TOOLMEM: Enhancing Multimodal Agents with Learnable Tool Capability Memory

Yunzhong Xiao[1]   **Yangmin Li**[1]   **Hewei Wang**[1]   **Yunlong Tang**[2]   **Zora Zhiruo Wang**[1]

[1]Carnegie Mellon University   [2]University of Rochester

yunzhonx@alumni.cmu.edu, yangmin2@alumni.cmu.edu, heweiw@alumni.cmu.edu,
yunlong.tang@rochester.edu, zhiruow@andrew.cmu.edu

Abstract

Agents utilizing tools powered by large language models (LLMs) or vision-language models (VLMs) have demonstrated remarkable progress in diverse tasks across text and visual modalities. Unlike traditional tools such as calculators, which give deterministic outputs, neural tools perform uncertainly across task scenarios. While different tools for a task may excel in varied scenarios, existing agents typically rely on fixed tools, thus limiting the flexibility in selecting the most suitable tool for specific tasks. In contrast, humans snowball their understanding of the capabilities of different tools by interacting with them, and apply this knowledge to select the optimal tool when solving a future task. To build agents that similarly benefit from this process, we propose TOOLMEM that enables agents to develop memories of tool capabilities from previous interactions, by summarizing their strengths and weaknesses and storing them in memory; at inference, the agent can retrieve relevant entries from TOOLMEM, and select the best tool to solve individual tasks more accurately. We evaluate TOOLMEM on learning varied text generation and text-to-image generation neural tools. Compared to no-memory, generic agents, we find TOOLMEM-augmented agents predict tool performance 14.8% and 28.7% more accurately across text and multimodal generation scenarios. Moreover, TOOLMEM facilitates optimal tool selection among multiple choices by 21% and 24% absolute increases in respective scenarios.

1 Introduction

Recent advances in agents have drastically reshaped the landscape of generative tasks, especially by utilizing powerful tools supported by large language models (LLMs) [Schick et al., 2023] or vision-language models (VLMs) [Gao et al., 2025, Carrasco et al., 2025, Radford et al., 2021]. These agents supported with task-specific neural tools have demonstrated impressive capabilities across various downstream applications such as instruction following and reasoning in text modality [Kim et al., 2025, Ouyang et al., 2022], as well as visual understanding and editing in cross-modal scenarios [Shen et al., 2023, Wang et al., 2024a, Tang et al., 2025a]. Unlike traditional tools such as `calculator` or `sql_executor` that always produce deterministic outputs for a given input, tools supported by neural models vastly expand the spectrum of tasks that cannot be tackled by deterministic tools, such as answering ad-hoc questions [Chen et al., 2017, Karpukhin et al., 2020] or search relevant news on the web [Nakano et al., 2022, Dunn et al., 2017]. Nonetheless, what are brought along with this wider applicability of neural tools is the uncertainty of their outputs—there is no guarantee that a `QA` tool can answer an arbitrary question correctly, especially when the answer to that question is open-ended [Lewis et al., 2021] or changes over time [Guu et al., 2020].

Moreover, current agents typically solve tasks using a pre-designated set of tools by human experts, and only get to learn to use these tools at test time by contextualizing on static textual descriptions

Figure 1: Example of identifying better-performing tools based on learned TOOLMEM. For example, compared to the image generation tool instantiated by *Midjourney*, *SDXL-Base* tool is better at rendering short text such as the queried "Eco Market". We thus store this information in TOOLMEM and prioritize selecting *SDXL-base* in later tasks that ask to render short texts.

for tool functionalities [Qin et al., 2023b, Guo et al., 2024]. More often than not, the agent may be provided with multiple tools possessing similar functionalities [Tang et al., 2023, Li et al., 2023], thus making it harder to distinguish the best tool to use for a certain task, because the agent has no prior knowledge about the individual expertise of these seemingly similar tools [Yao et al., 2023]. Taking Figure 1 as an example, the two text-to-image tools have different expertise: while both might be good at representing lively environment, the later one may excel at rendering text. Given the task of generating image with text, the agent better pick the later tool to ensure better performance.

Human gradually learn to distinguish and select between tools by interacting with tools and gathering knowledge about their individual properties through past experiences. Recent methods have explored some techniques to gather explicit knowledge from experiences by reflecting on [Shinn et al., 2023] and refining [Madaan et al., 2023] generated solutions, or constructing short-term workflows [Wang et al., 2024c] and long-term memory [Xu et al., 2025], and even curating new tools to assist target downstream tasks [Wang et al., 2023, 2024b]. Besides explicitly storing the knowledge, approaches like Toolformer [Schick et al., 2023] and ToolLLM [Qin et al., 2023a] also explored parametric updates via self-supervised fine-tuning and instruction tuning on tool-using experiences. However, these methods focus on learning task-specific procedures with a fixed tool, instead of distinguishing between a group of functionally similar tools to optimize downstream tool selection and performance. More concretely, agents lack a mechanism to build and update an internal, dynamic memory that encapsulates the strengths and weaknesses of diverse generative tools.

To bridge this gap, we introduce TOOLMEM, a framework that empowers agents to learn and apply tool-specific capability memories (§2). TOOLMEM is built upon three core components: (1) a structured capability memory initialized with a taxonomy that categorizes tool behaviors by proficiency levels, enabling consistent updates and targeted retrieval; (2) a feedback generation process that evaluates tool outputs using either human annotations or automated metrics—such as LLM-based judgment scores—to extract fine-grained insights into tool performance; and (3) a dynamic memory update mechanism that incorporates new experiences through retrieval-augmented generation (RAG), allowing the agent to refine, revise, and expand its tool knowledge over time. During inference, TOOLMEM-augmented agents retrieve relevant memory entries based on the current task and inject them into the input context, enabling more accurate tool selection and solution generation.

We first study: *Can agents learn and estimate tool capabilities through interaction?* (§3) We construct TOOLMEM about image generation tools on GENAI-BENCH [Li et al., 2024] and various text-oriented tools on BIGGEN BENCH [Kim et al., 2025], then evaluate if TOOLMEM-augmented agent can accurately estimate the performance of these tools. We find that TOOLMEM can predict solution quality using varied tools 14.8–28.7% more accurately than GENERIC agent without prior knowledge about tools.

Next, we ask: *Can agents leverage* TOOLMEM *to optimize performance through tool selection?* (§4) We examine TOOLMEM-augmented agents on selecting the best-performing tool for individual tasks

2

among multiple tool candidates, and find it outperforming GENERIC agent approach by 21% and 24% on GENAI-BENCH and BIGGEN BENCH benchmarks (§4).

In short, our work builds memory-adaptive tool-using agents, and uniquely addresses the dynamic nature of neural tools by enabling agents to grasp an evolving understanding of accessible tools.

2 TOOLMEM: Learning Tool Capability Memory

In this section, we introduce our TOOLMEM framework that initializes a structured memory (§2.1), learns and updates tool capability knowledge through experiences (§2.2), and eventually solves new tasks according to retrieved memory entries (§2.3). Overall, TOOLMEM equips agents to build tool capability memories progressively learned from past tool usage experiences, and leverages this dynamic knowledge repository to enhance its decision-making.

2.1 Structured Memory Initialization

The evolution of TOOLMEM is an iterative process that builds upon previous states. As such, the initial memory plays a critical role in guiding subsequent developments.

Our goal is for agents π to learn the capabilities of a set of tools $T = \{t\}$ with similar functionalities to solve a particular task (e.g., text-to-image generation, instruction following), where each tool t is instantiated with a neural model (e.g., Midjourney, GPT). For each tool t, we initialize its capability memory $\mathcal{M}_t = \emptyset$ and aim to collect a set of knowledge entries $M_t = \{m_t\}$, each describing some properties of tool behaviors in natural language (NL). While a tool t can be good at certain scenarios and bad at others, we categorize memory entries based on the varied proficiency levels the agent possesses, namely $C = \{$ *proficient at* $(^p)$, *good at* $(^g)$, *bad at* $(^b)$, *weak at* $(^w)$ $\}$. Moreover, these proficiency-aware categories can be associated with concrete numerical measures of tool performance, $+2, +1, -1, -2$, when more accurate measures are useful. We thus denote TOOLMEM as the union of all memory categories $\mathcal{M} = \cup_{c \in C} \mathcal{M}^c$. This structured memory initialization offers flexibility in memory update and retrieval, which we will provide more details on next.

2.2 Learning Tool Capabilities from Experiences

Constructing Memory from Experiences   Agents learn more about the tool capabilities by interacting with them in task-solving experiences. More concretely, the agent begins by receiving a task q (e.g., an NL instruction), as well as a corresponding solution s_t using a designated tool t (e.g., an image generated by a text-to-image tool t, an answer generated by a text generation tool t). As an informative learning signal of this task, a reward system R provides feedback $r_t = R(q, s_t)$ on the quality of the solution s_t in tackling the task q, which could be numerical quality measures (e.g., 1-5 Likert scale) or NL feedback (as exemplified in Figure 2). Together, the task q, tool-using solution s_t, and its quality feedback r_t form an experience $e_t = (q, s_t, r_t)$ from interaction with tool t, upon which we can construct capability memories about. In our study, we focus on building TOOLMEM from experiences annotated by human experts for image generation tools and by llm-as-a-judge for text generation tools. These annotations are provided as part of the training datasets, typically accompanied by high-quality solutions and reliable feedback. We introduce a memory induction module I_{LM} instantiated with an LM, to summarize the capabilities of tool t from a pertaining experience e_t via $I_{LM}(e_t) \rightarrow m_t$.

Updating Agent Memory   Given a new memory entry m derived from an experience e, a naive memory update would directly augment the existing memory repository by $\mathcal{M} \leftarrow \mathcal{M} \cup \{m\}$. However, this simplistic approach often introduces redundancy and inflates computational overhead. Specifically, entries that are highly similar or overlapping may accumulate without being appropriately consolidated, and prior incomplete or suboptimal entries might persist without receiving necessary updates.

To enable flexible memory refinement capabilities, we introduce a structured and refined memory updating procedure. Formally, we denote the agent's current memory as $\mathcal{M} = \cup_{c \in C} \mathcal{M}^c$ partitioned into C categories introduced in §2.1. Given the newly constructed experience $e = (q, s, r)$, we first perform a targeted retrieval within each memory category $\mathcal{M}^c$ to identify the top-k most semantically
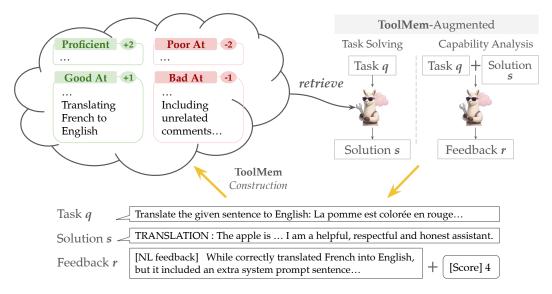
Figure 2: Illustration of the TOOLMEM construction process from past experiences (left), as well as TOOLMEM-augmented task-solving and capability analysis at downstream (right).

relevant memory entries. $\mathcal{M}^c_{\text{retrieved}} = \textit{Retriever}(e, \mathcal{M}^c), \forall c \in C$ where $\textit{Retriever}(\cdot, \cdot)$ calculates some similarity measure (e.g., cosine similarity) computed between embedded memory entries.

Next, we construct a condensed contextual set from these retrieved entries across all categories: $\mathcal{M}_{\text{retrieved}} = \bigcup_{c \in C} \mathcal{M}^c_{\text{retrieved}}$. Subsequently, the condensed contextual set $\mathcal{M}_{\text{context}}$ along with the new experience $e = (q, s, r)$ is passed into the TOOLMEM induction module I to for entry refinement $\mathcal{M}_{\text{updated}} = I(\mathcal{M}_{\text{retrieved}}, e)$ then memory update $\mathcal{M} \leftarrow (\mathcal{M} \setminus \mathcal{M}_{\text{context}}) \cup \mathcal{M}_{\text{updated}}$ The refinement process involves adding novel insights, updating incomplete entries, removing redundant information, and merging semantically related entries (see exact prompts in §B.

2.3 Solving Tasks with Retrieved Memory

Analogous to how humans draw upon past experiences when confronted with a new task, for a given task q' at test time, our task-solving agents can (i) retrieve top-k relevant memory entries from TOOLMEM utilizing a retriever model $\textit{Retriever}(q', \mathcal{M}) \rightarrow \mathcal{M}_{q'}$, and then (ii) augmented these relevant memory entries to input contexts to generate solutions $\pi(q', \mathcal{M}_{q'}) \rightarrow s'$ or estimate tool performance $\pi(q', \mathcal{M}_{q'}) \rightarrow r'$. As TOOLMEM contains multiple categories $\mathcal{M} = \cup_c \mathcal{M}^c$, we separately retrieve top-k relevant entries from each category $\mathcal{M}^c$ and provide all of them in context. This RAG strategy addresses two key challenges: (1) relevance filtering: focusing on contextually useful entries and discarding irrelevant ones; (2) scalability: preventing computation overload by restricting retrieval to a small number of entries per category.

3 Experiments: Tool Performance Prediction

In this section, we first introduce the baseline approaches in comparison to our TOOLMEM (§3.1), then demonstrate TOOLMEM's effectiveness on two representative tasks — text generation (§3.2) and text-to-image generation evaluation (§3.3). We focus on TOOLMEM usage under the *performance prediction* scenario in this section. Later in §4, we further demonstrate its utility when extrapolating to *model selection* under accuracy and efficiency considerations.

3.1 Baseline and TOOLMEM Approaches

We introduce the two baseline approaches and our TOOLMEM method for experimentation.

Generic Agent   Our first baseline is a vanilla agent without tool-specific memory beyond general information (e.g., tool's name, basic skills, input/output) at inference. This setup is similar to the

current tool-selection approach adopted by state-of-the-art generative agents [Wang et al., 2024a]. Agents must rely heavily on the parametric knowledge acquired through the training of their backbone LM to estimate tool capabilities.

Few-Shot Memory Without the need to induce TOOLMEM, we propose another stronger baseline on top of GENERIC that also retrieves top-12 training examples, i.e., (task, score, rubric) triplets, that are most relevant to the test example at hand. We denote this the FEW-SHOT setting and compare it to our TOOLMEM-augmented LM, to show the effectiveness of derived capability memory against implicit information from raw examples.

TOOLMEM Instead of employing memory with generic or raw experiences, our TOOLMEM contains carefully induced and updated memory from past experiences. Particularly, we induce memory entries from the same set of training examples used for retrieval in FEW-SHOT setting, and test TOOLMEM-augmented LM on the same test set.

For generations, we use GPT-4o (the `gpt-4o-mini-2024-07-18` checkpoint) for all approaches by default; we use a temperature $t = 0.0$ and sample $n = 1$ output. For retrieval, we use OpenAI's `text-embedding-ada-002`[1] model to generate embeddings and implement RAG using ChromaDB[2] that measures task-memory similarity using (negative) cosine distance. During TOOLMEM construction, we retrieve $k = 6$ entries from each memory category and perform memory update on them. At test time, the agent retrieves $k = 12$ entries from each memory category and grounds on them to generate task solutions; this setting scores the best in our ablation study on k value choices in §A.

3.2 Text Generation

Dataset: BIGGEN BENCH We evaluate TOOLMEM on diverse text-generation tasks from the BIGGEN BENCH benchmark [Kim et al., 2025], which includes 696 task instances spanning 8 capability dimensions, such as reasoning, safety, and tool usage, evaluated across 103 language models. We split the data into 211 training and 485 testing examples for our experiments.

Each task q consists of a natural language instruction composed of a system prompt and user input. The solution s_t is the textual response generated by a specific language model (tool) t. The quality feedback $r_t = R(q, s_t)$ includes a numerical score (on a 5-point Likert scale), a detailed score rubric, and NL feedback by GPT-4. We use feedback generated by GPT-4 to construct TOOLMEM for each tool, which are then used to improve downstream score prediction accuracy for new prompts.

We focus on six language model tools with varying capabilities and accessibility levels: (1) Proprietary models: *Claude-3-Sonnet*, *GPT-3.5-Turbo*;[3] (2) Top-tier open-source models: *Meta-Llama-3-70B-Instruct*, *Qwen-110B-Chat*; (3) Smaller models: the strongest sub-7B model, *Gemma-1.1-2B-It*, and the smallest chat-capable model, *Qwen1.5-0.5B-Chat*.

Evaluation: Score Prediction We evaluate TOOLMEM-augmented LMs under a score prediction evaluation setup on how accurately they measure the tool generation quality. In our vanilla evaluation setup, each method is tasked with predicting a quality score for the generated image or text given a task prompt. The goal is to predict the quality score of a prompt-response pair q, using the task-specific rubrics provided in the dataset, also rated on a 1–5 Likert scale. We take the `gpt4-04-turbo-score` as ground-truth, as no human scores are provided.

We measure the consistency of predicted scores s_{pred} to the ground-truth scores s by (i) the average mean absolute error (MAE) across all N test examples $MAE = \frac{1}{N}\sum_{i=1}^{N}|s_{pred}^{(i)} - s^{(i)}|$, (ii) the root mean squared error by $RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(s_{pred}^{(i)} - s^{(i)})^2}$, (iii) the Pearson correlation coefficient between predicted and true scores: $Pearson = \frac{\sum_{i=1}^{N}(s_{pred}^{(i)} - \bar{s}_{pred})(s^{(i)} - \bar{s})}{\sqrt{\sum_{i=1}^{N}(s_{pred}^{(i)} - \bar{s}_{pred})^2}\sqrt{\sum_{i=1}^{N}(s^{(i)} - \bar{s})^2}}$, where $\bar{s}_{pred}$ and $\bar{s}$ are the means of the predicted and ground-truth scores, respectively.

---

[1] https://platform.openai.com/docs/models/text-embedding-ada-002
[2] https://www.trychroma.com/
[3] Specific checkpoints are: `claude-3-sonnet-20240229`, `gpt-3.5-turbo-1106`.

Results and Analysis   As shown in Table 1, TOOLMEM reduces MAE by 14.8% and RMSE by 14.5% on average than GENERIC baseline; and improves ranking fidelity—Pearson correlation increases by 76.7%.

Tools with varied sizes   For the two smallest and least capable tools (*Qwen1.5-0.5B-Chat* and *gemma-1.1-2B-it*), TOOLMEM boosts Pearson correlation significantly, lifting it from near-random (-0.007 and 0.120) to 0.405 and 0.324, highlighting the valuable, tool-specific priors that are initially lack for these lightweight models and are augmented through TOOLMEM. Even for the strongest open-weight tool (*Meta-Llama-3-70B-Instruct*), TOOLMEM achieves a notable 38.9% increase in Pearson correlation, from 0.175 to 0.243, while slightly reducing RMSE.

Top closed tools have diminishing returns   For *gpt-3.5-turbo-1106* and *claude-3-sonnet*, baseline predictions already achieve relatively strong Pearson correlations (≥ 0.224), while TOOLMEM still manages to enhance correlation for GPT by 48.2%.

Instability of Raw Experiences   Across all tools, the FEW-SHOT strategy underperforms GENERIC across metrics and often doubles the error for the two top models (e.g., MAE $+95\%$ on Meta-Llama-3-70B). In contrast, TOOLMEM delivers robust improvements without such regressions, underscoring the value of a compact, induced capability memory over raw example reference.

Table 1: Performance prediction of text-generation tools.

Tool	Method	MAE ↓	RMSE ↓	Pearson ↑
Llama3 70B	GENERIC	**0.571**	1.036	0.175
	FEW-SHOT	1.115	1.566	0.163
	TOOLMEM	0.610	**1.025**	**0.243**
Qwen 110B	GENERIC	0.602	0.991	0.228
	FEW-SHOT	1.155	1.605	0.146
	TOOLMEM	**0.600**	**0.988**	**0.246**
Gemma 2B	GENERIC	1.004	1.528	0.120
	FEW-SHOT	1.008	1.429	0.271
	TOOLMEM	**0.932**	**1.252**	**0.324**
Qwen1.5 0.5B	GENERIC	1.769	2.136	-0.007
	FEW-SHOT	1.381	1.762	0.102
	TOOLMEM	**1.080**	**1.400**	**0.405**
Claude3	GENERIC	**0.546**	0.983	**0.313**
	FEW-SHOT	1.085	1.555	0.140
	TOOLMEM	0.551	**0.979**	0.310
GPT-3.5	GENERIC	0.625	1.095	0.224
	FEW-SHOT	0.924	1.392	0.204
	TOOLMEM	**0.588**	**0.998**	**0.332**

3.3   Text-to-Image Evaluation

Dataset: GENAI-BENCH   We evaluate text-to-image generation tools using the GENAI-BENCH benchmark [Li et al., 2024], which contains 1,600 diverse natural language instructions spanning compositional scenarios such as object counting, spatial relations, and negation. We randomly sample 200 training and 800 testing examples for experiments.

Each instruction is paired with six images generated by different tools, including *MidJourney*, *DALL·E 3*, *SDXL Turbo*, *DeepFloyd I-XL v1*, *SDXL 2.1*, and *SDXL Base*. Each text-image pair is annotated by three human raters on a 1–5 Likert scale, reflecting how well the image aligns with the prompt. As multiple human annotators can introduce variance in prediction scores, we take the first annotator's rating as the ground-truth score to ensure label consistency. Our goal is to predict the quality score r_t based on the task input q (image generation prompt) and the solution s_t (the generated image using tool t) with agents having no memory, few-shot examples, or TOOLMEM.

Evaluation: Score & Description Prediction   We similarly evaluate TOOLMEM-augmented LM to predict the alignment score of a prompt-image pair q, rated on a 5-point Likert scale.

While score prediction may be subject to individual human judgment and less robust, we alternatively evaluate image quality through text-image alignment: given a text prompt q (e.g. "Three birds flying in the sky"), we ask agent to predict a text description d_{pred} for the image s^{image} generated by tool t. We then evaluate the alignment between the image and the predicted description using the VQA score [Li et al., 2024], computed as $s_{vqa} = VQA(s^{image}, d_{pred})$. A higher VQA score indicates stronger alignment between the image and the textual description. We report average VQA Score in Table 2.

Results and Analysis Table 2 lists the six image–generation tools in descending order of capability. Overall, TOOLMEM consistently improves over the GENERIC baseline by decreasing MAE by 28.7% and RMSE by 26.6% across tools. The gains, however, are not uniform:

Top-tier closed models (DALLE 3, Midjourney 6). TOOLMEM beats the GENERIC agent (–7.4% and –12.2% MAE, respectively) but trails FEW-SHOT by 3.7% to 6.7%. This suggests that when the underlying generator is highly capable, large in-context exemplars are sufficient and an external capability memory brings limited extra benefit. However, TOOLMEM delivers consistent gains in VQA scores, with improvements of +2.1% for DALLE 3 and +2.0% for Midjourney 6. These steady gains highlight TOOLMEM's ability to fine-tune descriptions even for top-performing models, effectively capturing subtle yet crucial details that baseline methods might overlook.

Mid/low-tier open models (DeepFloyd_I_XL_v1, SDXL-Base, SDXL-Turbo, SD-2-1). Here TOOLMEM clearly shows its strength: it achieves the lowest MAE and RMSE on all four models, reducing MAE by 26.1–42.6% and RMSE by 25.5–36.6% relative to GENERIC, and outperforming FEW-SHOT by 6.7–18.0% MAE.

TOOLMEM also exhibits significant improvements for VQA scores, particularly in mid- and low-tier models where gains reach +3.17% for SDXL-Turbo and +4.23% for SDXL-2-1. This pattern reflects TOOLMEM's ability to adapt to each model's specific strengths and weaknesses, resulting in more accurate and context-aware descriptions. The learnable memory is especially effective for weaker tools that lack robust prior knowledge, helping narrow the gap between less capable generators and their stronger counterparts.

In summary, TOOLMEM offers the greatest benefit where it is most

Table 2: Predicting text-to-image tool performance on GENAI-BENCH.

Tool	Method	Score Prediction		Desc. Pred.
		MAE ↓	**RMSE ↓**	**VQA Score (%) ↑**
DALLE 3	GENERIC	1.009	1.405	81.30
	FEW-SHOT	**0.875**	**1.180**	81.64
	TOOLMEM	0.934	1.204	**82.98**
Midjourney 6	GENERIC	1.085	1.434	81.19
	FEW-SHOT	**0.919**	1.227	82.21
	TOOLMEM	0.953	**1.212**	**82.84**
DeepFloyd	GENERIC	1.334	1.703	79.17
	FEW-SHOT	1.058	1.393	79.74
	TOOLMEM	**0.986**	**1.268**	**81.30**
SDXL-Base	GENERIC	1.481	1.841	80.47
	FEW-SHOT	1.111	1.422	81.37
	TOOLMEM	**0.978**	**1.285**	**82.03**
SDXL-Turbo	GENERIC	1.566	1.932	79.22
	FEW-SHOT	1.101	1.400	80.01
	TOOLMEM	**0.996**	**1.331**	**81.73**
SDXL-2-1	GENERIC	1.645	1.981	75.70
	FEW-SHOT	1.151	1.477	75.91
	TOOLMEM	**0.944**	**1.255**	**78.90**

needed—narrowing the gap between weaker generators and their stronger peers—while remaining competitive on state-of-the-art closed models. Its dual effectiveness in both score prediction and precise description prediction highlights its capacity to enhance model evaluation by leveraging learned memory, ultimately providing higher VQA scores and more precise model-specific descriptions.

4   Experiments: Performant Tool Selection

In addition to predicting a single tool's performance, we can select the better-performing tool among multiple choices based on the TOOLMEM learned for these tools. We construct TOOLMEM and evaluate on the same datasets on GENAI-BENCH and BIGGEN-BENCH, and compare the same three methods (GENERIC, FEW-SHOT, TOOLMEM) as in §3.

Evaluation Metrics Let s_A and s_B denote the ground-truth scores of tools A and B, and $s_{pred,A}$ and $s_{pred,B}$ be their predicted scores. Since for equal pairs selecting either tool is equally acceptable, we evaluate tool selection performance over the subset of *unequal* pairs $D = \{i : s_A^i \neq s_B^i\}$. For each $i \in D$, we define $TP_< = |\{i \in D : s_A^i < s_B^i \wedge s_{pred,A}^i < s_{pred,B}^i\}|$, $P_< = |\{i \in D : s_{pred,A}^i < s_{pred,B}^i\}|$, and $R_< = |\{i \in D : s_A^i < s_B^i\}|$. We define $TP_>$, $P_>$, and $R_>$ analogously for the ">"

Table 3: Selecting better text generation tools. Best results for each tool pair are bold.

Tool Pair		GENERIC			FEW-SHOT			TOOLMEM		
		$F_{1<}$	$F_{1>}$	Acc.	$F_{1<}$	$F_{1>}$	Acc.	$F_{1<}$	$F_{1>}$	Acc.
Llama3	Claude3	0.00	0.09	0.03	0.20	0.06	0.07	**0.28**	**0.10**	**0.11**
Claude3	GPT-3.5-Turbo	0.03	0.06	0.03	**0.21**	0.22	0.13	0.06	**0.30**	**0.15**
Qwen 110B	Qwen 0.5B	0.00	0.13	0.07	0.00	0.09	0.05	0.00	**0.77**	**0.62**
Llama3	Qwen 110B	0.04	0.19	0.07	0.12	**0.17**	0.08	**0.16**	0.14	**0.09**
Gemma 2B	Qwen 0.5B	0.10	0.04	0.03	0.05	0.12	0.06	**0.06**	**0.39**	**0.23**
GPT-3.5-Turbo	Gemma 2B	0.00	0.24	0.11	**0.08**	0.27	0.14	0.00	**0.62**	**0.39**
Average		0.03	0.12	0.06	**0.11**	0.16	0.09	0.09	**0.39**	**0.27**

case. Then we compute $F_{1<} = \frac{2TP_<}{P_< + R_<}$, $F_{1>} = \frac{2TP_>}{P_> + R_>}$, and $\text{Acc} = \frac{TP_< + TP_>}{|D|}$, where $F_{1<}$ (resp. $F_{1>}$) is the F_1 score for predicting $s_A < s_B$ (resp. $s_A > s_B$) over the non-equal set D, and Acc is the proportion of cases where the predicted and ground-truth preference directions match.

Text Generation Evaluation On BIGGEN BENCH, we evaluate six tool pairs as listed in Table 3. The selected pairs include comparisons between top open-source and closed-source models (e.g., Meta-Llama-3-70B vs. Claude-3-Sonnet), similarly strong closed models (Claude-3-Sonnet vs. GPT-3.5-Turbo), and strong open models (Meta-Llama-3-70B vs. Qwen110B), models from the same family with large capacity gaps (Qwen110B vs. Qwen0.5B), and small-scale model comparisons (e.g., Gemma-2B vs. Qwen0.5B). These combinations cover diverse settings across model families, sizes, and access types.

As shown in Table 3, we find that TOOLMEM offers clear and consistent accuracy advantages over GENERIC and FEW-SHOT settings by 21% and 18% on average. TOOLMEM increases the $F_{1<}$ score, which measures the ability to identify the cases where weaker tool excels, increases from 0.00 to 0.28 from GENERIC in the Meta-Llama-3-70B vs. Claude-3-Sonnet comparison, on average it improves from GENERIC's 0.03 to 0.09. $F_{1>}$ improves more significantly from 0.12 to 0.39 on average, with TOOLMEM achieving the highest score on five of the six tool pairs. TOOLMEM boosts performance most substantially when tool pairs exhibit large capability gaps, e.g., TOOLMEM improves accuracy by +0.55 for Qwen110B vs. Qwen0.5B and +0.28 for GPT-3.5-Turbo vs. Gemma-2B. Even for tool pairs with smaller capability gaps (e.g., Meta-Llama-3-70B vs. Claude-3-Sonnet), TOOLMEM maintains an edge over both baselines.

Text-to-Image Generation Evaluation On GENAI-BENCH, we examine six representative tool pairs (in Table 4) to test whether TOOLMEM-augmented LM can accurately identify the superior tool for individual tasks. More capable model is on the left side of each pair. To showcase TOOLMEM utility in varied scenarios, we choose varied types of tool pairs. Specifically: (i) distinguishing comparably top-performing proprietary tools (DALL·E 3, Midjourney) and against open-weight tools (DALL·E 3, DeepFloyd), (Midjourney, DeepFloyd); (ii) comparing tools with huge gaps from the same (SDXL-Base, SDXL-Turbo) and different model families (DALL·E 3, SDXL-2.1).

Table 4 shows the consistent edge of TOOLMEM across varied tool pairs. Averaged over five tool pairs, accuracy on unequal pairs climbs from 0.09 (GENERIC) and 0.27 (FEW-SHOT) to 0.33 with TOOLMEM, an absolute gain of +0.24 over GENERIC (+266%) and +0.06 over FEW-SHOT (+22%).

The hardest class $F_{1<}$, which measures the situation to find worse cases for a more capable model, improves most dramatically—rising from 0.09 to 0.32 compared to GENERIC and FEW-SHOT approaches. Meanwhile, $F_{1>}$ also edges up from 0.15 to 0.46 when augmented with TOOLMEM.

The benefit brought by TOOLMEM shows more prominently as the capability gap of the tool pair grows. For instance, between comparable DALL·E 3 and Midjourney tools, the gain of Acc. for TOOLMEM compare to GENERIC is relatively small (+0.12), yet comparing DALL·E 3 to the weaker SDXL-2.1, TOOLMEM boosts accuracy from 0.08 to 0.53, achieves much larger gains.

Even within the same model family, TOOLMEM can effectively tell tools' capability differences, showcased by the +0.18 gain in the SDXL-Base vs. SDXL-Turbo comparison. These results confirm that TOOLMEM consistently chooses better tools when the tool choices exhibit different properties.

Table 4: Selecting better text-to-image generation tools. Best results for each tool pair are bold.

Tool Pair		GENERIC			FEW-SHOT			TOOLMEM		
		$F_{1<}$	$F_{1>}$	Acc.	$F_{1<}$	$F_{1>}$	Acc.	$F_{1<}$	$F_{1>}$	Acc.
DALLE 3	DeepFloyd	0.07	0.19	0.10	0.19	**0.52**	0.31	**0.31**	**0.52**	**0.35**
DALLE 3	Midjourney	0.00	0.31	0.14	0.17	0.42	0.23	**0.26**	**0.43**	**0.26**
DALLE 3	SD-2-1	0.00	0.16	0.08	0.20	0.66	0.47	**0.32**	**0.72**	**0.53**
SDXL-Base	SDXL-Turbo	0.14	0.03	0.05	0.24	**0.29**	0.18	**0.36**	0.26	**0.23**
Midjourney	DeepFloyd	0.23	0.07	0.08	0.26	0.28	0.18	**0.35**	**0.37**	**0.26**
Average		0.09	0.15	0.09	0.21	0.43	0.27	**0.32**	**0.46**	**0.33**

5 Related Work

Tool-Using Multimodal Agents Multimodal agents combine LLMs and LVLMs to solve complex tasks across modalities [Xie et al., 2024]. Early systems like Visual Programming [Gupta and Kembhavi, 2022] and ViperGPT [Surís et al., 2023] use LLMs to compose expert modules for visual reasoning. Visual ChatGPT [Wu et al., 2023] and HuggingGPT [Shen et al., 2023] improve agent flexibility via prompt routing and tool selection from a VFM pool. Later methods such as Toolformer [Schick et al., 2023], AssistGPT [Gao et al., 2023], and ToolLLM [Qin et al., 2023b] learn to invoke tools through self-supervision or in-context examples. While effective, these approaches assume fixed tool behaviors or rely on limited, static memory. In contrast, TOOLMEM enables adaptive, context-aware tool selection among choices, by building capability memories that capture tools' empirical strengths and weaknesses through experiences.

Generative AI Tools Unlike deterministic tools that produce predictable outputs, tools supported by generative neural models (e.g., *DALLE-3*, *GPT*) exhibit varied performances across task specifications [Tang et al., 2025b]. Some tools may excel at object counting, while others are better at spatial understanding [Sim et al., 2024]. While prior work has relied on human expertise to navigate these nuances, current agent systems typically neglect such variance during tool selection [Wang et al., 2024a]. In contrast, TOOLMEM records empirical performance signals for each tool, enabling agents to make accuracy-aware decisions during task solving.

Memory-Augmented Agents Effective memory systems enable agents to operate over extended tasks. Foundational work like neural Turing machines [Graves et al., 2014] introduced external memory modules for querying past states. Recent systems such as Reflexion [Shinn et al., 2023] store in-session NL-formatted feedback as memory, while AWM [Wang et al., 2024c] captures reusable workflows across tasks. MemGPT [Packer et al., 2024] and MemoryBank [Zhong et al., 2023] separate memory into short- and long-term slots with dynamic update strategies. While these methods enhance general reasoning and task procedures of agents, they do not model the behavior of external tools, especially those with functionalities unknown prior to testing. In contrast, TOOLMEM explicitly tracks tool-specific strengths and weaknesses, facilitating smarter tool selection for use.

6 Conclusion and Future Work

We introduced TOOLMEM, a closed-loop framework that equips multimodal agents with a learnable and evolving memory of tool capabilities. By integrating structured memory initialization based on a proficiency-level taxonomy, feedback-driven learning from LLM-generated critiques, and retrieval-augmented generation for memory refinement, TOOLMEM enables agents to continually improve their understanding of tool behaviors. Our experiments demonstrate that agents augmented with TOOLMEM achieve more accurate tool performance estimation and make better-informed tool choices, leading to improved outcomes in both text and image generation tasks.

This work opens up several promising directions for future exploration. One compelling avenue is extending ToolMem to support *a wider array of generative and decision-making tools* across domains such as multimodal editing and code synthesis. Another direction is developing *more advanced automated feedback* mechanisms to support autonomous memory construction. Additionally, analyzing the *theoretical underpinnings of memory* maintenance can offer new insights into long-term memory scalability. Finally, incorporating *human-in-the-loop* and *principled memory consolidation*

strategies may enhance learning from sparse or noisy experiences. We hope TOOLMEM can serve as a foundation for building adaptive agents that learn from prior interactions without repeated retraining, paving the way for more efficient, resilient, and generalizable AI systems.

References

Alejandro Carrasco, Marco Nedungadi, Victor Rodriguez-Fernandez, and Richard Linares. Visual language models as operator agents in the space domain. In *AIAA SCITECH 2025 Forum*. American Institute of Aeronautics and Astronautics, January 2025. doi: 10.2514/6.2025-1543. URL http://dx.doi.org/10.2514/6.2025-1543.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions, 2017. URL https://arxiv.org/abs/1704.00051.

Matthew Dunn, Levent Sagun, Mike Higgins, V. Ugur Guney, Volkan Cirik, and Kyunghyun Cho. Searchqa: A new q&a dataset augmented with context from a search engine, 2017. URL https://arxiv.org/abs/1704.05179.

Difei Gao, Lei Ji, Luowei Zhou, Kevin Qinghong Lin, Joya Chen, Zihan Fan, and Mike Zheng Shou. Assistgpt: A general multi-modal assistant that can plan, execute, inspect, and learn, 2023. URL https://arxiv.org/abs/2306.08640.

Zhi Gao, Bofei Zhang, Pengxiang Li, Xiaojian Ma, Tao Yuan, Yue Fan, Yuwei Wu, Yunde Jia, Song-Chun Zhu, and Qing Li. Multi-modal agent tuning: Building a vlm-driven agent for efficient tool usage, 2025. URL https://arxiv.org/abs/2412.15606.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines, 2014. URL https://arxiv.org/abs/1410.5401.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. Stabletoolbench: Towards stable large-scale benchmarking on tool learning of large language models, 2024.

Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training, 2022. URL https://arxiv.org/abs/2211.11559.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Realm: Retrieval-augmented language model pre-training, 2020. URL https://arxiv.org/abs/2002.08909.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. Dense passage retrieval for open-domain question answering, 2020. URL https://arxiv.org/abs/2004.04906.

Seungone Kim, Juyoung Suk, Ji Yong Cho, Shayne Longpre, Chaeeun Kim, Dongkeun Yoon, Guijin Son, Yejin Cho, Sheikh Shafayat, Jinheon Baek, Sue Hyun Park, Hyeonbin Hwang, Jinkyung Jo, Hyowon Cho, Haebin Shin, Seongyun Lee, Hanseok Oh, Noah Lee, Namgyu Ho, Se June Joo, Miyoung Ko, Yoonjoo Lee, Hyungjoo Chae, Jamin Shin, Joel Jang, Seonghyeon Ye, Bill Yuchen Lin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. The biggen bench: A principled benchmark for fine-grained evaluation of language models with language models, 2025. URL https://arxiv.org/abs/2406.05761.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021. URL https://arxiv.org/abs/2005.11401.

Baiqi Li, Zhiqiu Lin, Deepak Pathak, Jiayao Li, Yixin Fei, Kewen Wu, Tiffany Ling, Xide Xia, Pengchuan Zhang, Graham Neubig, and Deva Ramanan. Genai-bench: Evaluating and improving compositional text-to-visual generation, 2024. URL https://arxiv.org/abs/2406.13743.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://aclanthology.org/2023.emnlp-main.187/.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023. URL https://arxiv.org/abs/2303.17651.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. Webgpt: Browser-assisted question-answering with human feedback, 2022. URL https://arxiv.org/abs/2112.09332.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems, 2024. URL https://arxiv.org/abs/2310.08560.

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. Toolllm: Facilitating large language models to master 16000+ real-world apis, 2023a. URL https://arxiv.org/abs/2307.16789.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis, 2023b.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools, 2023. URL https://arxiv.org/abs/2302.04761.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face, 2023. URL https://arxiv.org/abs/2303.17580.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023.

Shang Hong Sim, Clarence Lee, Alvin Tan, and Cheston Tan. Evaluating the generation of spatial relations in text and image generative models, 2024. URL https://arxiv.org/abs/2411.07664.

Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning, 2023. URL https://arxiv.org/abs/2303.08128.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*, 2023.

Yunlong Tang, Jing Bi, Chao Huang, Susan Liang, Daiki Shimada, Hang Hua, Yunzhong Xiao, Yizhi Song, Pinxin Liu, Mingqian Feng, Junjia Guo, Zhuo Liu, Luchuan Song, Ali Vosoughi, Jinxi He, Liu He, Zeliang Zhang, Jiebo Luo, and Chenliang Xu. Caption anything in video: Fine-grained object-centric captioning via spatiotemporal multimodal prompting, 2025a. URL https://arxiv.org/abs/2504.05541.

Yunlong Tang, Junjia Guo, Pinxin Liu, Zhiyuan Wang, Hang Hua, Jia-Xing Zhong, Yunzhong Xiao, Chao Huang, Luchuan Song, Susan Liang, Yizhi Song, Liu He, Jing Bi, Mingqian Feng, Xinyang Li, Zeliang Zhang, and Chenliang Xu. Generative ai for cel-animation: A survey, 2025b. URL https://arxiv.org/abs/2501.06250.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023. URL https://arxiv.org/abs/2305.16291.

Zhenyu Wang, Aoxue Li, Zhenguo Li, and Xihui Liu. Genartist: Multimodal llm as an agent for unified image generation and editing, 2024a. URL https://arxiv.org/abs/2407.05600.

Zhiruo Wang, Daniel Fried, and Graham Neubig. Trove: Inducing verifiable and efficient toolboxes for solving programmatic tasks, 2024b. URL https://arxiv.org/abs/2401.12869.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory, 2024c. URL https://arxiv.org/abs/2409.07429.

Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. Visual chatgpt: Talking, drawing and editing with visual foundation models, 2023. URL https://arxiv.org/abs/2303.04671.

Junlin Xie, Zhihong Chen, Ruifei Zhang, Xiang Wan, and Guanbin Li. Large multimodal agents: A survey, 2024. URL https://arxiv.org/abs/2402.15116.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents, 2025. URL https://arxiv.org/abs/2502.12110.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory, 2023. URL https://arxiv.org/abs/2305.10250.

A Analysis: Number of Memory Entries to Retrieve

We further conduct an in-depth analysis on top-k retrieval granularity, by retrieving varying numbers of memory entries from TOOLMEM.

As shown in Figure 3, We also compare performance when using varying *top-k* values (0–24)[4]. Figure 3 shows that larger *top-k* values generally reduce MAE and RMSE but eventually plateau or degrade due to noise from excessive retrieval. Accuracy follows a similar trend, often peaking around *top-k* = 10–14. For example, TOOLMEM 's predictions for Midjourney_6 improve from an MAE of 0.98 at *top-k* = 0 to 0.91 at *top-k* = 18; SDXL_Turbo's MAE falls from 1.24 to 1.06 over the same range. These results suggest that leveraging a moderate number of high-relevance memory items helps balance predictive accuracy and computational efficiency.

Figure 3: TOOLMEM's score prediction accuracy on six text-to-image models across different top-k memory retrieval sizes, evaluated on 100 samples. The dashed line at $k = 12$ indicates most models' empirically optimal memory size.

B Prompt Templates

B.1 Text to Image Tools

Prompt for Feedback Generation

The text prompt used to generate the image is "{prompt}". The Scoring metric used here is a 5-point Likert scale:
1 (Does not match at all): The generated image or video completely fails to align with the text prompt. It either lacks the required elements or depicts a scenario that contradicts the prompt.
2 (Has significant discrepancies): There are major mismatches between the text prompt and the visual output. Key elements may be missing, or relationships and attributes are significantly misrepresented.
3 (Has several minor discrepancies): The output aligns with the prompt in a general sense but has noticeable errors or omissions in key details. While the main elements are present, finer aspects are not accurately rendered.
4 (Has a few minor discrepancies): The output is mostly aligned with the prompt, with only a few small and less critical details being inaccurate or missing. The visual matches the description well but not perfectly.
5 (Matches exactly): The generated image or video perfectly matches the text prompt. All elements, relationships, attributes, and details align seamlessly with the description, with no noticeable discrepancies.
This image scored {score} based on the human ratings. Now tell me why you think the image is or isn't following the text prompt.

---

[4]When *top-k* = 0, predictions are made using only the tool name and its overview, without retrieving any memory items.

Predict Image Description - FEW-SHOT

You are familiar with how the model {self.model_name} generates images.
Your current memory of {self.model_name} is: "{current_memory}"

Here are some examples of the model's performance on specific prompts:
{few_shot_memory}

The scoring metric used here is a 5-point Likert scale:
1 (Does not match at all): The generated image or video completely fails to align with the text prompt. It either lacks the required elements or depicts a scenario that contradicts the prompt.
2 (Has significant discrepancies): There are major mismatches between the text prompt and the visual output. Key elements may be missing, or relationships and attributes are significantly misrepresented.
3 (Has several minor discrepancies): The output aligns with the prompt in a general sense but has noticeable errors or omissions in key details. While the main elements are present, finer aspects are not accurately rendered.
4 (Has a few minor discrepancies): The output is mostly aligned with the prompt, with only a few small and less critical details being inaccurate or missing. The visual matches the description well but not perfectly.
5 (Matches exactly): The generated image or video perfectly matches the text prompt. All elements, relationships, attributes, and details align seamlessly with the description, with no noticeable discrepancies.

Based on these examples and your memory of {self.model_name}, predict how an image generated by {self.model_name} for the following prompt would look:

Prompt: "{prompt}"

Describe the expected image in no more than 50 English words. Be concise and specific. Only return the description text.

Predict Image Description - TOOLMEM

You are familiar with how the model {model_name} generates images.
Your current memory of {model_name} is: "{current_memory}"

Based on this memory, predict how an image generated by {model_name} for the following prompt would look:
Prompt: "{prompt}"

Describe the expected image in no more than 50 English words. Be concise and specific. Only return the description text.

Predict Score - GENERIC

Your current memory (knowledge) of the tool {model_name} is: "A text to image model"
Based on your memory of model {model_name}, predict the score for the following prompt: "{prompt}".

The scoring metric used here is a 5-point Likert scale:
1 (Does not match at all): The generated image or video completely fails to align with the text prompt. It either lacks the required elements or depicts a scenario that contradicts the prompt.
2 (Has significant discrepancies): There are major mismatches between the text prompt and the visual output. Key elements may be missing, or relationships and attributes are

significantly misrepresented.

3 (Has several minor discrepancies): The output aligns with the prompt in a general sense but has noticeable errors or omissions in key details. While the main elements are present, finer aspects are not accurately rendered.

4 (Has a few minor discrepancies): The output is mostly aligned with the prompt, with only a few small and less critical details being inaccurate or missing. The visual matches the description well but not perfectly.

5 (Matches exactly): The generated image or video perfectly matches the text prompt. All elements, relationships, attributes, and details align seamlessly with the description, with no noticeable discrepancies.

Now return a score between 1 and 5 for prompt {prompt}, based on your memory of the tool {model_name}. Return a single number only, nothing else.

Predict Score - FEW-SHOT

Your current memory (knowledge) of the tool {self.model_name} is: "A text to image model".

Here are some examples of the model's performance on specific prompts: {samples_prompt}.

Based on these examples and your memory of {self.model_name}, predict the score this model will get for answering the following task prompt: "{prompt}".

The scoring metric used here is a 5-point Likert scale:

1 (Does not match at all): The model's response completely fails to align with the task prompt. It either lacks the required elements or presents information that contradicts the prompt.

2 (Has significant discrepancies): There are major mismatches between the task prompt and the model's response. Key elements may be missing, or relationships and attributes are significantly misrepresented.

3 (Has several minor discrepancies): The response aligns with the prompt in a general sense but has noticeable errors or omissions in key details. While the main elements are present, finer aspects are not accurately rendered.

4 (Has a few minor discrepancies): The response is mostly aligned with the prompt, with only a few small and less critical details being inaccurate or missing. The response matches the description well but not perfectly.

5 (Matches exactly): The model's response perfectly matches the task prompt. All elements, relationships, attributes, and details align seamlessly with the description, with no noticeable discrepancies.

Now, return a score between 1 and 5 based on your memory of {self.model_name}'s capabilities and limitations. Return a single number only, nothing else.

Predict Score - TOOLMEM

Your current memory (knowledge) of the tool {model_name} is: "{current_memory}"

Based on your memory of model {model_name}, predict the score for the following prompt: "{prompt}".

The scoring metric used here is a 5-point Likert scale:

1 (Does not match at all): The generated image or video completely fails to align with the text prompt. It either lacks the required elements or depicts a scenario that contradicts the prompt.

2 (Has significant discrepancies): There are major mismatches between the text prompt and the visual output. Key elements may be missing, or relationships and attributes are significantly misrepresented.

3 (Has several minor discrepancies): The output aligns with the prompt in a general sense but has noticeable errors or omissions in key details. While the main elements are present,

finer aspects are not accurately rendered.

4 (Has a few minor discrepancies): The output is mostly aligned with the prompt, with only a few small and less critical details being inaccurate or missing. The visual matches the description well but not perfectly.

5 (Matches exactly): The generated image or video perfectly matches the text prompt. All elements, relationships, attributes, and details align seamlessly with the description, with no noticeable discrepancies.

Now return a score between 1 and 5 for prompt {prompt}, based on your memory of the tool {model_name}. Return a single number only, nothing else.

B.2 Text Generation Tools

Prompt for Memory Refinement

You are an assistant responsible for maintaining an incremental memory of the tool's capabilities, limitations, and use cases.

You will receive a current memory and new feedback. Your job is to update the memory by integrating the new feedback without losing previously stored information.

Rules:
1. Start with the current memory.
2. Consider the new prompt and feedback, but do not simply repeat the feedback, think deep and get general insights.
3. Incrementally refine and update the memory to reflect any new insights.
4. Do not remove previously known capabilities unless the new feedback explicitly contradicts them.
5. If no new insights are provided, leave the memory unchanged.

Instructions:
Follow this procedure strictly:
- Review the current memory for model {tool_name}:
"{current_memory}"
- Input task prompt to the tool was:
"{task_prompt}"
- Tool's answer was:
"{response}"
- Score rubric was:
"{score_rubric}"
- The score for tool's response was:
"{score}"
- Feedback for this answer received:
"{feedback}"

Integrate the key insights of the tool's limitations and capabilities from the feedback into the existing memory. Use complete sentences starting with phrases like `good at`, `bad at`, `proficient at`, or `poor at` to describe your findings.
- If the feedback suggests new limitations, capabilities, or clarifications, add them.
- If the feedback suggests that something previously stated is incorrect, correct it.
- If the feedback suggests the tool struggles with a certain aspect (like counting multiple objects), note it.
- If no contradiction or new insight is provided, keep the memory as is.
- Think deeply and determine what the feedback actually means for the model's capability, rather than staying on the surface of a specific example.

In your final answer, output **ONLY** the updated overall memory of the tool. Do not include extra explanations.

Predict Score - GENERIC

Your current memory (knowledge) of the tool {model_name} is: "A large language model"
Based on your memory of model {model_name}, predict the score this model will get for answering the following task prompt: "{prompt}".

The scoring metric used here is a 5-point Likert scale:

{rubric}

Now return a score between 1 and 5, based on your memory of {model_name}'s capabilities and limitations.
Return a single number only, nothing else.

Predict Score - FEW-SHOT

Your current memory (knowledge) of the tool {model_name} is: A large language model.
Here are some examples of the model's performance on some tasks:

Task Prompt: {task_prompt}
Rubric: {rubric}
Model's Score: {score}

Task Prompt: {task_prompt}
Rubric: {rubric}
Model's Score: {score}

Task Prompt: {task_prompt}
Rubric: {rubric}
Model's Score: {score}
...
Based on those samples and your memory of {model_name}, predict the score this model will get for answering the following task prompt: "{task_prompt}".

The scoring metric used here is a 5-point Likert scale:
{rubric}

Now return a score between 1 and 5, based on your memory of {model_name}'s capabilities and limitations.
Return a single number only, nothing else.

Predict Score - TOOLMEM

Your current memory (knowledge) of the tool {model_name} is: {current_memory}
Based on your memory of model {model_name}, predict the score this model will get for answering the following task prompt: "{prompt}".

The scoring metric used here is a 5-point Likert scale:

{rubric}

> Now return a score between 1 and 5, based on your memory of {model_name}'s capabilities and limitations.
> Return a single number only, nothing else.

Predict performance of SDXL-Turbo for prompt: An oil painting of a majestic lion hanging above the fireplace

Score: 4 (Has a few minor discrepancies)

Reason: SDXL_Turbo is *good at* generating majestic subjects (like lions) and general scenes with appropriate lighting and mood. However, it may struggle slightly with **accurately arranging elements** (like positioning the lion painting precisely *above* a fireplace) and **maintaining perfect detail** in complex scenes (e.g., oil painting texture + interior setting). Thus, the result is likely visually impressive and mostly accurate but may miss some spatial or stylistic precision.

SDXL-Turbo generated result:

Human evaluation score: 4

Predict performance of SDXL-Turbo for prompt: A pair of socks, one is striped, but the other is not

Score: 3 (Has several minor discrepancies)

Reason: SDXL_Turbo is *good* at depicting simple object descriptions and textures like stripes and socks. However, it's *bad at* distinguishing **singular vs. plural** accurately and often struggles with **fine-grained visual contrasts** between similar objects. This means it may depict both socks as striped or both as plain, or fail to show a clear visual difference, leading to minor but noticeable mismatches with the prompt.

SDXL-Turbo generated result:

Human evaluation score: 2

Figure 4: Qualitative examples for TOOLMEM. The left image illustrates the prediction for "An oil painting of a majestic lion hanging above the fireplace," while the right example shows the prediction for "A pair of socks, one is striped, but the other is not."

C   Case Study

To better understand TOOLMEM's capability in performance prediction, we present two representative examples in Figure 4, where TOOLMEM is prompted to explicitly forecast the quality and fidelity of the output images generated by the SDXL-Turbo tool.

Example 1: Majestic Lion Above the Fireplace.   For the prompt "*An oil painting of a majestic lion hanging above the fireplace*," TOOLMEM assigns a predicted score of 4 (minor discrepancies). In practice, SDXL-Turbo successfully produces a visually appealing scene with a detailed lion portrait. However, our system anticipates that fine spatial alignment may not be perfectly captured (e.g., the fireplace might be cropped or misaligned). Indeed, the actual generated image exhibits a slightly off positioning of the lion, confirming TOOLMEM's prediction. Notably, the final score aligns closely with human evaluations, underlining TOOLMEM's proficiency in assessing model outputs.

Example 2: Pair of Socks with Different Patterns.   For the prompt "*A pair of socks, one is striped, but the other is not*," TOOLMEM predicts a score of 3 (several minor discrepancies). While SDXL-Turbo generally handles simple object prompts and textures adequately, TOOLMEM notes the model's recurring difficulty in distinguishing subtle visual contrasts between highly similar objects. In the actual output, both socks often appear identical (both striped), matching TOOLMEM 's forecast. However, human evaluators deemed these inaccuracies more severe, giving a lower score of 2. This discrepancy suggests that although TOOLMEM detects the correct type of error, it occasionally assigns higher tolerance to such mismatches than human annotators.

Discussion.   These examples highlight both the strengths and limitations of TOOLMEM. On one hand, TOOLMEM demonstrates strong alignment with human judgments when identifying whether the main elements of a prompt are preserved or substantially distorted. On the other hand, its scoring

mechanism sometimes diverges from that of human evaluators, underscoring the subjective nature of visual quality assessments.

D Compute Resource and Reproducibility Details

The core computation in our experiments is conducted via OpenAI's API, GPT-4o-mini serves as the backbone language model for TOOLMEM, and text-embedding-ada-002 for RAG tasks. As our framework relies on API calls, it does not require any local GPU hardware for inferences.

All other models involved—such as Claude-3, DALL·E 3, Midjourney, SDXL, and others—are part of the precomputed outputs provided by the GENAI-BENCH and BIGGEN BENCH datasets. We do not rerun these tools; we only analyze and build memory around their existing outputs.

The only locally executed model in our pipeline is the clip-flant5-xl model used to calculate VQA Score, which evaluates image-description alignment. This model is run on an NVIDIA A100 GPU, and is highly efficient—each VQA score can be computed in about 0.2 seconds. As such, the compute requirement is minimal, and large batches can be processed in real time.

Overall, our pipeline is lightweight and fully reproducible. All heavy model inference is handled externally via OpenAI APIs, and the only local requirement is for lightweight VQA scoring, making TOOLMEM practical to deploy and evaluate with modest resources.

E Broader Impact and Limitation

This paper investigates a new capability for agents powered by Large Language Models (LLMs): building and utilizing structured memory representations of tool capabilities to improve tool selection and task-solving performance. By enabling agents to systematically record and retrieve performance feedback from past tool interactions, our proposed TOOLMEM framework pushes toward more autonomous and adaptive decision-making systems.

The broader impact of this line of work lies in its potential to make complex multi-tool reasoning more robust, transparent, and efficient. Applications that involve numerous generative or decision-support tools—such as multimodal assistants, educational platforms, or creative design systems—could benefit from better memory-guided tool selection, reducing trial-and-error usage and model waste. TOOLMEM may also lower the barrier for non-experts to leverage advanced model capabilities, as the system autonomously adapts based on prior observations.

However, several limitations and risks must be acknowledged. First, the memory quality heavily depends on the accuracy and reliability of the feedback used to construct it. In this work, we rely on GPT-4-generated scores and rubrics, which, although effective, may introduce biases or inconsistencies not present in human-grounded evaluations. Moreover, the framework assumes tool capabilities remain relatively stable over time. In real-world deployments, frequent model updates or usage context shifts may render historical memory entries stale or misleading.

Another concern is over-reliance on memorized preferences, which may lead to suboptimal or unfair tool choices in edge cases not well-represented in the agent's prior experiences. Without proper monitoring, such systems might reinforce early biases or miss the opportunity to explore potentially better alternatives.

Lastly, our experiments are conducted in controlled benchmark environments with known tool outputs and synthetic feedback. Generalizing TOOLMEM to open-ended tasks, dynamic tool ecosystems, or safety-critical applications requires additional safeguards and validation, which we leave for future work.

While promising, TOOLMEM represents an early step toward memory-augmented tool reasoning, and further exploration is needed to ensure both its robustness and its alignment with human-centered values in broader deployment contexts.