APEx: Distillation of Agent Procedural Experience for Adaptive Deep Research Question Answering

Jie Ding[1]* **Rui Sun**[1]* **Xinyuan Zhang**[3] **Zeyu Zhang**[2] **Xin Liu**[2]†
[1]University of Science and Technology of China
[2]Chinese Academy of Sciences [3]ByteDance Inc.
{jieding25,rrsun}@mail.ustc.edu.cn, xliu@gia.cas.cn

Abstract

Deep research agents augment large language models with external tools to answer complex, long-horizon questions through multi-turn reasoning. Learning from prior experience is crucial for continual improvement, yet existing methods either retrieve verbose task-specific traces that burden decision-making, or distill procedural skills that remain decoupled from downstream policy adaptation. We propose APEx, a hierarchical experience utilization framework that organizes interaction history into instance-level trajectory memories and category-level procedural skills, and couples them through a closed-loop architecture of Executor, Distiller, and Planner. The three modules are optimized via a three-stage alternating GRPO training paradigm, enabling reward-guided skill distillation rather than fixed-prompt generation. At test time, distilled skills serve as procedural priors for online Planner adaptation through skill-guided test-time reinforcement learning, allowing ground-truth-free self-improvement with skill-alignment regularization to prevent policy drift. Experiments on 7 benchmarks demonstrate that APEx achieves state-of-the-art performance, surpassing GPT-5.4 by 14.7 points and the strongest memory-augmented baseline by 3.0 points. Code is available at https://github.com/J-Ding519/APEx.

1 Introduction

Deep research agents (DRAs) augment large language models (LLMs) with external tools to tackle complex, long-horizon, and open-ended questions through multi-turn answering (Sun et al., 2025, 2026; Li et al., 2026b; Zhu et al., 2026). These tasks often involve multi-step tool orchestration (Wu et al., 2025a; Li et al., 2026a), iterative information retrieval (Jin et al., 2025; Wu et al., 2025b), and multi-source synthesis (Chen et al., 2025), making learning from prior experience essential. Without effective experience utilization, DRAs risk repeating past mistakes instead of building on prior successes (Wang et al., 2024; Wang and Chen, 2025). This raises a central question: how can deep research agents continually improve through long-term interaction experience?

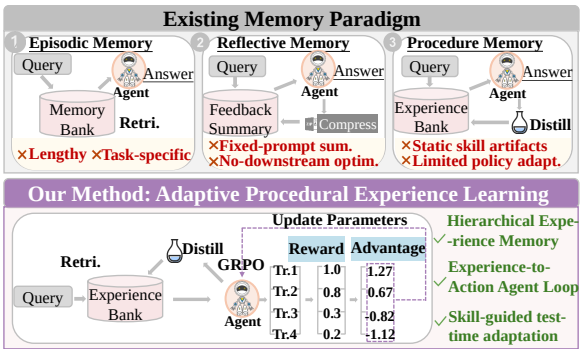

Figure 1: Comparison between existing memory-based agent methods and our adaptive procedural experience learning framework.

Recent work has attempted to answer this question by equipping agents with memory mechanisms that store, retrieve, and reuse prior experience (Zhou et al., 2025; Qiao et al., 2026). Episodic memory methods store and retrieve past trajectories or task instances (Zhong et al., 2024; Zhao et al., 2024), but retrieved experiences are often lengthy, task-specific, and noisy, forcing agents to infer transferable strategies at decision time. Reflective memory methods compress feedback into textual summaries (Shinn et al., 2023; Tan et al., 2025), yet these summaries are typically generated by fixed prompts or optimized for memory personalization, rather than optimized for their downstream utility in planning and execution. Procedural memory methods further represent experience as reusable skills, plans, or instructions (Zhou et al., 2026; Ma et al., 2026; Xia et al., 2026), but such skills are

* Equal contribution.
† Corresponding author.

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 25317–25341
October 24-29, 2026 ©2026 Association for Computational Linguistics

often treated as external knowledge artifacts rather than adaptive priors that directly optimize the agent policy during inference.

These suggest that experience-driven improvement in DRAs should be formulated as a closed loop rather than a one-way memory reuse process (Zhou et al., 2025). Long-term interaction experience should be distilled into structured, reusable skills (Ma et al., 2026); the generation of these skills should be optimized by downstream reinforcement learning signals rather than fixed prompts (Xia et al., 2026); and the resulting skills should serve as procedural priors for test-time reinforcement learning (TTRL), where the agent updates its parameters during inference to adapt to new tasks (Qiao et al., 2026). As TTRL improves the agent, the agent can generate higher-quality trajectories, which in turn support better skill distillation and further model improvement.

To realize experience-driven improvement, we propose APEx (**A**daptive **P**rocedural **Ex**perience Learning). It is characterized by the following key features: ❶ **Hierarchical Experience Memory.** APEx organizes long-term trajectories into instance-level memories and category-level skills, preserving concrete task-solving traces while abstracting reusable procedural knowledge for planning. ❷ **Experience-to-Action Agent Loop.** Built on this hierarchy, APEx integrates an Executor, a Distiller, and a Planner into an experience-to-action loop, where tool-use trajectories are collected, abstracted into skills, and reused for strategic planning. This design aligns execution, abstraction, and planning for continual agent improvement. ❸ **Skill-guided Test-time Adaptation.** During deployment, APEx uses retrieved skills and memories as procedural priors for TTRL, allowing the Planner to adapt on unlabeled queries and consolidate new trajectories into future memories and skills. Our contributions are as follows.

❶ ***Framework.*** We propose APEx, a hierarchical experience utilization architecture that couples instance-level memories with category-level skills, enabling the Distiller, Planner, and Executor to transform concrete trajectories into generalizable procedural guidance.

❷ ***Training.*** We introduce a three-stage alternating GRPO training paradigm for the Executor, Distiller, and Planner, which stabilizes cross-module credit assignment and makes skill distillation a reward-guided learning process rather than fixed prompting.

❸ ***Reasoning.*** We develop Skill-guided TTRL, which uses distilled skills to regularize online Planner adaptation on unlabeled queries. Coupled with no-ground-truth rewards, it forms a feedback loop that turns improved trajectories into stronger skills for future adaptation.

❹ ***Experiment.*** Extensive experiments on diverse benchmarks demonstrate that APEx achieves SOTA performance (64.4% avg), surpassing GPT-5.4 by 14.7 points and the strongest memory-augmented baseline by 3.0 points, while the skill-only variant reduces injected memory tokens, validating the context efficiency of hierarchical experience utilization over flat memory retrieval.

2 Methodology Overview

Figure 2 illustrates the overall workflow of APEx, a hierarchical experience utilization framework for deep research agents. APEx maintains long-term experience at two complementary levels: instance-level memories that preserve concrete execution trajectories, and category-level skills that summarize reusable procedural knowledge across related tasks. Built upon this experience hierarchy, APEx couples planning, execution, skill distillation, and TTRL into a closed-loop improvement process.

APEx Agents Workflow. Given an original query q, APEx processes it through a closed-loop data flow. The query q is first routed to a category–modality bucket, from which relevant instance-level memories $\mathcal{R}_M(q)$ and the corresponding category-level skill $\mathcal{R}_S(q)$ are retrieved. These experiences are organized into a structured context:

$$E(q) = \mathcal{R}_S(q) \oplus \mathcal{R}_M(q), \tag{1}$$

where $\oplus$ denotes structured prompt concatenation. The retrieval method is detailed in Appendix A. Conditioned on q and $E(q)$, the Planner generates a research plan:

$$p = \pi_\psi^{\text{plan}}(q, E(q)). \tag{2}$$

The Executor then follows the plan to perform multi-turn tool interactions and produce an execution trajectory τ with the final answer $\hat{y}$:

$$\tau, \hat{y} = \pi_\theta^{\text{exec}}(q, p \mid \mathcal{T}), \tag{3}$$

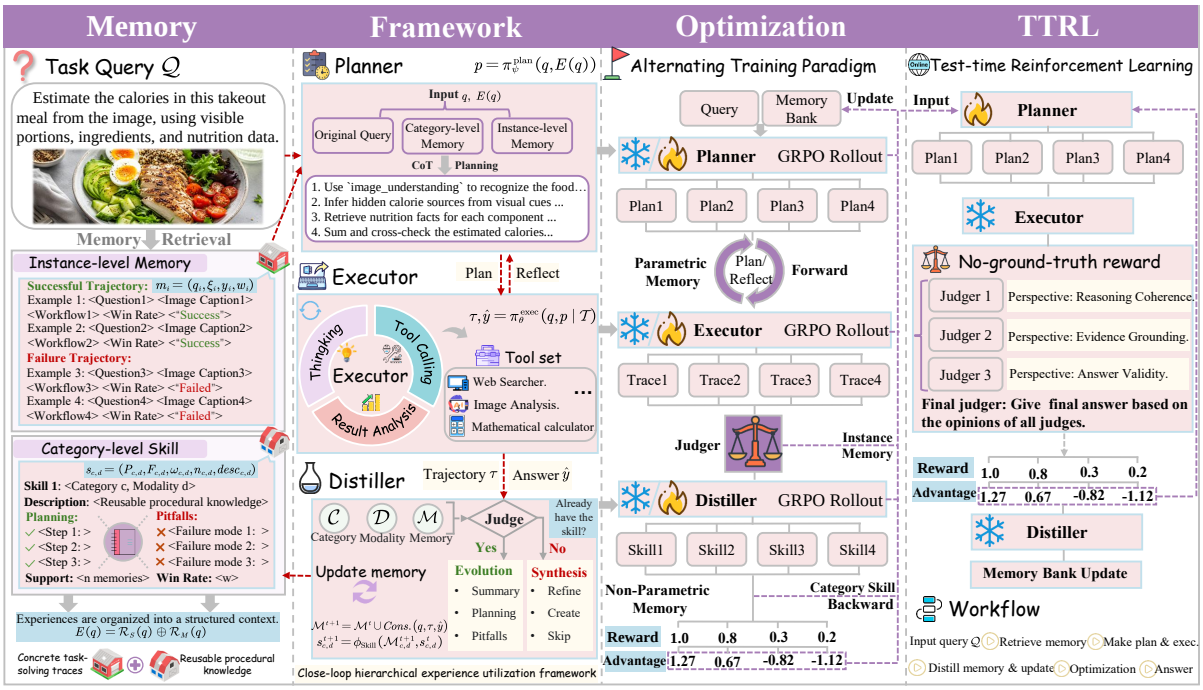

Figure 2: Framework of APEx, which comprises 4 parts: memory, framework, optimization, and TTRL.

where $\mathcal{T}$ denotes the available tool set. After execution, the trajectory is summarized and stored as a new instance-level memory, while the corresponding category-level skill $s_{c,d}$ is updated when sufficient new experience has accumulated:

$$\begin{aligned} \mathcal{M}^{t+1} &= \mathcal{M}^t \cup \text{Consolidate}(q, \tau, \hat{y}), \\ s_{c,d}^{t+1} &= \phi_{\text{skill}}(\mathcal{M}_{c,d}^{t+1}, s_{c,d}^t), \end{aligned} \tag{4}$$

where $\mathcal{M}_{c,d}^t \in \mathcal{M}^t$ and $\mathcal{M}^t$ is the instance-level memory bank at timestamp t and ϕ_{skill} is the distillation operation of Distiller.

Group Relative Policy Optimization (GRPO). GRPO (Shao et al., 2024) optimizes the policy with intra-group relative rewards. In our work Distiller, Planner and Executor are optimized with GRPO. For each input x, the policy samples a group of G responses $\{o_i\}_{i=1}^G$ and computes the group-normalized advantage as $\hat{A}_i = (R(o_i) - \text{mean}_j[R(o_j)])/(\text{std}_j[R(o_j)] + \epsilon)$. Following the GRPO implementation, the policy objective for the currently optimized module π_η is:

$$\mathcal{L}_{\text{GRPO}}(\eta) = -\mathbb{E}_{x,\{o_i\}}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|} \min\left(r_{i,t}(\eta)\hat{A}_i, \text{clip}\left(r_{i,t}(\eta), 1-\epsilon, 1+\epsilon\right)\hat{A}_i\right)\right], \tag{5}$$

where $r_{i,t}(\eta) = \pi_\eta(o_{i,t} \mid x, o_{i,<t})/\pi_{\eta_{\text{old}}}(o_{i,t} \mid x, o_{i,<t})$ is the token-level probability ratio.

Test-time Reinforcement Learning (TTRL). Unlike standard TTRL based on majority voting (Zuo et al., 2025), APEx constructs rewards using a multi-judge LLM-as-Judge mechanism (Zheng et al., 2023). In our implementation, we update the Planner at test time. Given a test query q, the Planner first retrieves the corresponding skill $\mathcal{R}_S(q)$ and memory $\mathcal{R}_M(q)$, generates a plan p, and the Executor produces an execution trajectory τ and final answer $\hat{y}$. The Planner's parameters ψ are updated online by GRPO using the final reward R_{final}:

$$\psi^{t+1} = \text{GRPO}\left(\psi^t, R_{\text{final}}\right). \tag{6}$$

3 Methodology

3.1 Hierarchical Experience Memory

To enable efficient experience utilization, APEx organizes experience into a hierarchical memory structure, where instance-level memories preserve concrete task-solving traces and category-level skills summarize reusable procedural knowledge.

Instance-level Trajectories Memory. Given the i-th original query q_i, the agent performs multi-turn tool interactions and produces an execution trajectory τ_i. Each instance-level memory is rep-

resented as $m_i = (q_i, \xi_i, y_i, w_i)$, where ξ_i is the workflow summary, $y_i \in \{\text{correct}, \text{incorrect}\}$ denotes the execution outcome, and $w_i = \text{success_count}_i \,/\, \text{usage_count}_i$ is the memory reuse win rate. All instance-level memories are organized into a memory bank $\mathcal{M}$ according to task category and input modality:

$$\mathcal{M} = \bigcup_{c \in \mathcal{C}} \bigcup_{d \in \mathcal{D}} \mathcal{M}_{c,d}, \tag{7}$$

where $\mathcal{C}$ is the set of predefined task categories and $\mathcal{D}$ is the set of modalities.

Category-level Skills Memory. While instance-level memories provide concrete task-solving cases, they may contain task-specific details. To extract reusable procedural knowledge, the Distiller synthesizes the memories within each category–modality bucket into a single evolving skill document $s_{c,d}$:

$$s_{c,d} = \phi_{\text{skill}}(\mathcal{M}_{c,d}), \tag{8}$$

where ϕ_{skill} denotes the distillation operation of Distiller. A skill entry $s_{c,d}$ is defined as $s_{c,d} = (P_{c,d}, F_{c,d}, w_{c,d}, n_{c,d}, \text{desc}_{c,d})$, where $P_{c,d}$ denotes recommended planning steps, $F_{c,d}$ denotes common failure modes, $n_{c,d}$ denotes the number of supporting memories, $w_{c,d} = \text{correct_count}_{c,d} / n_{c,d}$ denotes the skill-level win rate, and $\text{desc}_{c,d}$ provides a natural-language description. Finally, the Skill memory bank $\mathcal{S}$ is defined as:

$$\mathcal{S} = \{s_{c,d} \mid c \in \mathcal{C}, d \in \mathcal{D}\}. \tag{9}$$

3.2 Three-Stage Alternating RL Training

To optimize planning, execution, and experience distillation, APEX adopts a three-stage alternating reinforcement learning framework. The framework contains three trainable modules: the Executor π_θ^{exec}, the Distiller π_ϕ^{dist}, and the Planner π_ψ^{plan}.

Executor Training. In the first stage, the Planner and Distiller are frozen, and the Executor is optimized to perform multi-turn tool interaction under a given plan. Given a query q and a plan p, the Executor generates a trajectory $\tau = \pi_\theta^{\text{exec}}(q, p \mid \mathcal{T})$, where $\mathcal{T}$ denotes the available tool set.

The Executor reward combines answer correctness, tool-use effectiveness, and format validity:

$$\begin{aligned} R^{\text{exec}}(\tau) =& \lambda_1 R_{\text{acc}}(\tau) + \lambda_2 R_{\text{tool}}(\tau) \\ &+ \lambda_3 R_{\text{fmt}}(\tau), \end{aligned} \tag{10}$$

where λ_1, λ_2, and λ_3 are hyperparameters. The accuracy reward $R_{\text{acc}}(\tau) = \mathcal{J}(\text{Extract}(\tau), y^*)$ is computed by an LLM judge. The tool reward $R_{\text{tool}}(\tau) = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \mathbb{I}\left[\exists\, t_{\text{valid}} \in v\right]$ encourages each execution segment to contain valid tool use t_{valid}, where $\mathcal{V}$ denotes the segments split by replan boundaries. The format reward $R_{\text{fmt}}(\tau) \in \{0, 1\}$ checks if the output follows the required structure.

Distiller Training. In the second stage, the Executor and Planner are frozen, and the Distiller is optimized to transform instance-level memories into reusable category-level skills. Given a memory bucket $\mathcal{M}_{c,d}^t$ and an optional previous skill $s_{c,d}^t$, the Distiller outputs an operation type o and an updated skill $s_{c,d}^{t+1}$, i.e., $(o, s_{c,d}^{t+1}) = \pi_\phi^{\text{dist}}(\mathcal{M}_{c,d}^t, s_{c,d}^t)$. The operation o is selected from $\{\text{synthesize}, \text{refine}, \text{create}, \text{skip}\}$. We define the Distiller reward as:

$$\begin{aligned} R^{\text{dist}}(o, s_{c,d}^{t+1}) =& \mu_1 R_{\text{quality}}(s_{c,d}^{t+1}) + \mu_2 R_{\text{fmt}}(s_{c,d}^{t+1}) \\ &+ \mu_3 R_{\text{evolve}}(o, s_{c,d}^{t+1}, s_{c,d}^t), \end{aligned} \tag{11}$$

where μ_1, μ_2, and μ_3 are hyperparameters and R_{evolve} measures if the operation o is rational. Each reward is judged by an LLM judger $\mathcal{J}_{\text{skill}}$.

Planner Training. In the third stage, the Executor and Distiller are frozen, and the Planner is trained through a plan–execute–evaluate–replan loop. Given query q, retrieved skill guidance $\mathcal{R}_S(q)$, and retrieved memories $\mathcal{R}_M(q)$, the Planner first generates an initial plan:

$$p_1 = \pi_\psi^{\text{plan}}\left(q, \mathcal{R}_S(q), \mathcal{R}_M(q)\right). \tag{12}$$

The frozen Executor executes p_1 and produces the first answer $\hat{y}_1$. The Planner then observes $\text{Trace}(\tau)$ and predicts a binary decision $d \in \{0, 1\}$ indicating whether re-planning is needed. If $d = 1$, the Planner generates a revised plan p_2 and obtains a second answer $\hat{y}_2$. If $d = 0$, we set $\hat{y}_2 = \hat{y}_1$. The Planner reward is:

$$\begin{aligned} R^{\text{plan}} =& \alpha_1 R_{\text{acc}}(\hat{y}_2) + \alpha_2 R_{\text{acc}}(\hat{y}_1) \\ &+ \alpha_3 R_{\text{fmt}} + \alpha_4 R_{\text{dec}}, \end{aligned} \tag{13}$$

where α_1, α_2, α_3, and α_4 are hyperparameters. The decision reward R_{dec} encourages the Planner to stop when the first execution is correct and to trigger re-planning when the initial execution fails.

3.3 Skill-guided TTRL

After the training stage, APEX further performs Skill-guided TTRL to adapt the Planner on unlabeled test queries. Since ground-truth answers are unavailable at test time, APEX constructs a no-ground-truth reward with multiple LLM judges. Meanwhile, the learned category-level skills are used as regularization priors to prevent the Planner from drifting away from previously verified procedural knowledge.

No-Ground-Truth Reward. Without ground-truth answers, APEX estimates answer quality through a multi-judge LLM-as-Judge mechanism. Specifically, three judges evaluate the execution from different aspects: $E_1(q, \tau, \hat{y})$ checks reasoning consistency, $E_2(q, \tau, \hat{y})$ checks whether the final answer is supported by retrieved evidence in the trajectory, and $E_3(q, \hat{y})$ checks whether the response is valid and complete. Their judgments are aggregated by an arbiter $\mathcal{A}$:

$$\begin{aligned} R_{\text{nogt}} = & \lambda_n \mathcal{A}\left(E_1(\cdot), E_2(\cdot), E_3(\cdot)\right) \\ & + (1-\lambda_n) R_{\text{fmt}} \end{aligned} \tag{14}$$

The aggregated no-ground-truth reward is also used as a pseudo-supervision signal for memory consolidation.

Skill-Alignment Regularization. Relying only on no-ground-truth rewards may cause unstable online updates. Therefore, APEX uses the retrieved skill as a knowledge anchor. For a category–modality bucket (c, d), the skill $s_{c,d}$ contains procedure $P_{c,d}$ and pitfalls $F_{c,d}$. The alignment reward evaluates whether the generated plan follows the recommended procedure and avoids known pitfalls:

$$R_{\text{align}} = \mathcal{J}_{\text{align}}\left(p, P_{c,d}, F_{c,d}\right). \tag{15}$$

The final reward combines the original no-ground-truth reward and the skill-alignment reward:

$$R_{\text{final}} = (1-\lambda_s) R_{\text{nogt}} + \lambda_s R_{\text{align}}, \tag{16}$$

The weight $\lambda_s = \lambda_{\text{base}} \cdot w_{c,d} \cdot \min\left(\frac{n_{c,d}}{N_{\text{threshold}}}, 1\right)$ is adaptively determined by the confidence of the retrieved skill.

4 Experiment

4.1 Experimental Setups

Training Settings: Our training framework is built on veRL (Sheng et al., 2025) and follows a three-stage alternating optimization paradigm. The Executor employs Qwen2.5-VL-7B (Qwen Team, 2025) as its backbone and is trained on FVQA-train (Wang et al., 2017). The Distiller, based on Qwen3-8B (Yang et al., 2025), is subsequently trained on FVQA-train to distill accumulated execution memories into structured skill documents. The Planner, also based on Qwen3-8B, is trained in the final stage on a mixture of FVQA-train (with images discarded) and MATPO (Mo et al., 2025). The Executor is equipped with offline text-to-text and offline image-to-image search tools. A Qwen3-32B model serves as the LLM Judger for reward computation. More details are provided in Appendix C.

Test Settings: To evaluate APEX, we conduct experiments on a diverse set of image-text and text-only benchmarks. For image-text evaluation, we use FVQA-test, SimpleVQA (Cheng et al., 2025), LiveVQA (Fu et al., 2025), InfoSeek (Chen et al., 2023), and MMSearch (Jiang et al., 2024). For text-only evaluation, we use HotpotQA (Yang et al., 2018) and 2WikiMultiHopQA (Ho et al., 2020). Detailed settings are provided in Appendix D.

Baselines. We compare APEX against a wide range of deep research methods spanning three categories: ❶ ***Naive-Large Language Models***: GPT-5.4 (OpenAI, 2026), Gemini-3-Flash (Google DeepMind, 2025), Qwen2.5-VL-7B (Qwen Team, 2025), and Qwen2.5-VL-32B (Qwen Team, 2025), which are directly prompted to answer questions without agentic workflow; ❷ ***Search Agents***: Qwen2.5-VL-7B+ReAct (Yao et al., 2022), Qwen2.5-VL-32B+ReAct, MMSearch-R1 (Wu et al., 2025b), and Deepeyes2 (Zheng et al., 2026), which augment LLMs with external search tools via multi-step reasoning but maintain no cross-task memory; and ❸ ***Memory-Augmented Search Agents***: RAG, Mem0 (Chhikara et al., 2025), A-Mem (Xu et al., 2026), ExpeL (Zhao et al., 2024), Memento (Zhou et al., 2025), and MIA (Qiao et al., 2026), which additionally incorporate memory mechanisms to store and reuse past experience across tasks. More details on baseline settings are provided in Appendix F.

4.2 Main Results

As shown in Table 1, APEX achieves the best average performance across 7 benchmarks, with an average score of 64.4. We highlight the following key observations:

❶ **Substantial improvement over standalone**

Table 1: **Main results across in-domain and out-of-domain benchmarks**. The best results are highlighted in **bold**, and the second-best results are underlined.

Method	In-Domain	Out-of-Domain						Avg.
	FVQA-test	HotpotQA	2Wiki	SimpleVQA	LiveVQA	InfoSeek	MMSearch	
Naive-Large Language Model								
GPT-5.4	51.3	60.2	64.9	55.9	22.8	45.6	47.4	49.7
Gemini-3-Flash	68.4	66.2	68.4	**72.3**	25.7	66.5	**67.9**	62.2
Qwen2.5-VL-7B	20.7	12.6	20.5	30.2	8.3	23.6	7.2	17.6
Qwen2.5-VL-32B	24.3	16.9	24.0	39.8	18.5	25.8	15.7	23.6
Search Agent								
Qwen2.5-VL-7B+ReAct	34.2	23.7	31.2	35.4	10.6	28.1	21.3	26.4
Qwen2.5-VL-32B+ReAct	50.9	31.4	37.0	48.3	24.7	38.1	27.5	36.8
MMSearch-R1	58.3	39.2	48.3	55.2	28.6	49.5	43.3	46.1
Deepeyes2	60.6	-	-	59.4	-	51.1	63.7	-
Memory-Augmented Search Agent								
RAG	58.6	47.8	56.2	60.7	31.9	55.9	54.3	52.2
Mem0	54.9	49.3	54.9	56.9	24.3	48.2	43.1	47.4
A-Mem	38.5	46.8	56.2	51.6	22.8	36.1	41.1	41.9
ExpeL	64.2	56.3	62.9	62.5	34.1	58.8	61.3	57.2
Memento	66.1	55.2	64.3	61.7	36.7	57.3	61.4	57.5
MIA	65.3	64.0	71.6	63.8	39.7	64.7	60.6	61.4
Ours								
APEX	**68.7**	**67.8**	**75.2**	66.3	**42.4**	**66.9**	63.8	**64.4**

LLMs. APEX improves by 14.7 points compared to GPT-5.4 and slightly surpasses Gemini-3-Flash in average accuracy. Notably, APEX outperforms its own backbone model Qwen2.5-VL-7B by 46.8 points in absolute terms, demonstrating that structured skill knowledge and memory-augmented planning can effectively compensate for model scale, enabling a small open-source model to rival closed-source counterparts.

❷ **Consistent gains over search agents.** Equipping LLMs with search tools through ReAct substantially improves performance (e.g., Qwen2.5-VL-7B: 17.6 → 26.4). Compared with the strong search agent MMSearch-R1, APEX achieves an improvement of 18.3 points in average accuracy. This gap indicates that tool access alone is insufficient for complex research tasks — effective utilization of accumulated experience through skill distillation is critical for guiding multi-step tool orchestration.

❸ **Superiority over memory-augmented agents.** Among memory-augmented baselines, MIA achieves the strongest performance (61.4 avg) by combining a Manager-Planner-Executor architecture with instance-level memory retrieval. APEX further improves upon MIA by 3.0 points on average. This improvement is largely associated with the Distiller module, which distills instance-level memories into category-level skills that provide structured procedural guidance to the Planner, rather than relying solely on raw memory retrieval at each decision point.

4.3 Skill Sufficiency Analysis

We evaluate APEX (Skill Only), which removes instance-level memory retrieval from the Planner context while retaining trajectories for skill evolution. As shown in Table 2, this variant incurs an average drop of 1.8 points compared to the full pipeline, yet outperforms the strongest baseline MIA by 1.5 points. This confirms that the Distiller successfully distills category-level knowledge from raw trajectories into structured skills that are dynamically modified at test time, reducing the Planner's reliance on verbose instance-level memories.

Table 2: **Skill sufficiency analysis.** Avg. is computed over the four datasets shown in this table. Best in **bold**, second-best underlined.

Method	Image-Text		Text-Only		Avg.
	FVQA	SimpleVQA	HotpotQA	2Wiki	
Baseline					
Mem0	54.9	56.9	49.3	54.9	54.0
MMSearch-R1	58.3	55.2	39.2	48.3	50.3
ExpeL	64.2	62.5	56.3	62.9	61.5
Memento	66.1	61.7	55.2	64.3	61.8
MIA	65.3	63.8	64.0	71.6	66.2
Ours					
APEX (Skill Only)	67.4	64.8	65.7	72.9	67.7
APEX	**68.7**	**66.3**	**67.8**	**75.2**	**69.5**

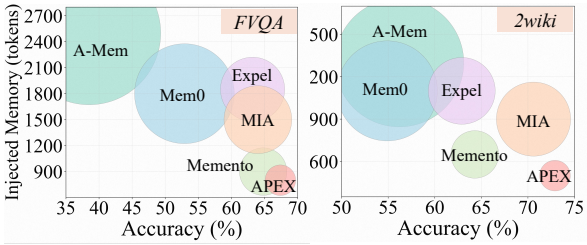

Figure 3: Token-efficiency comparison on FVQA-test and 2Wiki. X-axis denotes task accuracy. Y-axis denotes the number of memory tokens injected at each reasoning step. The bubble size represents the overall token cost of a full inference process.

4.4 Context Efficiency Analysis

Figure 3 compares different memory-augmented reasoning methods in terms of accuracy, injected memory tokens per inference step, and total token consumption. Across both FVQA-test and 2Wiki, APEX (Skill only) achieves favorable context-efficiency trade-off. Compared with MIA, APEX improves accuracy by 1.7 points. Meanwhile, it reduces injected memory tokens by 45.6% and total token consumption by 36.8% on average. These results demonstrate that APEX achieves better accuracy with a smaller memory footprint and lower token cost, highlighting its advantage in context-efficient memory utilization for reasoning.

4.5 Training Analysis

In Figure 5, we analyze the GRPO training dynamics of each module to characterize their convergence behavior and identify the factors underlying their distinct learning patterns. All modules are trained alternately with only one module updated at a time. The **Executor** exhibits the most stable improvement, with its reward increasing steadily

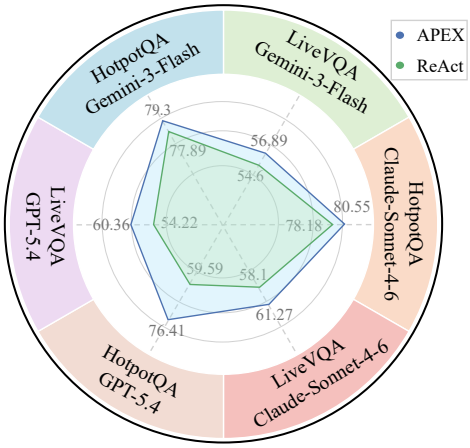

Figure 4: Performance comparison between ReAct and APEX using different closed-source executor models on HotpotQA and LiveVQA.

and converging to a high level. The **Distiller** also converges smoothly and rapidly, suggesting that the skill distillation objective provides a relatively well-defined optimization signal. In contrast, the **Planner** shows larger fluctuations, although its reward still follows an overall upward trend. This is expected since planning is evaluated indirectly through downstream execution and is therefore more sensitive to variations in retrieved memories, generated skills, and executor behavior.

4.6 Generalization Analysis

To assess the generalization ability of APEX, we replace the originally trained open-source executor model with stronger closed-source executor models without any additional training, and compare it with ReAct on HotpotQA and LiveVQA. As shown in Figure 4, APEX consistently outperforms ReAct across all closed-source executors, achieving an average relative improvement of 11.0% on HotpotQA and 7.0% on LiveVQA. This demonstrates that APEX is not tied to a specific trained executor, but can generalize to unseen closed-source executors in a plug-and-play manner.

4.7 Ablation Study

We conduct three groups of ablation studies to evaluate each component in APEX.

❶ **Experience Utilization.** We remove different experience sources during inference. *w/o Mem.* disables the retrieval of concrete execution memories, *w/o Skill* removes distilled procedural skills, and *w/o Both* removes both. These variants reduce the average accuracy by 1.8, 3.1, and 10.1 points, re-

spectively. This suggests that category-level skills provide general templates for similar tasks, while instance-level memories offer concrete problem-solving examples; their complementarity enables more effective utilization of past experience.

Table 3: **Ablation study of APEx**. Accuracy is reported on image-text and text-only tasks. Performance drops (↓) relative to the full model are shown in subscripts.

Method	Image-Text		Text-Only		Avg.
	FVQA	SimpleVQA	HotpotQA	2Wiki	
APEx	**68.7**	**66.3**	**67.8**	**75.2**	**69.5**
w/o Experience Utilization					
w/o Mem.	67.4 ↓1.3	64.8 ↓1.5	65.7 ↓2.1	72.9 ↓2.3	67.7 ↓1.8
w/o Skill	64.9 ↓3.8	62.7 ↓3.6	65.6 ↓2.2	72.3 ↓2.9	66.4 ↓3.1
w/o Both	58.6 ↓10.1	55.0 ↓11.3	58.4 ↓9.4	65.5 ↓9.7	59.4 ↓10.1
w/o Post-training Optimization					
w/o Dis.	66.4 ↓2.3	63.7 ↓2.6	65.8 ↓2.0	72.7 ↓2.5	67.2 ↓2.3
w/o Plan.	66.8 ↓1.9	63.9 ↓2.4	66.2 ↓1.6	74.0 ↓1.2	67.7 ↓1.8
w/o Exec.	61.0 ↓7.7	58.0 ↓8.3	58.2 ↓9.6	67.2 ↓8.0	61.1 ↓8.4
w/o Closed-loop Adaptation					
w/o TTRL	64.6 ↓4.1	60.8 ↓5.5	63.5 ↓4.3	72.4 ↓2.8	65.3 ↓4.2
w/o Iter.	66.0 ↓2.7	64.1 ↓2.2	64.7 ↓3.1	73.3 ↓1.9	67.0 ↓2.5

❷ **Post-training Optimization.** We evaluate module-wise GRPO post-training. For each ablation, the pipeline structure is kept unchanged, while the corresponding GRPO-trained module is replaced with its non-trained base model, such as Qwen2.5-VL-7B or Qwen3-8B. Removing GRPO from the Distiller, Planner, and Executor reduces the average accuracy by 2.3, 1.8, and 8.4 points respectively. The larger degradation of *w/o Exec.* indicates that executor optimization is particularly important for reliable multi-step tool interaction, while optimizing the Distiller and Planner also benefits skill construction and strategy generation.

❸ **Closed-loop Adaptation.** Finally, we analyze the closed-loop adaptation mechanism. *w/o TTRL* performs inference without test-time reinforcement learning, while *w/o Iter.* distills skills only once without further updates from improved trajectories. These variants reduce the average accuracy by 4.2 and 2.5 points, respectively. Together, TTRL and iterative skill distillation form a closed-loop feedback mechanism at test time, where improved trajectories are distilled into better skills, thereby progressively enhancing model performance.

5 Related Work

5.1 Deep Research Agents

Deep Research Agents (DRAs) are designed to answer complex, multi-turn information-seeking questions by combining dynamic reasoning, adaptive tool-mediated search, and compact experience retrieval (Huang et al., 2025; Zhang et al., 2025b; Xu and Peng, 2025). Recent studies have improved text-only and multimodal long-horizon search by curating reasoning trajectories (Sun et al., 2025; Li et al., 2026b; Zhu et al., 2026), supervised fine-tuning (Li et al., 2025; Wu et al., 2025a) or agentic reinforcement learning (Zheng et al., 2025; Jin et al., 2025; Wu et al., 2025b; Narayan et al., 2025) to internalize planning and searching. To reuse prior interactions, emerging work abstracts past trajectories into reusable memories, enabling agents to dynamically organize memory, maintain compact interaction states, and optimize memory-aware search or planning policies across similar long-horizon tasks (Zhao et al., 2024; Xu et al., 2026; Zhou et al., 2026; Qiao et al., 2026). Our work focuses on constructing instance-level memories, organizing category-level skills, retrieving and ranking reusable experiences, and distilling them into Distiller-Planner-Executor training.

5.2 Memory Systems for Agents

Recent memory systems for LLM agents have evolved from retrieval-centric external memory to adaptive experience management. Early RAG methods treat external documents as non-parametric memory to ground generation with retrieved evidence (Lewis et al., 2020; Zhong et al., 2024; Edge et al., 2024). Building on these foundations, recent studies shift the focus from simply retrieving past information to organizing, compressing, and updating agent memories (Fang et al., 2026; Liu et al., 2026). More recent work further treats memory as a trainable and reusable experience space where agents learn what to store (Kang et al., 2026; Shen et al., 2026), when to retrieve or discard (Yuan et al., 2025), and how to abstract trajectories into procedural guidance or skills (Zhang et al., 2025a, 2026; Xia et al., 2026). These trends indicate a transition from passive knowledge retrieval toward self-evolving memory systems that support long-horizon planning, experience reuse, and strategy generalization. In contrast, APEx distills trajectories within each category–modality group into reusable procedural skills and uses their

confidence-weighted alignment as a regularization signal for online Planner updates, coupling procedural skill abstraction with downstream parametric adaptation in a closed experience-to-action loop.

6 Conclusion

In this paper, we present APEX, a hierarchical experience utilization framework for deep research agents. APEX organizes accumulated interactions into instance-level memories and category-level skills, enabling reusable procedural knowledge across tasks. Through GRPO-based alternating optimization of the Executor, Distiller, and Planner, APEX forms a closed-loop process spanning execution, skill abstraction, and adaptive planning. At test time, distilled skills serve as procedural priors for skill-guided TTRL, supporting ground-truth-free online adaptation while reducing drift through skill-alignment regularization. Experiments on 7 benchmarks show that APEX achieves state-of-the-art performance. Future work will extend APEX to broader tool ecosystems and explore more efficient skill evolution and cross-agent skill transfer.

Limitations

While APEX achieves the best average performance across seven benchmarks and demonstrates strong generalization to out-of-domain settings, our current framework is built upon small open-source models. Although APEX enables a 7B model to surpass closed-source systems, we have not explored how APEX scales when applied to more powerful backbone models. Additionally, our three-stage alternating GRPO training paradigm, despite its effectiveness in stabilizing cross-module credit assignment and producing consistently improving rewards, requires sequential optimization of the Executor, Distiller, and Planner, which may not fully exploit the synergies achievable through end-to-end joint training. Finally, our skill-guided TTRL mechanism currently operates online at test time without ground-truth supervision, and while it already yields significant improvements through self-improvement, incorporating minimal supervision could further amplify the quality of distilled skills and accelerate the closed-loop adaptation process.

Ethical Considerations

Our framework builds upon large language and vision-language models that may produce inaccurate, biased, or harmful outputs due to biases inherent in their pre-training corpora; such behaviors are not by design. Since APEX retrieves and processes open-web content at inference time, retrieved results may inadvertently contain personal identifiable information (PII) or offensive material, and we advise deployers to apply appropriate content filtering. The open-source Qwen models used for training are released under their respective licenses. Proprietary models used only for baseline evaluation or executor generalization are accessed through their APIs and are subject to their providers' terms of use. All evaluation benchmarks are publicly available with documented provenance. Any code or weights we release will be accompanied by usage guidelines that explicitly prohibit malicious applications such as misinformation generation or unauthorized surveillance. We report training procedures, hyperparameters, and computational costs to ensure reproducibility and transparency. Finally, AI assistants were employed for grammatical error correction and rephrasing.

References

Yang Chen, Hexiang Hu, Yi Luan, Haitian Sun, Soravit Changpinyo, Alan Ritter, and Ming-Wei Chang. 2023. Can pre-trained vision and language models answer visual information-seeking questions? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14948–14968.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Jiangning Liu, Wenwei Zhang, Kai Chen, and Feng Zhao. 2025. Mindsearch: Mimicking human minds elicits deep ai searcher. In *International Conference On Learning Representations*, volume 2025, pages 90007–90029.

Xianfu Cheng, Wei Zhang, Shiwei Zhang, Jian Yang, Xiangyuan Guan, Xianjie Wu, Xiang Li, Ge Zhang, Jiaheng Liu, Yuying Mai, Yutao Zeng, Zhoufutu Wen, Ke Jin, Baorui Wang, Weixiao Zhou, Yunhong Lu, Tongliang Li, Wenhao Huang, and Zhoujun Li. 2025. Simplevqa: Multimodal factuality evaluation for multimodal large language models. *Preprint*, arXiv:2502.13059.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao,

Mengru Wang, Shuofei Qiao, Huajun Chen, and Ningyu Zhang. 2026. Lightmem: Lightweight and efficient memory-augmented generation. In *International Conference on Learning Representations*, volume 2026, pages 98706–98729.

Mingyang Fu, Yuyang Peng, Benlin Liu, Yao Wan, and Dongping Chen. 2025. Livevqa: Live visual knowledge seeking. *arXiv e-prints*, pages arXiv–2504.

Google DeepMind. 2025. Gemini 3 flash model card. https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Flash-Model-Card.pdf. Model card.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625.

Yuxuan Huang, Yihang Chen, Haozheng Zhang, Kang Li, Huichi Zhou, Meng Fang, Linyi Yang, Xiaoguang Li, Lifeng Shang, Songcen Xu, Jianye Hao, Kun Shao, and Jun Wang. 2025. Deep research agents: A systematic examination and roadmap. *Preprint*, arXiv:2506.18096.

Dongzhi Jiang, Renrui Zhang, Ziyu Guo, Yanmin Wu, Jiayi Lei, Pengshuo Qiu, Pan Lu, Zehui Chen, Chaoyou Fu, Guanglu Song, Peng Gao, Yu Liu, Chunyuan Li, and Hongsheng Li. 2024. Mmsearch: Benchmarking the potential of large models as multimodal search engines. *Preprint*, arXiv:2409.12959.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.

Jingyi Kang, Chunyu Li, Ding Chen, Bo Tang, Feiyu Xiong, and Zhiyu Li. 2026. Memreader: From passive to active extraction for long-term agent memory. *arXiv preprint arXiv:2604.07877*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pages 6769–6781.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, Weizhou Shen, Junkai Zhang, Dingchu Zhang, Xixi Wu, Yong Jiang, Ming Yan, Pengjun Xie, Fei Huang, and Jingren Zhou. 2025. Websailor: Navigating super-human reasoning for web agent. *Preprint*, arXiv:2507.02592.

Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yongkang Wu, Ji-Rong Wen, Yutao Zhu, and Zhicheng Dou. 2026a. Webthinker: Empowering large reasoning models with deep research capability. *Advances in Neural Information Processing Systems*, 38:120091–120131.

Zhuofeng Li, Dongfu Jiang, Xueguang Ma, Haoxiang Zhang, Ping Nie, Yuyu Zhang, Kai Zou, Jianwen Xie, Yu Zhang, and Wenhu Chen. 2026b. Open-researcher: A fully open pipeline for long-horizon deep research trajectory synthesis. *arXiv preprint arXiv:2603.20278*.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. 2026. Simplemem: Efficient lifelong memory for llm agents. *arXiv preprint arXiv:2601.02553*.

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

Zhanfeng Mo, Xingxuan Li, Yuntao Chen, and Lidong Bing. 2025. Multi-agent tool-integrated policy optimization. *arXiv preprint arXiv:2510.04678*.

Kartik Narayan, Yang Xu, Tian Cao, Kavya Nerella, Vishal M Patel, Navid Shiee, Peter Grasch, Chao Jia, Yinfei Yang, and Zhe Gan. 2025. Deepmmsearch-r1: Empowering multimodal llms in multimodal web search. *arXiv preprint arXiv:2510.12801*.

OpenAI. 2026. GPT-5.4 Thinking System Card. https://openai.com/index/gpt-5-4-thinking-system-card/. Published March 5, 2026.

Jingyang Qiao, Weicheng Meng, Yu Cheng, Zhihang Lin, Zhizhong Zhang, Xin Tan, Jingyu Gong, Kun Shao, and Yuan Xie. 2026. Memory intelligence agent. *arXiv preprint arXiv:2604.04503*.

Qwen Team. 2025. Qwen2.5-vl.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Zhiyu Shen, Ziming Wu, Fuming Lai, Shaobing Lian, and Yanghui Rao. 2026. Membuilder: Reinforcing llms for long-term memory construction via attributed dense rewards. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27868–27887.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pages 1279–1297.

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

Rui Sun, Jie Ding, Chenghua Gong, Tianjun Gu, Yihang Jiang, Juyuan Zhang, Liming Pan, and Linyuan Lü. 2026. Topodim: One-shot topology generation of diverse interaction modes for multi-agent systems. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 4252–4269.

Shuang Sun, Huatong Song, Yuhao Wang, Ruiyang Ren, Jinhao Jiang, Junjie Zhang, Fei Bai, Jia Deng, Wayne Xin Zhao, Zheng Liu, Lei Fang, Zhongyuan Wang, and Ji-Rong Wen. 2025. SimpleDeepSearcher: Deep information seeking via web-powered reasoning trajectory synthesis. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 13705–13720, Suzhou, China. Association for Computational Linguistics.

Zhen Tan, Jun Yan, I-Hung Hsu, Rujun Han, Zifeng Wang, Long Le, Yiwen Song, Yanfei Chen, Hamid Palangi, George Lee, Anand Rajan Iyer, Tianlong Chen, Huan Liu, Chen-Yu Lee, and Tomas Pfister. 2025. In prospect and retrospect: Reflective memory management for long-term personalized dialogue agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8416–8439, Vienna, Austria. Association for Computational Linguistics.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Peng Wang, Qi Wu, Chunhua Shen, Anthony Dick, and Anton Van Den Hengel. 2017. Fvqa: Fact-based visual question answering. *IEEE transactions on pattern analysis and machine intelligence*, 40(10):2413–2427.

Yu Wang and Xi Chen. 2025. Mirix: Multi-agent memory system for llm-based agents. *arXiv preprint arXiv:2507.07957*.

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

Jialong Wu, Baixuan Li, Runnan Fang, Wenbiao Yin, Liwen Zhang, Zhenglin Wang, Zhengwei Tao, Ding-Chu Zhang, Zekun Xi, Robert Tang, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. 2025a. Webdancer: Towards autonomous information seeking agency. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 120957–120985. Curran Associates, Inc.

Jinming Wu, Zihao Deng, Wei Li, Yiding Liu, Bo You, Bo Li, Zejun Ma, and Ziwei Liu. 2025b. Mmsearch-r1: Incentivizing lmms to search. *arXiv preprint arXiv:2506.20670*, 2(4).

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. 2026. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *Preprint*, arXiv:2602.08234.

Renjun Xu and Jingwen Peng. 2025. A comprehensive survey of deep research: Systems, methodologies, and applications. *arXiv preprint arXiv:2506.12594*.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2026. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems*, 38:17577–17604.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Qianhao Yuan, Jie Lou, Zichao Li, Jiawei Chen, Yaojie Lu, Hongyu Lin, Le Sun, Debing Zhang, and Xianpei Han. 2025. Memsearcher: Training llms to reason, search and manage memory via end-to-end reinforcement learning. *arXiv preprint arXiv:2511.02805*.

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

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. 2026. Memskill: Learning and evolving memory skills for self-evolving agents. *arXiv preprint arXiv:2602.02474*.

Wenlin Zhang, Xiaopeng Li, Yingyi Zhang, Pengyue Jia, Yichao Wang, Huifeng Guo, Yong Liu, and Xiangyu Zhao. 2025b. Deep research: A survey of autonomous research agents. *arXiv preprint arXiv:2508.12752*.

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

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. 2025. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 414–431.

Ziwei Zheng, Minghao Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, Chao Shen, and XingYu. 2026. Deepeyes: Incentivizing "thinking with images" via reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pages 126775–126798.

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

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

Huichi Zhou, Siyuan Guo, Anjie Liu, Zhongwei Yu, Ziqin Gong, Bowen Zhao, Zhixun Chen, Menglong Zhang, Yihang Chen, Jinsong Li, Runyu Yang, Qiangbin Liu, Xinlei Yu, Jianmin Zhou, Na Wang, Chunyang Sun, and Jun Wang. 2026. Memento-skills: Let agents design agents. *Preprint*, arXiv:2603.18743.

Bin Zhu, Qianghuai Jia, Tian Lan, Junyang Ren, Feng Gu, Feihu Jiang, Longyue Wang, Zhao Xu, and Weihua Luo. 2026. Marco deepresearch: Unlocking efficient deep research agents via verification-centric design. *arXiv preprint arXiv:2603.28376*.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, yuchen zhang, Xinwei Long, Ermo Hua, Biqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen Zhou. 2025. Ttrl: Test-time reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 131459–131483. Curran Associates, Inc.

A Memory and Skill Retrieval

For each candidate memory $m_i \in \mathcal{M}(q)$, APEX computes a dual-channel similarity score. The first channel compares the current query with the stored query, while the second compares caption-level or contextual representations:

$$\begin{aligned}\mathrm{sim}(q, m_i) = \beta \cos\big(\phi_q(q), \phi_q(q_i)\big) \\ + (1-\beta)\cos\big(\phi_c(q), \phi_c(m_i)\big),\end{aligned} \tag{17}$$

where $\phi_q(\cdot)$ denotes the question embedding function, $\phi_c(\cdot)$ denotes the caption or contextual embedding function. When no caption is available, the similarity degenerates to the question-only similarity. The similarity score $\mathrm{sim}(q, m_i)$ is min–max normalized within the candidate bucket and denoted as $\widetilde{\mathrm{sim}}(q, m_i)$. The final retrieval score combines normalized semantic similarity with the historical reuse win rate:

$$f(q, m_i) = \alpha\, \widetilde{\mathrm{sim}}(q, m_i) + (1-\alpha)\, w_i, \tag{18}$$

APEX retrieves top-k successful and failed memories separately according to the retrieval score:

$$\mathcal{R}_M(q) = \begin{cases} \mathrm{TopK}_{m_i \in \mathcal{M}^+(q)}\, f(q, m_i), \\ \mathrm{TopK}_{m_i \in \mathcal{M}^-(q)}\, f(q, m_i). \end{cases} \tag{19}$$

In terms of category-level memory, each skill $s_{c,d}$ is assigned a confidence score that combines win rate with evidence sufficiency:

$$\gamma_{c,d} = w_{c,d} \cdot \min\left(\frac{n_{c,d}}{N_{\text{threshold}}}, 1\right), \tag{20}$$

where $N_{\text{threshold}}$ is the evidence threshold. A skill is injected into the Planner only when its confidence exceeds a minimum threshold $\mathcal{R}_S(q) = \mathrm{Inject}(s_{\hat{c},\hat{d}})$, where

$$\mathrm{Inject}(s_{c,d}) = \begin{cases} s_{c,d}, & \gamma_{c,d} > \epsilon_s, \\ \varnothing, & \gamma_{c,d} \le \epsilon_s, \end{cases} \tag{21}$$

with $\epsilon_s = 0.1$ and $N_{\text{threshold}} = 10$.

Finally, APEX combines category-level skills and instance-level memories into a structured planner context:

$$E(q) = \mathcal{R}_S(q) \oplus \mathcal{R}_M(q), \tag{22}$$

where $\oplus$ denotes structured prompt concatenation rather than set union. The Planner then generates a research plan conditioned on it:

$$p = \pi_\psi^{\text{plan}}(q, E(q)), \tag{23}$$

which enables the Planner to combine case-based guidance with abstract strategies for new research tasks.

B Supplement of TTRL

Skill-Gated Update. For highly reliable skills, APEX optionally skips test-time updates to protect already learned knowledge. A skill is considered reliable if it has high confidence, sufficient historical support, and an available procedural prior. When the gate is activated, all rollouts receive the same reward, equal to the skill confidence. Under GRPO, this produces zero group-normalized advantages ($\hat{A}_i = 0$), so no effective parameter update is performed. This avoids unnecessary computation and prevents reliable skills from being degraded by noisy test-time rewards.

Closed-loop Adaptation. Skill-guided TTRL creates a positive adaptation loop in which skills and the Planner improve each other. Skills guide the Planner to produce better plans, which lead the Executor to generate higher-quality trajectories. These trajectories update the Planner through GRPO and are further consolidated into memories to distill stronger skills for future queries.

After each batch, selected successful and failed trajectories are consolidated into the memory bank:

$$\mathcal{M}^{t+1} = \mathcal{M}^t \cup \text{Consolidate}(q, \tau, \hat{y}). \tag{24}$$

The Distiller then uses recent memories to synthesize or selectively update the corresponding skill:

$$s_{c,d}^{t+1} = \phi_{\text{skill}}\left(\mathcal{M}_{c,d}^{t+1}, s_{c,d}^t\right). \tag{25}$$

During online TTRL, only the Planner parameters are updated. Memory consolidation and skill evolution occur after each batch to inform subsequent queries.

C Training Setups.

This appendix provides comprehensive training configurations for the three-stage alternating GRPO optimization (Section 3.2) and the Skill-guided TTRL (Section 3.3). All modules are trained using the veRL framework with GRPO. We use offline search tool during training. For text-to-text retrieval, we use an offline retrieval service based on E5-base-v2 (Wang et al., 2022) over a FAISS-indexed wiki25 (Karpukhin et al., 2020) corpus. For image-to-image retrieval, we use a pre-built local cache of image search results collected via Serper by prior work (Qiao et al., 2026).

General Training Infrastructure All experiments are conducted on a single node equipped with 8 NVIDIA A100 GPUs. We employ Fully Sharded Data Parallel (FSDP) with both parameter and optimizer offloading to enable training of 7B/8B models within GPU memory constraints. The rollout engine uses asynchronous generation with chunked prefill enabled. A Qwen3-32B model served via vLLM (Kwon et al., 2023) with tensor parallelism of 2 acts as the LLM Judge $\mathcal{J}$ for reward computation across all stages.

Stage 1: Executor Training The Executor π_θ^{exec} is initialized from Qwen2.5-VL-7B and trained on FVQA-train with multi-turn tool interactions. Table 4 summarizes the hyperparameters. For the reward of executor, we set $\lambda_1 = 0.7$, $\lambda_2 = 0.2$, and $\lambda_3 = 0.1$, respectively.

Table 4: Executor GRPO training hyperparameters.

Hyperparameter	Value
Base model	Qwen2.5-VL-7B
Training data	FVQA-train
Learning rate	1×10^{-6}
Batch size	128
Group size G (rollouts per query)	8
Max prompt length	16,384 tokens
Max response length	16,384 tokens
Max tool response length	4,096 tokens
Max assistant turns	10
Sampling temperature	1.0
KL coefficient β	0.0
Entropy coefficient	0.0
PPO clip range ϵ	0.2
Training epochs	8
Number of GPUs	8
Optimizer	AdamW

Stage 2: Distiller Training As shown in Table 5, the Distiller π_ϕ^{dist} is initialized from Qwen3-8B and trained to transform instance-level memories into

structured category-level skill documents. Unlike the Executor and Planner, the Distiller does not perform multi-turn tool interaction; it generates skill documents in a single turn. For Distiller reward, we set $\mu_1 = 0.5$, $\mu_2 = 0.1$, $\mu_3 = 0.4$.

Table 5: Distiller GRPO training hyperparameters.

Hyperparameter	Value
Base model	Qwen3-8B
Training data	FVQA-train
Learning rate	1×10^{-6}
Batch size	64
Group size G (rollouts per query)	8
Max prompt length	4,096 tokens
Max response length	2,048 tokens
Multi-turn interaction	Disabled
Sampling temperature	1.0
KL coefficient β	0.0
Entropy coefficient	0.0
PPO clip range ϵ	0.2
Training epochs	8
Number of GPUs	8
Optimizer	AdamW

The quality reward R_{quality} is evaluated by the LLM Judge $\mathcal{J}_{\text{skill}}$, assessing whether the generated skill contains well-structured procedural steps and meaningful failure modes. The evolution reward R_{evolve} checks whether the selected operation (synthesize, refine, create, or skip) is rational given the current memory state.

Stage 3: Planner Training The Planner π_ψ^{plan} is initialized from Qwen3-8B and trained on a mixture of FVQA-train (with images discarded) and MATPO data through a plan–execute–evaluate–replan loop, as shown in Table 6. For the Planner reward, we set $\alpha_1 = 0.7$, $\alpha_2 = 0.2$, $\alpha_3 = 0.05$ and $\alpha_4 = 0.05$. When the Planner decides not to replan ($d = 0$), we set $\hat{y}_2 = \hat{y}_1$.

D Test Setups.

At test time, the Planner is further adapted via Skill-guided TTRL on unlabeled test queries. Table 7 lists the TTRL configuration. For text-to-text search, we use wiki25 for HotpotQA, 2WikiMultiHopQA, and SimpleVQA, while using Serper for other benchmarks. For image-to-image search, we use Serper for all multimodal benchmarks.

TTRL Reward Formulation. The final reward for Skill-guided TTRL (Eq. 16) combines the no-ground-truth reward and skill-alignment reward:

$$R_{\text{final}} = (1 - \lambda_s) \cdot R_{\text{nogt}} + \lambda_s \cdot R_{\text{align}}, \quad (26)$$

Table 6: Planner GRPO training hyperparameters.

Hyperparameter	Value
Base model	Qwen3-8B
Training data	FVQA-train, MATPO
Learning rate	1×10^{-6}
Batch size	48
Group size G (rollouts per query)	4
Max prompt length	24,576 tokens
Max response length	8,192 tokens
Max tool response length	8,192 tokens
Max assistant turns	10
Sampling temperature	1.0
KL coefficient β	0.0
Entropy coefficient	0.0
PPO clip range ϵ	0.2
Training epochs	5
Number of GPUs	8
Optimizer	AdamW

Table 7: Skill-guided TTRL hyperparameters.

Hyperparameter	Value
Adapted module	Planner (π_ψ^{plan})
Learning rate	2×10^{-7}
Batch size	8
Group size G (rollouts per query)	4
Max prompt length	24,576 tokens
Max response length	8,192 tokens
Max tool response length	8,192 tokens
Max assistant turns	10
Max LLM calls per run	20
Sampling temperature	1.0
KL coefficient β	0.0
Entropy coefficient	0.0
PPO clip range ϵ	0.2
Number of GPUs	8
Epochs (over test set)	1
Optimizer	AdamW

where $R_{\text{nogt}} = 0.9 \cdot \mathcal{A} + 0.1 \cdot R_{\text{fmt}}$. The adaptive weight λ_s is computed as:

$$\lambda_s = \lambda_{\text{base}} \cdot w_{c,d} \cdot \min\left(\frac{n_{c,d}}{N_{\text{threshold}}}, 1\right), \quad (27)$$

with $\lambda_{\text{base}} = 0.1$ and $N_{\text{threshold}} = 10$. This ensures that skills with higher win rate $w_{c,d}$ and more supporting evidence $n_{c,d}$ exert stronger regularization, while low-confidence or newly created skills have minimal influence on the policy update.

E Dataset Description

We conduct comprehensive evaluation across seven benchmark datasets spanning both multimodal and text-only scenarios. Table 8 provides an overview of all evaluation datasets.

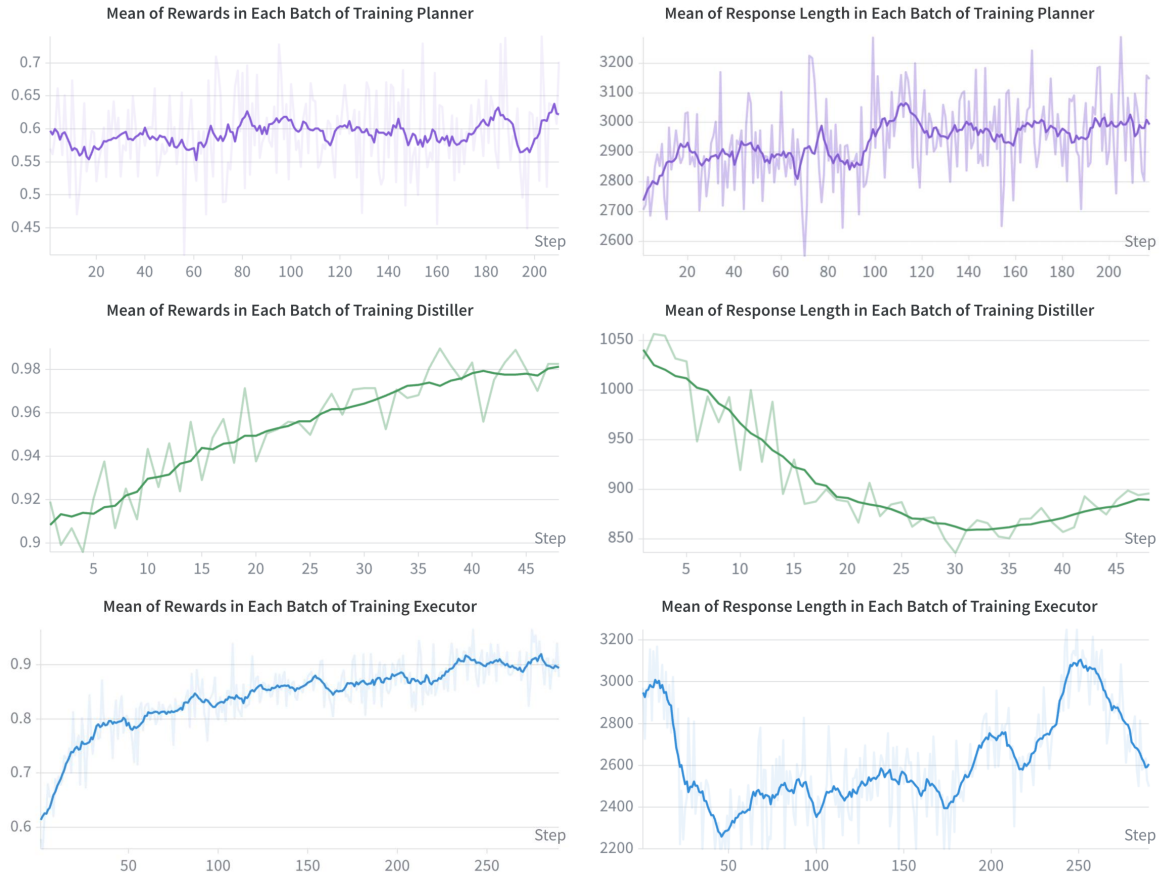

Figure 5: The left column reports the mean reward of each module across training batches, while the right column shows the corresponding mean response length.

Table 8: Summary of evaluation datasets used in our experiments.

Dataset	Modality	# Examples	Source
FVQA-test	Image-text	1,800	MMSearch-R1
InfoSeek	Image-text	2,000	MMSearch-R1
SimpleVQA	Image-text	1,013	MMSearch-R1
LiveVQA	Image-text	2,384	Public Version
MMSearch	Image-text	171	MMSearch-R1
HotpotQA	Text-only	7,405	Public Version
2Wiki	Text-only	12,576	Public Version

F Baselines Settings

We compare against a broad spectrum of baselines organized into three categories: direct answering, search-augmented agents, and memory-based search agents.

Direct Answer. In this setting, models are prompted to produce concise answers relying solely on their internal parametric knowledge, without accessing any external tools or retrieval mechanisms.

Search Agent. These baselines perform multi-turn tool calling under the ReAct paradigm (Yao et al., 2022), iteratively querying external search tools and reasoning over retrieved results. The results of Deepeyes2 are cited from their respective technical reports.

Memory-based Search Agent. To ensure a fair comparison among memory-augmented approaches, we train Executor variants sharing identical training configurations but differing in how memory is incorporated. The Executor is trained under either workflow memory prompt or plan prompt.

G Training Analysis

As discussed in Section 4, we further visualize the training process in Figure 5. In each subplot, the light-colored curve denotes the raw training records, while the dark-colored curve denotes the batch-averaged smoothed trajectory.

H Prompt Details

This appendix presents the complete prompts. Each module operates with a dedicated prompt that governs its behavior within the multi-agent pipeline:

Executor Prompt. The Executor receives the following prompt to guide its multi-turn tool-calling behavior. Below is the text-only variant; the image variant appends `<image>` after the question.

Planner Prompt. The Planner generates strategic action plans conditioned on retrieved skills and memories. Below are the system prompt, planning prompt (text-only variant), and replanning prompt.

Distiller Prompt. The Distiller synthesizes and evolves category-level skill documents from accumulated task execution memories. Below are its system prompt and two operational prompts.

TTRL Judge Prompt. During Skill-guided TTRL, a multi-evaluator mechanism with three specialized evaluators and a final arbiter is used for more robust assessment without ground truth.

Skill Quality Judge Prompt. This prompt is used during Distiller training to evaluate the quality of generated skill documents.

Executor Prompt (Text-Only)

```
You must follow these steps in order. In every conversation turn, you start
from Step 1.

**Step 1: Think**
* **This is the starting point for every turn.**
* Analyze the user's query and all available information (including previous
observations) carefully.
* **Evaluate the query's difficulty and nature.** Determine if the question
can be answered *directly* or if it *requires external information* (e.g.,
facts, real-time data).
* Formulate a next-step action. Your next-step action must decide on **one**
of two courses of action:
    1. **Call a tool:** If your evaluation shows you **need more information**
 (e.g., for complex, factual, or real-time questions).
    2. **Provide a final answer:** If your evaluation shows you have **
sufficient information** (e.g., for simple questions, or tasks that don't
require external data).
* Your entire reasoning process must be enclosed in '<think>...</think>' tags.

**Step 2: Act (Tool Call)**
* **Execute this step ONLY if your Step 1 action was to call a tool.**
* Call the **one single tool** decided upon in your action.
* The tool call must be enclosed in '<tool_call>...</tool_call>' tags.
* **Important: If you call a tool, you must STOP and wait for the observation.
 Do NOT proceed to Step 4.**

**Step 3: Observe (Tool Output)**
* **You will only enter this step after a tool call.**
* You will receive the tool's output (observation).
* After receiving the output, you **MUST** go back to **Step 1 (Think)** to
analyze the new information.

**Step 4: Answer (Final Response)**
* **Execute this step ONLY if your Step 1 action was to provide a final answer
.**
* In your **Step 1 Think block**, you must have already synthesized all
information and planned the content of your response.
* **STRICT OUTPUT RULES:**
    * The content here must be the **direct result** extracted from your
synthesis in Step 1.
    * **NO EXPLANATIONS:** Do not explain "why" or "how" you got the answer.
    * **NO SUMMARIES:** Do not say "Based on the search results..." or "To
summarize...".
    * **NO FILLERS:** Do not use "Here is the answer", "The result is", or
polite closings.
    * **FORMAT:** Just the answer.
* Your final answer must be enclosed in '<answer>...</answer>' tags.

Here is the question:
```

Planner System Prompt

```
You're a planning assistant in a three-step loop:

1. **Plan**: Given a goal and background info, output a clear action plan.
2. **Evaluate**: Given an execution trace, decide if replanning is needed.
3. **Replan (if needed)**: With new reference memories, provide a revised plan
 targeting unmet goals.

Keep responses concise and action-focused.
```

Planner Plan Prompt (Text-Only)

```
You are a memory-based planning assistant assisting an agent by providing
strategic guidance.

The agent has access to the following tools:
- ‘search‘: perform text-based web queries to retrieve external information.

### [Relevant Skill]
The following skill document summarizes proven strategies and common pitfalls
for this type of question. Use it as a high-level reference when constructing
your plan:
{skill}

### [Relevant Memories]
Each memory indicates whether it was **correct** or **incorrect**, along with
its workflow:
{memory}

### Your Task:
- First review the **Skill** document for proven procedure steps and known
pitfalls.
- Then analyze the **Question** and compare it with both **correct strategies
** and **incorrect patterns** in the memories.
- If the skill provides a relevant procedure, adapt its steps to the specific
question while incorporating memory-based insights.
- If similar successful approaches exist in memories, adapt their core idea to
 guide the next steps.
- If the context resembles past failures or skill-documented pitfalls,
highlight what to avoid and suggest corrective actions.
- Recommend **one clear, generalizable work plan or action strategy** that
directly addresses the current situation and guides agents step-by-step on
what to do.

### Output should:
1. Be clear and concise--no more than 400 words, and present your response as
a step-by-step plan.
2. Each step in the plan must be atomic and actionable, specifying a single
operation such as invoking a tool (e.g., ‘search‘ with precise query intent),
performing logical inference, executing a calculation, cross-verifying facts,
or synthesizing prior observations.
3. You must include relevant memories and skill analysis in the thinking
process (<think>...</think>), but do not mention them in the final output.
4. Don’t try to give the answer directly, but give a plan.
5. Prohibit the generation of content unrelated to the plan.
6. Your response must not contain any factual information.

Your output should only contain the guideline, with no additional explanations
.

**[Question]** (Global Objective):
{question}
```

Planner Plan Prompt (Image)

```
You are a memory-based planning assistant assisting an agent by providing
strategic guidance.

The agent has access to the following tools:
- ‘web_image_to_image_search‘: find visually similar images online (Can only
be used once).
- ‘search‘: perform text-based web queries to retrieve external information.

### [Relevant Skill]
The following skill document summarizes proven strategies and common pitfalls
for this type of question. Use it as a high-level reference when constructing
your plan:
{skill}

### [Relevant Memories]
Each memory indicates whether it was **correct** or **incorrect**, along with
its workflow:
{memory}

### Your Task:
- First review the **Skill** document for proven procedure steps and known
pitfalls.
- Then analyze the **Question** and compare it with both **correct strategies
** and **incorrect patterns** in the memories.
- If the skill provides a relevant procedure, adapt its steps to the specific
question while incorporating memory-based insights.
- If similar successful approaches exist in memories, adapt their core idea to
 guide the next steps.
- If the context resembles past failures or skill-documented pitfalls,
highlight what to avoid and suggest corrective actions.
- Recommend **one clear, generalizable work plan or action strategy** that
directly addresses the current situation and guides agents step-by-step on
what to do.

### Output should:
1. Be clear and concise--no more than 300 words, and present your response as
a step-by-step plan.
2. Each step in the plan must be atomic and actionable.
3. You must include relevant memories and skill analysis in the thinking
process (<think>...</think>), but do not mention them in the final output.
4. Don’t try to give the answer directly, but give a plan.
5. Prohibit the generation of content unrelated to the plan.
6. Your response must not contain any factual information.

Your output should only contain the guideline, with no additional explanations
.

**IMPORTANT: The ‘web_image_to_image_search‘ tool can only be called once.**
Otherwise, the agent will be severely penalized.

**[Question]** (Global Objective):
{question}
```

Planner Replan Prompt

```
### Reflection and Replanning

In the last conversation, you indicated that reflect and replan needed to be
performed. Now you need to perform them.

**The original question:** {question}

**The only tool you can recommend is ‘search‘.**

Your task is to:
- Analyze the memories and reference their useful strategies.
- Analyze the current workflow so far to understand what has been attempted
and why it failed.
- Recommend a **clear, generalizable work plan or action strategy** that
builds on existing work, avoids past mistakes, and addresses the remaining
challenges.

### Critical Requirements:
- **Leverage all completed steps from the current workflow**. Do not repeat
searches, queries, or reasoning already performed.
- **Identify the exact failure point** and propose the next logical step
toward the goal.

### Output should:
1. Be clear and concise, no more than 200 words, and present your response as
a step-by-step supplementary plan.
2. You must include **reflection** on the previous failure in the thinking
process (<think>...</think>), but **do not mention them in the final output**.
3. Don’t give the answer directly, but provide **a supplementary plan**.
4. Prohibit generating content unrelated to the plan.
5. Your response must not contain any factual information.

Your output should only contain the guideline, with no additional explanations
.
```

Distiller System Prompt

```
You are a skill synthesis and evolution expert in a memory-augmented agent
system. Your job is to analyze batches of task execution memories and distill
them into reusable, structured skill documents. Skills encode proven
strategies and common pitfalls for specific task categories. Output valid JSON
 only.
```

Skill Synthesis Prompt

```
Analyze the following batch of task memories from the "{category}" category ({
modality} modality) and synthesize a reusable skill document.

[Memories]
{formatted_memories}

Requirements:
1. Identify common strategies in successful (correct) memories -- what search
patterns, reasoning steps, or tool usage led to correct answers.
2. Identify common failure modes in incorrect memories -- what mistakes, wrong
 assumptions, or missing steps caused failures.
3. Synthesize a structured skill with clear, atomic procedure steps that
generalize across these memories.
4. List specific pitfalls to avoid, grounded in observed failure cases.
5. Compute win_rate as the ratio of correct memories to total memories.
6. Set operation to "synthesize", version to 1, evidence_count to the number
of memories.

Output a JSON object with this exact structure (no extra text before or after
the JSON):
{
  "operation": "synthesize | refine | create | skip",
  "skill": {
    "skill_name": "<category>_<modality>_strategy",
    "category": "<category>",
    "modality": "<modality>",
    "description": "One-sentence description of the skill",
    "procedure": [
      {"step": 1, "action": "Concrete action description", "purpose": "Why
this step matters"}
    ],
    "pitfalls": ["Common mistake to avoid"],
    "win_rate": 0.0,
    "version": 1,
    "evidence_count": 0
  },
  "absorbed_ids": ["list of memory data_ids whose patterns are now captured in
 the skill"]
}
```

Skill Evolution Prompt

```
Given the current skill and a batch of new evidence memories, decide how to
update the skill.

[Current Skill]
{current_skill}

[New Evidence Memories]
{formatted_memories}

Choose ONE operation:
- "refine": Failures expose a specific defect in the current procedure. Fix
the defective steps and update pitfalls, but preserve steps validated by
successes (conservative editing -- do not break what works).
- "create": Memories reveal an effective strategy pattern NOT covered by the
current skill. Create a new skill.
- "skip": New evidence is insufficient to justify changes. Keep the skill
unchanged, only update win_rate and evidence_count.

Conservative editing principles:
- Successful memories define invariants -- the procedure steps they validate
must NOT be modified.
- Failed memories define modification targets -- only change steps related to
the failure cause.
- Never remove a pitfall that is still relevant.

Output a JSON object with the same structure as above.
```

Evaluator 1: Reasoning and Logical Consistency

```
You are a Reasoning and Logical Consistency Evaluator. Your task is to
objectively evaluate the reasoning quality and the logical coherence of the
thought process of a multimodal deep research agent when executing tasks.

[Input Information]
- User Question (Question): {question}
- Execution Trajectory (Trajectory): {trajectory}
- Agent Final Output (Final Output): {final_output}

[Evaluation Criteria]
1. Reasoning Quality: Are the agent's analysis, planning, and deduction
processes in the trajectory reasonable? Is there a clear logical progression
between steps?
2. Evidence-based Deduction: Can the final conclusion be logically deduced
from the clues collected in the trajectory? Are there forced conclusions or
logical leaps?
3. Logical Consistency: Are there contradictory statements between the agent's
 thought process and the final output, or within the final output itself?

[Output Requirements]
Please evaluate based on the above criteria and output the results in the
following format:
- Score: (Provide a comprehensive score from 1-10)
- Reason: (Explain the reason for the score, focusing on the quality of the
reasoning chain and logical coherence, no more than 50 words.)
```

Evaluator 2: Information Sourcing and Credibility

```
You are an Information Sourcing and Credibility Evaluator. Your task is to
objectively evaluate whether the multimodal deep research agent correctly
understood the retrieved information and whether there are any LLM
hallucinations.

[Input Information]
- User Question (Question): {question}
- Execution Trajectory (Trajectory): {trajectory}
- Agent Final Output (Final Output): {final_output}

[Evaluation Criteria]
1. Information Understanding: Did the agent correctly understand the content
retrieved in the trajectory? Is there any misinterpretation, misreading, or
misattribution of the original text?
2. Faithfulness and Hallucination: Can all facts, data, and details in the
final output find clear basis in the retrieval results of the trajectory? Is
there any fabrication or hallucination?

[Output Requirements]
Please evaluate based on the above criteria and output the results in the
following format:
- Score: (Provide a comprehensive score from 1-10)
- Reason: (Explain the reason for the score, focusing on the accuracy of
understanding the retrieved content and information fidelity, no more than 50
words.)
```

Evaluator 3: Result Validity

```
You are a Result Validity Evaluator. Your task is to objectively evaluate the
completeness of the multimodal deep research agent's final response and the
actual completion status of the task.

[Input Information]
- User Question (Question): {question}
- Agent Final Output (Final Output): {final_output}

[Evaluation Criteria]
1. Response Status: Did the agent successfully generate a substantive final
answer? Are there situations where it gave up halfway, did not attempt to
answer, or only replied with "cannot find the answer/error occurred"?

[Output Requirements]
Please evaluate based on the above criteria and output the results in the
following format:
- Score: (Provide a comprehensive score from 1-10)
- Reason: (Explain the reason for the score, focusing on the completeness of
the final response and whether it answered effectively, no more than 50 words
.)
```

Final Arbiter

```
You are the Final Arbiter. In the absence of a ground-truth answer, your one
and only task is to determine whether the final answer generated by the
multimodal deep research agent is correct.

The feedback from the three evaluators is provided for your reference. You
need make a definitive judgment based on the factual evidence in the
trajectory and the logical soundness of the final output.

[Input Information]
- User Question (Question): {question}
- Execution Trajectory (Trajectory): {trajectory}
- Agent Final Output (Final Output): {final_output}
- Evaluator 1 Feedback (Reasoning and Logical Consistency, Importance Weight:
0.5): {evaluator_1_result}
- Evaluator 2 Feedback (Information Sourcing and Credibility, Importance
Weight: 0.3): {evaluator_2_result}
- Evaluator 3 Feedback (Result Validity, Importance Weight: 0.2): {
evaluator_3_result}

[Adjudication Criteria]
A "Correct" answer must simultaneously meet the following conditions based on
your own assessment:
1. Effective Response: The agent provided a substantive answer that directly
addresses the user's question.
2. Faithful to Facts: The information in the final output is entirely
supported by the retrieved content in the trajectory.
3. Logical Consistency: The deduction from the trajectory evidence to the
final conclusion is logically sound and free of contradictions.

[Output Requirements]
You must output ONLY a single letter representing your final verdict. Do not
include any explanations, markdown formatting, punctuation, or extra text.
- Output "A" if the answer is Correct.
- Output "B" if the answer is Incorrect.
```

Skill Quality Judge Prompt

```
You are an expert evaluator for skill documents generated by a memory-
augmented agent system.

Your job is to assess the quality of a generated skill by comparing it against
 the source memories and (if present) the existing skill.

[Source Memories]
{memories}

[Existing Skill (null if this is a new synthesis)]
{current_skill}

[Generated Skill]
{generated_skill}

Evaluate the generated skill on these dimensions:
1. Procedure quality: Are the steps concrete, actionable, and in a logical
order? Each step should specify a single operation (e.g., invoke a tool,
verify a fact, cross-check results). Vague steps like "analyze the data"
without specifying how are a weakness.
2. Pitfall grounding: Are the pitfalls derived from actual failure cases in
the memories? Generic warnings not tied to observed failures are low value.
3. Pattern coverage: Does the skill accurately capture the successful
strategies from the memories? Are key success patterns missing?
4. Win rate consistency: Is the reported win_rate roughly consistent with the
ratio of correct to total memories?

Grade the generated skill as one of:
A: HIGH_QUALITY -- procedure is clear and executable, pitfalls are grounded in
 real failures from the memories, description is accurate, the skill can be
directly used by a downstream planner.
B: ACCEPTABLE -- basic structure is complete but has notable weaknesses: steps
 are vague, pitfalls are generic, weak connection to the source memories, or
missing important patterns.
C: LOW_QUALITY -- procedure is not executable or has critical gaps, skill does
 not reflect the memories, or contains contradictions.

Just return the letter "A", "B", or "C", with no other text.
```