Experience Makes Skillful: Enabling Generalizable Medical Agent Reasoning via Self-Evolving Skill Memory

Haoran Sun[1,2]*, **Wenjie Li**[2]*, **Yujie Zhang**[2]*, **Zekai Lin**[1], **Fanrui Zhang**[3], **Kaitao Chen**[2], **Xingqi He**[1], **Yichen Li**[4], **Mianxin Liu**[2], **Lei Liu**[1]†, **Yankai Jiang**[2]†

[1]Fudan University
[2]Shanghai Artificial Intelligence Laboratory
[3]University of Science and Technology of China
[4]Huazhong University of Science and Technology

leiliu@fudan.edu.cn, jiangyankai@pjlab.org.cn

Abstract

Medical agent systems are increasingly expected to support interactive clinical decision making rather than only static question answering. In such settings, effective agents must reuse prior experience across evolving cases, yet existing memory mechanisms often retain raw historical traces that are redundant, noisy, and difficult to govern. More importantly, they rarely distinguish which memories are truly useful for future reasoning. This limits their ability to accumulate compact and reliable experience for long-horizon clinical reasoning. To close this gap, we propose **SkeMex**, a post-deployment self-evolution framework that improves medical agents through a skill-based memory without updating model weights. SkeMex distills informative interaction trajectories into structured skills that encode reusable procedural knowledge, and organizes them into a multi-branch repository spanning general, task-specific, and action-level experience. To determine which memories should be reused and retained, SkeMex estimates context-dependent utility from environment feedback and uses it to guide value-aware retrieval and repository governance. A closed-loop "Read–Write–Assess–Govern" lifecycle further supports continual evolution by writing new skills, updating utilities, promoting useful memories, and removing harmful entries. Experiments across diverse clinical tasks show that SkeMex consistently outperforms representative memory-based agents in both offline and online settings. It also generalizes across model backbones and supports transferable skill memory. Code will be available at https://github.com/manglu097/SkeMex

1 Introduction

Recent advances in medical large language models (LLMs) [8, 19, 49, 80] and agent-based systems [22, 28, 51, 59] have achieved strong performance on medical benchmarks, especially knowledge-intensive and exam-style datasets such as MedQA [20], MedMCQA [41], and MedBullets [7]. These benchmarks are widely used to evaluate medical reasoning and factual recall, and recent models now reach highly competitive results, sometimes approaching or exceeding human performance on selected subsets [54, 55]. However, most evaluations remain static and single-turn, with fixed

*These authors contributed equally.
†Corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

inputs and predefined criteria. Real clinical decision making is more dynamic. It unfolds over multiple steps and requires continuous interaction, evidence gathering, multimodal interpretation, hypothesis revision, and action adjustment under uncertainty [9, 62]. From a cognitive perspective, clinicians do not rely only on isolated factual knowledge [63]. Their expertise develops through accumulated experience across cases. Prior encounters are recalled, compared, and gradually abstracted into reusable patterns that guide future decisions [5, 23]. This process combines semantic knowledge, memory of past cases, and procedural knowledge about how to act in specific situations. It enables clinicians to adapt to new scenarios that are not fully covered by training data or textbooks.

To address this gap, recent work explores memory-augmented medical agents that store and reuse past interactions [25, 29, 47, 71]. Early methods mainly focus on intra-task memory, which records observations, reasoning steps, and tool interactions within a single clinical case to maintain long-horizon coherence [18, 79]. As LLMs gain longer contexts and stronger in-context reasoning [43, 61], memory is increasingly used not only to extend context, but also to organize and reuse experience across tasks. This shift motivates inter-task memory, where agents use prior cases, decisions, or failures to solve new problems. Case-based reasoning recalls and adapts similar past cases [15, 16, 23, 93], but storing raw cases often produces redundant, noisy, and instance-specific repositories. This limits generalization across diverse clinical scenarios [51, 59, 71]. Recent work therefore distills trajectories into skills, which provide compact units of reusable procedural knowledge [37, 77, 87]. These skills capture recurring patterns of reasoning and action, making experience more transferable. Despite this progress, current methods still face two limitations (Figure 1). First, some representative methods couple memory improvement with policy training or self-distillation, requiring parameter updates to incorporate experience-derived knowledge [65, 77]. This can be costly and may cause catastrophic forgetting or weak transfer across domains [75, 88]. Second, case-based and skill-based memories often lack mechanisms for evaluating long-term usefulness. Redundant or low-quality entries can therefore accumulate over time. These limitations make it difficult for current memory systems to support improvement from experience.

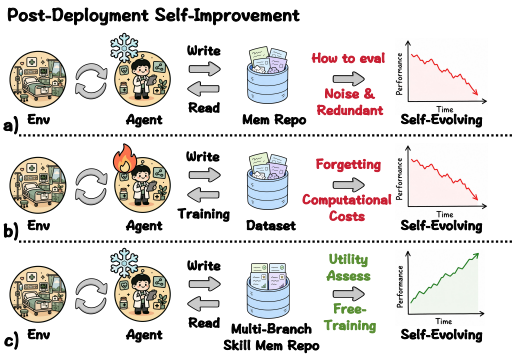

Figure 1: Comparison of (a) conventional memory, (b) training-based methods, and (c) our method.

In this work, we introduce **SkeMex**, a post-deployment self-evolution framework that enables medical agents to improve through external skill memory without modifying the backbone model. SkeMex treats interaction trajectories as sources of reusable experience and converts informative patterns into structured skills that encode actionable reasoning and decision-making procedures. These skills are organized in a multi-branch repository covering general reasoning, task-specific knowledge, and action-level operations. During inference, SkeMex performs value-aware retrieval, selecting skills based on both semantic relevance and empirically estimated utility in related clinical contexts. To support continual improvement, it further distills successful or informative interactions into new or updated skills, estimates skill utility from observed outcomes, and maintains the repository through a closed-loop "Read–Write–Assess–Govern" lifecycle. This lifecycle promotes high-utility skills, merges redundant ones, and removes low-quality or potentially harmful entries. Accordingly, SkeMex transforms raw interaction histories into an evolving memory that supports reliable experience reuse. It enables experience-driven learning without parameter updates, making agents more scalable and adaptable across clinical environments. Our contributions are summarized as follows:

- We propose **SkeMex**, a post-deployment self-evolution framework for medical agents that improves clinical reasoning through a skill-based memory without updating model weights.
- We formulate skill-memory evolution as a **non-parametric** reinforcement process, where clinical feedback provides reward signals to estimate context-dependent utility, a unified measure of memory effectiveness that guides both skill retrieval and repository governance.
- We introduce the **Read–Write–Assess–Govern** lifecycle, a closed-loop mechanism that converts trajectories into reusable skills and maintains a well-governed memory repository.

- We demonstrate that SkeMex is a **plug-and-play** framework that consistently improves performance across diverse clinical tasks, generalizes effectively across different model backbones, and supports transferable skill memory across heterogeneous task settings.

2 Related Works

LLM-based Medical Agents. Recent advances in medical foundation models, including Lingshu [80], Hulu-Med [19], and MedGemma [49], have expanded medical reasoning across text, imaging, and multimodal data. Building on these models, medical agents have incorporated retrieval, tool use, and multi-agent collaboration to handle heterogeneous clinical evidence. For example, i-MedRAG [78] and MedRAG [90] improve medical retrieval through iterative search and knowledge-guided reasoning. EHRAgent [51] uses code-based reasoning for structured EHR data and maintains long-term case memory, while MMedAgent [28] selects and composes specialized multimodal tools. At the system level, MedAgents [59], MDAgents [22], MAM [94], and MedAgent-Pro [69] decompose diagnosis into coordinated multi-agent workflows. Despite strong task-specific performance, most of these systems still process cases independently and lack mechanisms for accumulating reusable experience. Recent work has begun to incorporate memory and self-improvement. Agent Hospital [29] studies evolvable agents through simulated clinical practice, AMC [26] introduces structured memory for psychiatric tasks, and MACRO [10] extracts reusable tools from execution trajectories. STELLA [21] and HealthFlow [95] further explore evolving templates and policy refinement for biomedical research. However, these methods often tie memory to specific workflows or organize past cases with heuristic rules, which can limit cross-task generalization. Instead, SkeMex decouples memory evolution from fixed workflows. It distills reusable skills from trajectories, evaluates them with external feedback, and selectively retains high-value experience for continual cross-task learning.

Self-Evolving Memory. Memory mechanisms in LLM-based agents were initially introduced to address limited context windows, allowing systems to retain prior interactions or retrieved knowledge through static buffers or retrieval-augmented generation (RAG) [27, 40, 42, 66, 92]. As LLMs support longer contexts [43, 61], memory is no longer only a remedy for context limits. It has become a structured store of experience for summarizing and reusing knowledge from environmental interactions [12]. Early self-evolving memory systems stored raw trajectories or reflections to guide future actions [52, 64, 73, 89]. Recent work has moved toward structured and modular designs [31, 86]. Agent Workflow Memory [70] and Dynamic Cheatsheet [57] convert experiences into reusable procedural routines. In medicine, GSEM [17] represents clinical experience with a dual-layer memory graph, while HealthFlow [95] organizes successful and failed procedures into structured knowledge for tool use and decision making. More recently, skills have emerged as a compact and generalizable memory form [1]. Trace2Skill [37], SkillClaw [34], and EvoSkills [87] further show that trajectories can be distilled into hierarchical skill libraries. However, managing skill memories remains challenging. Recent methods often frame skill evolution as reinforcement learning or optimization. SkillRL [77] and Skill-SD [65] integrate this process into policy training, but require parameter updates. In medical domains, where reliability are critical, this can be costly and may risk catastrophic forgetting or weaken previously learned clinical behaviors. Contrastly, SkeMex decouples skill-memory evolution from model training. It estimates skill utility from environment feedback to guide retrieval and governance, and updates the repository without modifying parameters.

3 Method

3.1 Problem Formulation

We formulate SkeMex as a Memory-based Markov Decision Process (M-MDP) [93], where an LLM-based agent interacts with an environment while consulting and updating a memory bank.
Memory-Based Markov Decision Process. Following prior work [93], we formalize this process as

$$\mathcal{T}_{\mathrm{M-MDP}} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{E}, \gamma, \mathcal{M} \rangle, \tag{1}$$

where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, $\mathcal{P}$ is the transition kernel, $\mathcal{E}$ is the reward function, $\gamma \in [0, 1)$ is the discount factor, and $\mathcal{M}$ is the space of finite memory banks. At step t, the agent observes a state $s_t \in \mathcal{S}$, which includes the current problem, accumulated observations, and retained execution context. It then consults the memory bank $M_t = \{m_i\}_{i=1}^{N_t} \in \mathcal{M}$ and produces an

action $a_t \in \mathcal{A}$, covering reasoning decisions, tool calls, and final responses. Each memory unit is represented as $m_i = (k_i, c_i, u_i)$, where k_i is a retrieval key, c_i is reusable memory content, and $u_i \in \mathbb{R}$ is a utility statistic reflecting its historical contribution. In SkeMex, memory units are instantiated as skills. After executing a_t, the agent receives reward $r_t = \mathcal{E}(s_t, a_t)$ and transitions to $s_{t+1} \sim \mathcal{P}(\cdot \mid s_t, a_t)$. Meanwhile, the memory bank evolves through an update operator $\mathcal{U}$, written as $M_{t+1} = \mathcal{U}(M_t, s_t, m_t, a_t, r_t)$. Under this formulation, post-deployment improvement comes from better memory retrieval and memory evolution [53], rather than parameter updates [39, 46, 56].

Memory-Augmented LLM-based Agent. Based on the M-MDP, SkeMex is defined as a memory-augmented agent whose behavior depends on both the current state and the maintained memory bank. The memory bank M_t is updated across episodes by $\mathcal{U}$, accumulating distilled experience from prior interactions. Given (s_t, M_t), the agent retrieves memory units $m_t \subseteq M_t$ using $\mu(m_t \mid s_t, M_t)$, and then produces an action through the LLM $p_\theta(a_t \mid s_t, m_t)$. Formally, the overall policy is given by

$$\pi(a_t \mid s_t, M_t) = \sum_{m \in M_t} \mu\big(m \mid s_t, M_t\big)\, p_\theta\big(a_t \mid s_t, m\big). \tag{2}$$

Here, μ selects memory for the current state, while p_θ maps the state and retrieved memory into a reasoning decision, tool call, or final response. A trajectory is denoted by $\tau = \{M_0, s_0, m_0, a_0, r_0, \ldots, M_{T-1}, s_{T-1}, m_{T-1}, a_{T-1}, r_{T-1}\}$, making explicit that memory is retrieved before each decision and updated after feedback. We factorize the probability as

$$\begin{aligned} p(\tau) = p_0(M_0, s_0) \prod_{t=0}^{T-1} & \mu(m_t \mid s_t, M_t)\, p_\theta(a_t \mid s_t, m_t)\, \mathcal{E}(r_t \mid s_t, a_t) \\ & \mathcal{U}(M_{t+1} \mid M_t, s_t, m_t, a_t, r_t)\, \mathcal{P}(s_{t+1} \mid s_t, a_t), \end{aligned} \tag{3}$$

where $p_0(M_0, s_0)$ denotes the initial distribution over the memory bank and task state. $\mathcal{E}(r_t \mid s_t, a_t)$ denotes the environment reward, and $\mathcal{U}(M_{t+1} \mid M_t, s_t, m_t, a_t, r_t)$ captures how feedback updates the memory bank. This factorization explicitly reflects the modular structure of SkeMex.

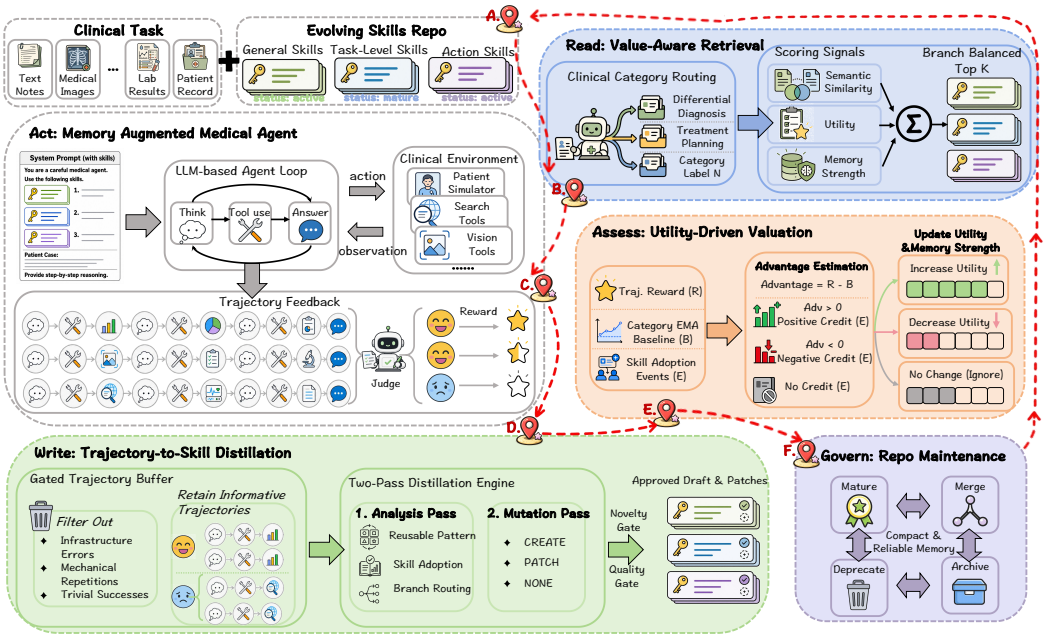

Figure 2: Overview of SkeMex. Components A–F constitute a closed-loop self-evolving cycle (A→B→C→D→E→F→A), enabling continual improvement through iterative memory operations.

3.2 Overview of SkeMex

SkeMex is designed to help medical agents accumulate and reuse useful experience from interaction, rather than rely on static contextual augmentation. Its core component is a continuously evolving

skills repo, which stores procedural knowledge distilled from prior trajectories. As shown in Figure 2, SkeMex follows a closed-loop **Read–Write–Assess–Govern** lifecycle. The repo first supports reasoning through value-aware retrieval (Section 3.3). Interaction trajectories and feedback then drive skill distillation (Section 3.4) and utility estimation (Section 3.5). Finally, repository governance maintains compactness and reliability by managing skills (Section 3.6). Through this cycle, SkeMex enables medical agents to improve from experience while keeping the skill repo compact and reusable.

In SkeMex, every memory unit is instantiated as a structured skill item. Therefore, M_t can be naturally understood as a skills repo. Unlike raw trajectories, which are often lengthy, noisy, and instance-specific [72, 86], skills provide a compact and reusable representation of experience. To organize skills at different levels of abstraction, the repo is divided into three branches:

$$M_t = M_t^{\text{gen}} \cup M_t^{\text{task}} \cup M_t^{\text{act}}. \tag{4}$$

The **general branch** (M_t^{gen}) stores transferable reasoning strategies and broad clinical principles. The **task-level branch** (M_t^{task}) captures patterns tied to specific task families or clinical categories. The **action-level branch** (M_t^{act}) records operational knowledge for tool use, such as parameter formatting. This structure separates general reasoning, task-specific knowledge, and concrete actions during retrieval and valuation. It prevents skills at different abstraction levels from competing in the same pool and allows each branch to be managed within its own scope. See Appendix K for details.

3.3 Value-aware Skill Retrieval

Although the M-MDP allows retrieval at every step, dense retrieval in long-horizon medical tasks can add overhead and fragment the context [27, 33]. Frequent context changes may also weaken consistency across steps. We therefore follow [88, 93] and retrieve skills once at the episode onset ($t = 0$). This provides a stable skill context for the full trajectory and reduces noise in utility estimation by using episodic return. Specifically, retrieval starts with clinical category routing. Given the initial query s_0, the agent extracts a category label κ_0, such as *Differential Diagnosis* or *Treatment Planning*. This label constrains the task-level branch M_0^{task} and removes skills outside the relevant clinical context. We then apply multi-channel screening to obtain a candidate subset $\widetilde{M}_0 \subseteq M_0$. The screening uses similarity and historical reliability to keep re-ranking efficient and reduce irrelevant candidates. Each candidate $m \in \widetilde{M}_0$ is then scored with semantic, utility, and temporal signals:

$$\text{Score}_{ret}(m \mid s_0, \kappa_0) = \lambda_{\text{sim}}\text{Sim}(s_0, m) + \lambda_u U(m \mid \kappa_0) + \lambda_h h_0(m), \tag{5}$$

where $\text{Sim}(s_0, m)$ measures semantic match, $U(m \mid \kappa_0)$ denotes historical utility under clinical category κ_0, and $h_0(m)$ is memory strength with temporal decay, following the Ebbinghaus forgetting curve [36]. Category-conditioned utility is only used for general-branch skills, since broad clinical principles can vary in effectiveness across domains. Instead, task-level and action-level skills are already tied to their contexts. The decay term favors recently reinforced skills and limits the effect of outdated or rarely validated ones. Finally, branch-aware top-K selection balances the three branches.

3.4 Trajectory-to-Skill Distillation

To make memory writing depend on useful experience, we introduce a gated trajectory buffer $\mathcal{B}^{(w)}$ for each learning window w. The gate removes infrastructure errors, mechanical repetitions, and trivial successes. It keeps trajectories that contain meaningful multi-step reasoning or informative failures after skill injection. The buffer also uses a soft isolation policy to preserve a balanced mix of successful and failed trajectories. Each trajectory competes for retention with $v(\tau) = (\alpha \log(1 + |\tau|) + \beta \cdot \mathbb{I}[\text{injected}])/(1 + n_\kappa)$, where $|\tau|$ is the number of reasoning steps, $\mathbb{I}[\text{injected}]$ indicates whether skills were retrieved, and n_κ penalizes over-represented clinical categories. This design focuses memory updates on informative clinical patterns, such as causes of misdiagnosis or improved treatment planning, rather than low-value interactions. Skill writing is performed through a two-pass process. Given a buffered trajectory $\tau \in \mathcal{B}^{(w)}$, the analysis pass extracts a reusable pattern z_τ, identifies adoption signals of previously injected skills, and determines a writing intent $o_\tau \in \{\text{CREATE}, \text{PATCH}, \text{NONE}\}$. It also assigns a target branch for the candidate skill. The mutation pass then turns z_τ into a new skill draft or applies a local update to an existing skill according

to the predicted intent. We formalize this process with a window-level writing operator $\mathcal{W}$:

$$\mathcal{W}(M^{(w)}, \tau) = \begin{cases} M^{(w)} \cup \{\hat{m}_{|\tau|}\}, & \text{if } o_\tau = \text{CREATE}, \\ \text{Patch}(M^{(w)}, \hat{m}_{|\tau|}), & \text{if } o_\tau = \text{PATCH}, \\ M^{(w)}, & \text{if } o_\tau = \text{NONE}, \end{cases} \tag{6}$$

where $\hat{m}_{|\tau|}$ is the candidate skill distilled from trajectory τ, and $M^{(w)}$ is the skills repo snapshot at the end of window w. All samples in the same window share this snapshot, keeping M_t consistent during writing. Before a CREATE draft is committed to the repo, a review pass applies novelty and quality gates. The novelty gate rejects drafts that overlap with existing skills in the same branch and clinical category, and redirects them to PATCH. Additionally, the quality gate requires each skill to include a clear situational trigger and concrete clinical steps, rather than vague principles.

3.5 Utility-driven Skill Valuation

Adding a skill to the repo does not guarantee its long-term usefulness. Its value must be tested against clinical outcomes over time. Updating utility from per-sample outcomes can be noisy [50], since medical tasks differ in difficulty. SkeMex therefore uses window-level valuation, aggregating feedback over multiple trajectories. To ensure fair credit assignment, skills are evaluated by their relative advantage rather than absolute reward. For each clinical category κ, we maintain an exponential moving average of rewards $\bar{r}^{(w)}(\kappa)$. The advantage of a trajectory τ is defined as $A_\tau = r_\tau - \bar{r}^{(w)}(\kappa_\tau)$. Within window w, we assign credit to each adoption event using the following contribution function:

$$c(\tau, m_i) = \begin{cases} \lambda_+ \cdot A_\tau, & \text{if } m_i \text{ is positively adopted}, \\ -\big(\lambda_- + \lambda_{\text{harm}} \cdot \max(0, -A_\tau)\big) - \rho_i, & \text{if } m_i \text{ is negatively adopted}, + \\ 0, & \text{if } m_i \text{ is ignored}, \end{cases} \tag{7}$$

where λ_+ scales positive credit, λ_- is a base penalty for harmful adoption, and λ_{harm} increases the penalty when $A_\tau < 0$. The term $\rho_i = \epsilon \cdot u_i$ is a risk-sensitive regularizer proportional to the current utility of m_i, discouraging high-utility skills from accumulating unsafe behavior. Furthermore, the utility of m_i is updated by aggregating contributions from all adoption events $\mathcal{E}_i^{(w)}$ in window w:

$$u_i^{(w+1)} = \text{clip}\left(u_i^{(w)} + \eta_i^{(w)} \cdot \frac{1}{|\mathcal{E}_i^{(w)}|} \sum_{\tau \in \mathcal{E}_i^{(w)}} c(\tau, m_i) \right), \tag{8}$$

where $\eta_i^{(w)}$ follows a cosine warmup schedule based on cumulative adoption count. This enables faster adjustment for new skills and steadier updates for mature ones. The clip operation constrains utility to $[0, 1]$, preventing extreme values from dominating retrieval. As described in Section 3.3, retrieval uses category-conditioned utility $U(m \mid \kappa_0)$ for general-branch skills, with $u_i(\kappa)$ updated per category and global utility u_i averaged across categories. Together, the category-aware baseline and branch-dependent valuation provide stable estimates for retrieval and memory maintenance.

3.6 Closed-loop Self-evolution Memory

The framework described above, including retrieval, distillation, and valuation, forms a unified self-reinforcing cycle. We summarize this overall process with the following operator composition:

$$M^{(w+1)} = \mathcal{G}\Big(\mathcal{V}\big(\mathcal{W}(M^{(w)}, \mathcal{B}^{(w)})\big)\Big), \tag{9}$$

where $\mathcal{B}^{(w)}$ denotes gated trajectories in window w, $\mathcal{W}$ is the trajectory-to-skill writing operator, $\mathcal{V}$ is the utility valuation operator, and $\mathcal{G}$ is the repo-level governance operator. At the start of each task, the agent retrieves high-utility skills from the current repo $M^{(w)}$. After task completion, the evaluated trajectory is added to $\mathcal{B}^{(w)}$. At the end of each window, $\mathcal{W}$ distills buffered trajectories into new or updated skills, and $\mathcal{V}$ updates utilities for skills with adoption signals. Governance $\mathcal{G}$ is applied every N windows to keep the repo compact and usable. It merges redundant skills, deprecates low-utility ones, promotes consistently effective skills to mature status, and removes the lowest-utility skills when a branch exceeds its capacity $C^{\text{gen}}, C^{\text{task}}, C^{\text{act}}$. The buffer is then cleared, and the next window starts with the updated repo. SkeMex does not update the backbone model. Instead, interaction feedback updates the repository, which then guides future retrieval and reasoning.

4 Experiments and Results

4.1 Experiment Settings

Datasets We evaluate SkeMex on nine medical benchmarks covering clinical interaction and knowledge-intensive reasoning. The first group includes AgentClinic [48], LiveClin [67], MedJourney [76], LiveMedBench [81], HealthBench [3], and MediQ [30]. These benchmarks involve diagnosis, treatment planning, patient interaction, and rubric-based clinical evaluation. The second group includes MedXpertQA [96], MMMU [83], and MMMU-Pro [84], with the latter two restricted to the Health & Medicine track. Several datasets include multimodal cases with medical images or clinical tables. To make skill accumulation informative, we prioritize diverse and relatively challenging data, since trivial examples provide limited reusable experience. Additionally, the accumulated skills are then evaluated on held-out in-domain data from the same benchmark families, while separate benchmark families are reserved for out-of-domain evaluation. Details are provided in Appendix D.

Implementation Details We use DeepSeek-V3.2 [32] as the main backbone and evaluate Qwen3.6-Plus [45] with the skill repo built by DeepSeek-V3.2 to test cross-model transfer. The same backbone is used for skill distillation, category classification, and governance. Semantic indexing uses text-embedding-3-large [38]. Retrieval uses top-$K = 6$ with $\lambda_{\text{sim}} = 0.4$, $\lambda_u = 0.4$, and $\lambda_h = 0.2$. The learning window contains 30 trajectories. Utility updates use $\lambda_+ = 1.0$, $\lambda_- = 0.1$, $\lambda_{\text{harm}} = 0.5$, and a cosine warmup schedule for $\eta_i^{(w)}$ from 0.05 to 0.20. The agent uses tools for medical search, knowledge lookup, clinical computation, multimodal analysis, and reasoning self-regulation. Interactive benchmarks also enable patient interaction and examination ordering. All agentic baselines share the same ReAct-style executor and benchmark-specific tool suite and permissions as SkeMex, providing a common execution setup for comparing memory mechanisms. We evaluate SkeMex in offline and online modes. Offline evaluation builds a skill repo from a static split and tests it on held-out in-domain and out-of-domain sets. Online evaluation treats each benchmark as a streaming task sequence, where the repo is updated during interaction. Details are provided in Appendix B & F.

Baselines & Metrics We compare SkeMex with four groups of baselines: (i) medical specialist models [19, 49, 80], which serve as domain-specific upper bounds, (ii) a memory-free ReAct agent with the same tools [82], (iii) retrieval-augmented reflection methods [14, 52], which reuse past reflections or critiques across tasks, and (iv) self-improving memory agents [11, 17, 42, 57, 58, 64, 70, 68, 73, 75, 86, 89, 91]. All memory-based methods use the same training split for experience accumulation and retrieve memory at inference time. Close-ended benchmarks are scored by exact-match accuracy. HealthBench and LiveMedBench use rubric-based scoring, where Gemini-3-Flash [13] judges responses against predefined clinical criteria. Details are provided in Appendix E.

4.2 Main Results

Offline Mode Table 1 reports the offline in-domain evaluation, where the skill repo is built from the training split and fixed during testing. This setting tests whether prior medical trajectories can be converted into reusable skills for held-out cases. SkeMex achieves the best average performance on both backbones. With DeepSeek-V3.2, it improves ReAct from 48.20% to 56.08%, yielding **+7.88** points and outperforming the strongest non-SkeMex memory baseline by 3.84 points. With Qwen3.6-Plus, it raises ReAct from 48.63% to 59.22%, yielding **+10.59** points and a 3.14-point lead over the strongest competing memory method. These gains are consistent across models and benchmarks, especially on tasks requiring evidence organization, multi-step reasoning, and verification. Figure 3 further evaluates the frozen repo on unseen benchmark families. SkeMex remains the strongest method, with a **+13.78** point gain over ReAct, compared with about +8.24 points for the strongest competing memory baseline. The gain is especially large on AgentClinic-Text, where SkeMex improves by +34.11 points. Several memory baselines fall below ReAct on MediQ or show limited gains on AgentClinic-MM, suggesting negative transfer from less structured memory. In contrast, SkeMex stays above ReAct across all out-of-domain benchmarks, showing more stable transfer.

Online Mode Table 2 isolates the major lifecycle stages on five benchmarks across two backbones, with tools, prompts, and inference budgets held fixed within each backbone. We disable skill writing by freezing the repository at initialization; disable utility valuation by assigning equal utilities and removing feedback-based updates; disable governance by removing maturation, merging, deprecation, and capacity control; and disable value-aware retrieval by using similarity-only ranking. Full SkeMex

Table 1: Main results of performance comparison (%) in the offline setting between SkeMex and baselines. The last column shows the improvement of memory-based methods over the memory-free ReAct baseline, highlighting the gains from memory. **Bold** numbers indicate the best performance.

Backbone	Method	Text				Multimodal			Avg.
		LiveClin	MedXpertQA	HealthBench	LiveMedBench	LiveClin	MedXpertQA	MMMU	
HuluMed-32B	CoT	76.24	32.97	9.11	35.58	58.92	38.51	58.27	44.23
Lingshu-32B	CoT	72.28	23.78	8.88	30.45	58.92	32.43	49.64	39.48
MedGemma-27B	CoT	84.16	29.73	14.83	40.64	59.46	41.89	44.60	45.04
DeepSeek-V3.2	CoT	83.17	32.43	22.42	44.93	57.30	42.57	42.45	46.47
	ReAct	85.15	33.51	19.06	48.64	58.38	46.62	46.04	48.20
	Reflexion	81.19	28.11	24.54	38.75	58.92	45.95	58.99	48.06 -0.14
	CRITIC	85.15	31.89	21.53	39.26	59.46	41.22	63.31	48.83 +0.63
	Voyager	83.17	29.73	23.74	52.74	**61.62**	45.95	59.71	50.95 +2.75
	DILU	82.18	32.43	23.37	53.94	59.46	46.62	62.59	51.51 +3.31
	ExPeL	82.18	33.51	23.63	37.72	59.46	45.95	57.55	48.57 +0.37
	GM	80.20	30.27	23.49	54.10	57.84	46.62	61.87	50.63 +2.43
	Memp	75.25	31.89	21.44	54.79	56.22	44.59	58.99	49.02 +0.82
	SkillWeaver	91.09	32.43	23.42	52.15	57.30	44.59	62.59	51.94 +3.74
	AWM	79.21	33.51	22.84	53.39	58.92	45.95	56.83	50.09 +1.89
	Agent KB	86.14	33.51	23.08	54.59	58.92	46.62	58.99	51.69 +3.49
	Evolver	**92.08**	32.43	21.59	51.86	57.84	46.62	60.43	51.84 +3.64
	DC	82.18	32.43	23.76	53.76	56.76	46.62	58.27	50.54 +2.34
	MobileE	82.18	33.51	21.97	48.96	58.92	42.57	58.27	49.48 +1.28
	CFM	81.19	33.51	23.26	54.29	59.46	44.59	61.15	51.07 +2.87
	GSEM	81.19	34.59	26.00	53.20	**61.62**	48.65	60.43	52.24 +4.04
	SkeMex	**92.08**	**35.68**	**27.65**	**57.95**	**61.62**	**50.68**	**66.91**	**56.08 +7.88**
Qwen3.6-Plus	ReAct	81.19	35.14	23.98	46.93	60.54	47.30	45.32	48.63
	Reflexion	85.15	42.16	26.89	48.72	70.27	48.65	58.99	54.40 +5.77
	CRITIC	86.14	44.32	26.29	47.75	68.65	48.65	58.27	54.30 +5.67
	Voyager	82.18	41.08	25.44	49.17	70.81	49.32	52.52	52.93 +4.30
	DILU	86.14	40.54	26.89	49.97	68.65	50.68	52.52	53.62 +4.99
	ExPeL	84.16	44.32	27.56	47.42	70.27	50.68	57.55	54.57 +5.94
	GM	86.14	42.16	28.66	48.69	70.81	54.73	57.55	55.54 +6.91
	Memp	82.18	43.78	26.74	49.06	70.81	52.03	55.40	54.29 +5.66
	SkillWeaver	86.14	43.24	27.36	50.20	71.35	51.35	58.99	55.52 +6.89
	AWM	84.16	44.32	26.60	50.39	70.81	52.70	58.27	55.32 +6.69
	Agent KB	83.17	42.70	27.23	49.01	70.27	52.03	54.68	54.15 +5.52
	Evolver	84.16	44.32	25.85	47.26	69.73	52.03	54.68	54.00 +5.37
	DC	85.15	42.16	25.06	48.90	70.81	50.00	53.24	53.62 +4.99
	MobileE	85.15	43.24	27.50	48.44	70.81	49.32	55.40	54.27 +5.64
	CFM	**91.09**	43.78	28.28	50.07	69.73	51.35	58.27	56.08 +7.45
	GSEM	83.17	44.32	27.57	47.61	70.81	52.03	58.27	54.83 +6.20
	SkeMex	**91.09**	**46.49**	**31.79**	**53.97**	**74.59**	**54.73**	**61.87**	**59.22+10.59**

achieves the highest average score across both backbones. Removing skill writing reduces the average by 7.88–9.06 points, while removing utility valuation reduces it by 6.06–6.87 points. Value-aware retrieval and governance contribute 5.58–6.26 and 3.20–3.69 points, respectively. These consistent gains support the complementary roles of experience distillation, feedback-based valuation, utility-guided reuse, and repository maintenance in the closed-loop lifecycle. The following analyses examine the designs within these stages.

4.3 Ablation Studies and Analyses

Skill-memory lifecycle. Table 2 evaluates lifecycle stages on five benchmarks across two backbones under matched tools, prompts, and inference budgets. The ablations freeze the initial repository (w/o writing), fix equal utilities without feedback updates (w/o valuation), remove maturation, merging, deprecation, and capacity control (w/o governance), or use similarity-only ranking (w/o value-aware retrieval). Full SkeMex performs best on average with both backbones. Removing writing and valuation reduces average scores by 7.88–9.06 and 6.06–6.87 points, respectively; removing value-aware retrieval and governance reduces them by 5.58–6.26 and 3.20–3.69 points. These results support the complementary roles of skill creation, feedback-based valuation, retrieval, and maintenance. We further examine their internal designs below.

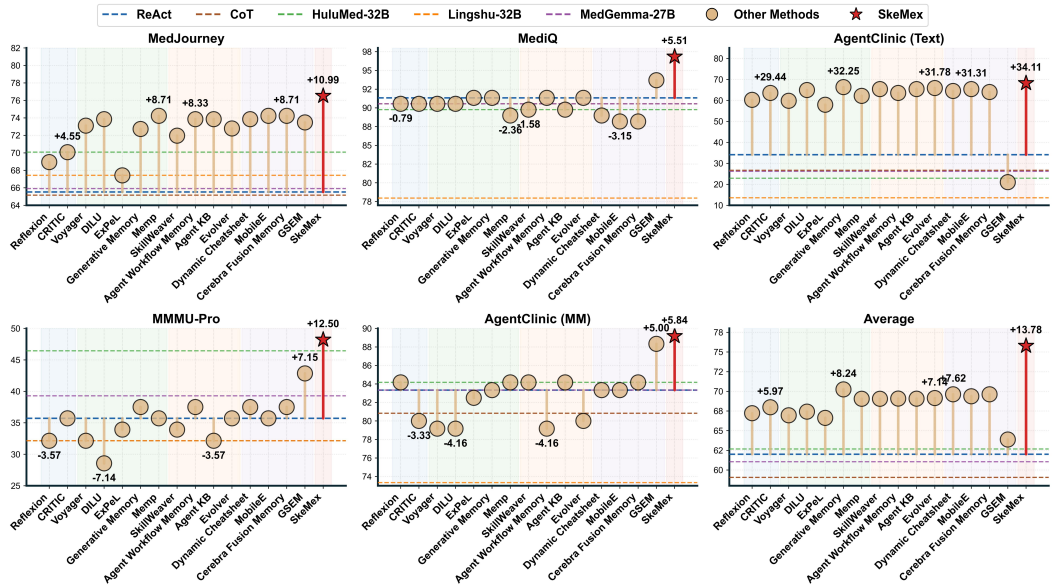

Figure 3: Main results on out-of-domain benchmarks (offline). Background colors denote different types of self-evolving memory methods, with blue for reflection-based methods and red for ours.

Buffer management and skill encoding Table 4 shows that repository quality depends strongly on which trajectories are written and how they are encoded. Full SkeMex achieves the best average score of 53.22%. Removing buffer gating gives the largest drop to 47.56%, indicating that noisy or irrelevant trajectories can corrupt skill extraction. Encoding quality is also important. Single-Prompt Encoding lowers the average to 50.97%, and removing draft review further reduces it to 48.82%. Other buffer variants also underperform the full model, suggesting that stable skill evolution benefits from selective trajectory filtering, informative case retention, and review before repository insertion.

Table 2: Ablation study on skill-memory lifecycle stages across two backbones. HB, AC, LMB, LC, and MXQA denote HealthBench, AgentClinic, LiveMedBench, LiveClin, and MedXpertQA; _T and _M indicate text-only and multimodal settings. Δ is the average score change relative to full SkeMex.

Setting	HB	AC_T	LMB	LC_M	MXQA_M	Avg.	Δ
DeepSeek-V3.2							
w/o Skill Writing	21.71	60.28	50.00	53.51	41.22	45.34	−7.88
w/o Utility Valuation	22.86	63.08	51.61	55.68	42.57	47.16	−6.06
w/o Governance	25.14	65.42	54.52	58.38	46.62	50.02	−3.20
w/o Value-aware Retrieval	19.26	62.62	53.90	55.14	47.30	47.64	−5.58
Full	**27.65**	**68.22**	**57.95**	**61.62**	**50.68**	**53.22**	–
Qwen3.6-Plus							
w/o Skill Writing	24.57	56.54	43.87	65.41	45.27	47.13	−9.06
w/o Utility Valuation	26.29	59.81	43.87	68.65	47.97	49.32	−6.87
w/o Governance	28.57	62.62	48.06	71.89	51.35	52.50	−3.69
w/o Value-aware Retrieval	26.29	60.75	43.55	69.73	49.32	49.93	−6.26
Full	**31.79**	**65.89**	**53.97**	**74.59**	**54.73**	**56.19**	–

Table 4: Ablation study on buffer and encoding. Dataset names are abbreviated for compactness.

Setting	HB	AC_T	LMB	LC_M	MXQA_M	Avg.
Buffer Management						
w/o Buffer Gating	20.47	65.42	54.47	56.22	41.22	47.56
w/o Buffer Splitting	25.58	64.02	55.47	60.00	47.97	50.61
w/o Capacity Control	26.47	66.36	56.87	59.46	50.00	51.83
FIFO Buffer	25.63	66.36	56.85	58.92	48.65	51.28
Encoding						
Single-Prompt	25.46	65.42	55.31	60.00	48.65	50.97
w/o Draft Review	23.57	65.42	55.66	56.22	43.24	48.82
Full	**27.65**	**68.22**	**57.95**	**61.62**	**50.68**	**53.22**

Table 5: Ablation study on branch combinations. G, T, and A denote general, task-level, and action-level branches, respectively.

#	Setting G	T	A	HB	AC_T	LMB	LC_M	MXQA_M	Avg.
1	✓			22.22	60.28	50.61	50.27	37.84	44.24
1		✓		19.13	60.75	49.47	52.43	45.27	45.41
2	✓	✓		23.97	64.95	50.77	54.05	42.57	47.26
2	✓		✓	23.04	64.49	52.49	60.00	43.24	48.65
2		✓	✓	19.12	61.21	46.07	52.97	45.95	45.06
3	✓	✓	✓	**27.65**	**68.22**	**57.95**	**61.62**	**50.68**	**53.22**

Utility-driven skill valuation Figure 4 reports performance drops relative to SkeMex, showing that reliable skill scoring needs more than a simple update rule. Removing baseline correction causes the largest degradation, with drops up to 7.00% on LiveMedBench and 6.09% on MedXpertQA-MM. This supports the role of category-aware reward normalization. A fixed learning rate also leads to broad declines, suggesting that adaptive updates are needed for both new and mature skills. Removing harm clamping causes a notable drop on LiveClin-MM, showing the value of harmful feedback control. The consistent decline without conditional utility indicates that skill value should depend on clinical context.

Table 3: Main results in the online setting for SkeMex and representative methods. AgentClinic and LiveClin are abbreviated as "AC" and "LC", with "_T" and "_M" denoting text-only and multimodal settings. Values show changes from the previous epoch, where red and green indicate decreases and gains.

Method	Epoch	AC_T	AC_M	LC_T	LC_M	LiveMedBench	Avg.
Agent KB	epoch@1	72.90	86.67	89.11	64.86	57.68	74.24
	epoch@2	73.83 +0.93	89.17 +2.50	90.10 +0.99	66.49 +1.63	57.16 -0.52	75.35 +1.11
	epoch@3	75.23 +1.40	87.50 -1.67	91.09 +0.99	67.57 +1.08	57.37 +0.21	75.75 +0.40
AWM	epoch@1	70.09	89.17	83.17	61.08	56.45	71.99
	epoch@2	73.36 +3.27	89.17 +0.00	79.21 -3.96	64.32 +3.24	56.08 -0.37	72.43 +0.44
	epoch@3	74.30 +0.94	91.67 +2.50	82.18 +2.97	63.24 -1.08	56.29 +0.21	73.54 +1.11
Evolver	epoch@1	72.43	85.83	93.07	65.41	56.91	74.73
	epoch@2	74.30 +1.87	88.33 +2.50	96.04 +2.97	65.95 +0.54	55.43 -1.48	76.01 +1.28
	epoch@3	75.70 +1.40	89.17 +0.84	99.01 +2.97	64.86 -1.09	56.09 +0.66	76.97 +0.96
Memp	epoch@1	72.43	87.50	84.16	64.32	55.18	72.72
	epoch@2	73.83 +1.40	87.50 +0.00	88.12 +3.96	64.32 +0.00	55.25 +0.07	73.81 +1.09
	epoch@3	73.36 -0.47	85.83 -1.67	89.11 +0.99	65.95 +1.63	55.77 +0.52	74.00 +0.19
SkillWeaver	epoch@1	70.56	89.17	94.06	60.00	56.25	74.01
	epoch@2	73.83 +3.27	85.83 -3.34	94.06 +0.00	61.08 +1.08	56.10 -0.15	74.18 +0.17
	epoch@3	74.30 +0.47	88.33 +2.50	93.07 -0.99	60.00 -1.08	56.18 +0.08	74.38 +0.20
SkeMex	epoch@1	72.90	90.00	95.05	65.41	58.59	76.39
	epoch@2	74.77 +1.87	90.83 +0.83	96.04 +0.99	66.49 +1.08	58.71 +0.12	77.37 +0.98
	epoch@3	76.17 +1.40	93.33 +2.50	96.04 +0.00	67.57 +1.08	59.68 +0.97	78.56 +1.19

Multi-branch memory structure. Table 5 shows that the three memory branches provide complementary signals. The full design achieves the highest average score of 53.22%, outperforming the best partial variant, General + Action, by 4.57 points. This indicates that task-specific medical knowledge is still needed alongside general reasoning and action-level guidance. Two-branch and single-branch variants lag behind the full model, showing that no single abstraction level is sufficient. Combining general, task-level, and action-level skills is important for robust medical problem solving. Additional ablations and extended analyses are provided in Appendix H and I.

5 Conclusion

In this paper, we introduced **SkeMex**, a post-deployment self-evolution framework that enables medical agents to improve through skill-based memory without updating model weights. By combining reusable skill distillation, utility-driven valuation, and closed-loop memory governance, SkeMex supports reliable experience accumulation and reuse across diverse clinical tasks. Extensive experiments demonstrate consistent improvements over ReAct and representative memory-based agents, as well as strong generalization across model backbones and task settings. We believe SkeMex offers a scalable step toward medical agents that can mature through continued clinical experience.

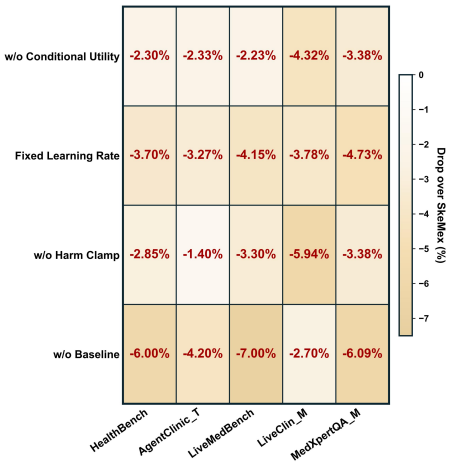

Figure 4: Ablation study on valuation module. Cell values show gaps from SkeMex.

6 Acknowledgements and Disclosures

This research was funded by the National Key Research and Development Program of China (2024YFA1307702), and the Shanghai Science and Technology Innovation Action Plan in Computational Biology (24JS2840200). The authors declare no competing interests and no financial relationships outside the submitted work that could be perceived as influencing it.

References

[1] Anthropic. Building agents with skills: Equipping agents for specialized work, 2026.

[2] Anthropic. Claude Sonnet 4.6. `https://www.anthropic.com/claude/sonnet`, 2026.

[3] Rahul K Arora, Jason Wei, Rebecca Soskin Hicks, Preston Bowman, Joaquin Quiñonero-Candela, Foivos Tsimpourlas, Michael Sharman, Meghan Shah, Andrea Vallone, Alex Beutel, et al. Healthbench: Evaluating large language models towards improved human health. *arXiv preprint arXiv:2505.08775*, 2025.

[4] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[5] Howard S Barrows and Paul J Feltovich. The clinical reasoning process. *Medical education*, 21(2):86–91, 1987.

[6] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270, 2004.

[7] Hanjie Chen, Zhouxiang Fang, Yash Singla, and Mark Dredze. Benchmarking large language models on answering and explaining challenging medical questions. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3563–3599, 2025.

[8] Chengfeng Dou, Chong Liu, Fan Yang, Fei Li, Jiyuan Jia, Mingyang Chen, Qiang Ju, Shuai Wang, Shunya Dang, Tianpeng Li, et al. Baichuan-m2: Scaling medical capability with large verifier system. *arXiv preprint arXiv:2509.02208*, 2025.

[9] Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and Jeff Dean. A guide to deep learning in healthcare. *Nature medicine*, 25(1):24–29, 2019.

[10] Lin Fan, Pengyu Dai, Zhipeng Deng, Haolin Wang, Xun Gong, Yefeng Zheng, and Yafei Ou. Evolving medical imaging agents via experience-driven self-skill discovery. *arXiv preprint arXiv:2603.05860*, 2026.

[11] Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. *arXiv preprint arXiv:2508.06433*, 2025.

[12] Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, et al. A survey of self-evolving agents: On path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*, 1, 2025.

[13] Google. A new era of intelligence with gemini 3, 2025.

[14] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. Critic: Large language models can self-correct with tool-interactive critiquing. *arXiv preprint arXiv:2305.11738*, 2023.

[15] Siyuan Guo, Cheng Deng, Ying Wen, Hechang Chen, Yi Chang, and Jun Wang. Ds-agent: Automated data science by empowering large language models with case-based reasoning. *arXiv preprint arXiv:2402.17453*, 2024.

[16] Siyuan Guo, Huiwu Liu, Xiaolong Chen, Yuming Xie, Liang Zhang, Tao Han, Hechang Chen, Yi Chang, and Jun Wang. Optimizing case-based reasoning system for functional test script generation with large language models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 4487–4498, 2025.

[17] Xiao Han, Yuzheng Fan, Sendong Zhao, Haochun Wang, and Bing Qin. Gsem: Graph-based self-evolving memory for experience augmented clinical reasoning. *arXiv preprint arXiv:2603.22096*, 2026.

[18] Xiaobin Hu, Yunhang Qian, Jiaquan Yu, Jingjing Liu, Xiaozhong Ji, Chengming Xu, Peng Tang, Chengming Xu, Peng Tang, Jiawei Liu, et al. The landscape of medical agents: A survey. 2026.

[19] Songtao Jiang, Yuan Wang, Sibo Song, Tianxiang Hu, Chenyi Zhou, Bin Pu, Yan Zhang, Zhibo Yang, Yang Feng, Joey Tianyi Zhou, et al. Hulu-med: A transparent generalist model towards holistic medical vision-language understanding. *arXiv preprint arXiv:2510.08668*, 2025.

[20] Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421, 2021.

[21] Ruofan Jin, Zaixi Zhang, Mengdi Wang, and Le Cong. Stella: Self-evolving llm agent for biomedical research. *arXiv preprint arXiv:2507.02004*, 2025.

[22] Yubin Kim, Chanwoo Park, Hyewon Jeong, Yik S Chan, Xuhai Xu, Daniel McDuff, Hyeonhoon Lee, Marzyeh Ghassemi, Cynthia Breazeal, and Hae W Park. Mdagents: An adaptive collaboration of llms for medical decision-making. *Advances in Neural Information Processing Systems*, 37:79410–79452, 2024.

[23] Janet L Kolodner. An introduction to case-based reasoning. *Artificial intelligence review*, 6(1):3–34, 1992.

[24] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[25] Yunghwei Lai, Weizhi Ma, and Yang Liu. Patient-zero: A unified framework for real-record-free patient agent generation. *arXiv e-prints*, pages arXiv–2509, 2025.

[26] Kunyao Lan, Bingrui Jin, Zichen Zhu, Siyuan Chen, Shu Zhang, Kenny Q Zhu, and Mengyue Wu. Depression diagnosis dialogue simulation: self-improving psychiatrist with tertiary memory. *arXiv preprint arXiv:2409.15084*, 2024.

[27] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[28] Binxu Li, Tiankai Yan, Yuanting Pan, Jie Luo, Ruiyang Ji, Jiayuan Ding, Zhe Xu, Shilong Liu, Haoyu Dong, Zihao Lin, et al. Mmedagent: Learning to use medical tools with multi-modal agent. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 8745–8760, 2024.

[29] Junkai Li, Yunghwei Lai, Weitao Li, Jingyi Ren, Meng Zhang, Xinhui Kang, Siyu Wang, Peng Li, Ya-Qin Zhang, Weizhi Ma, et al. Agent hospital: A simulacrum of hospital with evolvable medical agents. *arXiv preprint arXiv:2405.02957*, 2024.

[30] Shuyue S Li, Vidhisha Balachandran, Shangbin Feng, Jonathan S Ilgen, Emma Pierson, Pang W Koh, and Yulia Tsvetkov. Mediq: Question-asking llms and a benchmark for reliable interactive clinical reasoning. *Advances in Neural Information Processing Systems*, 37:28858–28888, 2024.

[31] Zhiyu Li, Shichao Song, Hanyu Wang, Simin Niu, Ding Chen, Jiawei Yang, Chenyang Xi, Huayi Lai, Jihao Zhao, Yezhaohui Wang, et al. Memos: An operating system for memory-augmented generation (mag) in large language models. *arXiv preprint arXiv:2505.22101*, 2025.

[32] Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

[33] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173, 2024.

[34] Ziyu Ma, Shidong Yang, Yuxiang Ji, Xucong Wang, Yong Wang, Yiming Hu, Tongwen Huang, and Xiangxiang Chu. Skillclaw: Let skills evolve collectively with agentic evolver. *arXiv preprint arXiv:2604.08377*, 2026.

[35] Moonshot AI. Kimi K2.6: Advancing Open-Source Coding. `https://www.kimi.com/blog/kimi-k2-6`, 2026.

[36] Jaap MJ Murre and Joeri Dros. Replication and analysis of ebbinghaus' forgetting curve. *PloS one*, 10(7):e0120644, 2015.

[37] Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Xiaoxi Jiang, and Guanjun Jiang. Trace2skill: Distill trajectory-local lessons into transferable agent skills. *arXiv preprint arXiv:2603.25158*, 2026.

[38] OpenAI. New embedding models and api updates, 2024.

[39] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[40] Charles Packer, Vivian Fang, Shishir_G Patil, Kevin Lin, Sarah Wooders, and Joseph_E Gonzalez. Memgpt: towards llms as operating systems. 2023.

[41] Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Conference on health, inference, and learning*, pages 248–260. PMLR, 2022.

[42] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22, 2023.

[43] Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all, April 2026.

[44] Qwen Team. Qwen3.6-Max-Preview: Smarter, sharper, still evolving, April 2026.

[45] Qwen Team. Qwen3.6-Plus: Towards real world agents, April 2026.

[46] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

[47] Zhiyao Ren, Yibing Zhan, Baosheng Yu, Liang Ding, Pingbo Xu, and Dacheng Tao. Healthcare agent: eliciting the power of large language models for medical consultation. *npj Artificial Intelligence*, 1(1):24, 2025.

[48] Samuel Schmidgall, Rojin Ziaei, Carl Harris, Eduardo Reis, Jeffrey Jopling, and Michael Moor. Agentclinic: a multimodal agent benchmark to evaluate ai in simulated clinical environments. *arXiv preprint arXiv:2405.07960*, 2024.

[49] Andrew Sellergren, Sahar Kazemzadeh, Tiam Jaroensri, Atilla Kiraly, Madeleine Traverse, Timo Kohlberger, Shawn Xu, Fayaz Jamil, Cían Hughes, Charles Lau, et al. Medgemma technical report. *arXiv preprint arXiv:2507.05201*, 2025.

[50] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[51] Wenqi Shi, Ran Xu, Yuchen Zhuang, Yue Yu, Jieyu Zhang, Hang Wu, Yuanda Zhu, Joyce C Ho, Carl Yang, and May Dongmei Wang. Ehragent: Code empowers large language models for few-shot complex tabular reasoning on electronic health records. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22315–22339, 2024.

[52] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

[53] David Silver and Richard S Sutton. Welcome to the era of experience. *Google AI*, 1:11, 2025.

[54] Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180, 2023.

[55] Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Mohamed Amin, Le Hou, Kevin Clark, Stephen R Pfohl, Heather Cole-Lewis, et al. Toward expert-level medical question answering with large language models. *Nature medicine*, 31(3):943–950, 2025.

[56] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.

[57] Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7080–7106, 2026.

[58] Xiangru Tang, Tianrui Qin, Tianhao Peng, Ziyang Zhou, Daniel Shao, Tingting Du, Xinming Wei, Peng Xia, Fang Wu, He Zhu, et al. Agent kb: Leveraging cross-domain experience for agentic problem solving. *arXiv preprint arXiv:2507.06229*, 2025.

[59] Xiangru Tang, Anni Zou, Zhuosheng Zhang, Ziming Li, Yilun Zhao, Xingyao Zhang, Arman Cohan, and Mark Gerstein. Medagents: Large language models as collaborators for zero-shot medical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 599–621, 2024.

[60] Tavily AI. Tavily AI GitHub Organization. `https://github.com/tavily-ai`, 2026.

[61] Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[62] Eric J Topol. High-performance medicine: the convergence of human and artificial intelligence. *Nature medicine*, 25(1):44–56, 2019.

[63] Endel Tulving et al. Episodic and semantic memory. *Organization of memory*, 1(381-403):1, 1972.

[64] 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. *arXiv preprint arXiv:2305.16291*, 2023.

[65] Hao Wang, Guozhi Wang, Han Xiao, Yufeng Zhou, Yue Pan, Jichao Wang, Ke Xu, Yafei Wen, Xiaohu Ruan, Xiaoxin Chen, et al. Skill-sd: Skill-conditioned self-distillation for multi-turn llm agents. *arXiv preprint arXiv:2604.10674*, 2026.

[66] Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. *Advances in Neural Information Processing Systems*, 36:74530–74543, 2023.

[67] Xidong Wang, Shuqi Guo, Yue Shen, Junying Chen, Jian Wang, Jinjie Gu, Ping Zhang, Lei Liu, and Benyou Wang. Liveclin: A live clinical benchmark without leakage. *arXiv preprint arXiv:2602.16747*, 2026.

[68] Zhenhailong Wang, Haiyang Xu, Junyang Wang, Xi Zhang, Ming Yan, Ji Zhang, Fei Huang, and Heng Ji. Mobile-agent-e: Self-evolving mobile assistant for complex tasks. *arXiv preprint arXiv:2501.11733*, 2025.

[69] Ziyue Wang, Junde Wu, Chang Han Low, and Yueming Jin. Medagent-pro: Towards multi-modal evidence-based medical diagnosis via reasoning agentic workflow. *arXiv e-prints*, pages arXiv–2503, 2025.

[70] Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024.

[71] Hao Wei, Jianing Qiu, Haibao Yu, and Wu Yuan. Medco: Medical education copilots based on a multi-agent framework. In *European Conference on Computer Vision*, pages 119–135. Springer, 2024.

[72] Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H Chi, et al. Evo-memory: Benchmarking llm agent test-time learning with self-evolving memory. *arXiv preprint arXiv:2511.20857*, 2025.

[73] Licheng Wen, Daocheng Fu, Xin Li, Xinyu Cai, Tao Ma, Pinlong Cai, Min Dou, Botian Shi, Liang He, and Yu Qiao. Dilu: A knowledge-driven approach to autonomous driving with large language models. *arXiv preprint arXiv:2309.16292*, 2023.

[74] David S Wishart, Yannick D Feunang, An C Guo, Elvis J Lo, Ana Marcu, Jason R Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, et al. Drugbank 5.0: a major update to the drugbank database for 2018. *Nucleic acids research*, 46(D1):D1074–D1082, 2018.

[75] Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, et al. Evolver: Self-evolving llm agents through an experience-driven lifecycle. *arXiv preprint arXiv:2510.16079*, 2025.

[76] Xian Wu, Yutian Zhao, Yunyan Zhang, Jiageng Wu, Zhihong Zhu, Yingying Zhang, Yi Ouyang, Ziheng Zhang, Huimin Wang, Zhenxi Lin, et al. Medjourney: Benchmark and evaluation of large language models over patient clinical journey. *Advances in Neural Information Processing Systems*, 37:87621–87646, 2024.

[77] Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, et al. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026.

[78] Guangzhi Xiong, Qiao Jin, Xiao Wang, Minjia Zhang, Zhiyong Lu, and Aidong Zhang. Improving retrieval-augmented generation in medicine with iterative follow-up questions. In *Biocomputing 2025: Proceedings of the Pacific Symposium*, pages 199–214. World Scientific, 2024.

[79] Gelei Xu, Xueyang Li, Yixiong Chen, Yuying Duan, Shuqing Wu, Haoxinran Yu, Ching-Hao Chiu, Juntong Ni, Ningzhi Tang, Toby Jia-Jun Li, et al. A comprehensive survey of ai agents in healthcare. *Journal of Biomedical Informatics*, page 105045, 2026.

[80] Weiwen Xu, Hou Pong Chan, Long Li, Mahani Aljunied, Ruifeng Yuan, Jianyu Wang, Chenghao Xiao, Guizhen Chen, Chaoqun Liu, Zhaodonghui Li, et al. Lingshu: A generalist foundation model for unified multimodal medical understanding and reasoning. *arXiv preprint arXiv:2506.07044*, 2025.

[81] Zhiling Yan, Dingjie Song, Zhe Fang, Yisheng Ji, Xiang Li, Quanzheng Li, and Lichao Sun. Livemedbench: A contamination-free medical benchmark for llms with automated rubric evaluation. *arXiv preprint arXiv:2602.10367*, 2026.

[82] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

[83] Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9556–9567, 2024.

[84] Xiang Yue, Tianyu Zheng, Yuansheng Ni, Yubo Wang, Kai Zhang, Shengbang Tong, Yuxuan Sun, Botao Yu, Ge Zhang, Huan Sun, et al. Mmmu-pro: A more robust multi-discipline multimodal understanding benchmark. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15134–15186, 2025.

[85] Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

[86] Guibin Zhang, Haotian Ren, Chong Zhan, Zhenhong Zhou, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng Yan. Memevolve: Meta-evolution of agent memory systems. *arXiv preprint arXiv:2512.18746*, 2025.

[87] Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, et al. Evoskills: Self-evolving agent skills via co-evolutionary verification. *arXiv preprint arXiv:2604.01687*, 2026.

[88] Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, et al. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*, 2026.

[89] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19632–19642, 2024.

[90] Xuejiao Zhao, Siyan Liu, Su-Yin Yang, and Chunyan Miao. Medrag: Enhancing retrieval-augmented generation with knowledge graph-elicited reasoning for healthcare copilot. In *Proceedings of the ACM on Web Conference 2025*, pages 4442–4457, 2025.

[91] Boyuan Zheng, Michael Y Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, et al. Skillweaver: Web agents can self-improve by discovering and honing skills. *arXiv preprint arXiv:2504.07079*, 2025.

[92] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 19724–19731, 2024.

[93] Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, et al. Memento: Fine-tuning llm agents without fine-tuning llms. *arXiv preprint arXiv:2508.16153*, 2025.

[94] Yucheng Zhou, Lingran Song, and Jianbing Shen. Mam: Modular multi-agent framework for multi-modal medical diagnosis via role-specialized collaboration. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 25319–25333, 2025.

[95] Yinghao Zhu, Yifan Qi, Zixiang Wang, Lei Gu, Dehao Sui, Haoran Hu, Xichen Zhang, Ziyi He, Junjun He, Liantao Ma, et al. Healthflow: A self-evolving ai agent with meta planning for autonomous healthcare research. *arXiv preprint arXiv:2508.02621*, 2025.

[96] Yuxin Zuo, Shang Qu, Yifei Li, Zhangren Chen, Xuekai Zhu, Ermo Hua, Kaiyan Zhang, Ning Ding, and Bowen Zhou. Medxpertqa: Benchmarking expert-level medical reasoning and understanding. *arXiv preprint arXiv:2501.18362*, 2025.

Appendix Contents

A Agent Architecture **18**

A.1 Runtime Components 18

A.2 Stepwise Execution Protocol 18

A.3 Integration of Skill Memory into the Agent Loop 19

A.4 Intra-Task Context Control 19

B Tool Suite **20**

C Additional Details of SkeMex **23**

D Dataset Details **23**

E Baselines & Metrics **27**

E.1 Baselines 27

E.2 Metrics 28

F Implementation Details **29**

G Sensitivity Analysis **32**

H Ablation Study **34**

I Further Analyses **36**

I.1 Offline OOD Generalization 36

I.2 Cross Backbone Generalization 37

I.3 Cross Backbone Skill Transfer 39

I.4 Execution Cost and Interaction Depth 40

I.5 Impact of Training Data Order 43

J Limitations **43**

K Case Study **44**

L Prompts **50**

L.1 Core Agent Interaction Prompts 50

L.2 Memory Evolution Pipeline Prompts 51

L.3 Automatic Evaluation Prompts 59

A Agent Architecture

In this section, we describe the execution backbone of our agent system and explains how the skill memory introduced in the main paper is connected to task execution. The agent is implemented as a bounded ReAct loop [82], where each task is solved through an iterative sequence of model decisions, tool executions, observations, and final response generation. The evolution module is attached to this backbone through a runtime skill injection interface, while the core agent loop remains responsible for maintaining the task state, enforcing the output protocol, invoking tools, and deciding when to terminate. This separation allows the learned skill memory to guide the agent without changing the basic semantics of execution. In other words, the system can use accumulated experience to influence the agent's decisions while keeping the underlying task solving process stable.

A.1 Runtime Components

For each input sample, the system constructs or reuses an `AgentLoop` instance. The loop is organized around several runtime components that jointly support structured reasoning, tool use, memory rendering, and context control. The language model interface receives the system prompt and the per step task prompt, then produces structured outputs under the required protocol. The tool registry exposes dataset specific tools that can be called by name, renders the available tool list into the system prompt, and executes one selected tool call at a time. The conversation memory stores the local interaction history of the current task, including the original user request, previous assistant outputs, and observations returned by tools. In addition, the evolution runtime context injects the task category and retrieved skills into the prompt at each step. This injected block is rendered dynamically and is not written into the conversation memory, which avoids repeated duplication across turns. When enabled, the context guard monitors the accumulated trajectory and adds control signals when the task becomes long, repetitive, or close to the context budget.

Table 6: Runtime components used by the agent execution backbone.

Component	**Role in execution**
Language model	Produces structured outputs under the system prompt and the current task prompt. Each output contains a reasoning block followed by either a tool call or a final response.
Tool registry	Renders the available tool list into the system prompt and executes one selected tool call at a time. It also receives per sample runtime context, such as case information or dataset metadata.
Conversation memory	Stores the task local interaction history and renders it into a plain text context block before each model call. The history contains the user request, previous assistant outputs, and observations returned by tools.
Evolution runtime context	Injects retrieved experience skills and task categories into the prompt at every step. This block is rendered freshly and is not written into the conversation memory, which avoids repeated duplication across turns.
Context guard	Provides intra-task context control through loop breaking, confirmed finding pinning, and budget aware history trimming.

At the beginning of each sample, the batch runner clears both the conversation memory and the recorded step history. This reset makes each benchmark instance an independent task and prevents information leakage from earlier samples. If the input sample contains images, their paths are appended to the user request as a numbered list. For vision capable models, the image contents are passed only to the first model call. Later steps rely on the textual conversation history, which avoids repeatedly sending the same visual input and keeps subsequent prompts more compact. Before execution starts, the tool registry is also updated with the sample specific runtime context, so that tools can access the information required by the current benchmark instance.

A.2 Stepwise Execution Protocol

The agent follows a strict structured output protocol that turns open ended model generation into a bounded and auditable action process. For multi step problems, the model may begin with an optional

<planning> block, but every turn must contain a <reasoning> block. This reasoning block must be immediately followed by exactly one action. The model either emits a <tool> block containing a tool name and a JSON input, or emits a <response> block that terminates the task with the final answer. By requiring every step to end in one of these two actions, the protocol converts free form generation into a small finite action interface and keeps the execution flow deterministic. At each step, the prompt is assembled from the system instruction, the rendered conversation history, the original user request, and the evolution runtime context when available. The system instruction defines the output grammar, the available tools, and the rule that only one tool may be called in a single turn. The conversation history provides the accumulated task context, including previous assistant outputs and tool observations. The original request is repeated to help the model retain the task objective after several interaction rounds. The evolution block adds retrieved skills and task categories when the evolution module has provided them. These skills are presented as prioritized experience references with explicit applicability conditions, so the model can consult them before selecting the next action and refer to the corresponding skill identifiers when a skill influences its reasoning.

After the model produces an output, the system records the complete content in memory and parses it according to the protocol. If the output contains a valid tool block, the parser extracts the tool name and JSON input, the tool registry executes the selected tool, and the returned observation is appended to the conversation memory before the loop proceeds to the next step. If the output contains a response block, the task terminates and the full trajectory is saved for later analysis. When the output violates the required format, the agent performs controlled recovery by appending a corrective observation rather than terminating immediately. For example, malformed JSON produces a parse error observation, an unknown tool produces an explicit tool name error, and a reasoning block without a following action prompts the model to complete the missing tool or response block. The loop is also constrained by a final step convergence rule. Once the maximum number of allowed steps is reached, the next prompt explicitly forbids further tool use and requires the model to provide a final response. If the model still fails to generate a valid response tag, the system removes structural tags and uses the remaining content as a fallback answer. This rule prevents long running tasks from ending without an output and makes the bounded step budget compatible with benchmark evaluation.

A.3 Integration of Skill Memory into the Agent Loop

The skill memory does not replace the agent policy, but enters the agent through a runtime context that is set immediately before solving each sample. The evolution runner first assigns a task category to the sample and retrieves a small set of relevant skills from the skill repository. It then passes the task category, the rendered skill block, and the identifiers of the injected skills to the agent through the context setter. During every step of the same task, this context is rendered into the prompt. Since the block is not persisted in conversation memory, the same skill content does not accumulate across turns. This design creates a clear boundary between retrieval and action. Retrieval determines which experiences are visible to the model, while the agent still decides whether the applicability conditions are satisfied in the current step. A retrieved skill therefore functions as a conditional reference rather than a hard command, which allows the agent to adopt it, ignore it, or override it according to the current observations and task state.

The recorded trajectory also provides evidence for later skill evolution. By inspecting the reasoning traces, tool calls, observations, and final answers, the trajectory encoder can identify which injected skills were adopted, ignored, or harmful. In this way, the execution backbone supplies the behavioral evidence needed by the evolution loop, while the evolution loop supplies compact task specific guidance to the execution backbone.

A.4 Intra-Task Context Control

Medical tasks often require multiple tool calls, intermediate observations, and careful synthesis. If the agent simply accumulates the full interaction history, several practical problems may arise. The agent may repeat the same ineffective action, important early evidence may become less visible after many later observations, and long tool outputs may exceed the intended prompt budget. To address these issues, the context guard introduces three mechanisms within a single task: loop breaking, confirmed finding pinning, and budget aware history trimming. These mechanisms do not modify the stored trajectory. Instead, they only adjust how the history is rendered into the next prompt, so the complete execution record remains available for downstream analysis. The loop breaking mechanism examines

recent actions before each step. If the same tool action with the same input has been repeated for a configured number of consecutive steps, the context guard appends a notice to the prompt, telling the model that the repeated action has not produced progress and that it should change strategy, use a different tool, or provide a final answer when sufficient evidence is already available. This mechanism is conservative because it only activates on exact repeated action signatures, which allows the agent to reuse the same tool when the inputs are meaningfully different.

The confirmed finding pinning mechanism is designed to preserve important early evidence. Once the task reaches a configured step index, the context guard extracts observations from the early steps and uses a lightweight model call to summarize only confirmed findings. The resulting summary is stored as pinned findings and appended to future prompt histories under a dedicated marker. This summary is generated once per task to limit overhead, and its purpose is not to introduce new information but to keep verified facts visible as the conversation becomes longer. This is especially useful in medical reasoning, where early observations may contain patient attributes, test results, or constraints that should remain available during final synthesis.

The budget aware history trimming mechanism further controls prompt length. Before each model call, the context guard estimates the token usage of the step history. If the estimate exceeds a configured fraction of the context budget, older observations are shortened while the most recent steps are preserved in fuller form. The rendered history also receives a context trim notice, which tells the model that earlier content has been compressed and that it should rely on pinned findings and recent steps. This differs from ordinary truncation because the complete trajectory remains stored in the agent record. Only the prompt view is compressed, which means downstream trajectory analysis and skill evolution can still access the untrimmed execution trace.

Table 7: The three intra-task context control mechanisms used by the agent.

Mechanism	**Trigger**	**Effect on the next prompt**
Loop breaking	The same action and input are repeated for the configured number of recent steps.	Adds a warning that asks the model to avoid repeating the action and to change strategy or answer.
Confirmed finding pinning	The task reaches the configured pinning step and early observations are available.	Adds a concise confirmed findings block distilled from early observations.
Budget aware trimming	The estimated prompt history length exceeds the configured budget ratio.	Shortens older observations, preserves recent steps, and adds a notice explaining that earlier observations were compressed.

Together, these mechanisms make the agent loop more stable without changing the external task interface. Loop breaking reduces redundant exploration, confirmed finding pinning protects salient evidence, and budget aware trimming keeps the prompt within a controlled length. Since all three mechanisms are injected as runtime context rather than stored as ordinary user messages, they guide the next decision while preserving the faithfulness of the underlying trajectory. This design is also compatible with the self evolution pipeline because the saved step history still records the original model outputs, tool calls, observations, errors, and final answer for later encoding and utility assessment.

B Tool Suite

Our agent is equipped with a modular tool suite that decomposes clinical problem solving into reusable capabilities. These tools cover six major categories: **external evidence retrieval, structured medical knowledge access, quantitative clinical computation, multimodal perception, reasoning control, and benchmark-specific interaction**. External evidence retrieval includes general web search and medical evidence retrieval. Structured medical knowledge access provides drug information lookup, drug interaction checking, and biomedical concept lookup. Quantitative clinical computation includes a safe numerical calculator, a medical unit converter, and a clinical score calculator. Multimodal perception separates diagnostic image analysis from OCR and chart reading. Reasoning control supports intermediate reflection and final answer verification. Benchmark-specific tools provide

patient dialogue, examination requests, and image access for interactive clinical benchmarks. All tools share a common execution interface: each call receives a structured payload, returns a compact observation, and exposes a parameter schema that can be injected into the agent prompt or requested when needed. This design keeps the action space flexible enough for heterogeneous medical tasks while preserving a uniform execution protocol across datasets. The detailed prompts for tools that involve LLM calls are provided in Appendix L.

- **General web search tool.** The general web search tool is used when the agent needs current or open-domain information, such as recent clinical updates, public medical resources, or general background facts that may not be reliably captured by the model itself. In our implementation, this tool is supported by the Tavily Search API [60]. Given a natural-language query, it retrieves relevant web results and returns a compact synthesized observation for the agent to use in subsequent reasoning.
- **Medical retrieval tool.** The medical retrieval tool is used when the agent needs domain-specific medical evidence for diagnosis, treatment, pharmacology, or biomedical reasoning. It provides access to curated medical sources such as PubMed, textbooks, StatPearls, Wikipedia, and combined biomedical corpora. In implementation, the tool uses a MedRAG-style [78] retrieval backend when local corpora are available, and can fall back to public scholarly search or PubMed-based retrieval when needed. Its purpose is to ground the agent's reasoning in medical references rather than relying only on internal model knowledge.
- **Drug information lookup tool.** The drug information lookup tool is used when the agent needs medication-specific information, including indications, mechanisms, contraindications, adverse reactions, warnings, and drug classes. It accepts a drug name and returns a concise summary of clinically relevant information. The implementation prioritizes local DrugBank-style resources [74] when available, with public sources such as RxNorm and OpenFDA labels used as fallbacks. This tool helps the agent reason about medication choice, adverse effects, and contraindications.
- **Drug interaction checking tool.** The drug interaction checking tool is used when a case involves multiple medications or when a proposed treatment must be checked for safety. It takes two or more drug names and reports known pairwise interactions when available. The implementation uses local structured interaction data when present and can fall back to public drug interaction resources [74]. This tool is intended to support medication reconciliation, treatment safety assessment, and adverse-effect reasoning.
- **Biomedical concept lookup tool.** The biomedical concept lookup tool is used when the agent needs to clarify a disease, symptom, procedure, or clinical term using standardized biomedical knowledge. It can provide definitions, semantic categories, and related concepts. When configured, the tool queries UMLS resources [6], and otherwise falls back to public NLM or MedlinePlus-style sources. This tool is useful for disambiguating clinical terminology and connecting patient-facing descriptions to standardized medical concepts.
- **Safe numerical calculator.** The safe numerical calculator is used for arithmetic and formula-based reasoning, especially when exact computation is needed for dosage, rate conversion, numerical comparison, or risk-score components. The implementation evaluates mathematical expressions through a restricted calculator rather than allowing arbitrary code execution. This helps reduce simple numerical errors while keeping the returned observation concise.
- **Medical unit converter.** The medical unit converter is used when clinical values or laboratory results are reported in different units. It supports common clinical conversions involving mass, volume, concentration, pressure, temperature, selected analyte-specific conversions, electrolyte equivalents, and HbA1c. The tool takes a value, source unit, target unit, and optionally a substance name for analyte-specific conversion. Its purpose is to normalize values before applying clinical thresholds or scoring rules.
- **Clinical score calculator.** The clinical score calculator is used when a task requires a standardized clinical risk score, severity score, or physiological index. It supports commonly used scores such as CHA_2DS_2-VASc, HAS-BLED, HEART, Wells scores, PERC, CURB-65, SOFA, qSOFA, NEWS2, Child-Pugh, MELD, Glasgow Coma Scale, $ABCD^2$, Glasgow-Blatchford, RCRI, BMI, eGFR, Cockcroft-Gault, and MDRD. The tool computes the score

Table 8: Dataset-specific tool availability. A check mark indicates that the corresponding tool is enabled.

Dataset	Web	Med. Ret.	Drug Info	Drug Int.	Concept	Calc.	Unit	Score	Image	OCR	Reflect	Verify	AC Reply	AC Exam	AC Image
AgentClinic-MM	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
AgentClinic-Text	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓	✓	✓	
LiveClin-MM	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			
LiveClin-Text	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓			
LiveMedBench	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓			
MMMU	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			
MMMU-Pro	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			
MedJourney	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓			
MedXpertQA-MM	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			
MedXpertQA-Text	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓			
MediQ	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓			
HealthBench	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓			

from the provided clinical variables and returns both the result and a brief interpretation. This helps reduce errors in threshold-based clinical decision making.

- **Medical image analysis tool.** The medical image analysis tool is used when the task contains clinically meaningful visual content, such as X-ray, CT, MRI, ECG images, pathology slides, fundus photographs, or other medical images. The tool receives a single image and asks a vision-language medical model [19] to produce structured clinical findings, an impression, and possible differential diagnoses. Its purpose is to support diagnostic reasoning when the image itself contains clinically relevant evidence.
- **OCR and chart reading tool.** The OCR and chart reading tool is used for document-like images where the main information is textual, numerical, or tabular. Examples include laboratory reports, medication charts, scanned notes, clinical forms, and tables embedded in images. Unlike the medical image analysis tool, this tool focuses on extracting displayed information [4] rather than interpreting visual pathology. This separation helps the agent distinguish between reading clinical data and diagnosing from medical images.
- **Reflection tool.** The reflection tool is used during intermediate reasoning when the agent needs to check whether its current reasoning is complete, whether additional evidence is needed, or whether important differentials or assumptions have been missed. It provides a compact critique and a suggested next step. In implementation, this tool uses a critic-style model when available, with simpler fallback checks when necessary. Its purpose is to improve reasoning control rather than to generate a separate final answer.
- **Verifier tool.** The verifier tool is used near the end of a trajectory to assess a proposed final answer before delivery. It compares the original question and candidate answer from an independent evaluation perspective, focusing on factual correctness, clinical safety, and completeness. This tool separates answer generation from answer checking, allowing the agent to audit its final response before producing it.
- **AgentClinic patient reply tool.** The AgentClinic patient reply tool is used in interactive clinical cases when the doctor-agent needs additional subjective history from the patient [48]. The agent asks a question, and the tool returns a first-person patient response based on patient-visible information in the benchmark case. The implementation prevents the simulated patient from revealing hidden diagnoses, objective test results, or information that a real patient would not know. This tool supports natural history taking while preserving benchmark constraints.
- **AgentClinic examination request tool.** The AgentClinic examination request tool is used when the doctor-agent needs objective clinical findings, such as vital signs, physical examination findings, laboratory results, imaging reports, or neurological examination results [48]. The agent submits a concise examination request, and the tool returns findings that are available in the bound case. If the requested information is not available, the tool

reports that limitation instead of inventing new findings. This separates subjective history gathering from objective clinical investigation.

- **AgentClinic image request tool.** The AgentClinic image request tool is used in multimodal AgentClinic cases when an associated image is available [48]. It retrieves the image path or URL from the benchmark case but does not interpret the image. The agent can then pass the retrieved image to the medical image analysis tool when visual interpretation is needed. This keeps benchmark data access separate from image-based clinical reasoning.

C Additional Details of SkeMex

This section provides an operational summary of SkeMex as introduced in the main text. We use the same M-MDP notation $T_{M\text{-}MDP} = \langle S, A, P, E, \gamma, M \rangle$, where $s_t \in S$ is the agent state, $a_t \in A$ is a reasoning action, tool action, or final answer action, and M is the external skill memory. Each memory unit is instantiated as a skill $m_i = (k_i, c_i, u_i)$, where k_i is the retrieval key, c_i is reusable procedural content, and u_i is an estimated utility statistic. At learning window w, the current skill repository is denoted by $M^{(w)}$, which contains general, task-level, and action-level branches. For a task instance x_n, the agent constructs an initial state $s_0(x_n)$, routes the task to a clinical category κ_n, retrieves a compact skill set $m_n \subseteq M^{(w)}$ using the retrieval distribution $\mu(m_n \mid s_0(x_n), M^{(w)})$, and then acts with the backbone policy $p_\theta(a_t \mid s_t, m_n)$. Since SkeMex retrieves skills once at the episode onset, m_n is kept fixed throughout the trajectory. SkeMex does not update θ. Instead, post-deployment improvement comes from evolving M. After the agent completes the task, the resulting trajectory is written as $\tau_n = (x_n, \kappa_n, m_n, H_n, \hat{y}_n, y_n, r_n)$, where $H_n = \{(s_t, a_t, o_t)\}_{t=0}^{T_n-1}$ stores the step history, o_t is the observation returned by the environment or tools, $\hat{y}_n$ is the final answer, y_n is the reference signal when available, and r_n is the task reward. A trajectory gate keeps only informative traces in the window buffer $\mathcal{B}^{(w)}$, such as nontrivial successes, useful failures, or trajectories that reveal whether retrieved skills helped or harmed the agent. At the end of each learning window, the writing operator $\mathcal{W}$ distills buffered trajectories into new skills or local patches to existing skills, the valuation operator $\mathcal{V}$ updates skill utilities from reward and adoption evidence, and the governance operator $\mathcal{G}$ maintains the repository by merging redundant skills, promoting stable skills, deprecating harmful or low-utility skills, and enforcing branch capacity constraints. Here, $\mathcal{W}(M^{(w)}, \mathcal{B}^{(w)})$ denotes applying the trajectory-level writing process over all gated trajectories in the window, followed by draft review. The overall memory update follows the closed-loop form

$$M^{(w+1)} = \mathcal{G}\Big(\mathcal{V}\big(\mathcal{W}(M^{(w)}, \mathcal{B}^{(w)})\big)\Big), \tag{10}$$

which corresponds to the Read, Write, Assess, and Govern lifecycle described in the main paper.

D Dataset Details

We evaluate SkeMex on nine medical benchmarks that jointly cover interactive clinical decision making, patient-centered clinical reasoning, knowledge-intensive medical question answering, and multimodal medical understanding. The selected benchmarks differ in task format, modality, evaluation protocol, and clinical scope, which allows us to test whether the skill repository can support both text-only and multimodal reasoning across heterogeneous medical settings.

- **AgentClinic.** AgentClinic [48] is a simulated clinical environment designed for evaluating medical agents in interactive scenarios. The doctor-agent can ask patients questions, collect additional history, request examinations, and use multimodal information before making diagnostic or treatment decisions. We use both the text-only and multimodal settings, which allows us to evaluate whether SkeMex helps in sequential information gathering and image-supported clinical reasoning.
- **LiveClin.** LiveClin [67] is a live clinical benchmark built from contemporary peer-reviewed case reports. It is designed to reduce data leakage and knowledge obsolescence by updating the benchmark over time. Its cases cover realistic clinical pathways and include both text-only and multimodal questions. We use LiveClin to evaluate whether the agent can handle recent, clinically grounded, and often complex patient cases.

Algorithm 1 Closed-loop skill memory evolution in SkeMex

Require: Initial skill repository $M^{(0)}$, task stream $\mathcal{D} = \{x_n\}_{n=1}^{N}$, retrieval budget K, window size L, governance period q
Require: Backbone policy p_θ, retrieval distribution μ, writing operator $\mathcal{W}$, valuation operator $\mathcal{V}$, governance operator $\mathcal{G}$

Initialize window index $w \leftarrow 0$ and buffer $\mathcal{B}^{(w)} \leftarrow \emptyset$
for $n = 1$ to N **do**
  Construct initial state $s_0 \leftarrow s_0(x_n)$
  Route the task to clinical category κ_n
  Read: retrieve skills $m_n \leftarrow \mathrm{TopK}_K\big(\mu(\cdot \mid s_0, M^{(w)})\big)$, using κ_n for category-aware scoring
  Inject m_n into the agent context and keep it fixed for the episode
  Initialize step history $H_n \leftarrow \emptyset$ and time step $t \leftarrow 0$
  while the episode has not terminated **do**
    Decode action $a_t \sim p_\theta(\cdot \mid s_t, m_n)$
    Execute a_t in the environment or tool interface and observe o_t
    Append (s_t, a_t, o_t) to H_n
    if a_t is a final-answer action **then**
      Terminate the episode
    else
      Update state $s_{t+1} \leftarrow \mathrm{Update}(s_t, a_t, o_t)$
      $t \leftarrow t + 1$
    end if
  end while
  Obtain final answer $\hat{y}_n$, optional reference y_n, and reward r_n
  Build trajectory $\tau_n = (x_n, \kappa_n, m_n, H_n, \hat{y}_n, y_n, r_n)$
  Write buffer: if $\mathrm{Gate}(\tau_n) = 1$, add τ_n to $\mathcal{B}^{(w)}$
  if $|\mathcal{B}^{(w)}| = L$ **then**
    Write: construct candidate memory $\widehat{M}^{(w)} \leftarrow \mathcal{W}(M^{(w)}, \mathcal{B}^{(w)})$
    Assess: update utilities $\widetilde{M}^{(w)} \leftarrow \mathcal{V}(\widehat{M}^{(w)}, \mathcal{B}^{(w)})$
    if $(w+1) \bmod q = 0$ **then**
      Govern: set $M^{(w+1)} \leftarrow \mathcal{G}(\widetilde{M}^{(w)})$
    else
      Set $M^{(w+1)} \leftarrow \widetilde{M}^{(w)}$
    end if
    $w \leftarrow w + 1$
    Reset buffer $\mathcal{B}^{(w)} \leftarrow \emptyset$
  end if
end for
return evolved skill repository $M^{(w)}$

- **MedJourney.** MedJourney [76] evaluates model performance across a broad patient journey rather than isolated exam-style questions. It covers multiple stages of healthcare delivery, including planning, access, clinical service, and ongoing care. We include MedJourney because it tests whether the agent can reason over patient-centered clinical workflows and produce contextually appropriate decisions.
- **LiveMedBench.** LiveMedBench [81] is a contamination-resistant benchmark built from real-world medical cases and evaluated with case-specific rubric criteria. It emphasizes open-ended clinical reasoning, but its rubric structure makes automatic evaluation more reliable than unconstrained free-form judging. We use LiveMedBench to assess whether SkeMex improves clinically complete and patient-specific answers under rubric-based evaluation.
- **HealthBench.** HealthBench [3] evaluates health-related conversations with physician-authored rubrics. It covers diverse healthcare scenarios, including clinical advice, safety, communication, guideline adherence, and user-facing health support. We include HealthBench because it provides a structured way to evaluate open-ended medical responses beyond simple exact-match accuracy.

- **MediQ.** MediQ [30] focuses on interactive question asking under incomplete clinical information. Instead of forcing the model to answer immediately, the benchmark emphasizes whether the agent can identify missing information and ask useful follow-up questions. We use MediQ to evaluate information-seeking behavior and reliable reasoning under uncertainty.
- **MedXpertQA.** MedXpertQA [96] is an expert-level medical reasoning benchmark spanning multiple specialties and body systems. It contains both text and multimodal subsets, with questions designed to require advanced medical knowledge and multi-step reasoning. We use both subsets to evaluate whether SkeMex can improve difficult specialty-level reasoning and multimodal interpretation.
- **MMMU.** MMMU [83] is a multidisciplinary multimodal benchmark collected from college-level exams, quizzes, and textbooks. We use only the Health & Medicine track. This subset includes medical images, diagrams, charts, tables, and knowledge-intensive questions, making it useful for testing multimodal medical understanding in an academic setting.
- **MMMU-Pro.** MMMU-Pro [84] strengthens MMMU by filtering out questions that can be answered from text alone, augmenting answer choices, and introducing settings that require stronger visual-textual integration. We use the Health & Medicine portion to evaluate whether the agent can handle more robust multimodal medical questions where superficial textual shortcuts are less effective.

To make skill accumulation informative and evaluation reliable, we apply a unified preprocessing pipeline before offline evolution and testing. This pipeline also helps reduce unnecessary experimental cost by removing samples that are unlikely to provide reusable experience or stable evaluation signals. First, when official difficulty labels, benchmark metadata, or subset annotations are available, we prioritize difficult or reasoning-intensive samples. For datasets without explicit difficulty annotations, we retain cases that require multi-step reasoning, integration of multiple clinical clues, tool use, or interpretation of multimodal evidence. This choice is important because trivial cases often provide limited reusable experience for skill construction. Second, we remove examples whose query description is inconsistent with the attached visual input. This includes cases where the recorded number of images does not match the prompt, image paths are missing or invalid, or the question refers to a figure although the sample is configured as text-only. Third, we exclude fully open-ended examples that do not provide answer options, reference answers, rubric criteria, or other reliable evaluation signals. Open-ended datasets are retained only when they include structured scoring rubrics or benchmark-provided evaluation criteria. Fourth, we normalize answer formats, option labels, image path fields, and modality tags across datasets so that the agent receives a consistent input format. Finally, we remove near-duplicate samples across skill accumulation and test splits to reduce leakage. This preprocessing strategy preserves challenging and clinically diverse cases while filtering out samples that are ambiguous, unevaluable, modality-inconsistent, or unlikely to contribute useful reusable skills. It also keeps the experimental cost manageable without weakening the main purpose of the benchmark pool: evaluating whether SkeMex can accumulate and transfer meaningful clinical experience.

After preprocessing, the evaluation pool contains 3,278 samples across 12 dataset configurations derived from the nine benchmarks. AgentClinic, LiveClin, and MedXpertQA are separated into text-only and multimodal configurations because they use different input modalities and tool availability. MMMU and MMMU-Pro are restricted to their Health & Medicine subsets. Table 9 reports the resulting data size and category distribution. For compactness, we use the following abbreviations: DD for differential diagnosis, TP for treatment planning, IMG for medical imaging interpretation, GFA for guideline update and formula adjustment, LNC for lifestyle and nutrition counseling, DOC for clinical documentation and protocol adherence, EPI for epidemiologic study design and control selection, DDI for drug interaction and medication safety review, ACAD for non clinical academic problem solving, PVH for preventive vaccination and travel health counseling, PERI for perioperative patient education and counseling, SURG for intraoperative surgical safety and procedure guidance, LIT for literature retrieval, OB for obstetric delivery assistance, GEN for genetic linkage and recombination analysis, BILL for clinical billing and administrative support, SOC for non clinical social interaction, SPAT for spatial reasoning and image navigation, and FLY for fitness to fly and physiologic risk assessment.

Table 9: Dataset size and category distribution after preprocessing.

Dataset	**Samples**	**Category distribution**
AgentClinic Text	214	DD 214
AgentClinic MM	120	DD 115, TP 2, IMG 3
HealthBench	350	DD 93, GFA 57, LNC 67, DOC 49, TP 32, LIT 14, BILL 9, PVH 8, SOC 7, OB 6, DDI 6, IMG 1, PERI 1
LiveClin Text	205	DD 78, TP 77, GFA 11, IMG 22, DDI 6, PERI 4, LNC 2, OB 2, PVH 1, LIT 1, DOC 1
LiveClin MM	371	DD 159, IMG 104, TP 69, SURG 13, GFA 8, OB 8, LNC 3, DOC 2, PERI 2, DDI 2, LIT 1
LiveMedBench	623	DD 254, TP 247, LNC 41, PVH 22, DDI 21, PERI 16, IMG 7, GFA 5, DOC 3, LIT 2, FLY 2, OB 1, SOC 1, SURG 1
MediQ	127	DD 74, TP 34, GFA 7, PVH 4, IMG 2, DDI 2, DOC 2, FLY 1, BILL 1
MedJourney	264	DD 115, TP 102, DDI 16, IMG 13, GFA 8, DOC 5, BILL 2, PVH 2, LNC 1
MedXpertQA Text	372	DD 182, TP 145, GFA 8, IMG 7, DOC 7, SURG 5, DDI 5, PVH 3, LNC 2, FLY 2, OB 2, PERI 2, LIT 1, EPI 1
MedXpertQA MM	299	DD 152, TP 99, IMG 36, DDI 4, EPI 4, SURG 3, GFA 1
MMMU	277	IMG 65, DD 65, EPI 61, ACAD 39, GEN 13, GFA 12, SPAT 10, DOC 5, SOC 4, TP 1, DDI 1, LIT 1
MMMU-Pro	56	IMG 21, DD 16, ACAD 12, EPI 6, GEN 1
Total	**3,278**	**19 task categories across text-only, multimodal, interactive, and rubric-based clinical settings**

E Baselines & Metrics

E.1 Baselines

We compare SkeMex with four groups of baselines. The first group consists of medical specialist models, which provide domain-specific reference points for medical reasoning and multimodal medical understanding. The second group contains a memory free tool-using agent, which isolates the effect of the proposed skill memory under the same tool access setting. The third group includes reflection based methods, which reuse prior critiques or self-reflections to support later reasoning. The fourth group contains self improving memory agents, which store, distill, or organize experience across tasks. Unless otherwise specified, all agentic baselines are equipped with the same tool suite as SkeMex when the corresponding benchmark permits tool use. For memory based baselines, we use the same training split for experience accumulation and retrieve stored experience at inference time according to each method's original design or the closest faithful adaptation.

- **Lingshu.** Lingshu [80] is a medical specialist foundation model designed for unified medical understanding and reasoning. It supports both text and multimodal medical tasks, which makes it a useful reference point for evaluating clinical and visual medical reasoning.
- **Hulu-Med.** Hulu-Med [19] is a medical-domain model developed for broad clinical and biomedical reasoning. It serves as another specialist baseline that reflects the performance of adapting the model itself to medical tasks.
- **MedGemma.** MedGemma [49] is a family of medical models for text and image comprehension. We include it as a specialist reference for benchmarks that involve medical knowledge and multimodal understanding.
- **Vanilla ReAct.** Vanilla ReAct [82] is a tool-using agent that interleaves reasoning and actions. We use the same task prompts and the same available tools as SkeMex, but remove external skill memory, skill retrieval, utility estimation, and repository governance. This baseline directly measures how much improvement comes from the evolving skill memory rather than from tool access alone.
- **Reflexion.** Reflexion [52] improves agents by writing natural-language reflections after receiving feedback. These reflections are reused in later trials to guide behavior. To make it suitable for our cross-task setting, we allow stored reflections to be retrieved for related medical tasks. This baseline tests whether reflective feedback alone is sufficient for reusable clinical improvement.
- **CRITIC.** CRITIC [14] uses external feedback to critique and revise model outputs. We adapt it by storing previous critiques and retrieving relevant critique records during later tasks. This baseline evaluates whether reusable critique memory can support medical reasoning without explicitly distilling experience into structured skills.
- **Voyager.** Voyager [64] is a lifelong agent that accumulates reusable skills from experience. Although it was originally designed for open-ended embodied exploration, it provides a useful reference for direct skill accumulation. In our adaptation, successful medical trajectories are converted into reusable procedures that can be retrieved for similar cases.
- **DILU.** DILU [73] stores and reuses prior decision-making experience to guide later actions. We instantiate it as an experience memory baseline in which previous medical reasoning traces are summarized and retrieved for the current task. This comparison tests whether experience reuse without explicit utility guided skill governance can match SkeMex.
- **ExPeL.** ExPeL [89] extracts natural-language lessons from prior task experiences and uses them to improve future behavior. It represents an experience distillation baseline that stores compact insights rather than raw trajectories. We include it to compare SkeMex with a method that learns reusable lessons but does not perform adoption-aware utility estimation or repository-level governance.
- **GenerativeMemory.** GenerativeMemory [42] stores observations and experiences, then retrieves and synthesizes them to support later behavior. We adapt it as a global experience memory baseline for medical tasks. This baseline tests whether general episodic memory and reflection can provide the same benefit as the procedural skill abstraction used in SkeMex.

- **Memp.** Memp [11] improves task solving by retrieving useful past experience and injecting it into the prompt. We include it as a lightweight memory baseline because it directly tests the value of prompt-level memory augmentation. Compared with Memp, SkeMex treats memory as an evolving repository of reusable skills whose utilities are updated from later outcomes.
- **SkillWeaver.** SkillWeaver [91] discovers and refines reusable skills from agent experience. It is one of the closer baselines to SkeMex because both methods focus on skill-level experience reuse. The comparison examines whether skill discovery alone is sufficient, or whether value-aware retrieval and memory governance provide additional benefits in heterogeneous medical benchmarks.
- **AgentWorkflowMemory.** AgentWorkflowMemory [70] stores successful workflows and reuses them when similar tasks appear. We adapt it by mining reusable medical reasoning and tool-use workflows from the training split. This baseline evaluates the benefit of workflow-level memory, while SkeMex further allows skills to be created, patched, valued, merged, or deprecated over time.
- **AgentKB.** AgentKB [58] builds a knowledge base for agents from previous interactions and task-solving experience. In our experiments, it represents a knowledge-base-oriented memory baseline where stored entries are retrieved to guide later reasoning. We include it to test whether a general agent knowledge base can provide the same benefit as a structured skill repository.
- **Evolver.** Evolver [75] is a self improving agent framework that refines behavior from accumulated experience. We include it as a representative evolution-oriented baseline. Its comparison with SkeMex highlights the difference between general experience refinement and our explicit lifecycle for reading, writing, assessing, and governing skill memory.
- **DynamicCheatsheet.** DynamicCheatsheet [57] maintains an adaptive cheatsheet that is updated across tasks and reused as persistent test-time memory. It represents a compact global memory baseline. SkeMex differs by retrieving task-relevant skills rather than relying on a single evolving cheatsheet.
- **MobileE.** MobileE [68] is a memory based agent method developed for interactive agent settings. We include it to examine whether experience reuse mechanisms designed for interactive environments transfer to medical reasoning benchmarks. In our adaptation, stored experiences are retrieved when they match the current clinical task context.
- **CerebraFusionMemory.** CerebraFusionMemory [86] integrates multiple memory signals to improve agent behavior. We include it as a stronger global memory baseline because it can combine heterogeneous past information. This comparison evaluates whether memory fusion alone can match SkeMex's explicit utility estimation and repository maintenance.
- **GSEM.** GSEM [17] is a graph based self evolving memory method for clinical reasoning. It is the closest baseline in spirit because it also structures medical experience memory. We include it to compare SkeMex with a clinical memory evolution method that organizes experience through graph structure, while SkeMex represents reusable knowledge as procedural skills and updates them through outcome based utility assessment.

These baselines cover several levels of adaptation. The no-memory setting is represented by Vanilla ReAct. Reflective memory methods include Reflexion and CRITIC. Experience distillation methods include Voyager, DILU, ExPeL, GenerativeMemory, and Memp. Skill and workflow memory methods include SkillWeaver, AgentWorkflowMemory, AgentKB, and Evolver. Global or persistent memory methods include DynamicCheatsheet, MobileE, and CerebraFusionMemory. Graph structured clinical memory is represented by GSEM. In addition, Lingshu, Hulu-Med, and MedGemma serve as medical specialist references. SkeMex is distinguished from these baselines by combining a structured skill repository, value-aware retrieval, category-normalized utility estimation, and periodic memory governance. This design allows the agent not only to reuse previous experience, but also to estimate whether each stored skill remains useful for future clinical tasks.

E.2 Metrics

We use dataset-specific metrics that match the answer format and evaluation protocol of each benchmark. For standard close-ended benchmarks, including most multiple-choice datasets, the

primary metric is accuracy. Let $\mathcal{I}$ denote the set of samples that are successfully evaluated, excluding only failures such as missing inputs or invalid benchmark records. The accuracy is computed as

$$\mathrm{Acc} = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} c_i, \tag{11}$$

where $c_i \in \{0, 1\}$ is the per-sample correctness indicator. For multiple-choice tasks, we normalize both model predictions and reference answers into uppercase option letters. Let $Y_i \subseteq \mathcal{O}_i$ be the set of correct options for sample i, where $\mathcal{O}_i$ is the option set, and let $\Phi(\hat{y}_i) \subseteq \mathcal{O}_i$ be the set of options parsed from the model response $\hat{y}_i$. The correctness indicator is

$$c_i = \mathbb{I}\{\Phi(\hat{y}_i) = Y_i\}. \tag{12}$$

For single-answer questions, this reduces to exact option matching. Common answer formats such as A, (A), $[A]$, $A.$, and A : are normalized before comparison. If a response cannot be parsed into a valid option, it is counted as incorrect rather than removed from the denominator. This makes the evaluation deterministic and avoids giving credit to ambiguous answer formats.

For open-ended datasets whose official answers are not simple option letters, such as AgentClinic Text and MedJourney, we use a semantic-equivalence judge. The judge receives the reference answer and the model prediction, then returns a binary verdict indicating whether the prediction is correct or semantically equivalent to the reference. The resulting verdict is used as c_i in the same accuracy computation above. Thus, these datasets are still reported as accuracy, but correctness is determined by semantic equivalence rather than exact string or option matching.

For HealthBench and LiveMedBench, we use rubric-based scoring because their outputs are free-form clinical responses. Each sample i contains a list of rubric criteria

$$\mathcal{C}_i = \{(q_{ij}, w_{ij})\}_{j=1}^{J_i}, \tag{13}$$

where J_i denotes the number of rubric criteria for sample i, q_{ij} is the j-th clinical criterion, and w_{ij} is its associated weight. We use Gemini-3-Flash [13] to evaluate all criteria for a sample in a single call. The judge returns a binary verdict $b_{ij} \in \{0, 1\}$ for each criterion in the original rubric order. Positive criteria indicate required clinical information or behavior, while negative criteria indicate undesirable content such as unsafe advice, unsupported claims, or clinically inappropriate recommendations. A positive criterion contributes its weight when satisfied. A negative criterion has a negative weight and contributes only when the response commits the specified error.

The normalized rubric score for sample i is

$$s_i = \mathrm{clip}_{[0,1]} \left(\frac{\sum_{j=1}^{J_i} b_{ij} w_{ij}}{\sum_{j=1}^{J_i} \max(w_{ij}, 0)} \right), \tag{14}$$

where $\mathrm{clip}_{[0,1]}(\cdot)$ clamps the value to the interval $[0, 1]$. If a sample has no positive-weight criterion, its score is set to 0. The benchmark-level rubric score is then computed as

$$\mathrm{Score} = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} s_i. \tag{15}$$

For HealthBench, this is reported as the mean sample score. For LiveMedBench, the same computation is reported as the mean case score.

The same scoring functions are also reused by the memory-evolution pipeline to convert task outcomes into scalar rewards. Accuracy-based datasets provide binary rewards through c_i, while rubric-based datasets provide continuous rewards through $s_i \in [0, 1]$. This keeps the reported benchmark metrics and the Assess-stage utility updates aligned under the same per-sample evaluation rule.

F Implementation Details

Data split. We evaluate SkeMex under both in-domain and out-of-domain settings. For benchmarks used in the in-domain evaluation, we split the available samples into approximately equal training and testing partitions. The training split is used only for experience accumulation, skill writing, utility

Table 10: Dataset statistics and train/test split used in our experiments.

Dataset	Total	Train	Test	Type
Out-of-domain (OOD)				
AgentClinic Text	214	0	214	OOD
AgentClinic MM	120	0	120	OOD
MediQ	127	0	127	OOD
MMMU-Pro	56	0	56	OOD
MedJourney	264	0	264	OOD
In-domain (ID)				
MMMU	277	138	139	ID
HealthBench	350	175	175	ID
LiveClin Text	205	104	101	ID
LiveClin MM	371	186	185	ID
LiveMedBench	623	313	310	ID
MedXpertQA Text	372	187	185	ID
MedXpertQA MM	299	151	148	ID

valuation, and repository governance. The corresponding test split is held out for evaluation. For out-of-domain benchmarks, no samples are used during skill accumulation, and the full processed subset is reserved for testing. This protocol allows us to evaluate both whether the skill repository improves performance on related held-out cases and whether the evolved skills can transfer to benchmark families that were not used during repository construction. Table 10 summarizes the data split used in our experiments.

Models and execution setup. We use DeepSeek-V3.2 [32] as the main backbone in the primary experiments. The same backbone is used for agent reasoning, trajectory-to-skill distillation, clinical category classification, utility-related judgment, and repository governance. To test cross-model transfer, we additionally evaluate Qwen3.6-Plus [45] with the skill repository accumulated by DeepSeek-V3.2, without re-distilling or rewriting the repository. This setting examines whether the learned skills encode transferable procedural knowledge rather than model-specific response patterns. Semantic indexing of skill items uses text-embedding-3-large [38]. Unless otherwise stated, model calls are executed through API endpoints, while the skill repository, retrieval indices, utility records, and trajectory traces are maintained locally.

Agent execution. Each task is executed as a bounded ReAct-style trajectory. At the beginning of an episode, the agent receives the task input, the available tool descriptions, and the retrieved skill snippets. At each step, the agent either continues reasoning, calls a tool with structured arguments, or emits a final answer. Tool outputs are appended to the trajectory as observations and can be used by later steps as well as by the trajectory-to-skill distillation module. We set the maximum number of agent steps to 7. If the agent does not produce a final answer within this budget, the trajectory is marked as a maximum-step failure and is excluded from skill writing by the buffer filter. For reproducibility and later analysis, each interaction is saved in a structured JSONL record containing the task input, retrieved skills, intermediate actions, tool observations, final answer, and evaluation result.

Skill retrieval. Skill retrieval is performed once at the beginning of each episode. The current task is first assigned to a clinical category, which is used to guide category-aware retrieval. The default retrieval budget is $K = 6$. Candidate skills are scored by combining semantic similarity, utility, and memory strength:

$$\lambda_{\text{sim}} = 0.4, \qquad \lambda_u = 0.4, \qquad \lambda_h = 0.2. \tag{16}$$

We use a minimum similarity threshold of 0.2 and apply a 10% bonus to mature skills. Retrieval is branch-aware, so that general skills, task-level skills, and action-level skills can all contribute to the final retrieved set when available. When the candidate pool is sufficiently large, a lightweight pre-screening step is applied before final ranking. In our configuration, pre-screening is enabled when

the number of candidates exceeds 5, considers up to 5 candidates per branch, and uses a maximum generation budget of 1024 tokens.

Skill writing and utility valuation. The learning buffer operates over fixed windows. In our implementation, the buffer window size is 30 trajectories and the maximum retained buffer capacity is 20 trajectories. The buffer removes trajectories that are unlikely to contain reusable information, including simple successes completed within two steps, repetitive loops, and trajectories terminated by the maximum-step limit. At the end of each window, retained trajectories are processed by the encoder, which may propose one of three outcomes: creating a new skill, patching an existing skill, or making no memory update. Newly created skills are initialized with utility 0.5 and must pass novelty and quality checks before being inserted into the repository.

Utility valuation uses category-normalized rewards and adoption-aware credit assignment. Positive adoption is scaled by $\lambda_+ = 1.0$. Negative adoption uses a base penalty $\lambda_- = 0.10$ and an additional harm scale $\lambda_{\text{harm}} = 0.5$. Category reward baselines are updated with exponential moving average coefficient $\alpha = 0.2$, require at least 3 samples before being treated as reliable, and default to 0.5 when no stable estimate is available. Skill utilities are clipped to $[0, 1]$. The adoption-count-dependent learning rate follows a warmup and decay schedule, with base learning rate 0.05, maximum learning rate 0.20, 5 warmup steps, and 20 decay steps.

Repository governance. Repository governance is triggered every 2 learning windows. It merges highly similar skills, deprecates consistently low-utility skills, promotes stable high-utility skills to mature status, and enforces branch-wise capacity constraints. The merge similarity threshold is 0.8. Skills with utility below 0.3 are eligible for deprecation, while skills with utility at least 0.75 and usage count at least 15 are eligible for mature status. The default capacities are 12 general skills, 8 task-level skills per category, and 5 action-level skills per tool. These limits keep the repository compact and prevent unbounded growth during continual evolution.

Context management. We use the context guard described in Appendix A.4 to prevent long trajectories and verbose tool observations from exceeding the model context budget. The default token budget is $16{,}384$. When the rendered context becomes too long, the system compresses earlier observations while preserving the latest reasoning state, the retrieved skills, and pinned key findings. The trim ratio is 0.8. The system keeps the last 3 steps in full and compresses older tool observations to at most 200 characters. The loop detector uses a repeat threshold of 3, and key findings are pinned after 5 steps when applicable.

Tool execution and external services. The tool suite combines API-based tools and locally served components. General web search is implemented through the Tavily Search API [60]. Medical concept lookup, medication lookup, PubMed-oriented retrieval, reflection, verification, and patient simulation are executed through external APIs or API-compatible endpoints when available. The medical retrieval tool uses a MedRAG-style backend with MedCPT as the retriever, PubMed as the default corpus, top-$k = 3$ evidence retrieval, HNSW indexing, and corpus caching enabled [78]. The multimodal image-analysis tool is served by a locally deployed Hulu-Med-32B endpoint [19], while OCR and chart reading use the Qwen-VL-OCR API [4]. The agent controller communicates with all of these services only through tool calls, so large tool models do not need to run on the same machine as the controller.

Computational resources. When LLMs and tool models are accessed through APIs, SkeMex is lightweight on the controller side. A CPU-only machine with 8 to 16 CPU cores and 32GB memory is sufficient for running the agent loop, maintaining the skill repository, storing trajectory traces, and performing embedding-based retrieval over the skill index. The main memory bottleneck comes from local medical retrieval when corpora, dense retrieval indices, HNSW structures, and corpus caches are loaded. In that setting, we recommend 64 to 128GB system memory depending on the number and size of cached corpora. The medical image-analysis tool is deployed as a separate vLLM service [24] on a single A100 GPU. This deployment keeps the vision-language inference cost isolated from the CPU-side SkeMex controller. The controller only sends image-analysis requests through the tool interface, while repository maintenance, trajectory logging, and embedding-based retrieval remain lightweight CPU-side operations.

Table 11: Core hyperparameters of SkeMex used in all main experiments.

Module	Hyperparameter	Value
Retrieval		
	Number of retrieved skills K	6
	Similarity weight λ_{sim}	0.4
	Utility weight λ_u	0.4
	Memory-strength weight λ_h	0.2
Memory update		
	Learning window size L	30
Utility estimation		
	Positive advantage scale λ_+	1.0
	Negative base penalty λ_-	0.10
	Negative harm scale λ_{harm}	0.5
	Category EMA coefficient α	0.2
Governance		
	Governance period	Every 2 windows
	Merge similarity threshold	0.8
	Mature utility threshold	0.75

G Sensitivity Analysis

Overall setup. We conduct sensitivity analysis on two rubric based benchmarks, HealthBench and LiveMedBench, to examine whether SkeMex depends strongly on a narrow set of hyperparameter choices. Unless otherwise specified, we use the default configuration with retrieval budget $K = 6$, retrieval weights $(\lambda_{\text{sim}}, \lambda_u, \lambda_h) = (0.4, 0.4, 0.2)$, learning window size $L = 30$, category baseline update coefficient $\alpha = 0.20$, and maximum utility update step $\eta_{\max} = 0.20$. This default setting obtains 27.65 on HealthBench and 57.95 on LiveMedBench, with an average score of 42.80. In each analysis, we vary one hyperparameter while keeping all others fixed, so that the effect of each design choice can be examined in isolation.

Retrieval budget. We first vary the number of retrieved skills K, as shown in Figure 5. The average score increases from 42.52 at $K = 3$ to 42.80 at the default value $K = 6$, and reaches 42.93 at $K = 9$. When K is further increased to 12, the average score slightly decreases to 42.70. This trend suggests that retrieving too few skills may provide insufficient procedural guidance, while retrieving too many skills may introduce redundant or weakly relevant information into the prompt. The overall variation is small across all tested values, which indicates that SkeMex is not highly sensitive to the exact retrieval budget as long as a moderate number of skills is available.

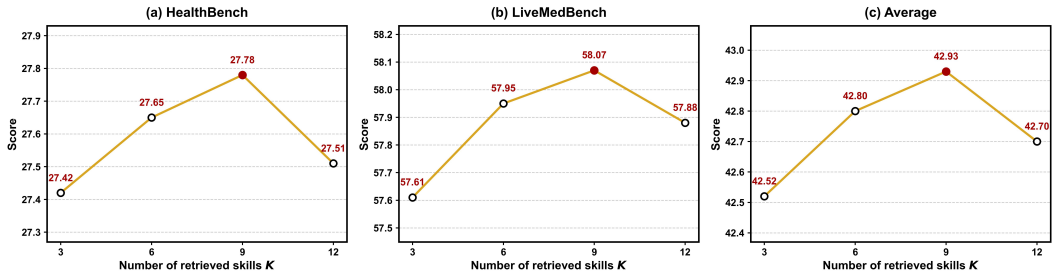

Figure 5: Sensitivity analysis with respect to the number of retrieved skills K.

Retrieval weights. Figure 6 evaluates how the relative weights of semantic similarity, empirical utility, and memory strength affect retrieval quality. The default setting achieves an average score of 42.80. Increasing the semantic similarity weight gives a slightly higher average score of 42.85, mainly

due to the improvement on LiveMedBench. Increasing the utility weight produces a comparable average score of 42.79 and yields the best HealthBench score among the tested settings. By contrast, the memory heavy setting decreases the average score to 42.57. These results suggest that semantic relevance and estimated utility are both important for selecting useful skills, while placing too much emphasis on memory strength alone can reduce retrieval precision by favoring recently reinforced skills that may not be the best match for the current case.

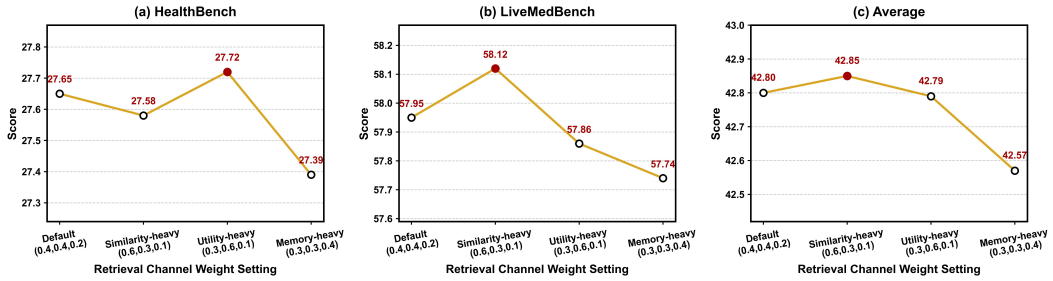

Figure 6: Sensitivity analysis with respect to retrieval channel weights $(\lambda_{\text{sim}}, \lambda_{\text{u}}, \lambda_{\text{h}})$.

Learning window size. The effect of the learning window size L is reported in Figure 7. The average scores are 42.78, 42.82, 42.80, 42.78, and 42.64 for $L \in \{10, 20, 30, 40, 60\}$, respectively. The results do not show a monotonic pattern, suggesting that the optimal window size is closely tied to both the amount and the distribution of training data. Smaller windows allow the repository to react more quickly to recent feedback, which can be useful when new trajectories are diverse and informative. However, when the window contains only a small number of samples from each clinical category, utility estimates may become noisy and sensitive to local fluctuations. Larger windows aggregate more trajectories before updating the repository, which can improve stability when the training stream is sufficiently large and balanced, but may slow adaptation when the data distribution shifts or when rare categories are underrepresented. The default value $L = 30$ lies in a stable region and performs nearly the same as the best tested value, indicating a reasonable balance between update stability and responsiveness under our data scale and category distribution.

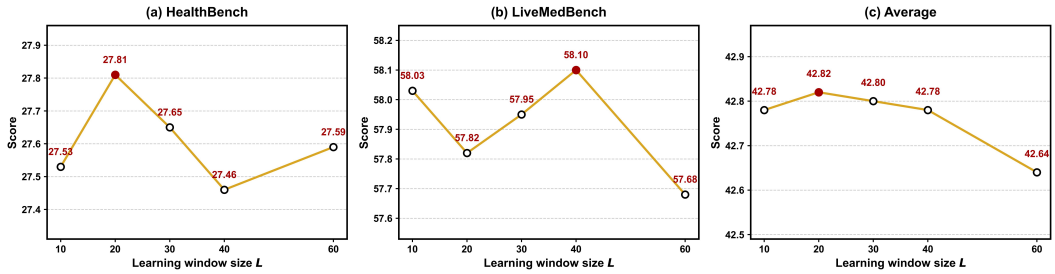

Figure 7: Sensitivity analysis with respect to the learning window size L on rubric-based benchmarks.

Category baseline update coefficient. Figure 8 studies the coefficient α used to update category reward baselines. The best average score is obtained at $\alpha = 0.10$, with an average of 42.86, while the default value $\alpha = 0.20$ achieves 42.80. Larger values such as $\alpha = 0.40$ and $\alpha = 0.80$ lead to slightly lower average scores, although the degradation remains limited. Since a larger α makes the category baseline more responsive to the current window, overly large values may cause the baseline to track short term reward fluctuations too closely. A moderate value therefore provides a better balance between stability and responsiveness.

Utility update step. Figure 9 examines the maximum utility update step $\eta_{\max}$. The average score changes only mildly across different values and reaches 42.83 at $\eta_{\max} = 0.30$, which is close to the default score of 42.80. When the update step is increased to 0.40, the average score drops to 42.54. This suggests that excessively large utility updates can amplify short term feedback noise and make skill valuation less stable. The default setting remains within a robust operating range, avoiding both overly conservative updates and overly reactive utility shifts.

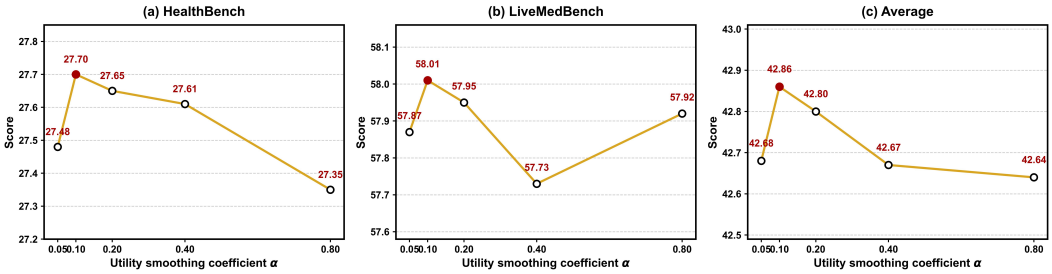

Figure 8: Sensitivity analysis with respect to the utility smoothing coefficient α on rubric-based benchmarks.

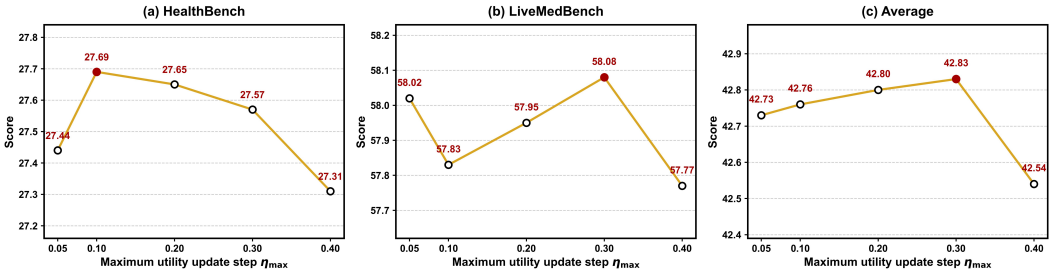

Figure 9: Sensitivity analysis with respect to the maximum utility update step $\eta_{\max}$ on rubric-based benchmarks.

Finally, the sensitivity results show that SkeMex remains stable across a broad range of hyperparameter choices. The largest variations appear when the retrieved skill set is too small, when memory strength is overemphasized in retrieval, or when utility updates become too aggressive. Even in these cases, the performance differences remain modest. These findings suggest that the default configuration provides a balanced trade off among retrieval coverage, retrieval precision, adaptation speed, and utility estimation stability.

H Ablation Study

Table 12: Ablation study on value-aware skill retrieval across different configurations.

Setting	HealthBench	AgentClinic_T	LiveMedBench	LiveClin_M	MedXpertQA_M	Avg.
w/o Prescreen	24.62	67.76	56.23	58.92	45.95	50.69
w/o Memory Channel	26.20	67.29	54.61	58.38	47.30	50.76
Similarity Only	19.26	62.62	53.90	55.14	47.30	47.64
Utility Only	24.65	64.02	51.74	55.14	45.95	48.30
LLM Ranking Only	25.46	65.89	54.78	60.54	47.97	50.93
Full (SkeMex)	**27.65**	**68.22**	**57.95**	**61.62**	**50.68**	**53.22**

Value-aware skill retrieval. Table 12 studies how different retrieval signals contribute to selecting useful skills from the repository. The full model achieves the best average score of 53.22%, showing that effective skill retrieval requires more than surface-level semantic matching. Removing the memory-strength channel lowers the average to 50.76%, while using LLM ranking alone reaches 50.93%. These drops indicate that neither a recency-aware memory signal nor structured retrieval scores can be fully replaced by direct LLM reranking. Although LLM ranking can identify seemingly relevant skills, it does not explicitly account for whether a skill has been recently reinforced or has shown stable utility across prior trajectories.

The decline becomes larger when retrieval relies on a single signal. Similarity Only obtains 47.64%, which is 5.58 points below the full model, and Utility Only obtains 48.30%, which is 4.92 points

lower. This pattern suggests that the two signals capture different aspects of skill usefulness. Semantic similarity can retrieve skills that are topically close to the current case, but such skills may still be ineffective if they have not led to successful outcomes in similar settings. Utility alone can favor historically successful skills, but may select overly general procedures that do not match the current clinical context. The full retrieval score avoids these failure modes by jointly considering semantic relevance, empirical effectiveness, and memory strength.

Prescreening also contributes to retrieval quality. Removing it reduces the average score to 50.69%, suggesting that early candidate filtering helps remove irrelevant or weakly matched skills before final ranking. This is particularly important in a growing skill repository, where noisy candidates can crowd out more useful skills if all entries are passed directly to the final selection stage. Overall, the results show that value-aware retrieval benefits from a balanced combination of relevance, utility, and memory-strength signals. This combination allows SkeMex to retrieve skills that are not only related to the case, but also empirically reliable and recently validated.

Table 13: Ablation study on closed-loop self-evolution memory lifecycle.

Setting	**HealthBench**	**AgentClinic_T**	**LiveMedBench**	**LiveClin_M**	**MedXpertQA_M**	**Avg.**
w/o Deprecation	23.43	67.76	56.08	55.14	43.92	49.27
w/o Memory Merging	23.78	64.02	54.08	57.84	48.65	49.67
w/o Capacity Control	25.77	64.95	56.56	60.00	47.97	51.05
w/o Maturation	23.17	66.36	51.18	56.76	40.54	47.60
Full (SkeMex)	**27.65**	**68.22**	**57.95**	**61.62**	**50.68**	**53.22**

Closed-loop memory lifecycle. Table 13 evaluates the contribution of repository governance mechanisms in the closed-loop memory lifecycle. The full SkeMex obtains the highest average score of 53.22%, which shows that memory evolution requires not only writing new skills but also maintaining the repository after skills have been created. Among the ablations, removing maturation causes the largest drop, reducing the average score to 47.60%. This 5.62-point decline suggests that the system benefits from distinguishing repeatedly validated skills from newly created or less stable drafts. Without maturation, retrieval may overuse skills that have not yet accumulated enough positive evidence, which weakens long-term reliability.

A substantial degradation is also observed when deprecation is removed. The average score falls to 49.27%, indicating that obsolete, harmful, or consistently low-utility skills can introduce noise into later retrieval. This result highlights the importance of allowing the repository to forget or down-weight entries that no longer provide useful guidance. Disabling memory merging produces a similar decline, with the average score decreasing to 49.67%. This shows that redundancy is not only a storage issue, but also a retrieval issue. When highly similar skills remain separate, they can crowd the candidate pool and make it harder for the agent to retrieve diverse and complementary guidance.

Capacity control has a smaller but still meaningful effect. Without capacity limits, the average score drops to 51.05%, suggesting that unconstrained repository growth gradually weakens retrieval quality even when other lifecycle operations remain active. As more skills accumulate, the repository can become harder to search and more vulnerable to low-value candidates unless branch-wise capacity is regulated. Taken together, these results show that a reliable skill memory depends on the full lifecycle of creation, validation, consolidation, pruning, and capacity control. In SkeMex, governance is therefore not a peripheral cleanup step, but a core component that keeps the evolving repository compact, selective, and clinically dependable.

I Further Analyses

I.1 Offline OOD Generalization

Table 14: Main results on out-of-domain benchmarks in the offline setting. All datasets are excluded from training and used solely for evaluation. The last column reports the improvement of memory-based methods over the memory-free ReAct baseline, highlighting the benefits of memory. **Bold** numbers indicate the best performance.

Backbone	Method	Text			Multimodal		Avg.
		MedJourney	MediQ	AgentClinic_T	MMMU-Pro	AgentClinic_M	
Hulu-Med-32B	CoT	70.08	89.76	22.90	46.43	84.17	62.67
Lingshu-32B	CoT	67.42	77.95	13.55	32.14	73.33	52.88
MedGemma-27B	CoT	65.91	90.55	26.17	39.29	83.33	61.05
DeepSeek-V3.2	CoT	65.15	90.55	26.64	32.14	80.83	59.06
	ReAct	65.53	91.34	34.11	35.71	83.33	62.01
	Reflexion	68.94	90.55	60.28	32.14	84.17	67.22 +5.21
	CRITIC	70.08	90.55	63.55	35.71	80.00	67.98 +5.97
	Voyager	73.11	90.55	59.81	32.14	79.17	66.96 +4.95
	DILU	73.86	90.55	64.95	28.57	79.17	67.42 +5.41
	ExPeL	67.42	91.34	57.94	33.93	82.50	66.63 +4.62
	GM	72.73	91.34	66.36	37.50	83.33	70.25 +8.24
	Memp	74.24	88.98	62.15	35.71	84.17	69.05 +7.04
	SkillWeaver	71.97	89.76	65.42	33.93	84.17	69.05 +7.04
	AWM	73.86	91.34	63.55	37.50	79.17	69.08 +7.07
	Agent KB	73.86	89.76	65.42	32.14	84.17	69.07 +7.06
	Evolver	**78.79**	91.34	65.89	35.71	80.00	70.35 +8.34
	DC	73.86	88.98	64.49	37.50	83.33	69.63 +7.62
	MobileE	74.24	88.19	65.42	35.71	83.33	69.38 +7.37
	CFM	74.24	88.19	64.02	37.50	84.17	69.62 +7.61
	GSEM	73.48	93.70	21.03	42.86	88.33	63.88 +1.87
	SkeMex	76.52	**96.85**	**68.22**	**48.21**	**89.17**	**75.79 +13.78**
Qwen3.6-Plus	ReAct	68.56	92.13	34.58	41.07	85.00	64.27
	Reflexion	79.17	96.06	62.15	44.64	90.00	74.40 +10.13
	CRITIC	79.92	96.06	61.21	42.86	86.67	73.35 +9.08
	Voyager	79.92	95.28	64.02	44.64	90.00	74.77 +10.50
	DILU	78.79	95.28	64.02	46.43	89.17	74.74 +10.47
	ExPeL	79.17	95.28	64.49	44.64	89.17	74.55 +10.28
	GM	78.79	93.70	63.08	46.43	89.17	74.23 +9.96
	Memp	78.41	96.85	63.08	41.07	90.83	74.05 +9.78
	SkillWeaver	79.92	96.85	63.08	41.07	88.33	73.85 +9.58
	AWM	79.92	96.85	64.02	44.64	88.33	74.75 +10.48
	Agent KB	79.55	96.85	64.02	44.64	89.17	74.84 +10.57
	Evolver	79.17	96.06	63.55	46.43	90.83	75.21 +10.94
	DC	78.79	96.06	55.14	44.64	90.83	73.09 +8.82
	MobileE	78.41	96.85	64.02	46.43	89.17	74.97 +10.70
	CFM	79.92	96.85	60.75	41.07	90.00	73.72 +9.45
	GSEM	80.30	96.85	29.91	39.29	89.17	67.10 +2.83
	SkeMex	**81.44**	**97.64**	**65.89**	**51.79**	**94.17**	**78.18 +13.91**

Table 14 reports the complete offline out-of-domain results for both backbones, complementing Figure 3 where only the DeepSeek-V3.2 results are visualized due to space constraints. In this setting, all evaluation datasets are excluded from skill-repo construction, and the frozen repo is directly transferred to unseen benchmark families. SkeMex achieves the best average performance on both backbones. With DeepSeek-V3.2, SkeMex improves ReAct from 62.01% to 75.79%, yielding a

+13.78 point gain and outperforming the strongest competing memory baseline by 5.44 points. With Qwen3.6-Plus, SkeMex improves ReAct from 64.27% to 78.18%, yielding a **+13.91** point gain and a 2.97-point lead over the strongest competing memory method. These results show that the offline skill repo transfers consistently across model families, rather than benefiting only a single backbone.

The dataset-level results further support this conclusion. Under DeepSeek-V3.2, SkeMex obtains the best score on MediQ, AgentClinic-Text, MMMU-Pro, and AgentClinic-MM, with especially large gains over ReAct on AgentClinic-Text and MMMU-Pro by +34.11 and +12.50 points, respectively. Under Qwen3.6-Plus, SkeMex achieves the best score on all five out-of-domain benchmarks, including MedJourney, MediQ, AgentClinic-Text, MMMU-Pro, and AgentClinic-MM. The largest improvements over ReAct again appear on AgentClinic-Text, MMMU-Pro, and AgentClinic-MM, with gains of +31.31, +10.72, and +9.17 points, respectively. In contrast, several memory baselines exhibit less stable transfer. For example, GSEM obtains only +1.87 and +2.83 average gains under DeepSeek-V3.2 and Qwen3.6-Plus, and falls below ReAct on AgentClinic-Text in both cases. Overall, the cross-backbone results indicate that structured skill abstraction and utility-guided retrieval help reduce negative transfer and preserve robust generalization to unseen medical benchmark families.

I.2 Cross Backbone Generalization

We further examine whether SkeMex remains effective when the underlying backbone model changes. This analysis tests whether the learned skill repository captures reusable medical experience across model families, rather than exploiting response patterns specific to a single backbone. We evaluate three additional backbones, Qwen3.6-Max-Preview [44], Kimi-2.6 [35], and GLM-5.1 [85], on four representative benchmarks: HealthBench, MMMU-Pro, AgentClinic, and MedXpertQA. For each backbone, we compare SkeMex with ReAct and three memory based baselines under the same evaluation protocol. The three memory based baselines are selected as the strongest self-evolving memory methods from their respective categories in our main experiments.

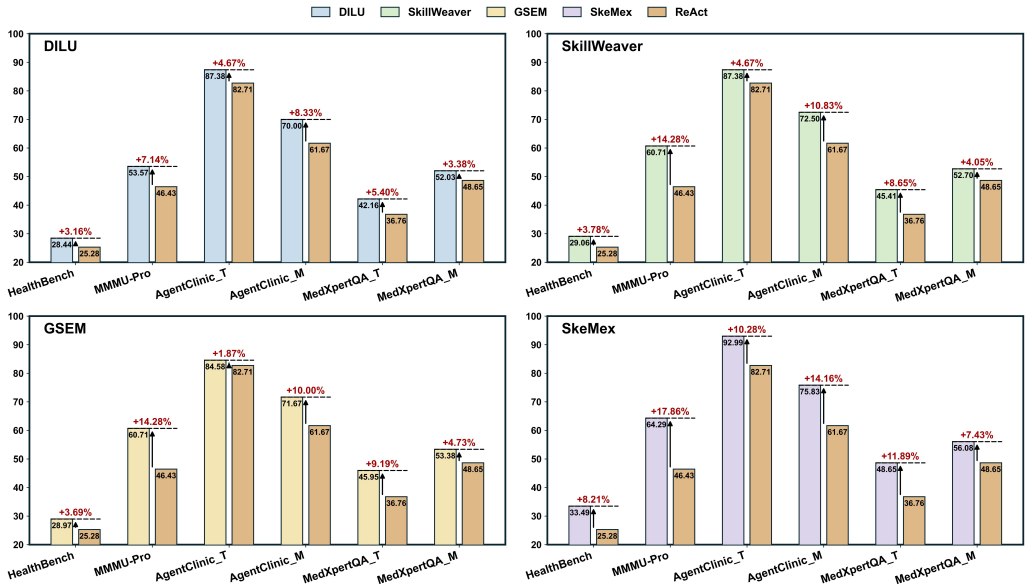

Figure 10: Cross backbone generalization results on Qwen3.6-Max-Preview.

Figures 10, 11, and 12 show that the advantage of SkeMex is consistent across all three additional backbones. On Qwen3.6-Max-Preview, SkeMex achieves the best result on every benchmark. It improves over ReAct by 8.21 points on HealthBench, 17.86 points on MMMU-Pro, 10.28 points on AgentClinic-Text, 14.16 points on AgentClinic-MM, 11.89 points on MedXpertQA-Text, and 7.43 points on MedXpertQA-MM. The average score increases from 50.25% for ReAct to 61.89% for SkeMex, giving an average gain of **11.64** points. SkeMex also outperforms the strongest competing memory baseline by 3.93 points on average, which suggests that its gain cannot be attributed to memory augmentation alone.

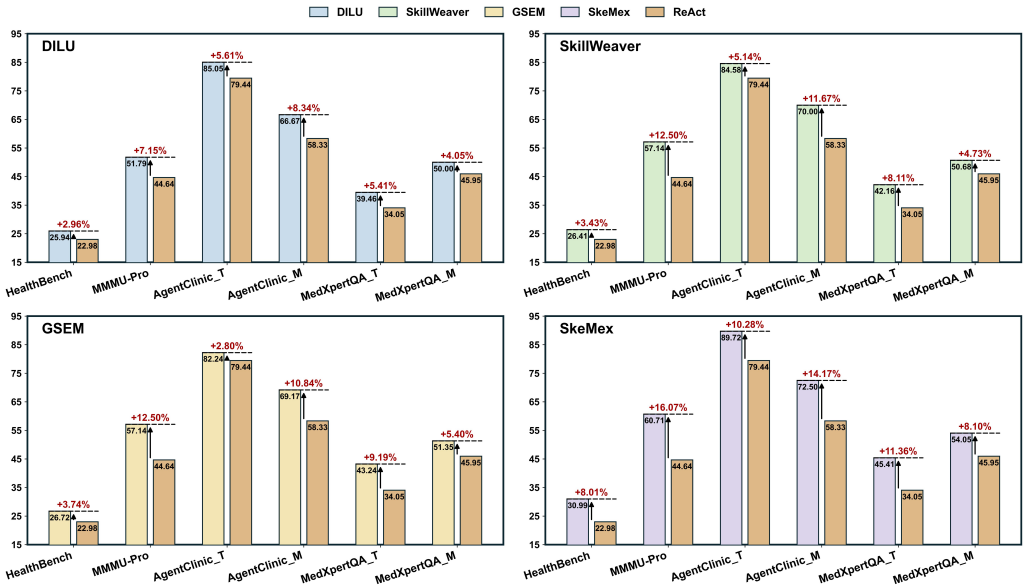

Figure 11: Cross backbone generalization results on Kimi-2.6.

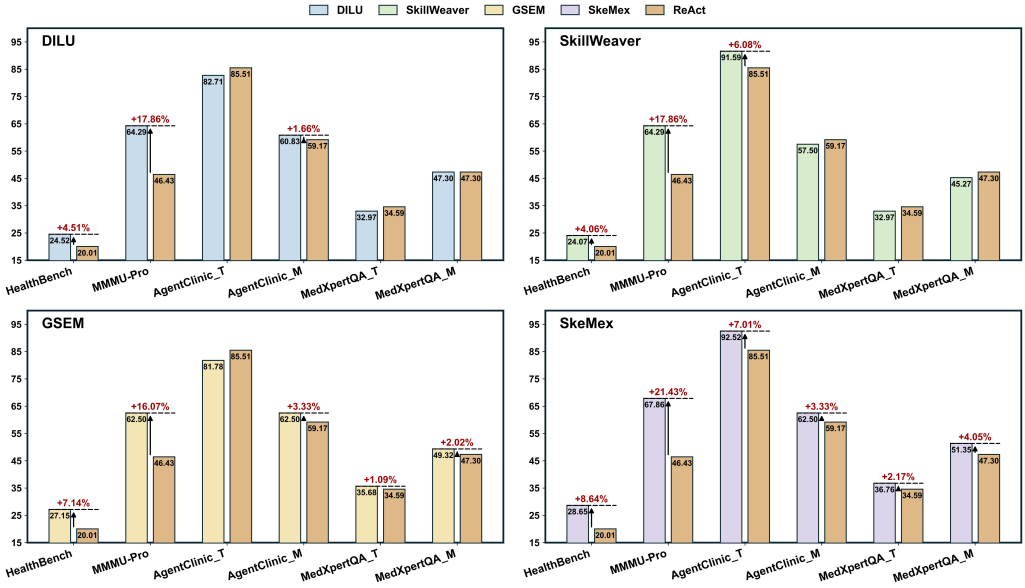

Figure 12: Cross backbone generalization results on GLM-5.1.

The results on Kimi-2.6 follow a similar pattern. SkeMex again obtains the best score on all six benchmarks, improving over ReAct by 8.01 points on HealthBench, 16.07 points on MMMU-Pro, 10.28 points on AgentClinic-Text, 14.17 points on AgentClinic-MM, 11.36 points on MedXpertQA-Text, and 8.10 points on MedXpertQA-MM. Its average score reaches 58.90%, compared with 47.56% for ReAct, which corresponds to an average gain of **11.33** points. The close agreement between Qwen3.6-Max-Preview and Kimi-2.6 indicates that SkeMex provides stable improvements across both general clinical reasoning benchmarks and multimodal clinical evaluation benchmarks.

GLM-5.1 provides a more challenging setting because competing memory baselines are less stable across datasets. DILU falls below ReAct on AgentClinic-Text and MedXpertQA-Text. SkillWeaver falls below ReAct on AgentClinic-MM, MedXpertQA-Text, and MedXpertQA-MM. GSEM falls below ReAct on AgentClinic-Text. In contrast, SkeMex produces nonnegative gains on all six datasets.

It improves over ReAct by 8.64 points on HealthBench, 21.43 points on MMMU-Pro, 7.01 points on AgentClinic-Text, 3.33 points on AgentClinic-MM, 2.17 points on MedXpertQA-Text, and 4.05 points on MedXpertQA-MM. SkeMex achieves the best result on five benchmarks and ties for the best result on AgentClinic-MM. Its average score improves from 48.84% for ReAct to 56.61%, giving an average gain of **7.77** points.

Across the eighteen backbone and dataset pairs, SkeMex achieves the best result in seventeen cases and ties for the best result in the remaining case. When all three backbones are pooled together, SkeMex improves the average score from 48.88% to 59.13%, yielding a **10.25** point gain over ReAct. The averaged dataset-wise gains are also consistently positive, with 8.29 points on HealthBench, 18.45 points on MMMU-Pro, 9.19 points on AgentClinic-Text, 10.55 points on AgentClinic-MM, 8.47 points on MedXpertQA-Text, and 6.53 points on MedXpertQA-MM. These results indicate that SkeMex does not depend on a particular backbone model. Instead, the structured skill abstraction and utility-guided retrieval provide transferable experience that can be reused by diverse medical agents while preserving stable empirical gains.

I.3 Cross Backbone Skill Transfer

We further examine whether the skills learned by SkeMex are specific to the backbone that produced them. In this analysis, we use the skill repository constructed from the DeepSeek-V3.2 training split in the main experiments and keep it fixed during evaluation. We then replace the test-time backbone with Claude Sonnet-4.6 [2] and Qwen3.6-35B-A3B [43]. We choose these two models to make the transfer setting more representative and challenging. Claude Sonnet-4.6 is a strong closed-source frontier model, while Qwen3.6-35B-A3B is a strong open-source mixture-of-experts backbone. Evaluating both allows us to test whether the same skill repository can transfer across different model families, deployment regimes, and architectural designs. This setting is stricter than building a separate repository for each target model, because the target backbone must consume and apply skills generated by another model family. It therefore directly tests whether SkeMex learns transferable medical problem solving procedures rather than backbone specific response traces.

Table 15: Generalization of skills learned by DeepSeek-V3.2 to other backbone models. LC, MXQA, HB, and LMB denote LiveClin, MedXpertQA, HealthBench, and LiveMedBench, respectively. **Bold** numbers indicate the best result for each target backbone and dataset.

Backbone	Method	Text				Multimodal			Avg.
		LC	MXQA	HB	LMB	LC	MXQA	MMMU	
Claude Sonnet-4.6	ReAct	81.99	36.29	24.63	47.88	61.84	48.15	46.32	49.59
	GM	86.99	43.41	29.41	49.69	72.16	55.58	58.60	56.55
	SkillWeaver	86.99	44.39	28.46	51.15	72.65	52.25	60.24	56.59
	CFM	90.99	44.93	29.28	51.02	70.98	52.30	59.22	56.96
	SkeMex	**92.04**	**47.49**	**32.79**	**54.92**	**75.94**	**55.83**	**62.87**	**60.27**
Qwen3.6-35B-A3B	ReAct	81.37	33.64	23.38	45.88	59.34	46.20	44.17	47.71
	GM	85.09	40.76	28.01	47.59	69.46	**54.03**	56.30	54.46
	SkillWeaver	85.09	41.99	26.71	49.05	70.20	50.55	57.74	54.48
	CFM	**91.17**	42.78	27.58	49.12	68.73	50.65	57.07	55.30
	SkeMex	90.04	**45.49**	**30.99**	**53.12**	**73.34**	53.78	**60.87**	**58.23**

Table 15 reports the full dataset-level results. When the fixed DeepSeek-V3.2 skill repository is reused by Claude Sonnet-4.6, SkeMex achieves the best score on all seven benchmarks. The average score improves from 49.59% with ReAct to 60.27%, giving a **10.68** point gain. SkeMex also exceeds CFM, the strongest competing memory baseline, by 3.31 points on average. The gains over ReAct are broadly distributed across benchmarks, including 10.05 points on LiveClin-Text, 11.20 points on MedXpertQA-Text, 8.16 points on HealthBench, 7.04 points on LiveMedBench, 14.10 points on LiveClin-MM, 7.68 points on MedXpertQA-MM, and 16.55 points on MMMU. These improvements cover text-only clinical reasoning, rubric-based evaluation, and multimodal medical tasks, indicating that Claude Sonnet-4.6 can interpret and apply skills produced by a different backbone.

The results on Qwen3.6-35B-A3B show a similar transfer effect under a different model architecture. SkeMex improves the average score from 47.71% with ReAct to 58.23%, corresponding to a **10.52** point gain. It also outperforms CFM, the strongest competing baseline on average, by 2.93 points. At the dataset level, SkeMex ranks first on MedXpertQA-Text, HealthBench, LiveMedBench, LiveClin-MM, and MMMU. It is slightly below CFM on LiveClin-Text by 1.13 points and slightly below GM on MedXpertQA-MM by 0.25 points, while still achieving the strongest overall average. More importantly, its gains over ReAct remain positive on every benchmark: 8.67 points on LiveClin-Text, 11.85 points on MedXpertQA-Text, 7.61 points on HealthBench, 7.24 points on LiveMedBench, 14.00 points on LiveClin-MM, 7.58 points on MedXpertQA-MM, and 16.70 points on MMMU. The large gains on LiveClin-MM and MMMU suggest that transferred skills are particularly helpful when the target model needs to integrate heterogeneous clinical evidence or perform multimodal reasoning.

These results provide direct evidence that SkeMex skills can transfer across backbones. The skill repository is created by DeepSeek-V3.2, but its benefits persist when consumed by both a closed-source frontier model and an open-source mixture-of-experts model. This makes it unlikely that the repository only stores cached answers or backbone-specific response traces. Instead, the learned skills appear to capture reusable clinical procedures, such as decomposing patient information, identifying relevant evidence, selecting appropriate tools, and applying domain-specific reasoning heuristics. This interpretation is supported by the consistent gains across text-only, rubric-based, and multimodal benchmarks. This analysis complements the cross-backbone generalization study in Appendix I.2. That study evaluates SkeMex after changing the backbone under each target setting, while this study fixes the skill source to DeepSeek-V3.2 and changes only the model that consumes the skills. The strong results in both settings suggest that SkeMex separates accumulated medical experience from the particular model that produced it. This portability is important for medical agent systems, because experience accumulated by one capable backbone can be reused to improve other backbones without rebuilding the entire skill repository from scratch.

I.4 Execution Cost and Interaction Depth

We analyze the execution behavior of different methods from two perspectives: the average number of interaction steps and the wall clock time required to complete each task. Table 16 and Table 17 report the detailed results for all methods evaluated with the DeepSeek-V3.2 backbone. The goal of this analysis is to understand how skill memory changes the agent's problem solving process, especially whether the performance gain comes with additional reasoning depth or runtime overhead. Since the measured time is affected by API latency, external service availability, and tool response speed, these numbers should be interpreted as approximate wall clock runtimes rather than exact computational complexity.

Interaction depth. A clear pattern in Table 16 is that SkeMex uses more interaction steps than the other methods. Its average number of steps is 4.77, compared with 3.17 for the memory free ReAct baseline and 3.97 to 4.61 for other memory based methods. This higher interaction depth reflects a more deliberate solving process. Retrieved skills often encourage the agent to decompose the task, verify intermediate evidence, use tools when necessary, and avoid premature final answers. This behavior is especially useful for difficult medical cases, where the correct answer often depends on combining multiple pieces of clinical evidence rather than reacting to the most salient clue. The effect is particularly visible on challenging benchmarks. On MMMU-Pro, ReAct terminates after only 1.482 steps on average, which suggests that it often answers quickly without sufficient exploration. SkeMex increases the average number of steps to 4.625 and improves performance from 35.71% to 48.21%, as shown in Table 14. A similar pattern appears on LiveMedBench, where SkeMex takes 6.139 steps compared with 3.578 steps for ReAct. These examples suggest that the additional steps are not merely redundant actions. Instead, they correspond to more structured clinical reasoning, evidence gathering, and answer verification guided by the retrieved skills.

Runtime overhead. The deeper interaction process also increases wall clock time. As shown in Table 17, SkeMex takes 116.06 seconds per task on average, which is higher than ReAct at 54.48 seconds and Evolver at 81.65 seconds. To better understand this overhead, we also compare the average time per step. SkeMex takes approximately 24.33 seconds per step, while ReAct takes 17.19 seconds and Evolver takes 20.41 seconds. The larger per step cost mainly comes from two sources. First, retrieved skills are rendered into the agent context, which increases the input length of model

Table 16: Average number of interaction steps across datasets. AC, LC, MXQA, HB, LMB, MJ, MQ, and MP denote AgentClinic, LiveClin, MedXpertQA, HealthBench, LiveMedBench, MedJourney, MediQ, and MMMU-Pro, respectively. Suffixes "_T" and "_M" indicate text-only and multimodal settings. Methods are grouped following the same categories as in the main offline comparison table.

Method	Out-of-domain					In-domain							Avg.
	AC_M	AC_T	MJ	MQ	MP	LC_M	LC_T	LMB	MMMU	MXQA_M	MXQA_T	HB	
ReAct	3.650	4.397	2.789	2.591	1.482	3.785	3.635	3.578	1.897	3.385	3.692	3.120	3.17
Reflexion	4.017	5.780	3.042	2.756	5.500	5.000	4.129	5.597	4.698	4.745	4.573	3.411	4.44
CRITIC	4.242	5.590	3.625	2.622	4.179	5.232	4.139	6.265	3.353	5.236	4.681	3.594	4.40
Voyager	4.483	5.795	3.205	2.756	3.946	5.124	4.238	5.922	3.309	3.939	4.443	3.469	4.22
DILU	3.267	5.005	2.576	2.898	5.429	4.022	3.792	3.116	4.698	4.155	5.086	3.629	3.97
ExPeL	3.567	5.888	3.303	2.906	4.661	5.032	4.277	5.529	3.576	5.054	5.049	3.114	4.33
GM	3.733	5.505	3.330	2.937	4.464	3.692	3.832	6.052	3.324	3.892	4.676	3.257	4.06
Memp	4.725	5.808	3.436	2.701	5.571	5.038	4.109	6.039	4.827	4.912	4.649	3.514	4.61
SkillWeaver	3.758	5.603	3.144	2.622	4.143	4.984	4.139	6.000	3.439	4.980	4.319	3.377	4.21
AWM	4.392	5.626	3.239	2.937	5.482	5.227	4.178	5.852	4.813	4.169	4.530	3.291	4.48
Agent KB	4.792	5.603	3.687	2.780	5.321	4.751	3.812	5.672	4.741	3.327	4.054	3.806	4.36
Evolver	3.475	5.626	3.379	2.772	4.125	3.984	4.020	5.784	3.137	3.939	4.578	3.189	4.00
DC	4.600	5.659	3.701	2.976	5.339	3.896	4.307	6.090	4.763	3.784	4.649	3.337	4.43
MobileE	3.592	5.210	3.193	2.795	5.625	3.816	4.149	5.364	4.748	4.027	4.584	3.366	4.21
CFM	3.533	5.491	1.148	2.567	4.054	4.076	4.356	3.326	3.583	3.899	4.476	3.269	3.65
GSEM	4.817	5.332	3.220	3.031	5.196	3.973	4.337	5.797	4.468	3.838	4.768	3.754	4.38
SkeMex	4.167	6.145	4.208	3.094	4.625	5.184	5.396	6.139	4.223	4.973	5.784	3.337	4.77

calls. Second, skill retrieval requires embedding based matching and multi channel scoring at the beginning of each episode. These operations are lightweight compared with model inference, but they still add overhead to the full trajectory.

Accuracy and efficiency. Although SkeMex requires more steps and longer runtime, the additional cost is accompanied by consistent accuracy gains across both in domain and out of domain benchmarks. In clinical reasoning settings, this trade off is meaningful because reliability is often more important than raw response speed. The results suggest that SkeMex spends additional computation on useful intermediate reasoning rather than unproductive loops. The context guard and buffer filter also reduce the chance that repeated or failed trajectories are reinforced into memory.

The overhead is not uniform across datasets. On HealthBench, SkeMex reduces the average time to 52.36 seconds, which is lower than ReAct at 64.19 seconds and also below the overall method average of 65.05 seconds, while maintaining a comparable number of steps at 3.337. This indicates that skill memory can sometimes streamline execution when the retrieved skills are highly relevant to the task. In such cases, the agent may avoid slow unguided exploration and move more directly toward the required clinical criteria. Overall, the execution analysis shows that SkeMex generally trades some runtime efficiency for a more structured and empirically grounded reasoning process, while in certain settings the retrieved skills can also improve efficiency by making the trajectory more focused.

Table 17: Average time consumption in seconds across datasets. Due to API latency and occasional service instability, the measured time may contain small fluctuations and should be interpreted as approximate wall-clock runtime. AC, LC, MXQA, HB, LMB, MJ, MQ, and MP denote AgentClinic, LiveClin, MedXpertQA, HealthBench, LiveMedBench, MedJourney, MediQ, and MMMU-Pro, respectively. Suffixes "_T" and "_M" indicate text-only and multimodal settings. Methods are grouped following the same categories as in the main offline comparison table.

Method	**Out-of-domain**					**In-domain**							**Avg.**
	AC_M	**AC_T**	**MJ**	**MQ**	**MP**	**LC_M**	**LC_T**	**LMB**	**MMMU**	**MXQA_M**	**MXQA_T**	**HB**	
ReAct	68.85	109.75	64.26	50.52	46.32	18.04	18.02	18.03	18.02	84.31	93.51	64.19	54.48
Reflexion	101.20	133.15	51.74	41.65	78.97	89.76	95.97	112.10	61.49	80.04	81.84	54.71	81.88
CRITIC	64.35	124.50	74.74	38.41	77.92	85.28	76.01	153.10	62.54	128.38	93.86	72.36	87.62
Voyager	92.02	115.46	50.58	53.70	95.89	122.13	116.01	124.17	72.13	110.28	107.90	84.22	95.37
DILU	45.41	81.91	41.41	55.87	103.55	78.32	67.89	60.20	79.15	108.08	121.47	76.83	76.67
ExPeL	51.39	85.36	35.59	47.09	90.19	101.96	76.39	73.99	60.05	77.16	100.25	44.35	70.32
GM	55.10	119.69	71.20	66.08	81.80	70.61	67.31	142.81	50.09	91.54	97.10	54.09	80.62
Memp	57.56	91.65	59.54	44.01	202.33	104.59	77.78	148.91	119.40	95.70	113.38	78.52	99.45
SkillWeaver	70.70	109.13	55.66	49.26	105.43	117.61	110.68	150.23	73.23	101.86	107.63	77.99	94.12
AWM	106.60	133.54	62.46	54.00	74.46	86.69	86.43	143.74	65.06	94.20	80.78	53.88	86.82
Agent KB	74.51	94.03	61.86	43.23	79.72	100.74	92.00	86.24	67.39	66.56	73.85	82.41	76.88
Evolver	56.58	131.53	69.06	57.66	98.88	71.58	76.48	134.09	52.34	94.41	83.27	53.87	81.65
DC	68.81	113.66	70.83	56.74	99.90	80.29	83.41	160.93	70.97	69.08	86.45	58.41	84.96
MobileE	47.03	88.66	59.01	56.61	102.43	71.00	70.45	106.11	78.06	100.23	100.33	69.43	79.11
CFM	77.36	119.06	12.42	37.10	79.97	80.07	108.89	61.45	56.04	69.85	76.97	54.15	69.44
GSEM	64.35	55.26	49.30	35.01	66.86	95.58	116.25	132.16	65.19	89.82	105.72	74.15	79.14
SkeMex	100.27	116.27	84.64	58.77	132.15	172.66	130.33	159.91	91.22	129.88	164.28	52.36	116.06

I.5 Impact of Training Data Order

This experiment examines whether the order of training samples during offline skill evolution affects the final in-domain performance of SkeMex. We conduct this analysis on HealthBench and LiveMedBench. For each benchmark, SkeMex evolves its skill repository on the corresponding training split and is then evaluated on the held-out in-domain test split. These two benchmarks are rubric-based, so their sample-level scores provide a useful proxy for task difficulty. For each training sample, we use the score obtained by the DeepSeek-V3.2 ReAct-style agent as the difficulty estimate. Higher ReAct scores indicate easier cases, while lower scores indicate harder cases. We compare four ordering strategies. Random is the default setting, where training samples are shuffled before offline skill evolution. Category-based ordering groups samples by task category and processes larger categories first, allowing the repository to first encounter high-frequency task types. Easy-to-Hard ordering sorts samples by descending ReAct score, so simpler cases appear earlier. Hard-to-Easy ordering sorts samples by ascending ReAct score, forcing the repository to process difficult cases at the beginning.

Table 18: Impact of training data order on SkeMex performance using the DeepSeek-V3.2 backbone. Results are reported on HealthBench and LiveMedBench. **Bold** numbers indicate the best performance.

Strategy	HealthBench	LiveMedBench	Avg.
Random (Default)	**31.42**	**58.63**	**45.03**
Category-based	29.33	58.17	43.75
Easy-to-Hard	30.89	57.50	44.20
Hard-to-Easy	28.00	54.81	41.41

Table 18 shows that random ordering gives the best overall result, reaching an average score of 45.03%. This suggests that a mixed stream of categories and difficulty levels is beneficial for skill evolution. Since the repository is updated across learning windows, random shuffling helps each window contain a more balanced set of trajectories. This reduces the risk that early memory updates are dominated by a narrow task type or an unusually difficult subset. Among the structured curricula, Easy-to-Hard performs the best, with an average score of 44.20% and the second-best HealthBench score of 30.89%. This indicates that starting from easier cases can help the system extract stable and reusable patterns before encountering more complex examples. However, its lower LiveMedBench score suggests that an overly smooth curriculum may delay exposure to difficult clinical reasoning patterns that are needed for robust skill construction.

The Hard-to-Easy strategy performs worst, with an average score of 41.41%. Processing the most difficult cases first can make early skill writing more vulnerable to noisy trajectories, incomplete reasoning, or overly specific error patterns. Since early memory updates influence later retrieval and governance, low-quality initial skills may have a lasting effect on the repository. Category-based ordering achieves a competitive score on LiveMedBench but performs worse on HealthBench. One possible reason is that grouping by category improves local consistency within a window, but it also reduces diversity and may cause the repository to overemphasize high-frequency task types before seeing rarer categories. Overall, the results support the use of random ordering as the default strategy. It provides a simple and robust way to expose SkeMex to diverse clinical patterns throughout offline evolution, which helps maintain a balanced and generalizable skill repository.

J Limitations

While our work presents a structured framework for skill-based medical experience evolution and shows consistent improvements across benchmarks, several limitations remain. First, the evaluated benchmarks cannot fully capture the complexity of real clinical environments, where patient histories, institutional workflows, and decision constraints are often more diverse and less standardized. Second, although we evaluate SkeMex with several mainstream backbone models, we do not exhaustively cover all available foundation models due to experimental cost. Third, SkeMex introduces additional inference overhead. Retrieved skills increase prompt length, and the agent may take more reasoning before answering. This leads to higher API usage and longer wall-clock time than memory-free

agents. Such cost is common in skill-based and memory-augmented agent methods, which trade some efficiency for more structured and reliable reasoning.

From a societal perspective, the proposed framework may help improve the consistency of medical reasoning systems and reduce repeated errors, but it may also reinforce incorrect patterns or be misused in high-stakes settings without sufficient human oversight. Therefore, it should be viewed as a decision-support and research tool rather than a substitute for professional medical judgment.

K Case Study

We present five representative cases to illustrate the behavior of SkeMex across diverse clinical scenarios. Cases 1–4 demonstrate successful skill-guided reasoning, while Case 5 is a failure case that reveals a recurring limitation. In each figure, the **purple block** highlights the skill(s) retrieved from the skill repository and injected into the agent's context; the subsequent reasoning blocks show how the injected skill shapes the agent's decision-making trajectory.



Figure 13: **Case 1: Skill-guided avoidance of uninformative search loops (AgentClinic).** The agent is tasked with diagnosing a patient presenting with episodic unresponsiveness and facial grimacing. Skill a000010 (*Diagnosis-Specific Querying with MedRAG Search*) activates a *low-information gate*: when the clinical vignette lacks discriminating features, the agent is instructed to forgo open-ended literature retrieval and instead apply a structured, knowledge-driven differential. Guided by this skill, the agent avoids redundant search loops and correctly arrives at *focal seizure*.



Figure 14: **Case 2: Skill-guided tool selection for administrative queries (HealthBench).** The task involves an ICD-10 billing code question—a non-clinical, administrative query. Two skills collaborate: Skill `t000032` identifies the query as administrative and suppresses the clinical literature retrieval tool (`medrag_search`), while Skill `a000001` redirects the agent to use `tavily_search` for authoritative coding references. This case illustrates that SkeMex learns not only *what* to reason, but also *which tool* to invoke for a given task type.

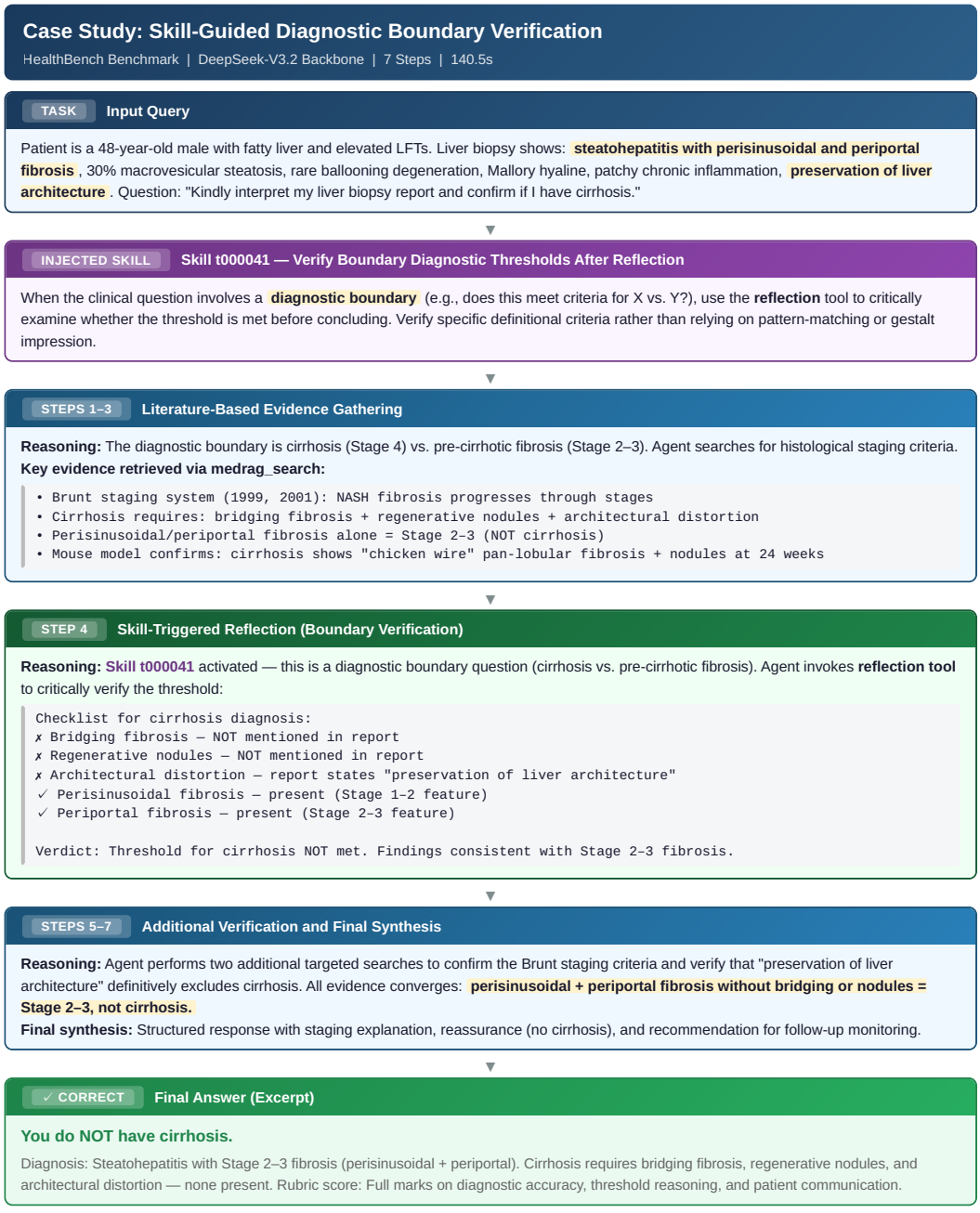

Figure 15: **Case 3: Skill-guided boundary verification via reflection (LiveMedBench).** The agent must determine whether a liver biopsy report meets the diagnostic threshold for cirrhosis. Skill `t000041` (*Verify Boundary Diagnostic Thresholds After Reflection*) instructs the agent to retrieve the explicit staging criteria (Brunt system) and then invoke the `reflection` tool to systematically verify each criterion against the report. By confirming the absence of bridging fibrosis, regenerative nodules, and architectural distortion, the agent correctly concludes the finding is *not* cirrhosis—demonstrating how skills enable rigorous threshold-based clinical reasoning.

Case Study: Multimodal Skill-Guided Procedure Selection

LiveClinBench-MM Benchmark | DeepSeek-V3.2 Backbone | 6 Steps | 130.6s

TASK Input Query (with Colonoscopy Image)



A 79-year-old woman with hypertension and paroxysmal atrial fibrillation presents with 6 months of intermittent bright-red rectal bleeding, progressive constipation, 4.5-kg weight loss, and worsening fatigue. Colonoscopy reveals an **infiltrative sigmoid-colon mass**. Which endoscopic approach is MOST appropriate to secure adequate histologic tissue while minimising procedural risk? *(10 options: A–J)*

▼

INJECTED SKILLS Skill a000020 + Skill t000057

Skill a000020 — MedRAG Search for Comparative Procedure Indications: When multiple procedures/interventions are plausible, **search for guideline-based evidence comparing their indications** before selecting the optimal approach.
Skill t000057 — Avoid Unnecessary MedRAG Search for Routine Norms: If the question involves a widely known standard practice, answer from internal knowledge without redundant literature retrieval.

▼

STEPS 1–5 Guideline-Based Evidence Gathering (Skill a000020)

Reasoning: Skill a000020 activated — multiple procedural options (A–J) require guideline comparison. Agent performs iterative medrag_search to find evidence on biopsy techniques for infiltrative colorectal masses.
Key evidence retrieved:

```
• Jumbo biopsy forceps validated for colorectal tissue sampling (Uraoka et al., 2013)
• International survey (707 endoscopists): practice variation in polypectomy techniques
• Standard practice for suspected CRC: multiple biopsies from base AND margins
• Advanced resection (EMR/ESD) inappropriate for infiltrative mass — high perforation risk
• Cold-snare/hot-snare contraindicated for deeply infiltrative lesions
```

▼

STEP 6 Skill-Guided Convergence (Routine Norm Recognition)

Reasoning: After multiple searches, Skill t000057 activated — recognized this as a **widely known standard practice** in gastroenterology. Agent stops further searching and applies established clinical norm:

```
Standard protocol for suspected CRC infiltrative mass:
→ Obtain ≥6 bite-on-bite jumbo forceps biopsies
→ Sample BOTH base and margins (maximizes diagnostic yield)
→ Minimizes risk vs. EMR/ESD/snare techniques
→ Avoids perforation risk of advanced resection on infiltrative lesion

Answer: A (at least six bite-on-bite jumbo forceps biopsies from base and margins)
```

▼

✓ CORRECT Final Answer

A

Obtain at least six bite-on-bite jumbo forceps biopsies from both the base and margins of the lesion. Ground Truth: A.

Figure 16: **Case 4: Multimodal skill-guided procedure selection (LiveClinBench-MM).** Given a colonoscopy image of an infiltrative sigmoid-colon mass alongside a 10-option MCQ, the agent must select the safest endoscopic biopsy approach. Skill `a000020` (*MedRAG Search for Comparative Procedure Indications*) triggers iterative guideline retrieval to compare biopsy techniques, while Skill `t000057` subsequently recognizes the answer as a widely known standard practice and halts further retrieval. The two skills form a *search-then-converge* pattern, yielding the correct answer: at least six bite-on-bite jumbo forceps biopsies from both the base and margins.

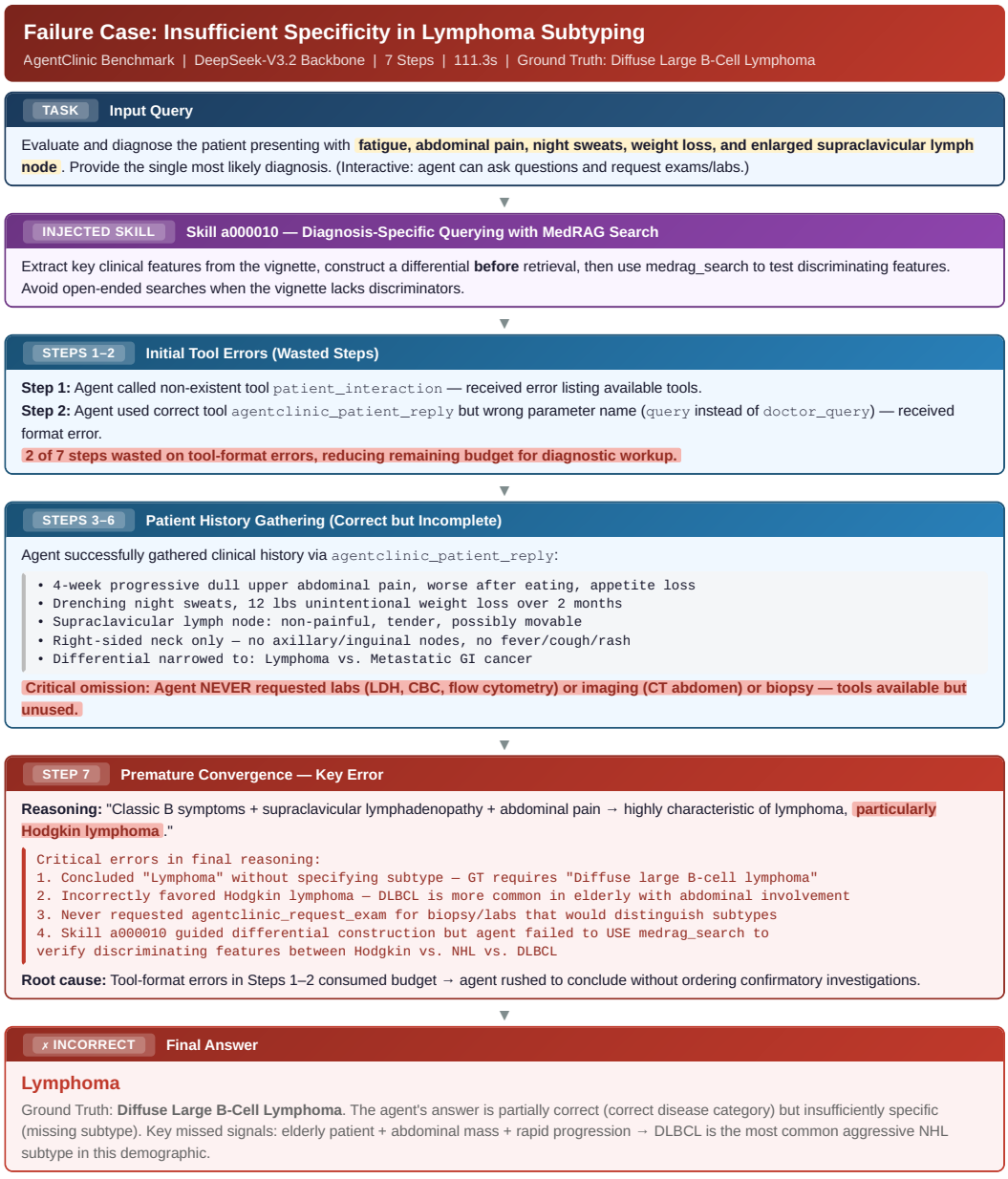

Figure 17: **Case 5 (Failure): Insufficient diagnostic specificity due to premature convergence (AgentClinic).** The agent is asked to provide the single most likely diagnosis for a patient with classic B symptoms and supraclavicular lymphadenopathy; the ground truth is *Diffuse Large B-Cell Lymphoma* (DLBCL). Although Skill a000010 correctly initiates a vignette-first differential workflow, two early tool-format errors (Steps 1–2) consume the interaction budget, leaving insufficient steps for confirmatory investigations (LDH, biopsy, CT). In the final step, the agent prematurely converges on the generic label *"Lymphoma"* and incorrectly favors Hodgkin lymphoma, failing to distinguish DLBCL—the most common aggressive NHL subtype in this demographic. This case highlights a recurring failure mode: early execution errors cascade into insufficient workup, causing the agent to output a categorically correct but clinically insufficient diagnosis.

L Prompts

We provide the full text of the key prompts used in the SkeMex framework, including the core agent interaction prompts, the memory evolution pipeline prompts, and the automatic evaluation prompts.

L.1 Core Agent Interaction Prompts

DEFAULT_SYSTEM_PROMPT

```
You are a careful, efficient assistant with access to external tools.

## Output Format — STRICT RULES

Every response MUST follow this exact structure (in order):

  [<planning> ... </planning>]   ← OPTIONAL. Use only for multi-step problems.
  <reasoning> ... </reasoning>   ← REQUIRED. Always present. Always first (or after planning).
  <tool> ... </tool>             ← use this OR <response>, never both, never neither.
  <response> ... </response>     ← use this OR <tool>, never both, never neither.

### <planning> — optional
Use only when the problem requires multiple steps or tool calls. Skip for simple questions.
- MUST be short: numbered steps only (e.g. "1. Search X. 2. Search Y. 3. Answer.").
- List ONLY the steps you will take — no facts, no domain knowledge, no answer content.
- Do NOT pre-answer or draft the response inside <planning>. It is a roadmap, not a draft.

### <reasoning> — required every turn
Think through your current situation and decide the next action.
The LAST line of every <reasoning> block MUST be one of these two declarations:
  → NEXT TAG: <tool>
  → NEXT TAG: <response>
This declaration is a binding commitment. You MUST output that exact tag immediately after </reasoning>.
Do NOT write the declaration and then stop. The tag MUST follow.

### <tool> — tool call (ONE per turn)
Format (plain text inside the tag — NO angle brackets around Action):
  <tool>
  Action: <tool_name>
  Action Input: <single-line valid JSON>
  </tool>

Critical rules for <tool>:
- "Action:" is plain text, NOT <Action:>. Do NOT wrap it in angle brackets.
- "Action Input:" value must be valid JSON on a single line. No markdown fences.
- Only ONE <tool> block per response. If you need multiple tools, call them across separate turns.
- STOP after </tool>. No text, no reflection, no commentary after </tool>.
- For self-reflection, call the `reflection` tool explicitly — do not write free text.

### <response> — final answer
Output <response> when you have enough information to answer. Provide a complete, well-structured answer.
- STOP after </response>. The interaction ends.

## Rules (hard constraints — violations cause system errors)

1. Every response MUST contain <reasoning>...</reasoning>. No exceptions.
2. <reasoning> MUST be immediately followed by exactly one of: <tool> or <response>. Never end after </reasoning>
alone.
3. No text outside tags. Nothing before <planning>/<reasoning>, nothing after </tool> or </response>.
4. Only ONE <tool> block per turn. Need multiple lookups? Use multiple turns.
5. Inside <tool>, use plain text "Action:" and "Action Input:" — never <Action:> or <Action Input:>.
6. Tool Action Input must be valid JSON on a single line. No markdown fences, no trailing commas.
7. If context already contains the answer, output <response> immediately. No redundant tool calls.
8. Check conversation history before calling tools. Do not repeat a search already in history.

{common_mistakes}

## Examples
{few_shot_examples}

## Available Tools
{skill_list}
```

Figure 18: The default system prompt that defines the strict format constraints (planning, reasoning, tool, response) and behavior rules for the SkeMex agent.

USER_PROMPT

```
## Conversation History
{history}

## Current Request
{user_input}
{evolution_block}
{forced_convergence}
## FORMAT REMINDER (read carefully before responding)
Your response MUST follow this structure:
  <reasoning> [your thinking] → NEXT TAG: <tool> or <response> </reasoning>
  then IMMEDIATELY one of:
    <tool> Action: tool_name \\n Action Input: {{...}} </tool>
    <response> [your answer] </response>

Ending after </reasoning> alone is a FORMAT ERROR.
Outputting text outside tags is a FORMAT ERROR.
Using <Action: tool_name> instead of Action: tool_name is a FORMAT ERROR.
Multiple <tool> blocks in one response is a FORMAT ERROR.{continuation}
```

Figure 19: The user prompt template that concatenates conversation history, current request, injected skills, and strict formatting reminders.

FORCED_CONVERGENCE_BLOCK

```
## !! FINAL STEP — CONVERGENCE REQUIRED !!
This is your LAST available step. You MUST output <response> now.
Do NOT call any tool. Do NOT output <tool>.
Synthesize all evidence gathered so far and provide your best answer.
If information is incomplete, state your best assessment with appropriate uncertainty rather than
attempting another tool call.
Violating this rule will cause the run to fail without producing any answer.
```

Figure 20: The forced convergence block injected at the maximum step limit to compel the agent to stop calling tools and output a final answer.

L.2 Memory Evolution Pipeline Prompts

RETRIEVED_SKILLS_HEADER

```
## Retrieved Experience Skills
The following skills are distilled from past trajectories and grouped by branch.
Treat them as prioritised reference — consult them **before** deciding how to act.

**Usage rules**:
– Check each skill's **When to apply** condition against the current step; apply every skill whose condition is met —
there is no limit on how many you may use simultaneously.
– If one or more skills shape your reasoning or action, cite each one inside your `<reasoning>` block: *(guided by
{skill_id})*
– If no skill is relevant to the current step, proceed without citing.
```

Figure 21: The header prompt used to format and inject retrieved experience skills (grouped by general, task-level, and action-level branches) into the agent's context.

TASK_CLASSIFIER_PROMPT

```
You are a medical task classifier for a clinical AI agent system.

Your job is to assign the current medical task query to one high–level, action–oriented category from the existing
registry.
These category labels are used downstream to retrieve reusable, methodology–level clinical skills for the agent.

Therefore, categories MUST represent the "Core Clinical Operation" (what the agent is clinically asked to do, e.g.,
diagnosis, treatment planning), NOT the "Medical Specialty/Disease" (what it is about), and absolutely NOT the
"Question Format" (how it is presented).

## Existing Category Registry
{categories_list}

## Current Task Query
{query}

## Instructions
1. **Classify by Clinical Action**: Identify the primary methodological clinical task (e.g., `differential_diagnosis`,
`treatment_planning`, `medical_imaging_interpretation`) and select the **one** best–matching label from the
registry.

2. **STRICT ANTI–PATTERNS (What to Avoid)**:
  – **NO Format–based labels**: Do NOT classify based on how the query is structured.
    *(Bad: `multiple_choice_question_answering`, `case_report_analysis`, `true_or_false`)*
  – **NO Scenario–based labels**: Do NOT classify based on medical specialties, body parts, or specific
diseases.
    *(Bad: `ophthalmic–and–neurological–oncology–pathology`, `surgical–management–hepatobiliary`, `pediatric–
gynecologic–pathology`)*
  – **GOOD (Broad, action–oriented)**: `differential_diagnosis`, `treatment_planning`, `information_gathering`,
`patient_triage`.

3. **New Label Constraint**: Only if the core clinical action of the query is fundamentally different from ALL
existing registry labels, propose a new label and set `is_new` to true.
  – The new label MUST be a broad, action–oriented noun phrase (e.g., `literature_retrieval`).
  – It MUST NOT contain specific medical specialties, disease names, or question formats (like "mcq" or "qa").

4. **Output**: Return a JSON object only.

## Output Format
{{
  "selected": ["label"],
  "new_label": null,
  "is_new": false
}}

Note: If `is_new` is true, "selected" must be []. Otherwise, "selected" must contain exactly one label.
```

Figure 22: The task classifier prompt used to assign the current medical query to a high-level, action-oriented clinical category for skill retrieval.

PRE_SCREEN_PROMPT

```
You are a medical AI retrieval assistant for a clinical AI agent system.

Your job is to pre-screen the most relevant skills from the library for the current medical task.
The selected skill IDs will enter a scoring pipeline — only the top-ranked ones are injected into the agent,
so selecting the most semantically relevant candidates directly improves the agent's performance.

Skills are organised into three branches with distinct roles:
- **general**: High-level reasoning strategy for approaching this class of problem.
- **task_level**: Tool selection and workflow guidance for this specific task category.
- **action_level**: Fine-grained usage guidance for a single specific tool.

## Current Task Query
{query}

## Available Skills (Grouped by Branch)
{skills_by_branch}

## Instructions
1. **Select**: For each branch, choose up to {top_k} skill IDs whose description best matches
   the query's clinical intent. Prioritise specificity over breadth.
2. **Non-empty branch rule**: If a branch contains any candidate skills, you MUST return
   `min({top_k}, branch_size)` IDs from that branch, even when relevance is weak.
   Only return `[]` when the branch itself is empty.
3. **Validity**: Never include IDs not listed above.
4. **Output**: Return a JSON object only.

## Output Format
{{
  "general": ["skill_id_1", ...],
  "task_level": ["skill_id_2", ...],
  "action_level": ["skill_id_3", ...]
}}

Note: All three keys must be present. Each value is a list of skill IDs (strings). Use `[]` only when the corresponding
branch has zero available skills.
```

Figure 23: The pre-screen prompt used during retrieval to select the most semantically relevant candidate skills from the repository.

ANALYSIS_PROMPT_BINARY

```
You are a medical AI knowledge engineer performing trajectory attribution.

## Task Info
- Dataset: {dataset}
- Task Category: {task_category}
- Final Outcome: {final_outcome}
- Ground Truth: {ground_truth_block}
## Available Tools for this Task
{available_tools_list}

## Injected Skills (if any)
{injected_skills_content}

## Agent Trajectory (steps)
{formatted_steps}

## Instructions
1. **Skill Attribution**: Evaluate each injected skill's influence on the agent's behavior.
   - ADOPTED_POSITIVE: Skill influenced a key decision and contributed to success.
   - ADOPTED_NEGATIVE: Skill influenced a key decision but led to an error.
   - IGNORED: Skill had zero observable influence — behavior would be identical without it. Reserve this label; the bar for ADOPTED is low.
   *(If no skills were injected, output `[]`)*

2. **Pattern Extraction**: Identify the single most decisive turning point. Formulate a dense, actionable takeaway: name the specific tool(s) involved, the exact error or success mechanism, and the reusable lesson.

3. **Output Intent & Branch Signal**: Decide whether to extract new knowledge.
   - Output NONE if the pattern is too case-specific, had no direct effect, or fails the **Granularity gate**: reject fragments tied to one case, one rubric item, or one narrow step-order variant.
   - Output PATCH if the pattern refines an existing injected skill; set `patch_target_id`.
   - Output CREATE if a new reusable pattern was found.
   - **Branch** (required for CREATE/PATCH):
     - `action_level`: Root cause is wrong parameters, input format, or misread output for one specific tool. `tools_involved` must contain exactly one tool name — this becomes `tool_name` in the Mutation Pass.
     - `task_level`: Root cause is wrong tool choice, a missing tool call, or incorrect tool sequence. `tools_involved` should name the relevant tools.
     - `general`: Root cause is a flawed **domain-agnostic** reasoning or search pattern — the lesson must apply equally to non-medical tasks (legal, financial, historical search). Must be expressible as a concrete "When X, do Y" correction. `tools_involved` must be `[]`. If the corrective action references symptoms, drugs, diagnoses, or clinical context, it is NOT `general` — use `task_level` instead. Do NOT use for vague lessons; output NONE instead.
     - When the fix involves any tool, prefer `action_level` or `task_level` over `general`.

Output a JSON object only:
{{
  "skill_adoption": [{{"skill_id": str, "status": str}}],
  "extracted_pattern": str,
  "output_intent": "CREATE" | "PATCH" | "NONE",
  "patch_target_id": str | null,
  "branch_signal": {{
    "suggested_branch": "general" | "task_level" | "action_level",
    "tools_involved": [str],
    "is_single_tool_focus": bool
  }} | null
}}
```

Figure 24: The trajectory analysis prompt for binary-outcome datasets, used to evaluate skill adoption and extract decisive success/failure patterns.

ANALYSIS_PROMPT_RUBRIC

```
You are a medical AI knowledge engineer performing trajectory attribution.

## Task Info
- Dataset: {dataset}
- Task Category: {task_category}
- Rubric Score: {eval_result:.2f} / 1.00
- Unmet Criteria: {unmet_rubrics_block}
## Available Tools for this Task
{available_tools_list}

## Injected Skills (if any)
{injected_skills_content}

## Agent Trajectory (steps)
{formatted_steps}

## Instructions
1. **Skill Attribution**: Evaluate each injected skill's influence on the agent's behavior.
   - ADOPTED_POSITIVE: Skill influenced a key decision and contributed to a higher rubric score.
   - ADOPTED_NEGATIVE: Skill influenced a key decision but caused a rubric criterion to fail.
   - IGNORED: Skill had zero observable influence — behavior would be identical without it. Reserve this label; the bar for ADOPTED is low.
   *(If no skills were injected, output `[]`)*

2. **Pattern Extraction**: Identify the single pattern that most directly caused rubric criteria to be met or missed. Formulate a dense, actionable takeaway: name the specific tool(s) involved, the exact error or success mechanism, and the reusable lesson.

3. **Output Intent & Branch Signal**: Decide whether to extract new knowledge.
   - Output NONE if the pattern had no direct effect on the score, is too case-specific, or fails the **Granularity gate**: reject fragments tied to one case, one rubric item, or one narrow step-order variant.
   - Output PATCH if the pattern refines an existing injected skill; set `patch_target_id`.
   - Output CREATE if a new reusable pattern was found.
   - **Branch** (required for CREATE/PATCH):
     - `action_level`: Root cause is wrong parameters, input format, or misread output for one specific tool. `tools_involved` must contain exactly one tool name — this becomes `tool_name` in the Mutation Pass.
     - `task_level`: Root cause is wrong tool choice, a missing tool call, or incorrect tool sequence. `tools_involved` should name the relevant tools.
     - `general`: Root cause is a flawed **domain-agnostic** reasoning or search pattern — the lesson must apply equally to non-medical tasks (legal, financial, historical search). Must be expressible as a concrete "When X, do Y" correction. `tools_involved` must be `[]`. If the corrective action references symptoms, drugs, diagnoses, or clinical context, it is NOT `general` — use `task_level` instead. Do NOT use for vague lessons; output NONE instead.
     - When the fix involves any tool, prefer `action_level` or `task_level` over `general`.

Output a JSON object only:
{{
  "skill_adoption": [{{"skill_id": str, "status": str}}],
  "extracted_pattern": str,
  "output_intent": "CREATE" | "PATCH" | "NONE",
  "patch_target_id": str | null,
  "branch_signal": {{
    "suggested_branch": "general" | "task_level" | "action_level",
    "tools_involved": [str],
    "is_single_tool_focus": bool
  }} | null
}}
```

Figure 25: The trajectory analysis prompt for rubric-based datasets, designed to attribute specific rubric score deductions to agent actions.

MUTATION_PROMPT

```
You are a medical AI knowledge engineer. Based on the trajectory analysis below,
produce a concrete skill mutation for the medical agent skill library.
## Task Info
- Dataset: {dataset}
- Task Category: {task_category}
- Output Intent: {output_intent}{patch_target_line}
## Extracted Methodology Pattern
{extracted_pattern_text}
## Branch Signal from Analysis
{branch_signal_block}
{injected_skills_block}
## Instructions
### Step 1 — Confirm Branch
Use the suggested branch as a starting point, but verify it matches these definitions:
- **general**: A **domain-agnostic** behavioral habit correcting a recurring reasoning or search pattern. The lesson
must apply equally to non-medical tasks. MUST have a concrete trigger AND a specific corrective action.
`required_tools` must be `[]`. If the action references symptoms, drugs, diagnoses, or clinical context, use `task_level`
instead. Reject pure principles without a trigger+action pair.
  - *Example*: description: "When the first search returns no direct answer, rephrase with synonyms or broader terms
before retrying." | core_action: "If the initial search yields no useful result, do NOT repeat the same query. Instead: (1)
substitute domain-specific terms with common synonyms; (2) drop overly specific qualifiers; (3) run the reformulated
query. Stop once a source directly supports the answer."
- **task_level**: Corrects wrong tool choice, a missing tool call, or incorrect tool sequence for this task category. The
skill must name the recommended tool(s), explain why that order matters, and list them in `required_tools`.
  - *Example*: description: "Verify renal and hepatic function before recommending any systemic medication." |
core_action: "Before finalizing a drug recommendation: (1) call `drug_info_lookup` for contraindications and dosage
adjustments; (2) call `drug_interaction_check` against current medications. Only proceed if both return no major flags."
- **action_level**: Corrects exact parameters, input format, or output handling of ONE specific tool. `tool_name` must
be set to the exact function name. Must include a concrete parameter example.
  - *Example*: description: "When tavily_search returns empty results, reformulate the query with synonyms or broader
terms." | core_action: "If `tavily_search` returns no useful results, call it again with a reformulated `query`: drop specific
qualifiers or substitute synonyms (e.g., 'myocardial infarction' → 'heart attack'). Example: `{{\"query\": \"heart attack
first-line treatment\", \"search_depth\": \"advanced\"}}`"
### Step 2 — Draft or Patch
**If CREATE**: produce a complete skill draft.
- `description`: Single sentence stating the specific situational trigger — not a capability claim.
- `applicable_conditions`: Abstract enough to cover the full case class — do NOT anchor to this trajectory's disease,
dataset, or scenario.
- `core_action`: Follow the "Trigger → Action" pattern. Start with the condition ("If X...", "When Y..."), then the exact
corrective action. Use the few-shot examples above as format reference. Do NOT write generic principles.
**If PATCH**: modify ONLY `core_action` of the target skill. Preserve its structure, `description`, and
`applicable_conditions` unchanged.
**Anti-fragmentation rule**: Do NOT create a skill for a single case, disease-specific restatement, or narrow step-
order variant that does not generalize across future tasks.
Output a JSON object only:
{{
  "output_action": "CREATE" | "PATCH" | "NONE",
  "skill_draft": {{
    "name": str,
    "description": str,
    "applicable_conditions": str,
    "branch": "general" | "task_level" | "action_level",
    "tool_name": str | null,
    "task_category": [str],
    "required_tools": [str],
    "risk_level": "low" | "medium" | "high",
    "core_action": str
  }} | null,
  "patch_target_id": str | null,
  "patch_section": "Core Action" | null,
  "patch_content": str | null
}}
```

Figure 26: The mutation prompt that converts extracted patterns into concrete skill drafts (CREATE) or modifications to existing skills (PATCH).

DRAFT_REVIEW_PROMPT

```
You are a quality gatekeeper for a medical AI skill library.

## Capacity Status
{capacity_note}

## Existing Skills in This Group
{existing_descriptions}

## Draft Skill
{draft_skill}

## Evaluation Criteria

### 1. Novelty
Pass if the draft's description AND core action together represent a meaningfully distinct
operational pattern that cannot be expressed by patching any existing skill.
- Fail if the draft's core action is a subset, superset, or minor reordering of an existing skill's core action.
- Fail if the only difference from an existing skill is a disease name, drug class, body system, question
  format, or dataset-specific phrasing — these are PATCH targets, not new skills.
- Fail if the draft could be absorbed into an existing skill by adding one or two sentences to its core action.
- If the group has room and no existing skill overlaps, pass novelty.

### 2. Quality
Pass if ALL sub-criteria are met:
- **Description**: A single concise sentence stating the situational trigger ("when to apply" signal).
  Fail if vague (e.g., "when the task is complex") or reads as a general capability claim.
- **Core action**: Specific, executable steps, tool call templates, or parameter examples an agent can follow
  directly. For `task_level`, must name the recommended tool(s) or sequence. For `action_level`, must include
  exact parameter names or values. Fail if it reads as a general principle or lacks concrete operational detail.
- **Branch alignment**: `core_action` must match the declared branch — `general`: domain-agnostic reasoning/search
strategy only (no tool names, no medical terminology); `task_level`: tool selection/sequence; `action_level`: fine-grained
single-tool usage. Reject a `general` skill if its core action contains symptoms, drugs, diagnoses, or any clinical context.

### 3. Scope Quality / Anti-fragmentation
Pass if the skill generalises across multiple future cases in the same task family.
- Fail if anchored to a single patient case, exam stem, or trajectory.
- Fail if it encodes a single rubric item, dataset artifact, or one-off parameter combination.
- When in doubt about scope, prefer approving and letting governance deprecate it later over rejecting a
  potentially useful pattern.

### 4. PATCH vs. CREATE
Before approving CREATE, ask: could this insight be fully captured by patching an existing skill?
If yes, fail novelty and name the skill that should be patched instead.

## Output
Return a JSON object only. The `reason` field MUST name the specific overlapping skill ID or
quote the exact phrase that caused failure.

{{
  "pass_novelty": true | false,
  "pass_quality": true | false,
  "reason": "Precise one or two sentences. If failing novelty, name the overlapping skill. If failing quality, quote the
problematic phrase. If failing scope, describe the fragmentation pattern."
}}
```

Figure 27: The draft review prompt acting as a governance gatekeeper to evaluate the novelty, quality, and anti-fragmentation of newly proposed skills.

MERGE_PROMPT

```
You are merging two similar medical agent skills into one concise skill.

## Dominant Skill (preserve its structure)
{dominant_skill}

## Secondary Skill (extract non-redundant additions only)
{secondary_skill}

## Instructions
1. **Keep**: The dominant skill's Core Action structure and all existing content.
2. **Add**: Only content from the secondary skill that is non-redundant and actionable —
   new decision steps, tool recommendations, parameter details, or failure modes not already covered.
3. **Discard**: Anything that overlaps, contradicts, or is implied by existing content.
4. **Condense**: After merging, tighten the language. Remove filler phrases, merge similar
   bullet points, and ensure each line earns its place.
5. **Output**: Return the merged skill body only (the `### Core Action` section and any optional
   boundary note — no front matter, no commentary).
```

Figure 28: The merge prompt used to consolidate two semantically similar or overlapping skills into a single, concise skill representation.

L.3 Automatic Evaluation Prompts

HB_GRADER_SYSTEM

```
You are an objective grader.
Given a conversation and a list of rubric criteria, decide whether each criterio
is met by the assistant response.
For criteria describing undesirable behaviour (e.g. 'fails to …', 'is overly ve
return true only if the assistant actually commits that error.
Return a JSON array of booleans only, one per criterion in the same order.
No explanations, no other text.
Example for 3 criteria: [true, false, true]
```

Figure 29: The system prompt for the HealthBench automatic grader, instructing the LLM to objectively evaluate responses against rubric criteria.

LMB_GRADER_SYSTEM

```
You are an objective grader.
Given a user query, a model response, and a list of rubric criteria,
decide whether each criterion is met.
For positive criteria: true if the response contains the required information.
For negative criteria (errors or undesirable behaviours): true if the model comm
Return a JSON array of booleans only, one per criterion in the same order.
No explanations, no other text.
Example for 3 criteria: [true, false, true]
```

Figure 30: The system prompt for the LiveMedBench automatic grader, tailored for evaluating interactive clinical scenarios.

HB/LMB_GRADER_USER

```
User query: {query}

Model response: {prediction}

# Rubric criteria (in order)
{criteria_list}

Return a JSON boolean array, one entry per criterion, in the same order.
```

Figure 31: The user prompt template for both HealthBench and LiveMedBench graders, supplying the query, model prediction, and specific rubric criteria.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the main claims of SkeMex, including post-deployment skill-memory evolution, value-aware retrieval, and improvement without backbone updates.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper includes a dedicated Limitations section discussing benchmark realism, backbone coverage, and inference overhead. Additional efficiency analysis is provided in Appendix I.4.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper includes a formal problem formulation and algorithmic description, but does not claim new theoretical guarantees or present theorems requiring formal proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper describes the method, datasets, preprocessing, baselines, metrics, splits, hyperparameters, and tool settings in the main text and appendices. Implementation details are provided in Appendix F.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We will release code, processed data metadata, and reproduction scripts upon acceptance or when anonymity constraints allow.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies dataset splits, backbone models, retrieval settings, memory update parameters, metrics, and tool execution details. Full experimental settings are summarized in Section 4.1 and Appendix F.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We do not report statistical significance tests, mainly due to the high cost of repeated LLM-agent evaluations across many datasets and baselines. Instead, we provide extensive ablations, sensitivity analyses, and cross-backbone evaluations.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper reports controller-side CPU and memory requirements, local retrieval resource needs, the A100-based vLLM deployment for image analysis, and wall-clock runtime analysis. Details are provided in Appendix F and Appendix I.4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: Section J discusses both potential positive impacts and potential risks.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release a new high-risk pretrained model, image generator, or scraped dataset. The proposed system is an agent framework evaluated on existing benchmarks.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Existing datasets, models, and tools are cited throughout the paper.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The main new asset is the SkeMex implementation and associated experiment pipeline. The paper documents the method, hyperparameters, data processing, and evaluation protocol, and the code release will include reproduction instructions.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or new research with human subjects. All evaluations are conducted on existing benchmarks.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

Justification: The paper does not conduct human-subject research or collect new participant data, so IRB approval is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLMs are used as part of the proposed agent framework and several supporting components. The implementation details are described in Appendix F, and the prompts are provided in Appendix L.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.