Hongru Wang, Zeming Liu, Yuhang Guo, Yuanfang Guo, Yunhong Wang, and Haifeng Wang. 2025. *Toolspectrum : Towards personalized tool utilization for large language models*. [Preprint](#), arXiv:2505.13176.
- Nadezhda Chirkova, Thibault Formal, Vassilina Nikoulina, and Stéphane Clinchant. 2025. *Provence: efficient and robust context pruning for retrieval-augmented generation*. [Preprint](#), arXiv:2501.16214.
- Confident AI. 2025. *Deepeval: The open-source llm evaluation framework*.
- Yu Du, Fangyun Wei, and Hongyang Zhang. 2024. *Any-tool: Self-reflective, hierarchical agents for large-scale api calls*. [Preprint](#), arXiv:2402.04253.
- Xiang Fei, Xiawu Zheng, and Hao Feng. 2025. *Mcp-zero: Active tool discovery for autonomous llm agents*. [Preprint](#), arXiv:2506.01056.
- Google. 2025a. Gemini. <https://gemini.google.com/>.
- Google. 2025b. Google model provider long-term memory. <https://ai.google/>. Google AI LLMs persist user memory across sessions. Accessed July 19, 2025.
- Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2024a. *Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings*. [Preprint](#), arXiv:2305.11554.
- Yupu Hao, Pengfei Cao, Zhuoran Jin, Huanxuan Liao, Yubo Chen, Kang Liu, and Jun Zhao. 2024b. *Citi: Enhancing tool utilizing ability in large language models without sacrificing general performance*. [Preprint](#), arXiv:2409.13202.
- Yupu Hao, Pengfei Cao, Zhuoran Jin, Huanxuan Liao, Yubo Chen, Kang Liu, and Jun Zhao. 2025. *Evaluating personalized tool-augmented llms from the perspectives of personalization and proactivity*. [Preprint](#), arXiv:2503.00771.
- Kelly Hong, Anton Troynikov, and Jeff Huber. 2025. Context rot: How increasing input tokens impacts llm performance. <https://research.trychroma.com/context-rot>. Chroma Research.
- Andrej Karpathy. 2025. Keynote: Software is changing (again). Presented at AI Startup School. San Francisco, CA. Slides available at <https://drive.google.com/file/d/1a0h1...>
- Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. 2025. *Llms get lost in multi-turn conversation*. [Preprint](#), arXiv:2505.06120.
- LangChain. 2025. Context engineering. <https://blog.langchain.com/context-engineering-for-agents/>.
- Letta. 2025. Letta: Stateful agents with transparent long-term memory. <https://github.com/letta-ai/letta>. Open-source framework based on MemGPT. Accessed July 19, 2025.
- Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. *Api-bank: A comprehensive benchmark for tool-augmented llms*. [Preprint](#), arXiv:2304.08244.
- Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong Wang, Yuxian Wang, Wu Ning, Yutai Hou, Bin Wang, Chuhan Wu, Xinzhi Wang, Yong Liu, Yasheng Wang, and 8 others. 2024. *Toolace: Winning the points of llm function calling*. [Preprint](#), arXiv:2409.00920.
- Elias Lumer, Pradeep Honaganahalli Basavaraju, Myles Mason, James A. Burke, and Vamse Kumar Subbiah. 2025a. *Graph rag-tool fusion*. [Preprint](#), arXiv:2502.07223.
- Elias Lumer, Anmol Gulati, Vamse Kumar Subbiah, Pradeep Honaganahalli Basavaraju, and James A. Burke. 2025b. *Scalemcp: Dynamic and auto-synchronizing model context protocol tools for llm agents*. [Preprint](#), arXiv:2505.06416.
- Elias Lumer, Vamse Subbiah, James Burke, Pradeep Basavaraju, and Austin Huber. 2025c. *Toolshed: Scale tool-equipped agents with advanced rag-tool fusion and tool knowledge bases*. In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence - Volume 3: ICAART*, pages 1180–1191. INSTICC, SciTePress.
- Elias Lumer, Vamse Subbiah, James Burke, Pradeep Basavaraju, and Austin Huber. 2025d. *Toolshed: Scale tool-equipped agents with advanced rag-tool fusion and tool knowledge bases*. In *Proceedings*

- of the 17th International Conference on Agents and Artificial Intelligence – Volume 3: ICAART, pages 1180–1191. INSTICC, SciTePress.
- Elias Lumer, Vamse Kumar Subbiah, James A. Burke, Pradeep Honaganahalli Basavaraju, and Austin Huber. 2024. Toolshed: Scale tool-equipped agents with advanced rag-tool fusion and tool knowledge bases. [Preprint](#), arXiv:2410.14594.
- Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of llm agents. [Preprint](#), arXiv:2402.17753.
- Mem0. 2025. Mem0: The memory layer for personalized ai. <https://mem0.ai/>. A universal memory layer for AI agents. Accessed July 19, 2025.
- Meta Platforms. 2025. Meta llama. <https://llama.meta.com/>.
- Model Context Protocol. 2025. Tools documentation. <https://modelcontextprotocol.io/docs/concepts/tools>.
- OpenAI. 2024. Function calling.
- OpenAI. 2025a. Openai. <https://openai.com/>.
- OpenAI. 2025b. Openai model provider long-term memory. <https://openai.com/>. OpenAI LLMs persist memory across user sessions. Accessed July 19, 2025.
- Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2024. Memgpt: Towards llms as operating systems. [Preprint](#), arXiv:2310.08560.
- Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Xufang Luo, Hao Cheng, Dongsheng Li, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Jianfeng Gao. 2025. On memory construction and retrieval for personalized conversational agents. [Preprint](#), arXiv:2502.05589.
- Joon Sung Park, Joseph C. O’Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. [Preprint](#), arXiv:2304.03442.
- Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2023. Gorilla: Large language model connected with massive apis. [Preprint](#), arXiv:2305.15334.
- Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2024. Graph retrieval-augmented generation: A survey. [Preprint](#), arXiv:2408.08921.
- Perplexity. 2025. Perplexity ai: Persistent memory in conversational ai. <https://www.perplexity.ai/>. Perplexity stores long-term user interactions. Accessed July 19, 2025.
- Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Toollm: Facilitating large language models to master 16000+ real-world apis. [Preprint](#), arXiv:2307.16789.
- Mathieu Ravaut, Aixin Sun, Nancy F. Chen, and Shafiq Joty. 2024. On context utilization in summarization with large language models. [Preprint](#), arXiv:2310.10570.
- Philipp Schmid. 2025. The new skill in ai is not prompting, it's context engineering. <https://www.philschmid.de/context-engineering>.
- Lianlei Shan, Shixian Luo, Zezhou Zhu, Yu Yuan, and Yong Wu. 2025. Cognitive memory in large language models. [Preprint](#), arXiv:2504.02441.
- Ning Shang, Li Lyna Zhang, Siyuan Wang, Gaokai Zhang, Gilsinia Lopez, Fan Yang, Weizhu Chen, and Mao Yang. 2025. Longrope2: Near-lossless llm context window scaling. [Preprint](#), arXiv:2502.20082.
- Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talei Khoei. 2025. Agentic retrieval-augmented generation: A survey on agentic rag. [Preprint](#), arXiv:2501.09136.
- Vellum.ai. 2025. Llm leaderboard. <https://www.vellum.ai/llm-leaderboard>.
- Qingyue Wang, Yanhe Fu, Yanan Cao, Shuai Wang, Zhiliang Tian, and Liang Ding. 2025. Recursively summarizing enables long-term dialogue memory in large language models. [Preprint](#), arXiv:2308.15022.
- Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2025a. Longmemeval: Benchmarking chat assistants on long-term interactive memory. [Preprint](#), arXiv:2410.10813.
- Mengsong Wu, Tong Zhu, Han Han, Chuanyuan Tan, Xiang Zhang, and Wenliang Chen. 2024. Seal-tools: Self-instruct tool learning dataset for agent tuning and detailed benchmark. [Preprint](#), arXiv:2405.08355.
- Yaxiong Wu, Sheng Liang, Chen Zhang, Yichao Wang, Yongyue Zhang, Huifeng Guo, Ruiming Tang, and Yong Liu. 2025b. From human memory to ai memory: A survey on memory mechanisms in the era of llms. [Preprint](#), arXiv:2504.15965.
- Wujiang Xu, Kai Mei, Hang Gao, Juntao Tan, Zujie Liang, and Yongfeng Zhang. 2025. A-mem: Agentic memory for llm agents. [Preprint](#), arXiv:2502.12110.
- Lifan Yuan, Yangyi Chen, Xingyao Wang, Yi R. Fung, Hao Peng, and Heng Ji. 2024. Craft: Customizing llms by creating and retrieving from specialized toolsets. [Preprint](#), arXiv:2309.17428.

Zep. 2025. Zep: A context engineering platform for ai agents. <https://www.getzep.com/>. A long-term memory service using temporal knowledge graphs. Accessed July 19, 2025.

Weizhi Zhang, Xinyang Zhang, Chenwei Zhang, Liangwei Yang, Jingbo Shang, Zhepei Wei, Henry Peng Zou, Zijie Huang, Zhengyang Wang, Yifan Gao, Xiaoman Pan, Lian Xiong, Jingguo Liu, Philip S. Yu, and Xian Li. 2025. [Personaagent: When large language model agents meet personalization at test time](#). [Preprint](#), arXiv:2506.06254.

Yuanhang Zheng, Peng Li, Wei Liu, Yang Liu, Jian Luan, and Bin Wang. 2024. [Toolrrank: Adaptive and hierarchy-aware reranking for tool retrieval](#).

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2023. [Memorybank: Enhancing large language models with long-term memory](#). [Preprint](#), arXiv:2305.10250.

Dongsheng Zhu, Weixian Shi, Zhengliang Shi, Zhaochun Ren, Shuaiqiang Wang, Lingyong Yan, and Dawei Yin. 2025. [Divide-then-aggregate: An efficient tool learning method via parallel tool invocation](#). [Preprint](#), arXiv:2501.12432.

Yuchen Zhuang, Jingfeng Yang, Haoming Jiang, Xin Liu, Kewei Cheng, Sanket Lokegaonkar, Yifan Gao, Qing Ping, Tianyi Liu, Binxuan Huang, Zheng Li, Zhengyang Wang, Pei Chen, Ruijie Wang, Rongzhi Zhang, Nasser Zalmout, Priyanka Nigam, Bing Yin, and Chao Zhang. 2025. [Hephaestus: Improving fundamental agent capabilities of large language models through continual pre-training](#). [Preprint](#), arXiv:2502.06589.

## Appendix

### A Full LLM Evaluation Table by Mode

We show the complete quantitative results across all models and agent modes in Appendix A, Table 2.

### B Prompts

We include the full prompt templates used by the LLMs in each mode to manage tool search, addition, and removal behavior, as shown in Appendix B, Figures 5, 6, and 7.

### C Autonomous Agent Mode — Tool Count Graphs

This section presents individual tool count trajectories for each LLM evaluated under MemTool’s Autonomous Agent Mode across 100 multi-turn conversations. These graphs visualize each model’s ability to regulate its short-term memory window.

Mode	LLM	Avg Removal Ratio 3T	Avg Residual 3T	Tool Correctness	Task Completion
Autonomous Agent	GPT-o3	<b>0.941</b>	7.44	0.75	<b>0.90</b>
	Gemini 2.5 Pro	0.924	6.51	0.81	0.80
	Claude 3.7 Sonnet	0.905	14.25	<b>0.86</b>	0.89
	Gemini 2.5 Flash	0.905	<b>5.08</b>	0.74	0.65
	Claude Opus 4	0.878	13.85	<b>0.86</b>	0.84
	Claude Sonnet 4	0.840	24.44	0.80	0.83
	GPT-4.1	0.834	48.12	<b>0.86</b>	0.88
	GPT-4.1 Mini	0.733	58.93	0.78	0.80
	GPT-4o	0.713	37.48	0.68	0.76
	GPT-4o Mini	0.449	121.06	0.78	0.88
	LLaMA 3 70B	0.244	123.33	0.42	0.72
	Claude 3.5 Sonnet	0.062	124.00	0.38	0.59
	GPT-4.1 Nano	0.000	0.00	0.13	0.60
Workflow	GPT-4o	<b>0.938</b>	7.19	0.71	0.70
	GPT-4.1	0.934	7.48	0.82	0.83
	LLaMA 3 70B	0.932	8.64	0.51	0.71
	Gemini 2.5 Pro	0.929	6.90	0.69	0.66
	Gemini 2.5 Flash	0.928	<b>6.60</b>	0.50	0.60
	GPT-o3	0.925	7.59	<b>0.88</b>	<b>0.84</b>
	GPT-4.1 Mini	0.922	7.72	0.72	0.81
	Claude 3.5 Sonnet	0.917	7.83	0.82	0.82
	Claude Opus 4	0.917	7.92	0.71	0.78
	Claude Sonnet 4	0.917	8.00	0.77	0.81
	GPT-4o Mini	0.916	8.43	0.74	0.72
	GPT-4.1 Nano	0.904	8.96	0.64	0.66
Hybrid	GPT-4o	<b>0.943</b>	7.15	0.77	0.76
	GPT-4.1	0.941	7.29	0.82	0.80
	LLaMA 3 70B	0.938	7.52	0.60	0.76
	Gemini 2.5 Pro	0.938	6.52	0.74	0.75
	Claude 3.5 Sonnet	0.935	7.32	0.81	0.83
	GPT-4o Mini	0.934	7.77	0.81	0.80
	GPT-o3	0.932	9.33	0.82	0.87
	Gemini 2.5 Flash	0.932	<b>6.15</b>	0.59	0.66
	GPT-4.1 Mini	0.929	7.26	0.81	0.79
	Claude 3.7 Sonnet	0.921	7.97	<b>0.83</b>	<b>0.88</b>
	Claude Opus 4	0.920	10.46	0.82	0.81
	Claude Sonnet 4	0.912	10.27	0.82	0.81
	GPT-4.1 Nano	0.869	6.94	0.48	0.63

Table 2: LLM performance by mode, sorted within each mode by Avg Removal Ratio 3T. Avg Removal Ratio 3T measures the percentage of tools removed within a rolling 3-turn window after addition, indicating how well a model manages short-term memory over time. Avg Residual 3T captures the average number of tools remaining after three turns, reflecting tool accumulation behavior.

```

1 PROMPT="""You are a highly intelligent financial Assistant that has access to 5,000 external financial tools for the fortune 1,000 companies to answer the user question.
2 To access these tools, you need to use the 'SYS_SEARCH_TOOLS' tool to search for the tools that match the user question.
3 These tools will get added to your memory context and you can decide which ones to use.
4 However, since these tools stay in your memory throughout the conversation, you need to be careful on not loading too many tools into your context.
5 You cannot have more than 128 tools in your active memory context.
6 IMPORTANT: Remove tools using 'SYS_REMOVE_TOOLS' when you do not use them for more than 2 subsequent user turns.
7
8 The current date is {current_date} , use this date as context for the user questions.
9
10 <Background about your 5,000 tools in a database>
11 These tools are highly decomposed and specific for the company and financial metric.
12 - For example, a tool could be "Apple Revenue Tool" or "Amazon Stock Price".
13 </Background about your 5,000 tools in a database>
14
15 <When to search for new tools>
16 You should search for new tools when:
17 1. You do not have any tools loaded in your context.
18 2. You have tools loaded, but they are not relevant to the user question.
19 </When to search for new tools>
20
21 <When to remove tools>
22 You should remove tools when:
23 1. As soon as the user asks a question, and the tools you have loaded are not relevant to the user question, remove them immediately, so they don't build up!!!
24 2. Since 128 tools is the maximum, you should be super careful to have a high tool count loaded!
25 </When to remove tools>
26
27 <Key Requirements>
28 1. Understand the user question and which tools are needed to answer it, and start to think about the keywords to search for them.
29 2. IMPORTANT: Remove inactive tools
30   * As soon as the user asks a question, and the tools you have loaded are not relevant to the user question, remove them immediately with SYS_REMOVE_TOOLS.
31   * Once the irrelevant tools are removed, if you need more tools to answer the question, you can search for them.
32   * If new tools are needed, call 'SYS_SEARCH_TOOLS' to attach new tools to your context.
33   The 'SYS_SEARCH_TOOLS' takes a list of search keywords for each tool you want to search for and add.
34   If you want to add multiple tools at once, like search for multiple companies financial metric tools, then make a list of search queries for each one within the tool.
35 3. Once new tools are added, figure out which one you need.
36 NOTE: if still you believe there isn't the right tool to answer the question, you can call 'SYS_SEARCH_TOOLS' again to add more, or simply tell the user you cannot
37 answer the question with the available tools.
38 3. Answer the question
39   * Call the appropriate task-specific tools you have loaded.
40   * Compose a concise, clear answer for the user.
41 4. Never remove the 'SYS_SEARCH_TOOLS' and 'SYS_REMOVE_TOOLS' tools from your context, as they are essential for managing your memory and tool usage.
42 </Key Requirements>
43
44 <YOUR CURRENT TOOL COUNT>
45 Tool count: {num_tools}
46 Take the necessary actions to answer the user question as well as manage your tool memory effectively
47 </YOUR CURRENT TOOL COUNT>
48 """

```

Figure 5: Autonomous Agent System Prompt, with the SearchTool (SYS\_SEARCH\_TOOL) and RemoveTool (SYS\_REMOVE\_TOOL)

```

11 |     system_prompt = """You are a highly intelligent financial Assistant that has access to 5,000 external financial tools for the fortune 1,000 companies to answer the user question.
12 | Your task is to create search keywords to find the tools to use for the user question.
13 |
14 <Background about your 5,000 tools in a database>
15 These tools are highly decomposed and specific for the company and financial metric.
16 - For example, a tool could be "Apple Revenue Tool" or "Amazon Stock Price".
17 - Each tool is a combination of a company and financial metric. So your keyword should reflect both the company and financial metric to search for.
18 </Background about your 5,000 tools in a database>
19
20 If you need more tools, search for them, with a list of search keywords based on the user question.
21 When the user question comes in, decide if you need new tools, and if so, create a list of search keywords to find the tools to use for the user question."""
22 |     human_message = f"""Existing tools: ``"
23 |     {existing_tools}
24 |     ``"
25 | """
26 User Query: `(user_query)` """

```

Figure 6: Search Tools prompt for Workflow Mode LLM call to add more tools by generating search keywords based on the user question.

```

14 |     system_prompt = """You are a highly intelligent financial Assistant that has access to 5,000 external financial tools for the fortune 1,000 companies to answer the user question.
15 | Your task is to look at the existing tools and decide if you need to remove any of them based on the user question.
16 | If the existing tools are not directly relevant to answering the user question, you should remove them.
17 |
18 <Background about your 5,000 tools in a database>
19 These tools are highly decomposed and specific for the company and financial metric.
20 - For example, a tool could be "Apple Revenue Tool" or "Amazon Stock Price".
21 </Background about your 5,000 tools in a database>
22
23 When the user question comes in, decide if any of the existing tools are not relevant to answering the user question.
24 If any of the existing tools are not relevant, then create a list of tools to remove. Their names need to match verbatim, otherwise the removal will not work."""
25 |     human_message = f"""Existing tools: ``"
26 |     {existing_tools}
27 |     ``"
28 | """
29 User Query: `(user_query)` """

```

Figure 7: Remove Tools prompt for Workflow Mode and Hybrid Mode LLM call to remove irrelevant tools by responding with the exact tool names based on the user question.

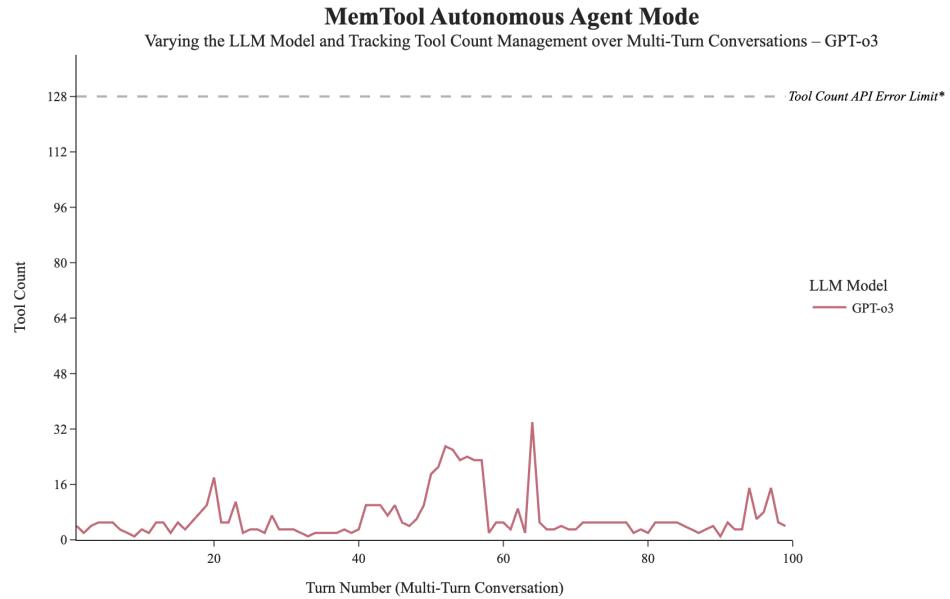


Figure 8: OpenAI o3 — Tool Count over Time (Autonomous Agent Mode)

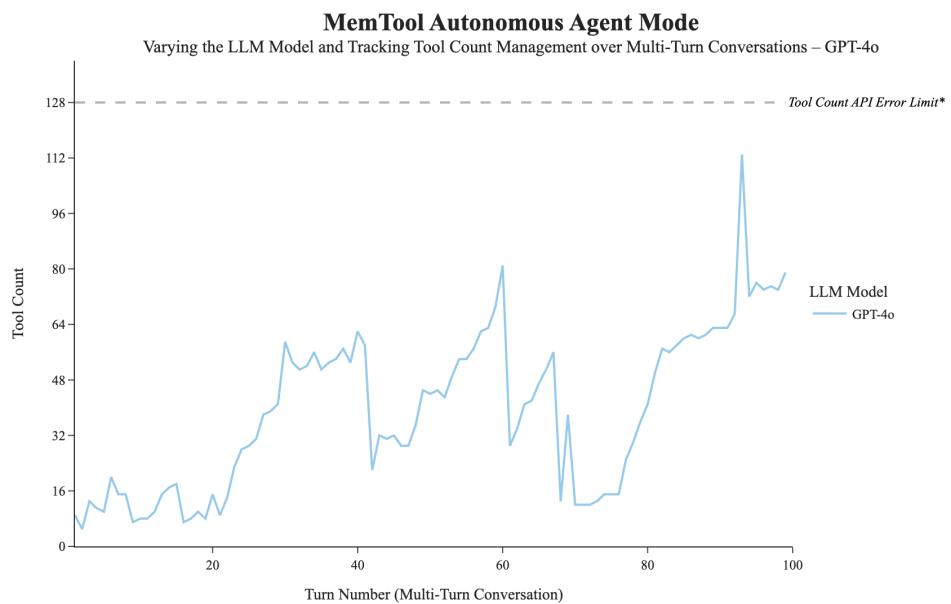


Figure 9: GPT-4o — Tool Count over Time (Autonomous Agent Mode)

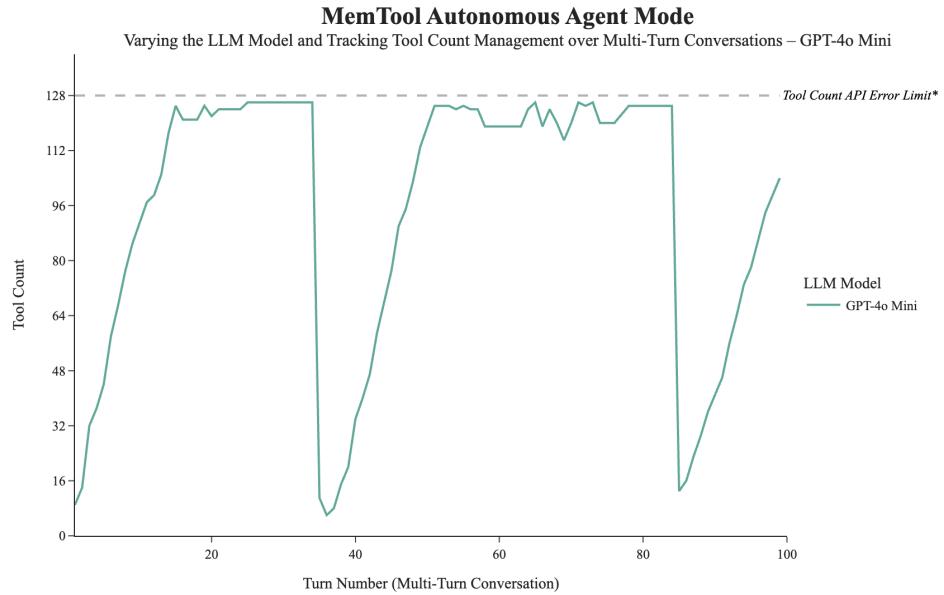


Figure 10: GPT-4o Mini — Tool Count over Time (Autonomous Agent Mode)

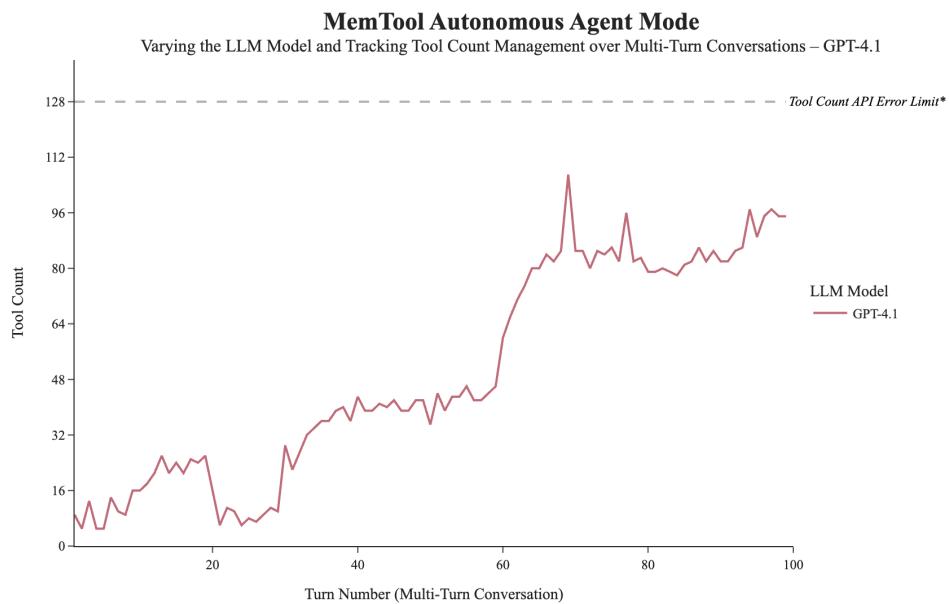


Figure 11: GPT-4.1 — Tool Count over Time (Autonomous Agent Mode)

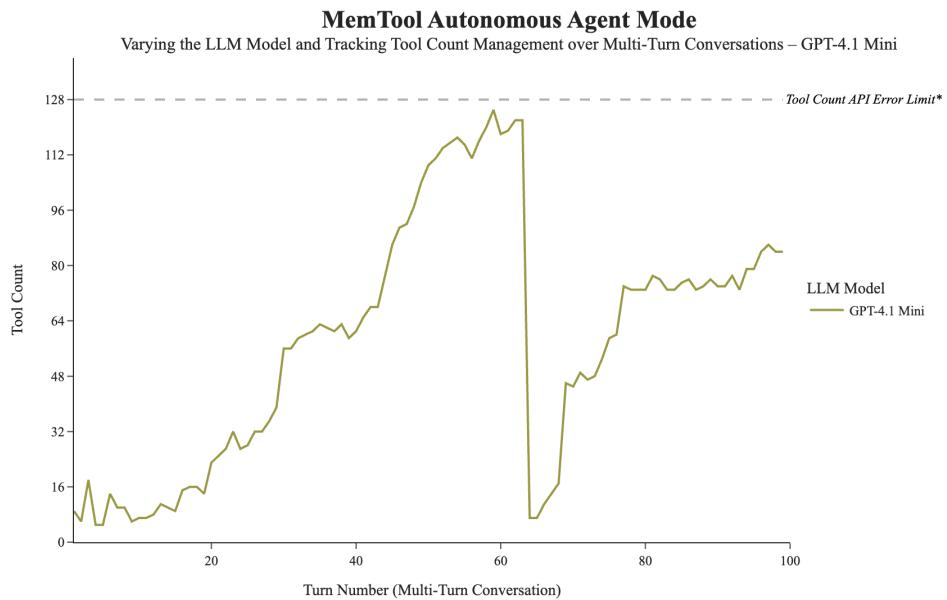


Figure 12: GPT-4.1 Mini — Tool Count over Time (Autonomous Agent Mode)

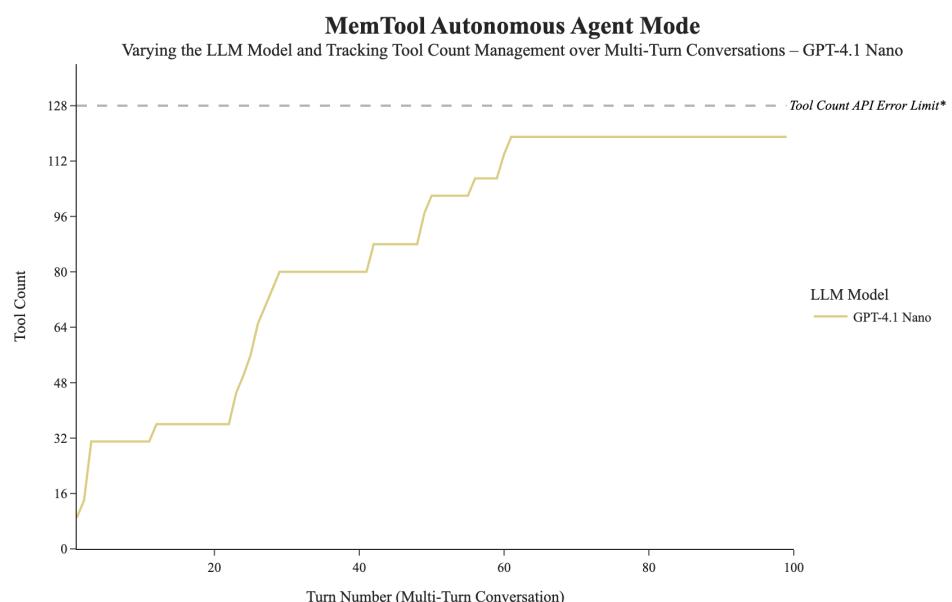


Figure 13: GPT-4.1 Nano — Tool Count over Time (Autonomous Agent Mode)

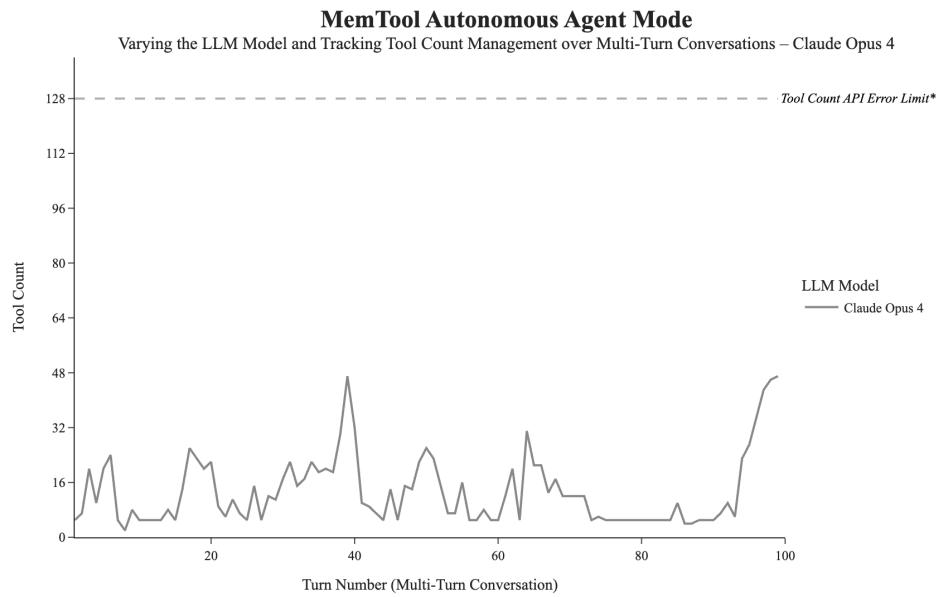


Figure 14: Claude Opus 4 — Tool Count over Time (Autonomous Agent Mode)

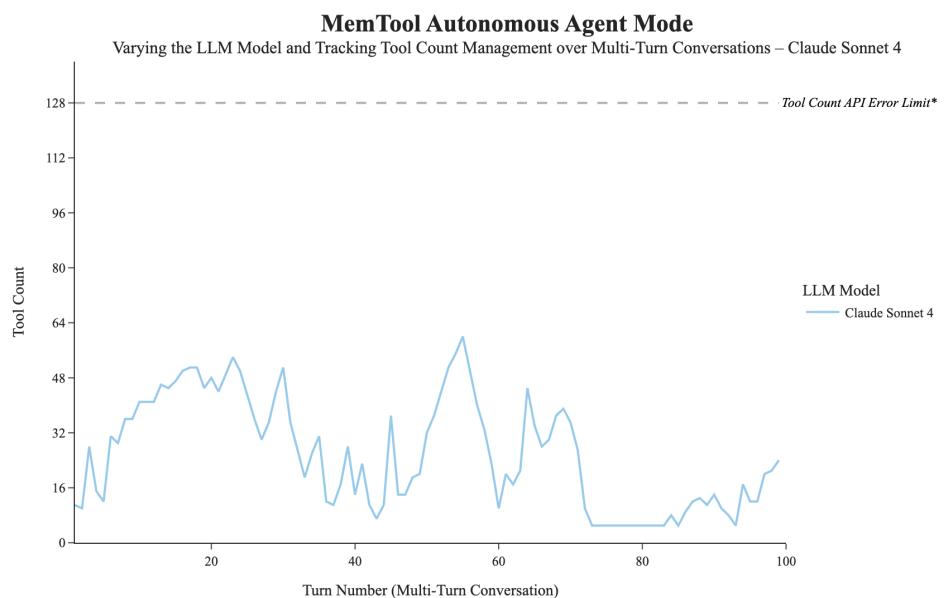


Figure 15: Claude Sonnet 4 — Tool Count over Time (Autonomous Agent Mode)

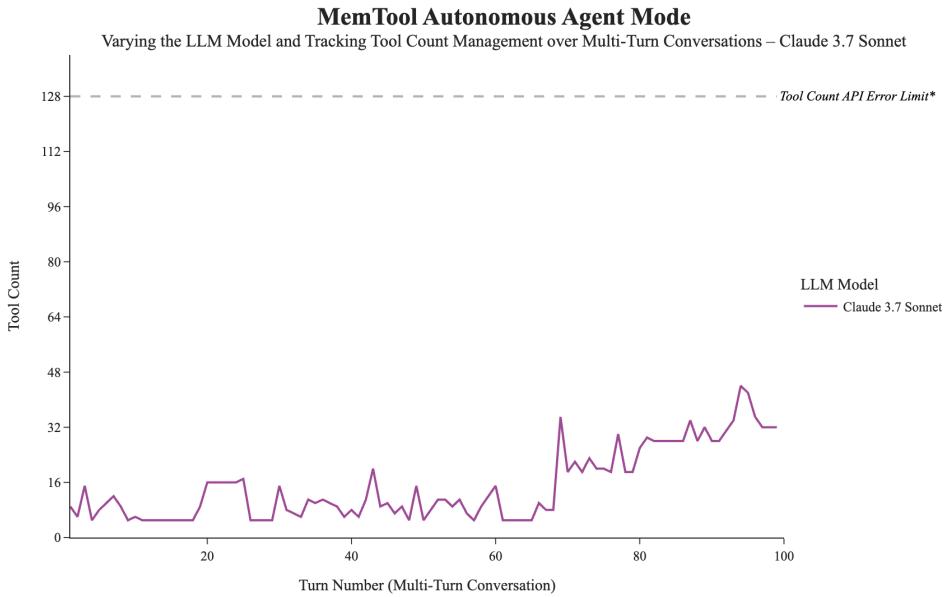


Figure 16: Claude 3.7 Sonnet — Tool Count over Time (Autonomous Agent Mode)

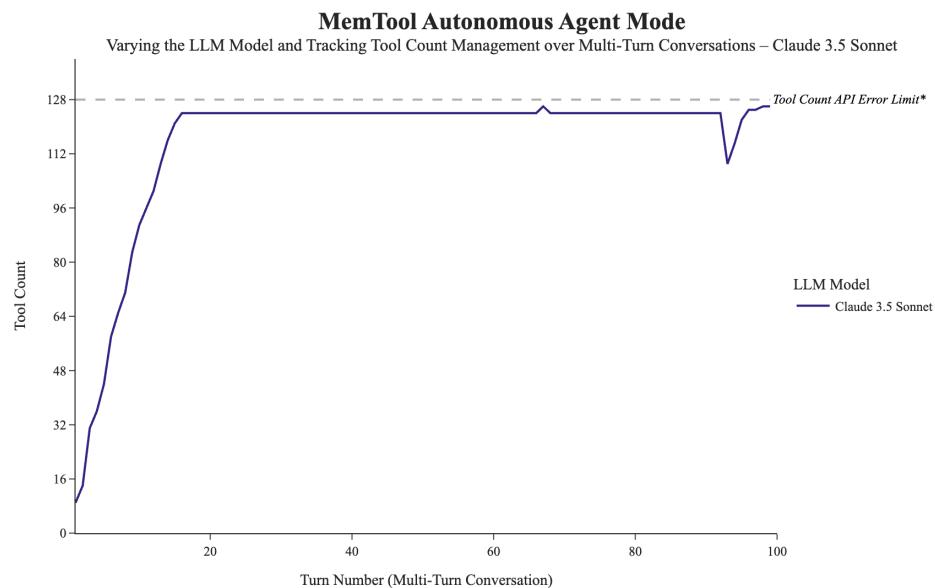


Figure 17: Claude 3.5 Sonnet — Tool Count over Time (Autonomous Agent Mode)

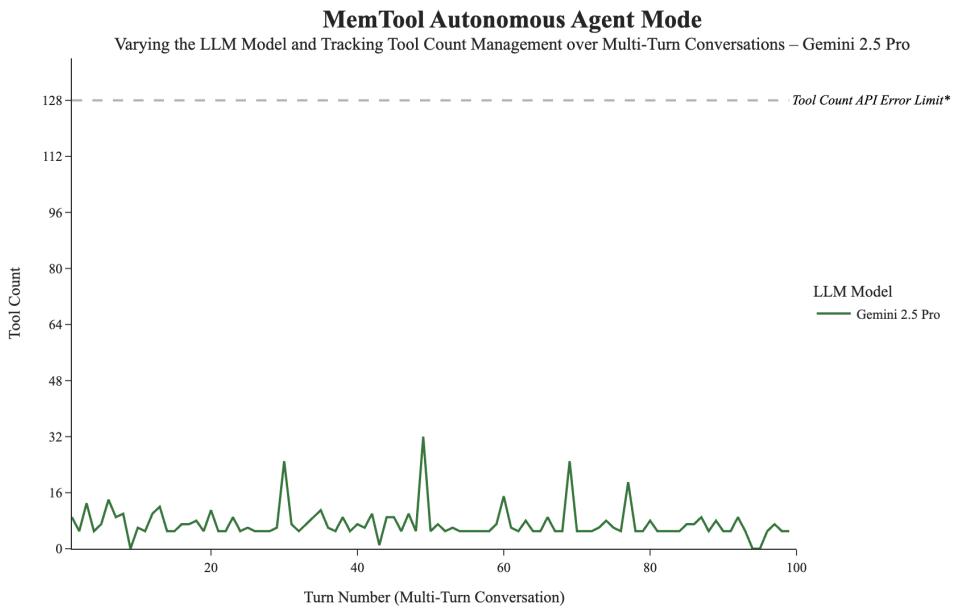


Figure 18: Gemini 2.5 Pro — Tool Count over Time (Autonomous Agent Mode)

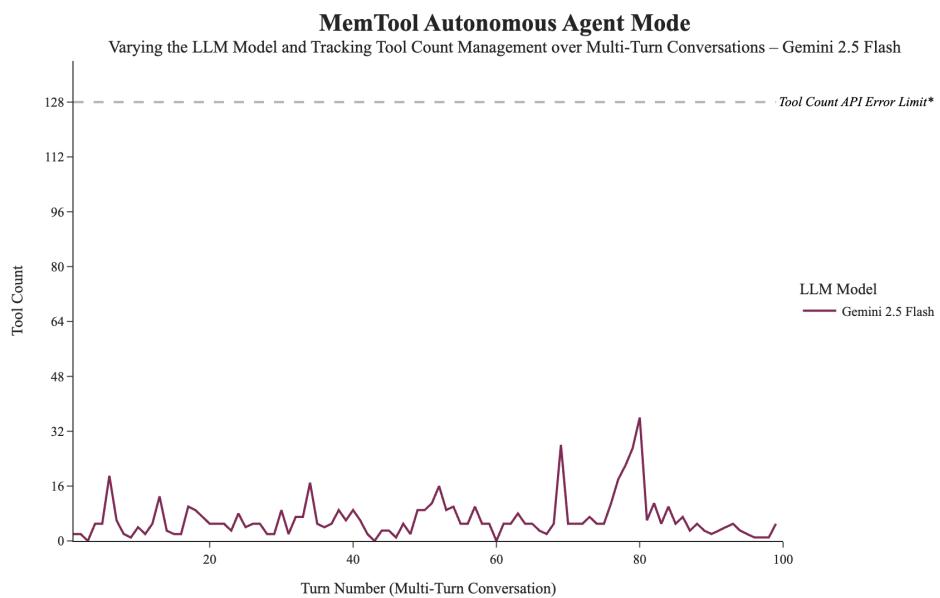


Figure 19: Gemini 2.5 Flash — Tool Count over Time (Autonomous Agent Mode)

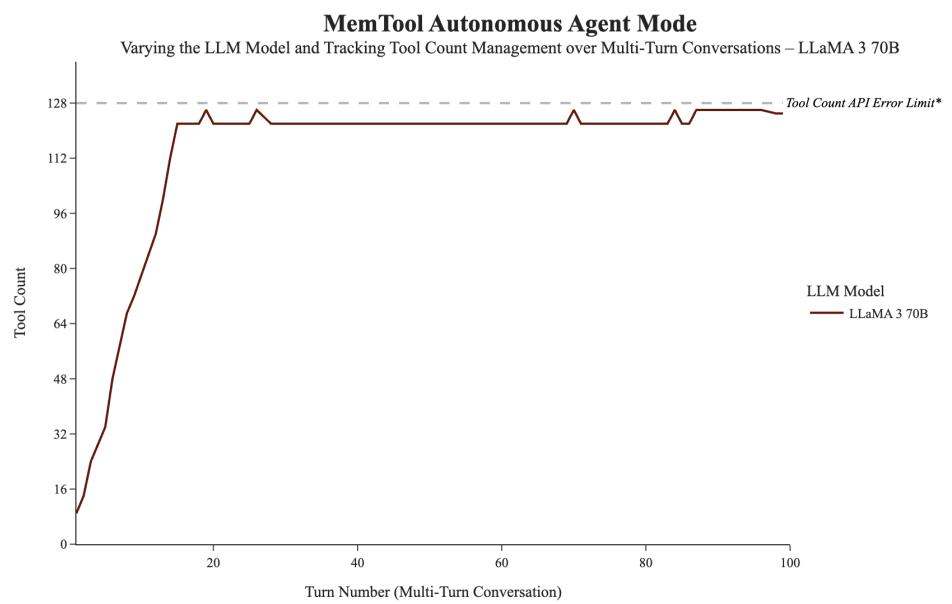


Figure 20: LLaMA 3 70B — Tool Count over Time (Autonomous Agent Mode)