LLM Agents, Agent Memories, and Harness Technology

JIAWEI HAN
COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

JUNE 29, 2026

Outline

- LLM Agents: A General Introduction
- Early Agents: LLM External Tools
- LLM Search Agent
- LLM Agent Memory
- Some Recent Research on Agents

LLM Agents

- LLM agent: A (semi)-autonomous system in which *an LLM serves as the central decision-making component, coordinating* ***reasoning***, ***tool use***, ***memory***, *and* ***actions*** *to achieve goals*
- LLM agents represent a major evolution from traditional LLMs. Instead of functioning solely as next-token predictors, agents are designed to *perceive, reason, plan, act, and learn* while ***interacting with external environments, tools, humans, and other agents.***

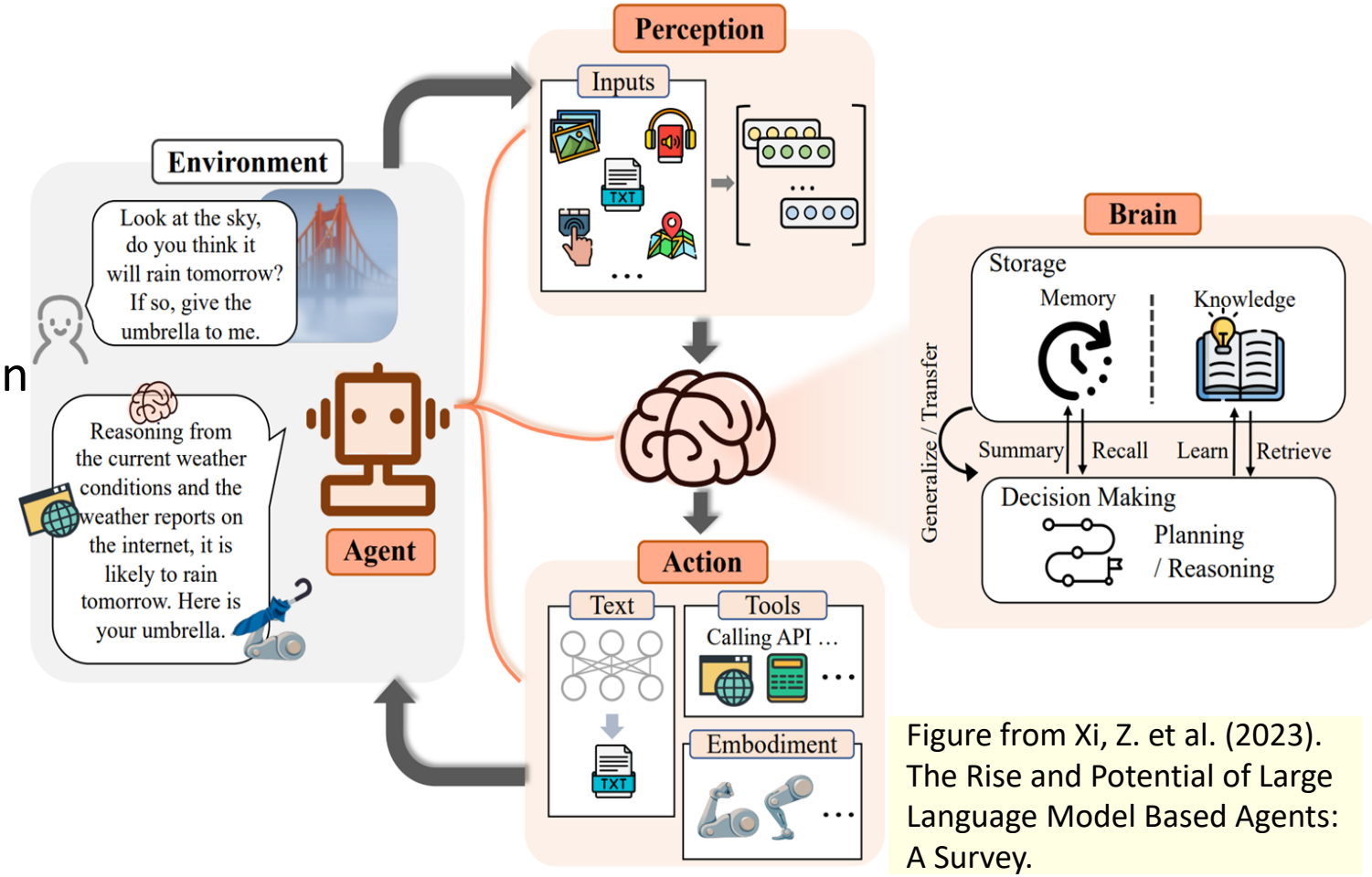


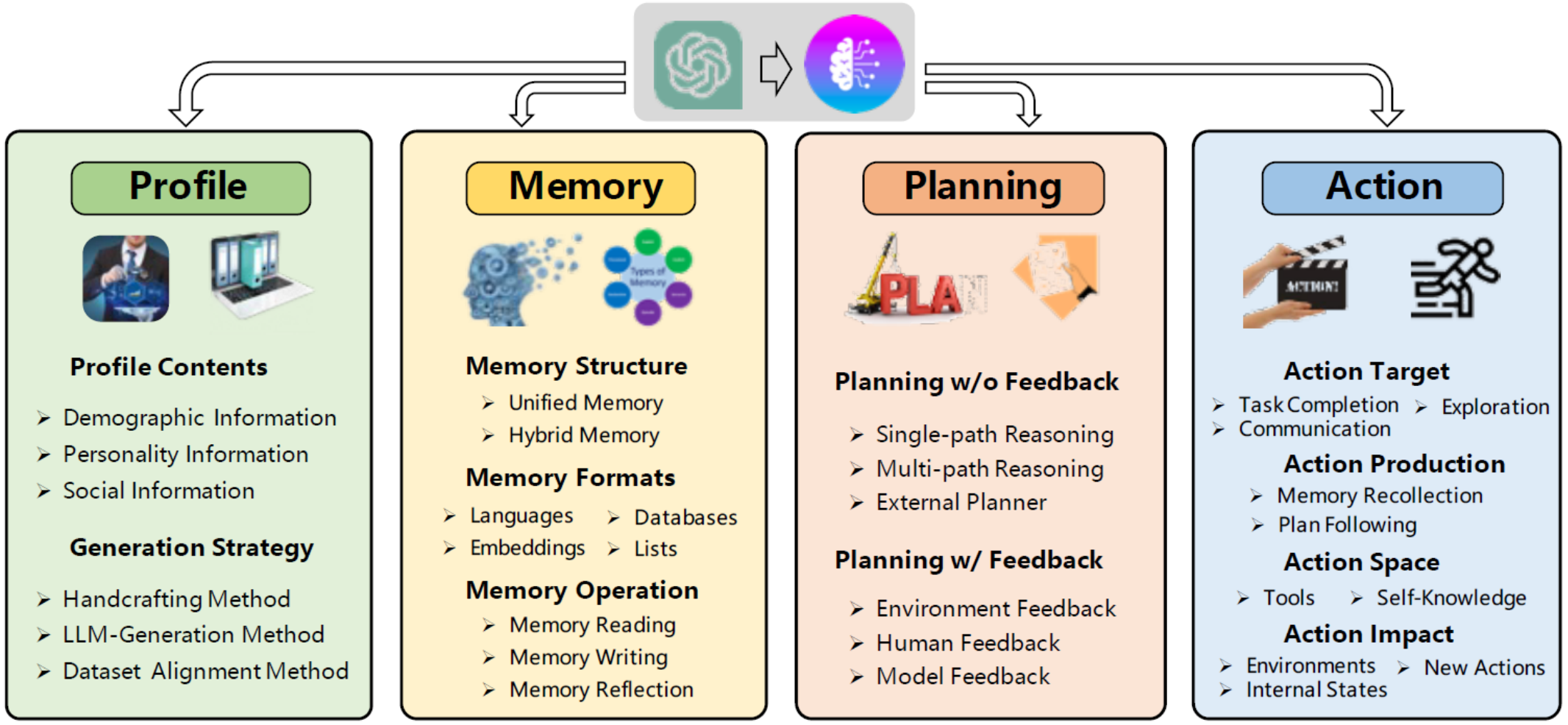
Figure from Xi, Z. et al. (2023). The Rise and Potential of Large Language Model Based Agents: A Survey.

- The emergence of LLM agents has transformed LLMs from passive assistants into active problem solvers capable of handling complex, long-horizon tasks.

A Brief History of LLM Agents

- **Pre-LLM Intelligent Agents:** Possessed planning and action capabilities but lacked flexible natural language reasoning.
  - Research roots can be traced to: Classical AI planning, Reinforcement learning agents, Belief-Desire-Intention (BDI) agents, Autonomous software agents, Multi-agent systems, Robotics agents
- **Tool-Augmented LLMs:** Set the foundation of agent architectures
  - Early milestones: Toolformer, ReAct, WebGPT
    - Key idea: LLMs can Think, Invoke tools, Observe results, Continue reasoning
- **Autonomous Agents**
  - 2023 witnessed the emergence of: AutoGPT, BabyAGI, MetaGPT, CAMEL
  - These systems demonstrated: Goal decomposition, Long-horizon execution, Self-reflection, Multi-agent collaboration
- **Emerging Research Frontiers (2025–2026)**
  - Agentic RAG / Reasoning-Acting Integration / Multi-Agent Collaboration / Long-Term Memory Agents / Computer-Use Agents / Embodied Foundation Agents / Self-Improving Agents

LLM-based Autonomous Agent: Architecture Design Framework



From: L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin et al., "A survey on large language model based autonomous agents," arXiv:2308.11432.

- Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou et al., "The rise and potential of large language model-based agents: A survey," arXiv:2309.07864, 2023.
- Z. Durante, Q. Huang, N. Wake, R. Gong, J. S. Park, B. Sarkar, R. Taori, Y. Noda, D. Terzopoulos, Y. Choi, K. Ikeuchi, H. Vo, L. Fei-Fei, and J. Gao, "Agent ai: Surveying the horizons of multimodal interaction," arXiv:2401.03568

Other survey papers on LLM agents

Research Challenges in Agent Technology

- **Reliability**
  - Agents still: Hallucinate // Misplan // Misuse tools
- **Long-Horizon Tasks**
  - Performance degrades as: Task Length ↑ Success ↓
- **Memory Management**
  - Problems: Memory pollution // Retrieval errors // Context overload
- **Cost and Efficiency**
  - Current agent systems often require: Many LLM calls // Long reasoning traces // Multiple tools
- **Safety and Alignment**
  - Risks include Autonomous harmful actions // Tool abuse // Security vulnerabilities // Deceptive behavior
- **Evaluation Crisis**
  - The field lacks: Standardized benchmarks // Reproducibility // Real-world evaluation protocols

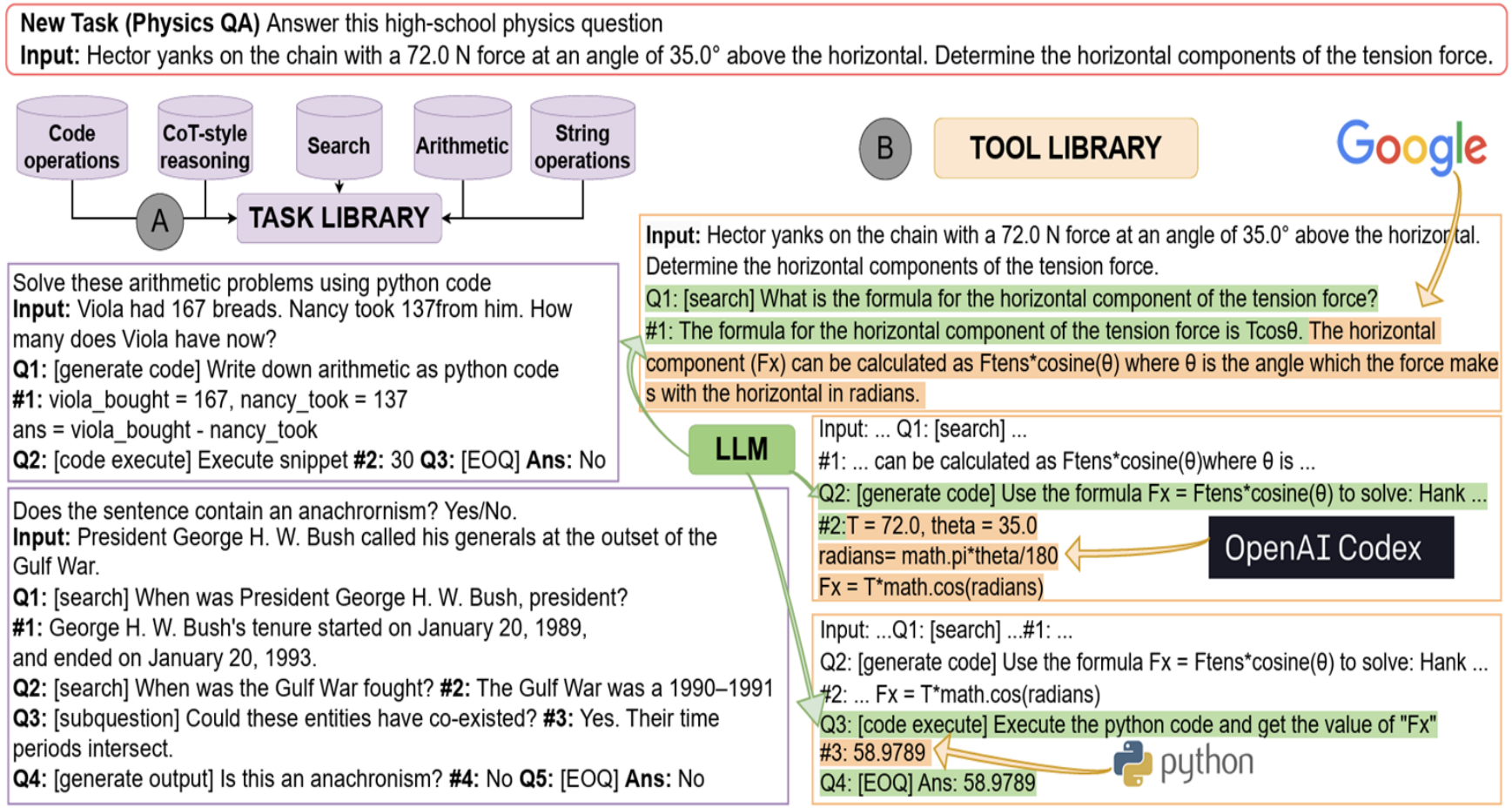
Outline

- LLM Agents: A General Introduction
- Early Agents: LLM External Tools
- LLM Search Agent
- LLM Agent Memory
- Some Recent Research on Agents

LLM External Tool: ART

- An LLM can access external tools (e.g., external functions or services) to augment its functionality
- Several tool-aware prompting approaches have been developed to make usage of tools more scalable
- Automatic Multistep Reasoning and Tool-use (ART): Combines automated chain of thought prompting with the use of external tools

• Given a task and input, the system first identifies similar tasks from a task library.
• These tasks are then used as examples in the prompt, guiding the LLM on how to approach and execute the current task
• Effective when tasks require a combination of internal reasoning and external data processing or retrieval.



B. Paranjape, S. Lundberg, S. Singh, H. Hajishirzi, L. Zettlemoyer, and M. T. Ribeiro, "Art: Automatic multi-step reasoning and tool-use for large language models," 2023.

LLM External Tool: Toolformer

- Toolformer: Train an LLM to decide what tool to use, when, and how to use it (even what parameters the API needs)
- Toolformer is built upon a pre-trained GPT-J model with 6.7B parameters. Subset of CCNet as language modelling dataset. It incorporates various tools: a calculator, a QA system, a search engine, a translation system & a calendar
- Toolformer enables language models to learn tool usage in a self-supervised manner in 3 steps

The New England Journal of Medicine is a registered trademark of [QA("Who is the publisher of The New England Journal of Medicine?") → Massachusetts Medical Society] the MMS.

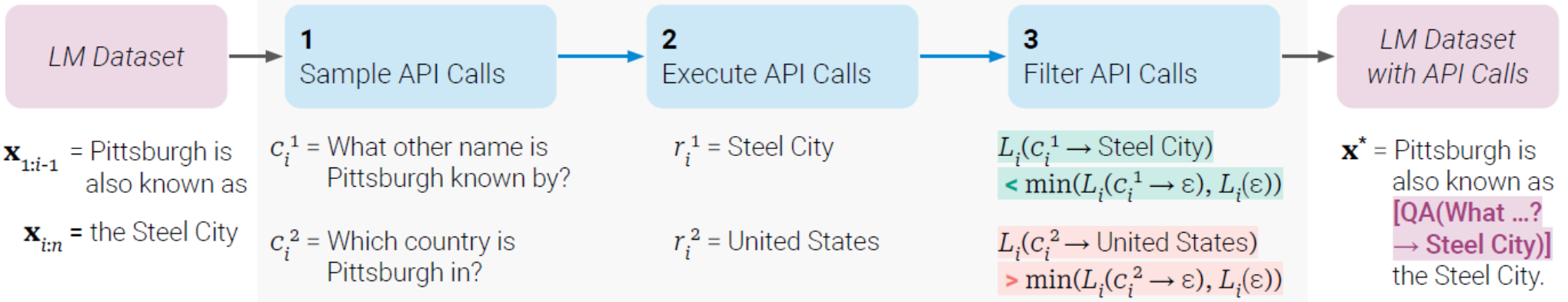
Out of 1400 participants, 400 (or [Calculator(400 / 1400) → 0.29] 29%) passed the test.

The name derives from "la tortuga", the Spanish word for [MT("tortuga") → turtle] turtle.

The Brown Act is California's law [WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.] that requires legislative bodies, like city councils, to hold their meetings open to the public.

Examples of using Toolformer

- Sampling API calls using in-context learning
- Executing and filtering these API calls
- Fine-tuning the model on the filtered API calls



Pls see Aditya Sinha's class presentation for more detail

T. Schick, J. Dwivedi-Yu, R. Dess`ı, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language models can teach themselves to use tools," 2023

Tools Play a Magic Role at Mitigating Hallucination

- Hallucination: LLMs can produce very plausible but untruthful answers

Q: "*List the title, venue and authors of highly cited papers on heterogeneous information network*"

A: "*Heterogeneous Information Network Analysis and Mining: A Comprehensive Survey*", by *Jiawei Han, Micheline Kamber, and Jian Pei, KDD 2011* (cited over *4,300* times as of March 2023), ….

- There is no easy way for prompting or chain-of-thought to do it right!
- But ChatGPT gives a perfect and comprehensive answer in June 2026!
  - What is the magic?
    - Exploring the power of tools like Google Scholar

Below are some of the most influential and highly cited papers in the heterogeneous information network (HIN) literature. They include foundational work on meta-paths, similarity measures, network mining, surveys, and representation learning.

Title	Venue	Authors
PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks	VLDB Endowment (PVLDB 4(11), 2011)	Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, Tianyi Wu Illinois Experts +1
HeteSim: A General Framework for Relevance Measure in Heterogeneous Networks	IEEE Transactions on Knowledge and Data Engineering	Chuan Shi, Xiangnan Kong, Yue Huang, Philip S. Yu, Bin Wu arXiv
Mining Heterogeneous Information Networks: A Structural Analysis Approach	SIGKDD Explorations	Yizhou Sun, Jiawei Han PMC
A Survey of Heterogeneous Information Network Analysis	IEEE Transactions on Knowledge and Data Engineering	Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, Philip S. Yu arXiv
metapath2vec: Scalable Representation Learning for Heterogeneous Networks	ACM SIGKDD Conference on Knowledge Discovery and Data Mining	Yuxiao Dong, Nitesh V. Chawla, Ananthram Swami Microsoft
PReP: Path-Based Relevance from a Probabilistic Perspective in Heterogeneous Information Networks	ACM SIGKDD Conference on Knowledge Discovery and Data Mining	Yu Shi, Po-Wei Chan, Honglei Zhuang, Huan Gui, Jiawei Han PMC
MetaGraph2Vec: Complex Semantic Path Augmented Heterogeneous Network Embedding	Pacific-Asia Conference on Knowledge Discovery and Data Mining	Daokun Zhang, Jie Yin, Xingquan Zhu, Chengqi Zhang arXiv

↓

Historical impact

Outline

- LLM Agents: A General Introduction
- Early Agents: LLM External Tools
- LLM Search Agent 
- LLM Agent Memory
- Some Recent Research on Agents

The Landscape of Agentic Tasks: Where LLM Agents Are Deployed Today

Agentic coding

edit, run, and debug code across a repo

Computer / browser use

click, type, and navigate real GUIs

Search & deep research

gather and synthesize evidence

Data analysis

query, transform, and chart data

Workflow automation

orchestrate tools and APIs

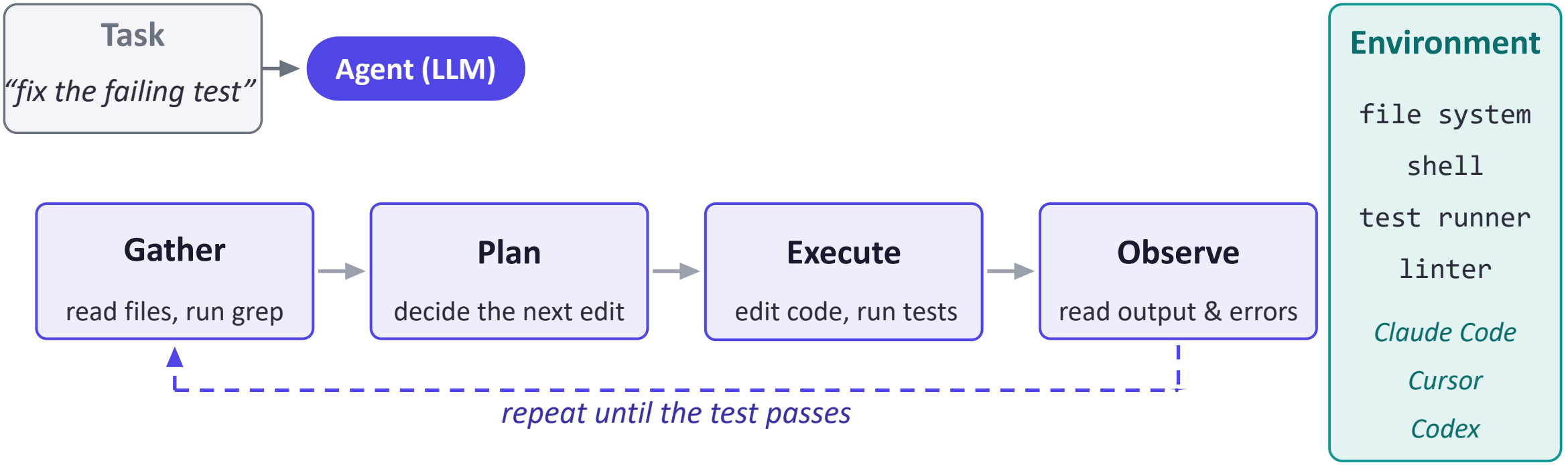
Scientific discovery

propose and test hypotheses

Common thread. Everyone is a loop: The agent gathers context, decides an action, observes the result, and repeats.

Anatomy of an Agentic Task: Coding

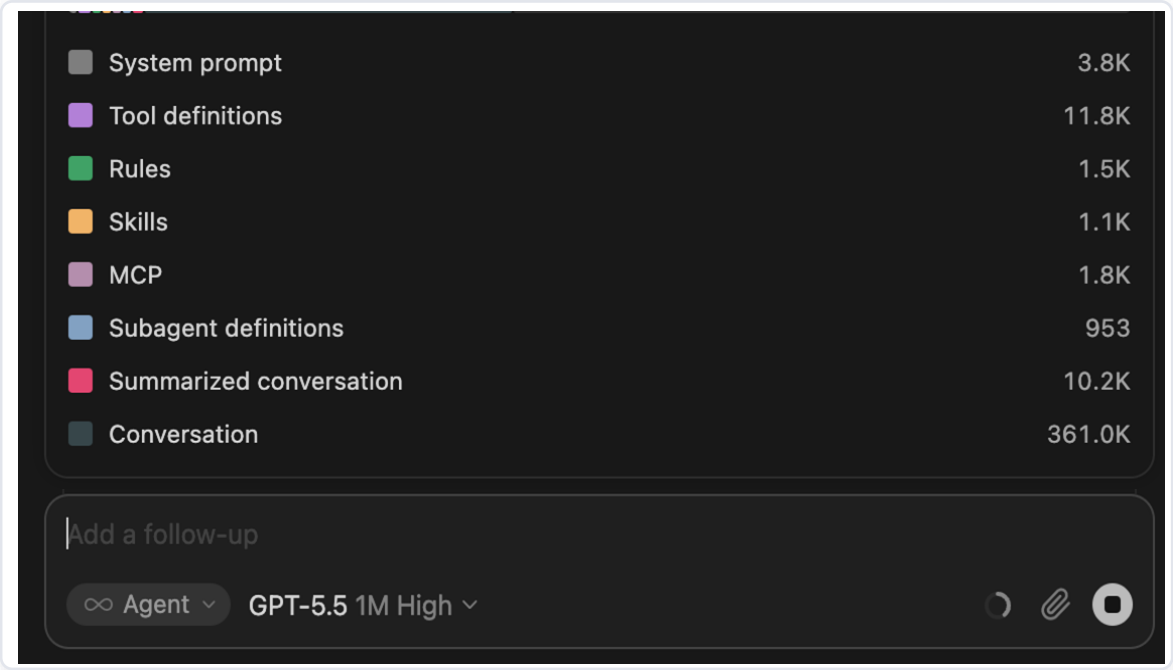
Coding: THE GATHER → PLAN → ACT → OBSERVE LOOP



The agent's real skill is managing context—deciding what to read, keep, and act on at each step (search!)

What Fills the Context Window

AN AGENT STEP IS MOSTLY CONTEXT ENGINEERING



INSTRUCTIONS

system prompt, rules, tool definitions

CAPABILITIES

tools, MCP servers, sub-agents, skills

STATE

the conversation, files, and a running summary

Context composition of a coding agent (Cursor).

LLM vs. Naive RAG vs. Agent: Three Levels of Access to the World

LLM

- Answers from parametric memory only
- No external access
- Fast, but frozen & can hallucinate

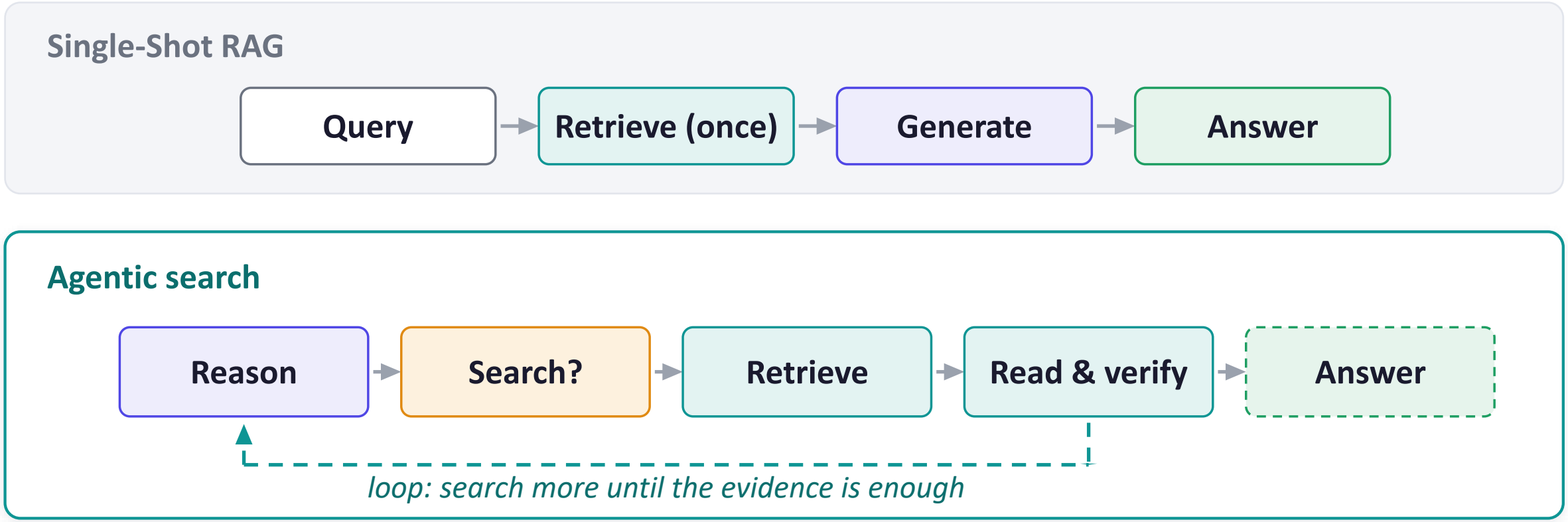
(Naive) RAG

- Retrieves once, then answers
- Single-shot external context
- Better grounding, but static retrieval

Agent

- Decides when & what to retrieve, in a loop
- Multi-step tool use + reasoning
- Gathers, verifies, and acts until done

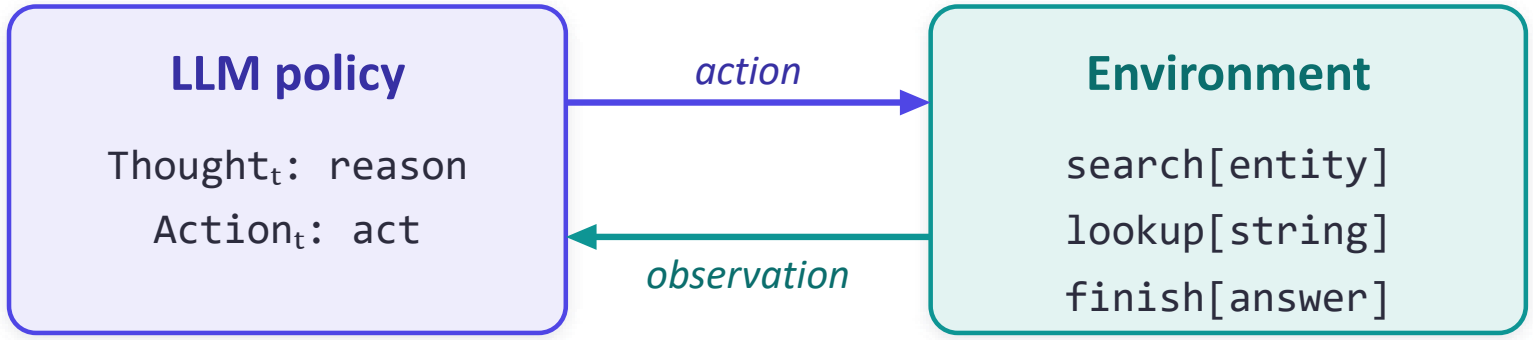
Agentic Search vs. Single-Shot RAG: Search Becomes a Decision, Repeated



The rest of this part: how agents learn to run this loop well — first without RL (prompting & reflection), then with RL.

ReAct: Reasoning × Acting

Non-RL Foundations · Interleave Thoughts with Tool Actions



repeat the Thought → Action → Observation cycle until finish[answer]

Example trace

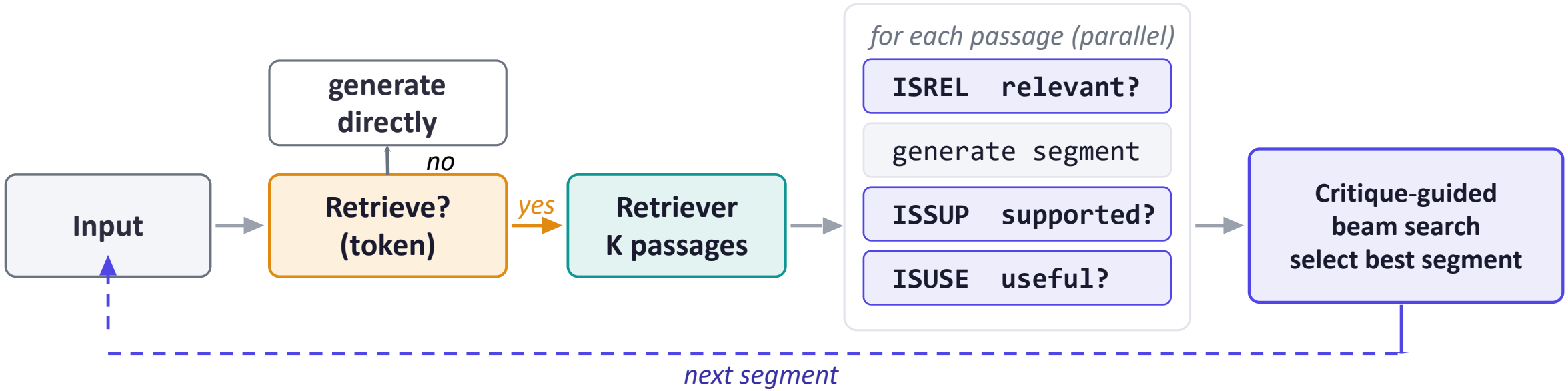
```
Thought I need the composer
of the opera…
Action search[opera]
Obs …premiered 1902…
Thought now find the
composer…
Action lookup[composer]
Obs …by Debussy…
Action finish[Debussy]
```

Key idea. Reasoning and acting in one trace — reasoning plans the next action; observations ground the next thought. Pure prompting, no training.

Yao et al., "ReAct: Synergizing Reasoning and Acting in Language Models," ICLR 2023. (Original schematic.)

Self-RAG: Retrieve, Generate, Critique

Non-RL Foundations · A Model That Reflects With Special Tokens



Four reflection tokens (added to the vocabulary)

Retrieve on-demand **ISREL** relevance **ISSUP** support **ISUSE** utility

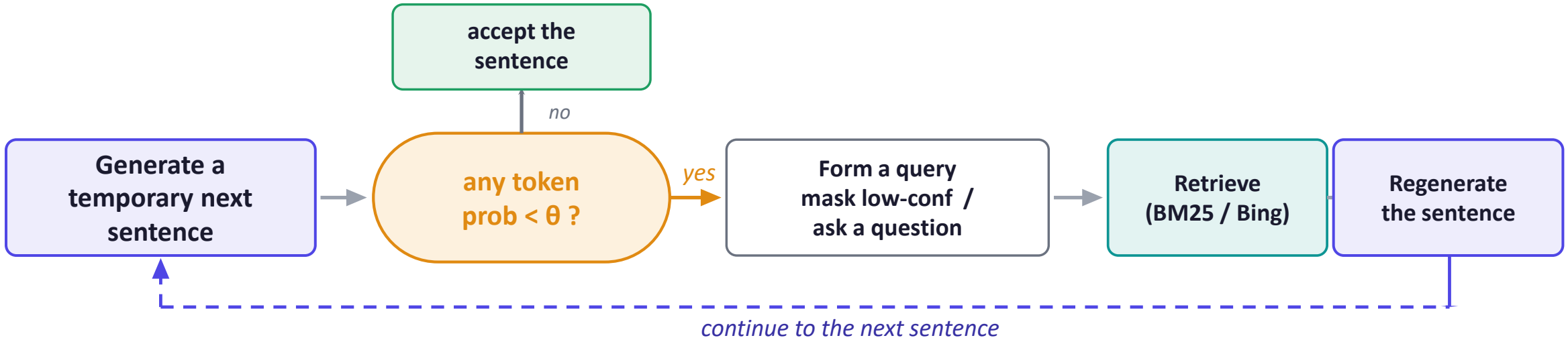
Trained offline

GPT-4 labels → critic model → augments corpus → generator learns to emit the tokens itself.

Asai et al., "Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection," ICLR 2024.

FLARE: Forward-Looking Active Retrieval

Non-RL Foundations · Retrieve Only When The Model Is Unsure




```
Joe Biden attended the University of ____ ← low-confidence span triggers a search
```

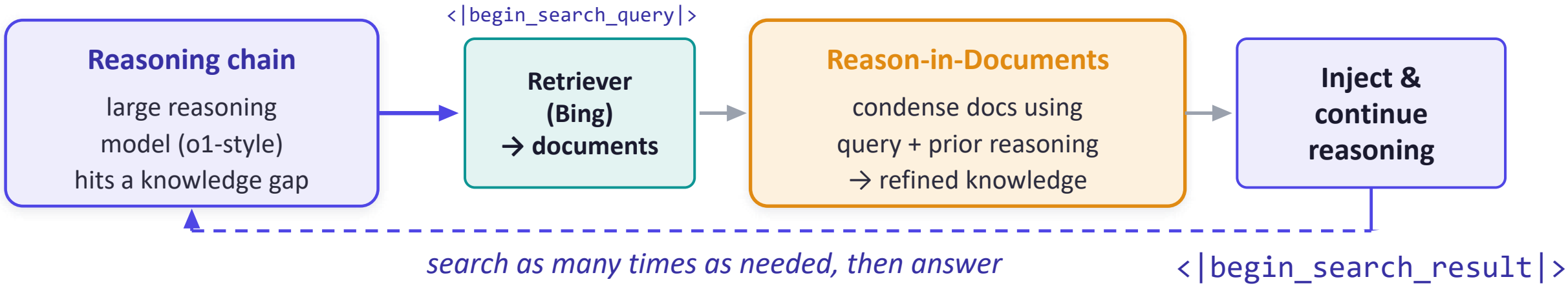
The model's own look-ahead becomes both the trigger and the query.

Two variants. FLARE-direct uses the sentence itself; FLARE-instruct prompts the model to emit `[Search(q)]`. Training-free.

Z. Jiang et al., "Active Retrieval Augmented Generation (FLARE)," EMNLP 2023. (Original schematic.)

Search-o1: Agentic Search in the Reasoning Chain

Non-RL Foundations · A Reasoning Model That Searches Mid-thought



The novelty: the **Reason-in-Documents** step distills long retrieved text into a concise fragment, so external knowledge enters the chain **without breaking its coherence**.

Li et al., "Search-o1: Agentic Search-Enhanced Large Reasoning Models," EMNLP 2025. (Original schematic.)

Non-RL Agentic Search: A Summary

Smarter Loops Over a Frozen Model — Then RL

Method	Core idea	How	Signature
ReAct	interleave thought + action	prompting	`search/lookup/finish`
Self-RAG	retrieve & self-critique	trained tokens	`Retrieve / ISREL / ISSUP / ISUSE`
FLARE	retrieve when unsure	prompting	`confidence threshold θ`
Search-o1	search inside the reasoning chain	prompting (LRM)	`Reason-in-Documents`

The limitation. All of these make a ***frozen*** model search more cleverly — the model never gets **better at searching**.

Next: use reinforcement learning to train the policy itself — so the agent learns search strategies that transfer.

Two Ways to Make an Agent Self-Evolve: Change the Weights, or Accumulate Experience

Parametric Evolution — *update θ*

Post-train the policy (RL, DPO, SFT) so the agent gets durably better at the task.

- Capability baked into the weights
- Strong, persistent gains — the engine of search agents
- Costly; risks catastrophic forgetting

DeepRetrieval · Search-R1 · s3 · Harness-1

Experience I/O — *memory & skills*

Keep weights frozen; write & read an external store of experience the agent grows over time.

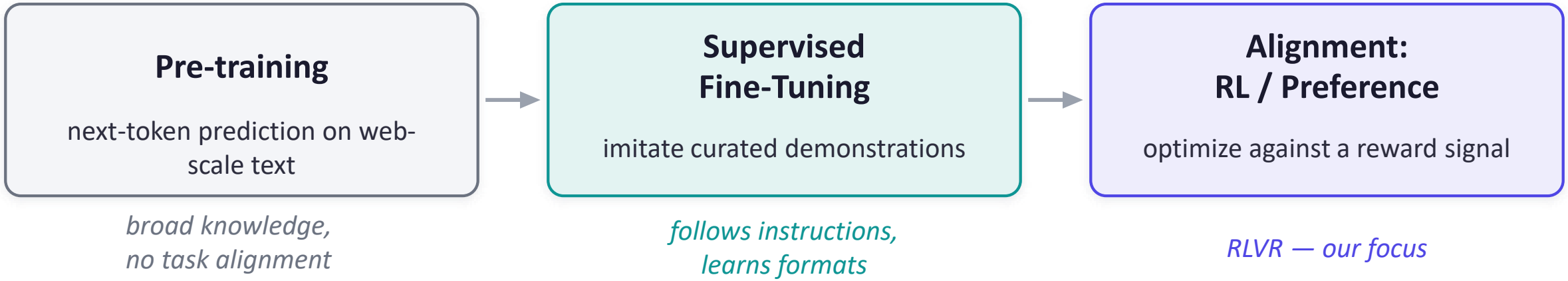
- Capability lives outside the weights
- Cheap, modular, continual — no retraining
- Bounded by the frozen model's ability

dynamic stores · reflective memory · skill libraries

Framing: Jiang et al., "Adaptation of Agentic AI: A Survey of Post-Training, Memory, and Skills," 2026.

Parametric Evolution: The Post-Training Pipeline

Where Reinforcement Learning Enters



Key idea. Pre-training and SFT teach broad ability; **RL then optimizes behavior against an outcome** — which is exactly what lets us train an agent to search well.

Ouyang et al., "Training LMs to follow instructions with human feedback," NeurIPS 2022.

From RLHF to RLVR: Replace the Reward Model with a Verifiable One

deepseek

DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning

DeepSeek-AI

research@deepseek.com

RLVR reward

$$r = 1[\hat{y} = y^\star]$$

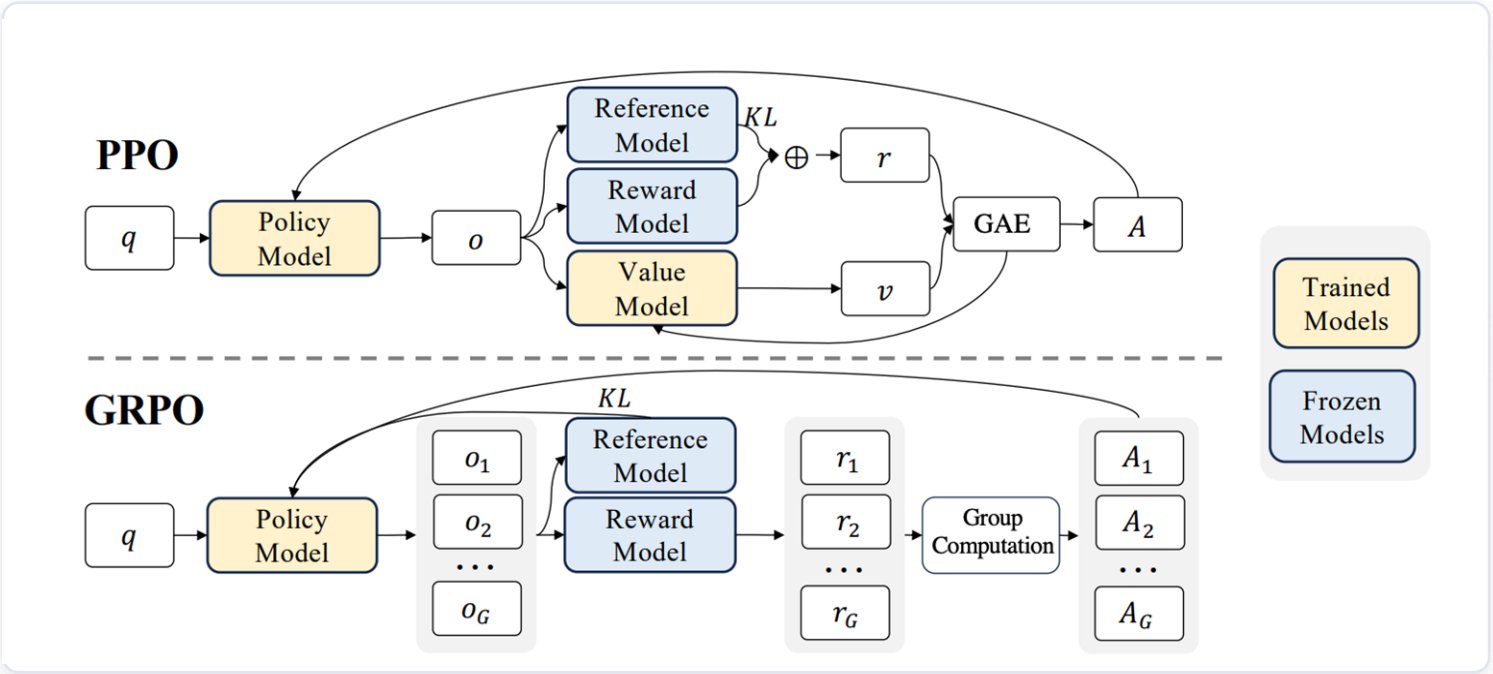
A rule-based, checkable reward (is the answer correct? does the code pass?) — no reward model needed.

DeepSeek-R1-Zero. Pure RL with verifiable rewards elicited emergent multi-step reasoning — the recipe behind RL search agents.

Why it matters for search. Retrieval recall and answer correctness are **verifiable** — perfect RLVR rewards.

Guo et al., "DeepSeek-R1," 2025. RLHF → RLVR

The RL Algorithms: PPO & GRPO: How the Policy Is Actually Updated



PPO — actor + critic

$$\mathcal{L}^{\text{CLIP}} = \mathbb{E}[\min(\rho_t A_t,\ \text{clip}(\rho_t, 1-\epsilon, 1+\epsilon)\, A_t)]$$

Clipped objective with a learned value network (critic); stable but heavier.

GRPO — critic-free

$$A_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}$$

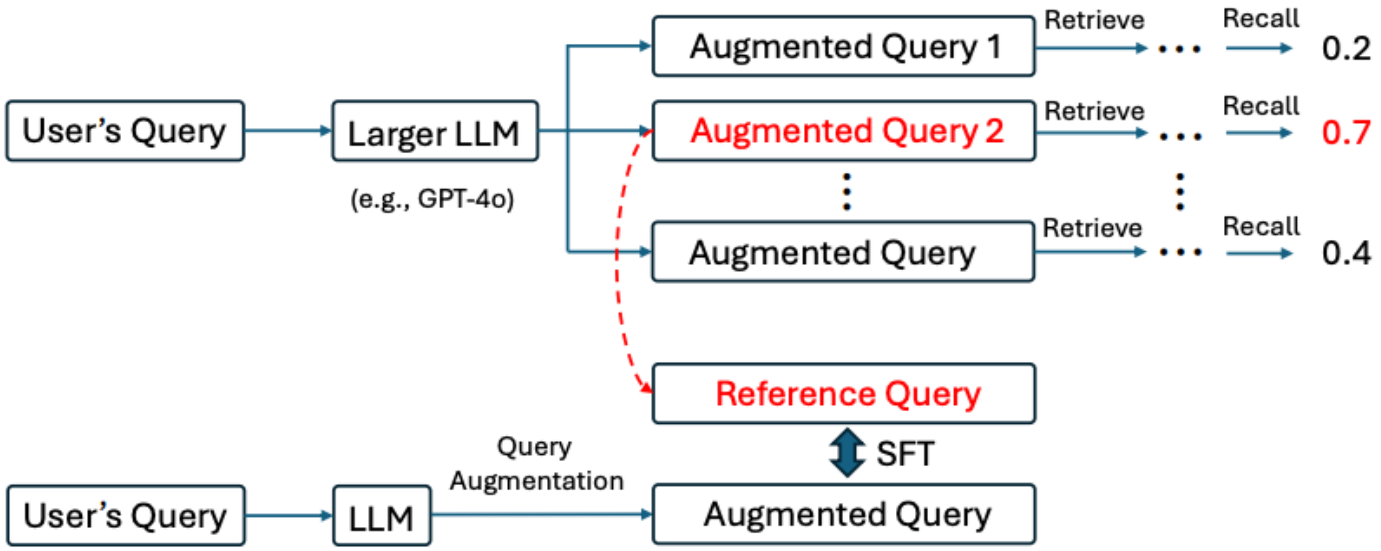
Drop the critic: normalize rewards across a group of samples. Lighter; popularized by DeepSeek-R1.

For search agents: either optimizer works — what matters is the **reward**. What reward makes an agent search **well**?

Schulman et al., "PPO," 2017. Shao et al., "DeepSeekMath (GRPO)," 2024

Why RLVR Matters to Retrieval?
— Retrieval Is Verifiable; Rewriting Is Exploration

Previous Approaches (Distillation from Larger LLMs):

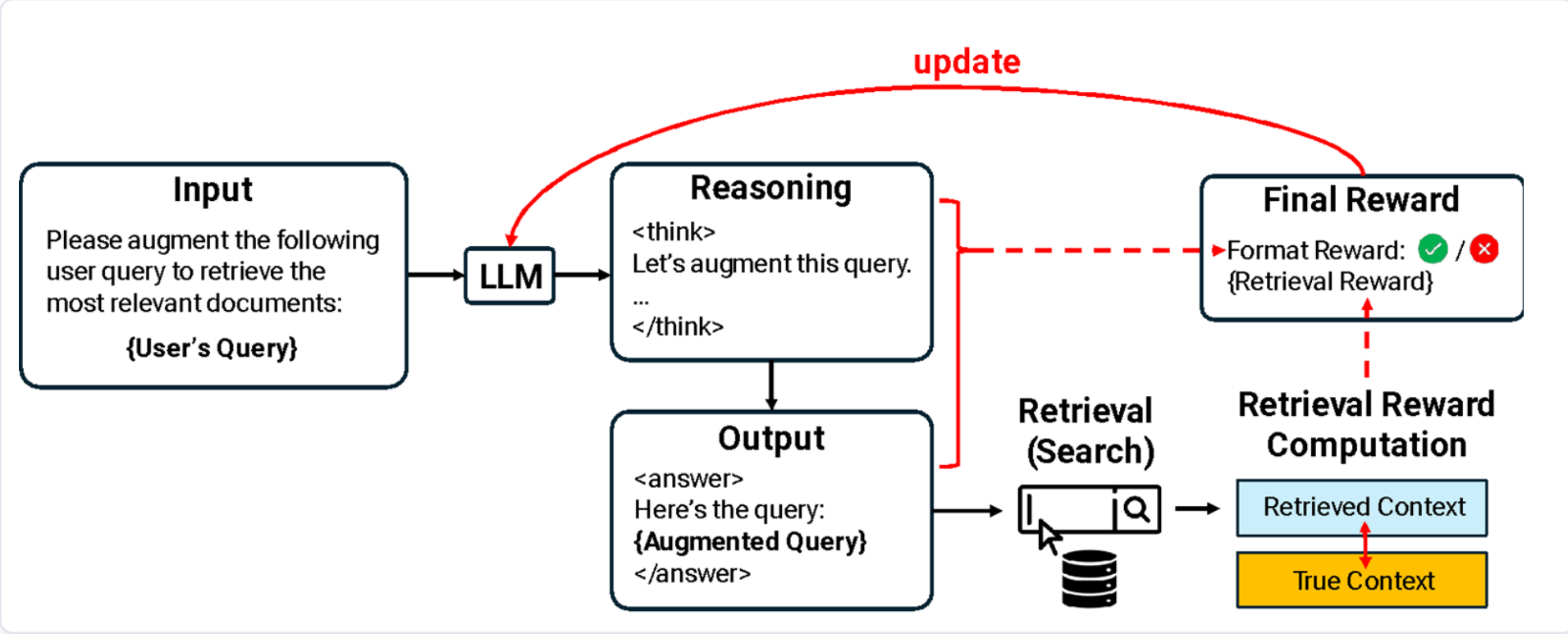


- Costly and highly rely on the quality of reference query (often suboptimal)

Key idea. Recall is a checkable reward, and rewriting the query is an **exploration problem, not a supervised one** — reward rewrites by how well they retrieve.

P. Jiang, et al., "DeepRetrieval," COLM 2025.

DeepRetrieval: Learn to Search by Exploration—Method

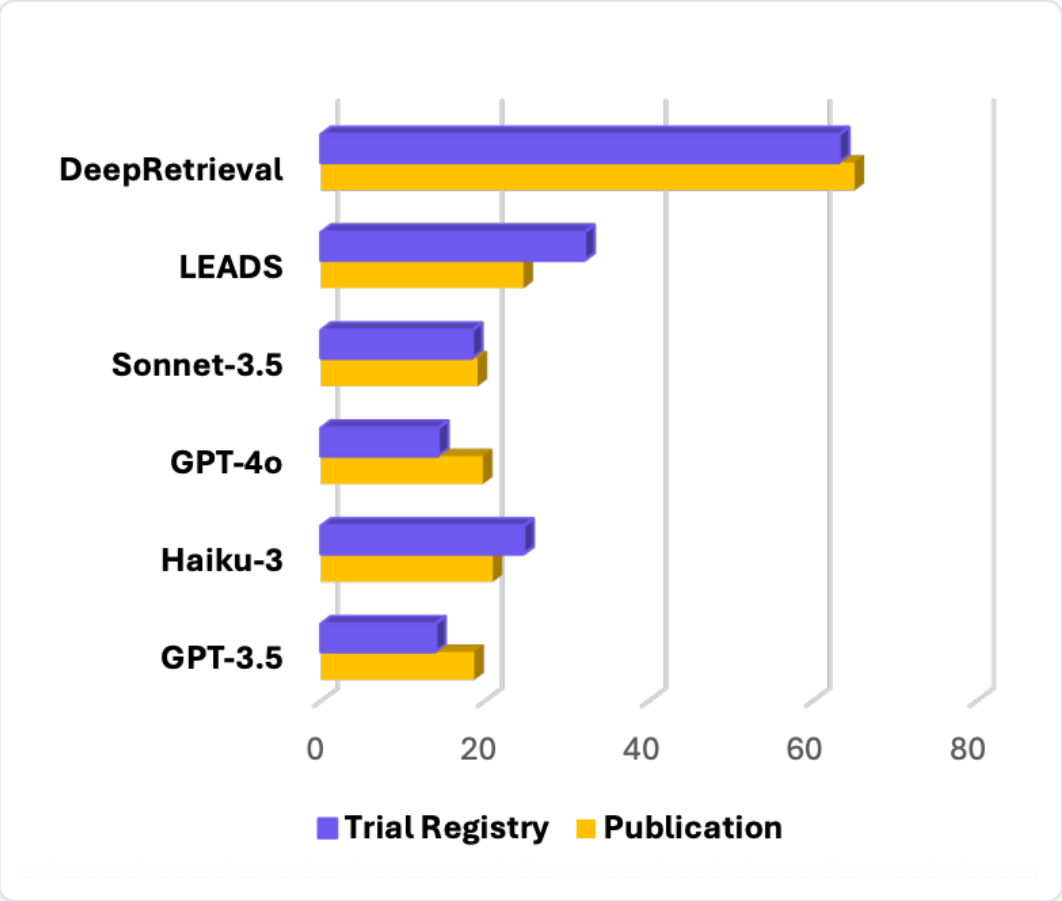


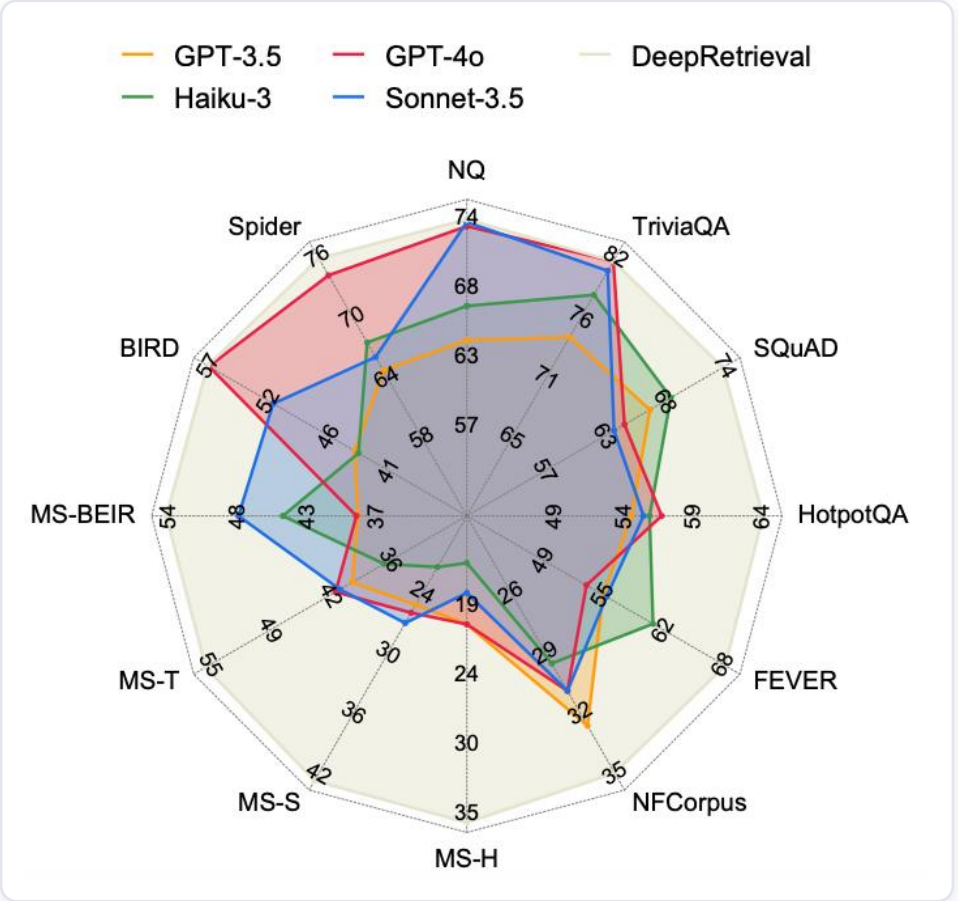
The LLM reasons, then emits an augmented query; the retrieval metric (recall / NDCG) is the RL reward.

No golden queries. A 3B policy discovers search strategies by trial-and-reward against the real engine.

P. Jiang, et al., "DeepRetrieval: Hacking Real Search Engines and Retrievers with LLMs via RL," COLM 2025.

DeepRetrieval: Learn to Search by Exploration—Results

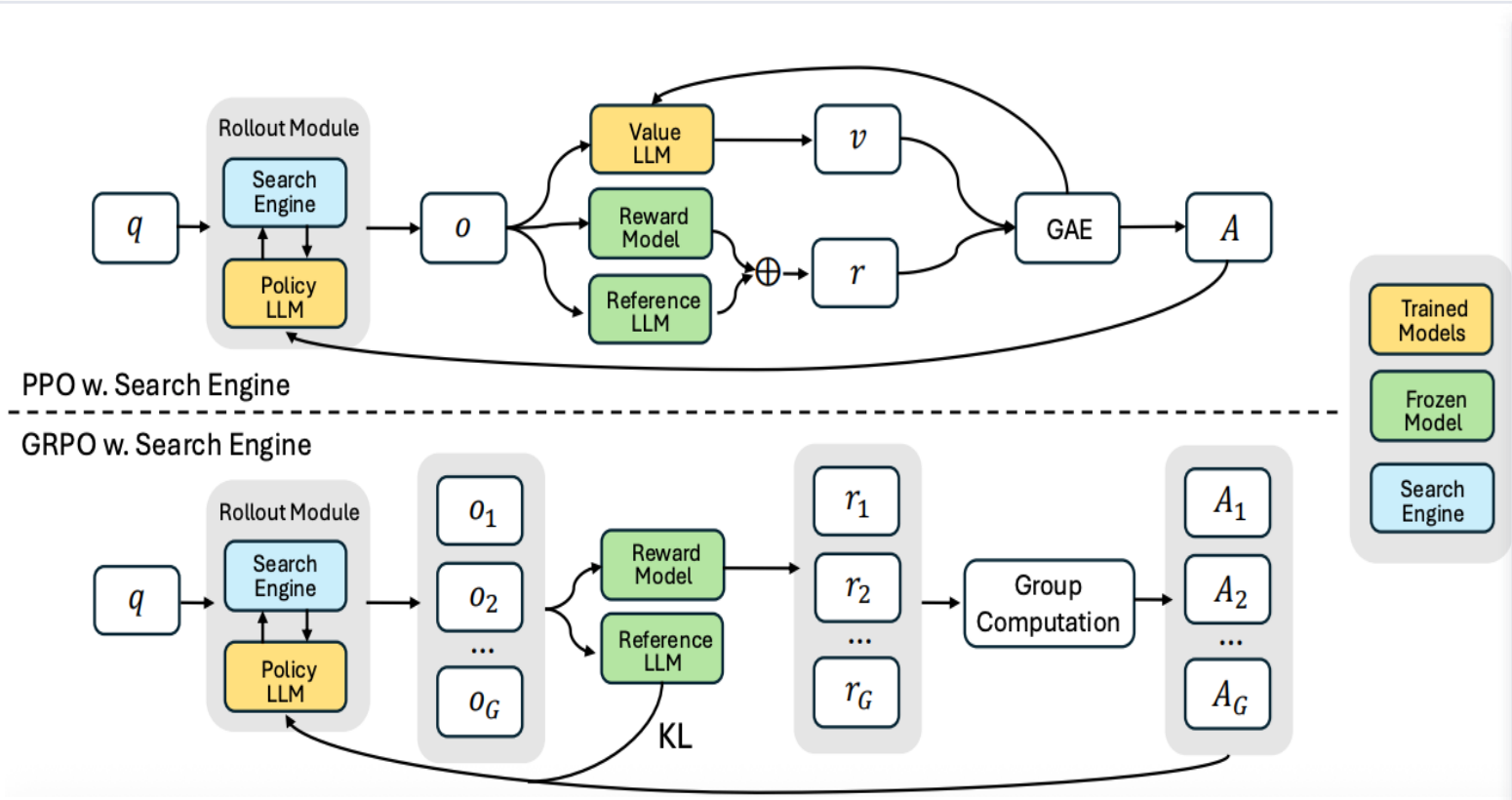




Recall on literature search (left) and across QA benchmarks (right) — the 3B policy leads.

P. Jiang, et al., "DeepRetrieval: Hacking Real Search Engines and Retrievers with LLMs via RL," COLM 2025.

Search-R1: End-to-End RL for Reasoning + Search—Method



Algorithm 1 LLM Response Rollout with Multi-Turn Search Engine Calls

Require: Input query x, policy model π_θ, search engine $\mathcal{R}$, maximum action budget B.
Ensure: Final response y.

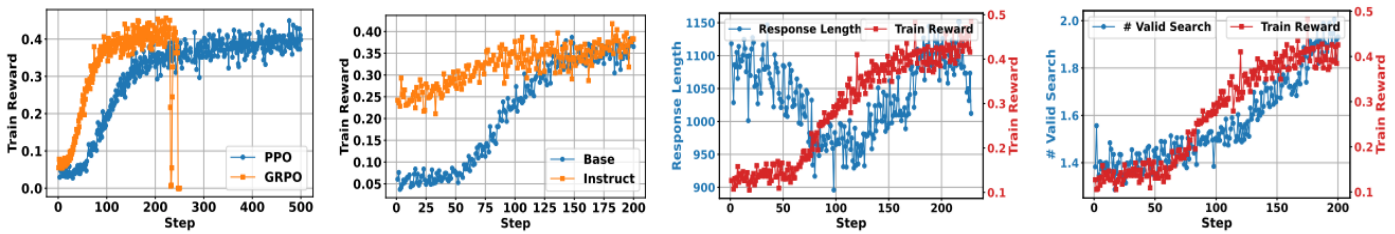
Initialize rollout sequence $y \leftarrow \emptyset$
Initialize action count $b \leftarrow 0$
while $b < B$ **do**
  Initialize current action LLM rollout sequence $y_b \leftarrow \emptyset$
  while True **do**
    Generate response token $y_t \sim \pi_\theta(\cdot \mid x, y + y_b)$
    Append y_t to rollout sequence $y_b \leftarrow y_b + y_t$
    if y_t in [</search>, </answer>, <eos>] **then** break
    end if
  end while
  $y \leftarrow y + y_b$
  if <search> </search> detected in y_b **then**
    Extract search query $q \leftarrow \text{Parse}(y_b, \text{<search>}, \text{</search>})$
    Retrieve search results $d = \mathcal{R}(q)$
    Insert d into rollout $y \leftarrow y + \text{<information>}d\text{</information>}$
  else if <answer> </answer> detected in y_b **then**
    return final generated response y
  else
    Ask for rethink $y \leftarrow y +$ "My action is not correct. Let me rethink."
  end if
  Increment action count $b \leftarrow b + 1$
end while
return final generated response y

Reward = answer EM. The policy interleaves reasoning and search; retrieved tokens are masked from the loss.

B. Jin, et al., "Search-R1: Training LLMs to Reason and Leverage Search Engines with RL," COLM 2025.

Search-R1: Accuracy Across QA Benchmarks—Results

Methods	General QA			Multi-Hop QA				
	NQ†	TriviaQA*	PopQA*	HotpotQA†	2wiki*	Musique*	Bamboogle*	Avg.
Qwen2.5-7b-Base/Instruct								
Direct Inference	0.134	0.408	0.140	0.183	0.250	0.031	0.120	0.181
CoT	0.048	0.185	0.054	0.092	0.111	0.022	0.232	0.106
IRCoT	0.224	0.478	0.301	0.133	0.149	0.072	0.224	0.239
Search-o1	0.151	0.443	0.131	0.187	0.176	0.058	0.296	0.206
RAG	0.349	0.585	0.392	0.299	0.235	0.058	0.208	0.304
SFT	0.318	0.354	0.121	0.217	0.259	0.066	0.112	0.207
R1-base	0.297	0.539	0.202	0.242	0.273	0.083	0.296	0.276
R1-instruct	0.270	0.537	0.199	0.237	0.292	0.072	0.293	0.271
Search-R1-base	**0.480**	**0.638**	**0.457**	**0.433**	0.382	**0.196**	**0.432**	**0.431**
Search-R1-instruct	0.393	0.610	0.397	0.370	**0.414**	0.146	0.368	0.385



(a) PPO vs. GRPO (b) Base vs. Instruct (c) Response length (d) # Valid search

Search-R1 (RL) beats inference, RAG, and SFT baselines on single- and multi-hop QA.

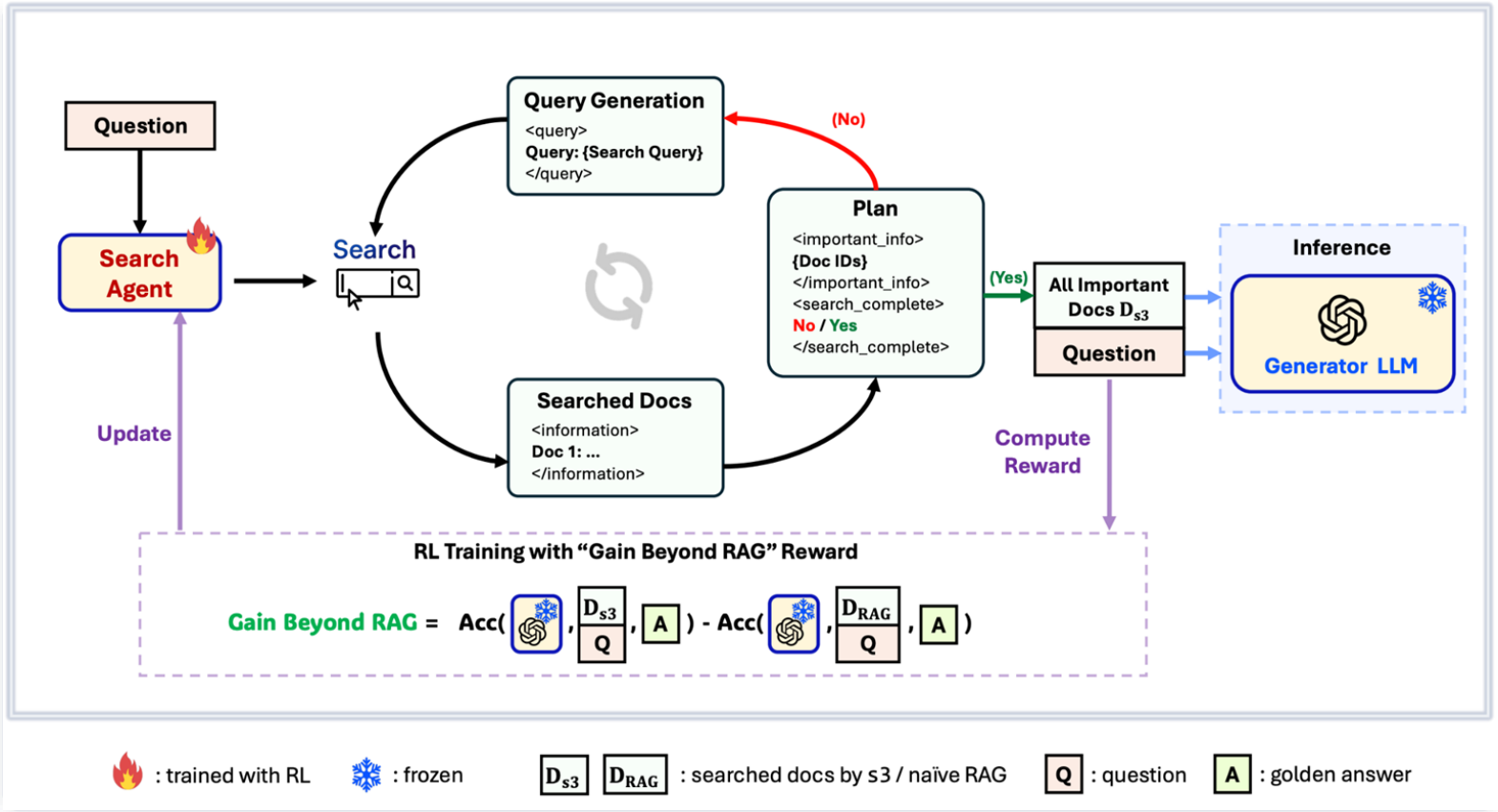
P. Jiang, et al., "DeepRetrieval: Hacking Real Search Engines and Retrievers with LLMs via RL," COLM 2025.

A Wave of RL Search Agents

Reasoning × Search	Search Environments	Reward & Process
R1-Searcher two-stage RL to incentivize autonomous search *Song et al., 2025*	**ZeroSearch** train against a simulated LLM search engine *Sun et al., 2025*	**StepSearch** step-level process rewards for multi-hop *2025*
ReSearch learn to reason while searching via RL *Chen et al., 2025*	**DeepResearcher** end-to-end RL in real web environments *Zheng et al., 2025*	**Atom-Searcher** fine-grained atomic-thought rewards *2025*
DecoupleSearch separate reasoning & search policies *2025*	**WebDancer / Sailor** autonomous deep-web research agents *Tongyi, 2025*	**Survey → AI Search** a fuller map of RL-based search *survey, 2025*

See references for the full landscape.

s3: Optimize the Searcher, Freeze the Generator—Method



Gain Beyond RAG (GBR)

$$r_{\text{GBR}} = \text{Acc}(g \mid \mathcal{D}_{s3}) - \text{Acc}(g \mid \mathcal{D}_{\text{RAG}})$$

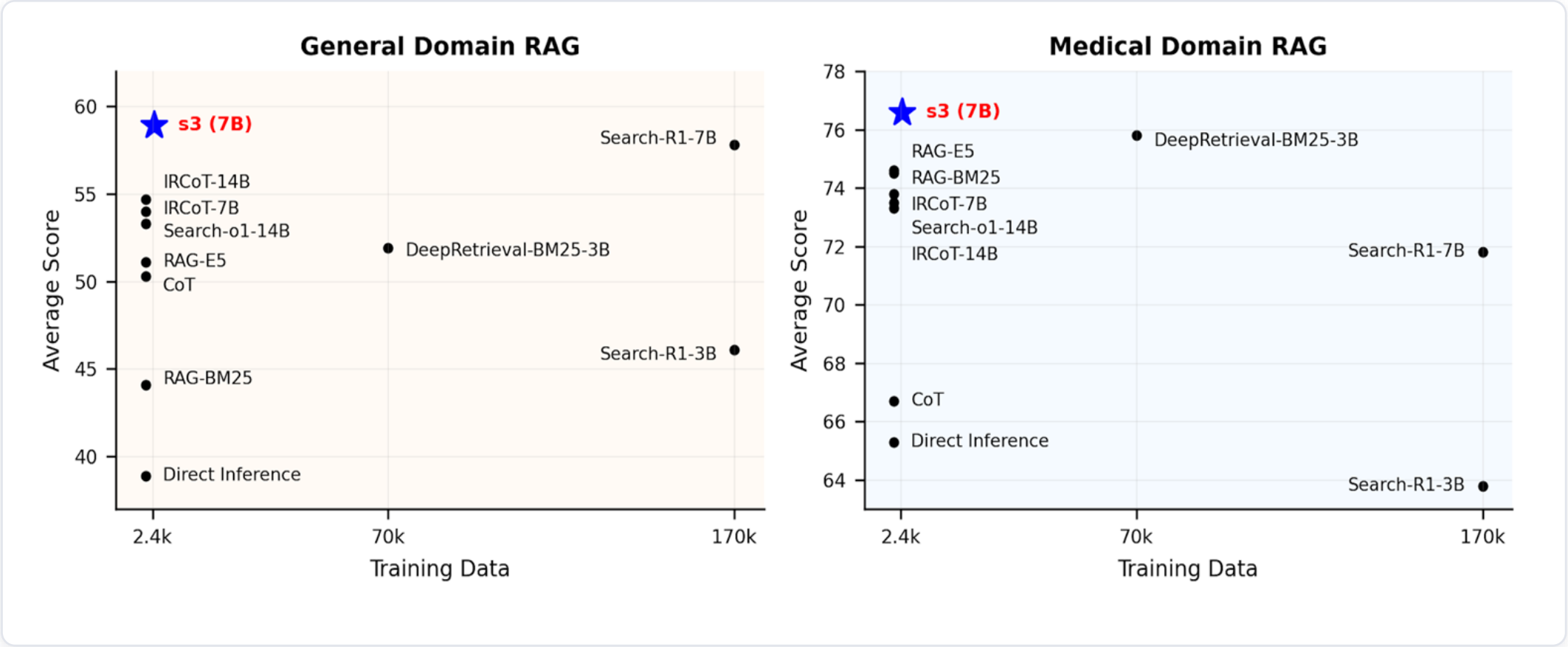
Reward the searcher only when its context helps a frozen generator beat naive RAG.

Why not just EM (Exact Match)?

EM penalizes correct-but-paraphrased answers and entangles search quality with the generator. GBR is robust.

P. Jiang, et al., "s3: You Don't Need That Much Data to Train a Search Agent via RL," EMNLP 2025.

s3: Optimize the Searcher, Freeze the Generator—Results



With ~70× less training data, s3 leads on general-domain QA and transfers to medicine with no medical training.

Decoupling + a help-based reward = efficiency ***and*** transfer — you don't retrain for every new domain.

P. Jiang, et al., "s3: You Don't Need That Much Data to Train a Search Agent via RL," EMNLP 2025.

The Long-Horizon Search Problem

One policy is asked to do everything

- **Plan** — a multi-step search over many turns
- **Remember** — every document it has retrieved
- **Rank** — which findings actually matter
- **Compress** — so the context does not overflow
- **Verify** — whether each claim is supported
- **Stop** — decide when the evidence is enough

As the transcript grows…

the model spends its attention **rebuilding its own memory every turn** instead of deciding what to do next.

For RL, the reward becomes **poorly conditioned** — one diluted signal stretched across 40+ turns is too weak to learn from.

The fix. Move the bookkeeping out of the model and into the environment.

P. Jiang, et al., "Harness-1: A Stateful Harness for Training Long-Horizon Search Agents," 2026.

Formalism: The (PO)MDP (What "Environment" Means Precisely)

Markov Decision Process

$$(\mathcal{S},\ \mathcal{A},\ P,\ R,\ \gamma)$$

- States $\mathcal{S}$, actions $\mathcal{A}$, transition P, reward R, discount γ
- Markov property: the next state depends only on the current state and action
- Goal: a policy maximizing long-run reward

Partial observability (POMDP)

$$o_t \sim O(\cdot \mid s_t)$$

- The agent sees an observation o_t, not the full state
- Must infer & remember — a belief over states
- LLM agents live here: the context window is their observation, and memory is how they cope

Why it matters. Long-horizon search is a POMDP — the agent never sees "all the evidence" at once, so **what it remembers becomes the bottleneck**.

The Idea: Split the Two Jobs (Stateful Cognitive Offloading)

POLICY *the LLM*

keeps the decisions

- what to search for
- what to keep
- what to verify
- when to stop

split

HARNESS *the environment*

keeps the state

- candidate pools
- curated evidence
- entity links & graph
- verification, history & budget

The paper calls it ***stateful cognitive offloading***: free-form reasoning stays in the model; bookkeeping moves to the harness.

P. Jiang, et al., "Harness-1," 2026. Policy: gpt-oss-20b.

The Harness: Working Memory

Harness: Search State, Kept by the Environment

P — Candidate pool

every document retrieved so far

C — Curated set

kept docs, tagged by importance · cap 30

G — Evidence graph

entities bridging documents

V — Verification

claims checked against source text

Z — Compression

observations deduplicated & shrunk

B — Budget render

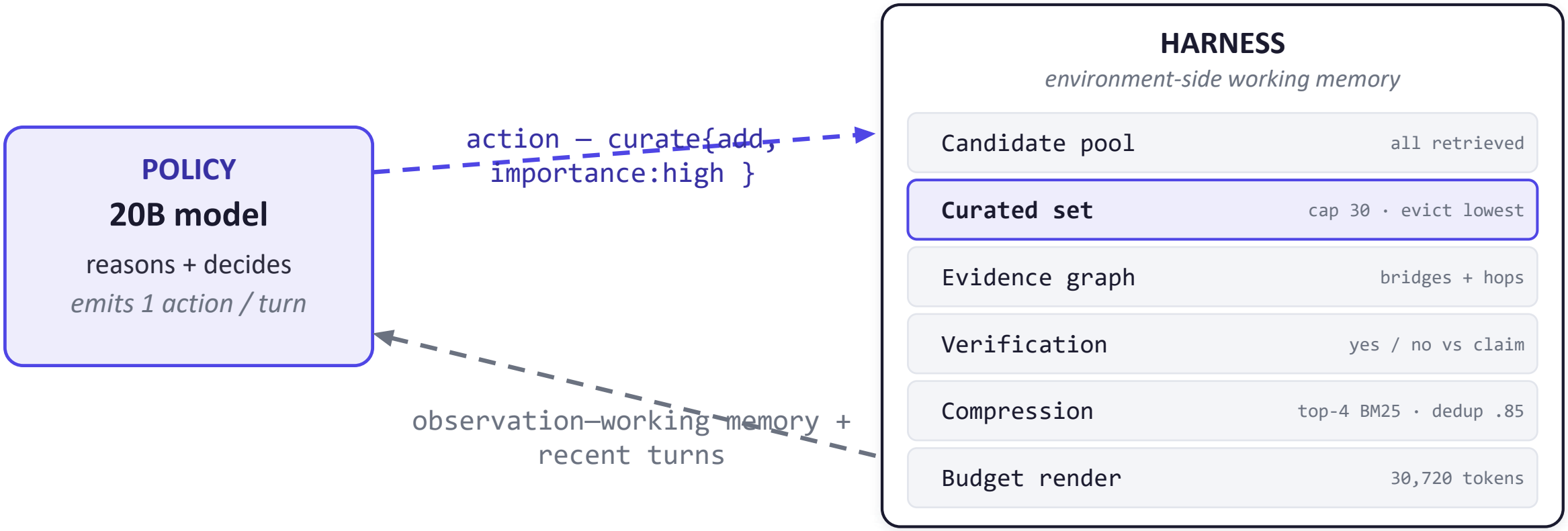
kept within the context limit

The model keeps only the decisions → `search · keep · verify · stop`

P. Jiang, et al., "Harness-1," 2026 (harness working memory).

One Turn: the Policy Decides, the Harness Remembers

(State, Action) → (State', Observation)



8 actions / 5 classes:

Retrieve	Inspect	Curate	Verify	End
fan_out · search · grep	read · review	add / remove · tag	claim	submit set

P. Jiang, et al., "Harness-1," 2026. *Free-form reasoning stays in the model; bookkeeping moves to the harness.*

What the Model Sees: Not A Transcript — A Structured State

```
== Working Memory · summarizing turns 0–12 ==

Query
"Which Brussels synagogue, completed in 1878,
 was designed by Désiré De Keyser?"

Curated Set (14 / 30)                    ↺ auto
very_high 22816 Grande Synagogue  ✓ verified
high      91442 Désiré De Keyser, architect
high      62390 Brussels synagogue register
fair      88114 19th-c Brussels architecture
low       30119 Belgian heritage  evicts first at 30

Document Pool — 31 total · 17 uncurated
[ ] 99012 synagogues of Belgium   [ ] 50441 …

[Evidence Graph] bridges                 ↺ auto
Brussels → 22816, 62390, 88114, 91442
1878 → 22816, 91442, 62390
De Keyser → 22816, 91442
bridge docs: 22816, 91442 · 5 singletons

Verification                             ↺ auto
claim: 1878 · De Keyser · Brussels synagogue
↳ 22816 yes — states 1878, De Keyser
↳ 62390 no — architect unnamed

Search History                           ↺ auto
T9  fan_out_search → 11 new · +6 curated
T10 grep_corpus "De Keyser" → 3 hits
T11 verify → 1 yes, 1 no

[Context  21,402 / 30,720]
```

Six kinds of state — extracted, ranked, verified, deduped, budgeted — re-derived for the model ***every single turn***.

P. Jiang, et al., "Harness-1," 2026 (rendered working-memory state).

The Recipe: Make Search Trainable (SFT Teaches the Interface • RL Teaches the Decisions)

1 · Supervised warm-start teacher acts in the harness → 899 demos (recall ≥ 0.10)

→

2 · On-policy RL CISPO, terminal reward, 40-turn cap → 3,453 queries

01

Warm-started curation

The first good search seeds the set with its top 8 — the model is always editing, never starting from a blank page.

02

Compact state

Importance tags, the evidence graph, and verification records — all rendered small enough to fit the budget.

03

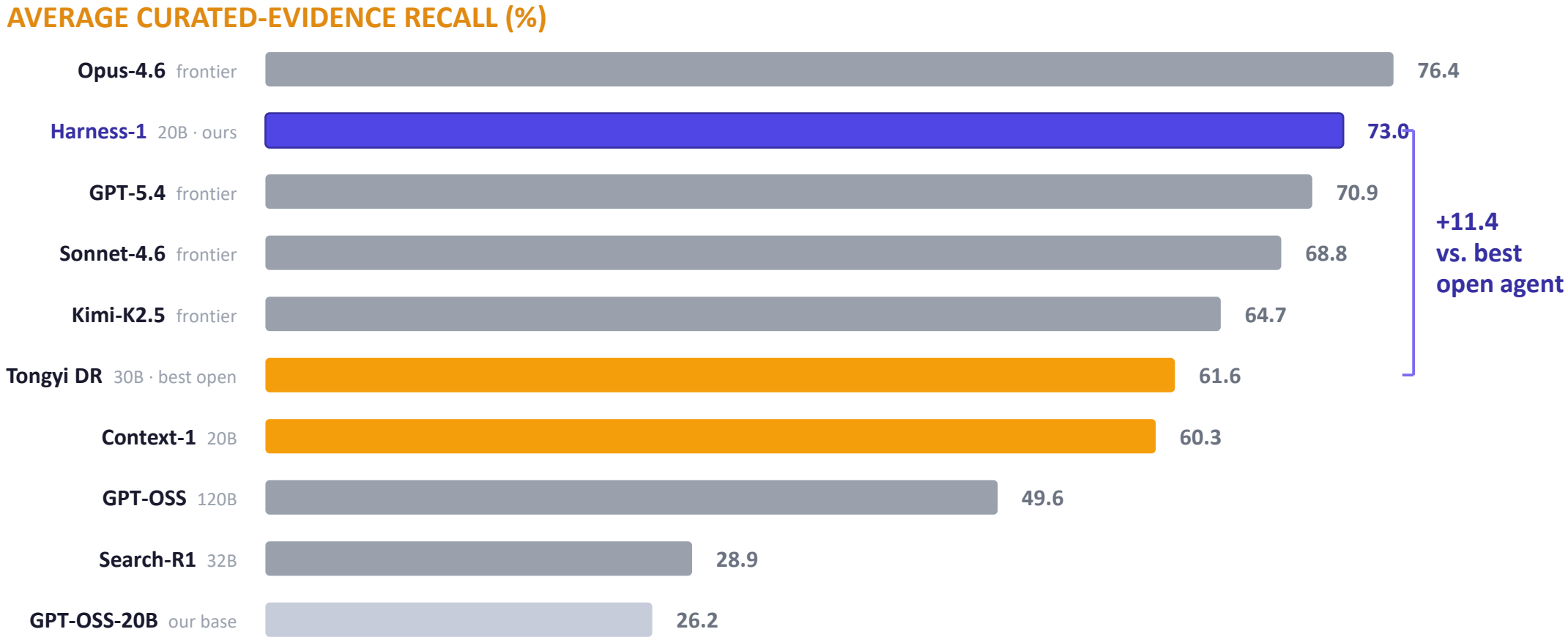
Diversity incentives

Reward a rhythm, not just discovery: search → curate → review → verify.

The reward shapes a small, complete, verified curated set *(recall weighted 4×)* — and rewards the search rhythm, not just raw discovery.

P. Jiang, et al., "Harness-1," 2026 (training recipe & reward).

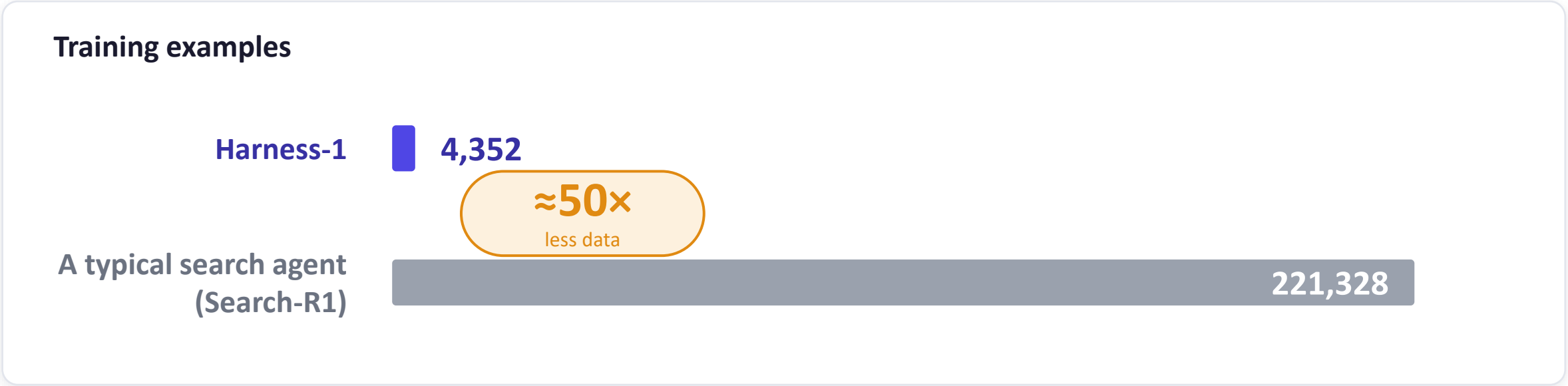
Results: Evidence Recall on 8 Hard Benchmarks



A 20B model matches the frontier. Only Opus-4.6 scores higher; the harness + RL add **+46.8 recall over the untrained base** (26.2 → 73.0).

P. Jiang, et al., "Harness-1," 2026. Average over 8 difficult evidence-retrieval benchmarks.

Trained on About 4k Examples: Need Much Less Training Data



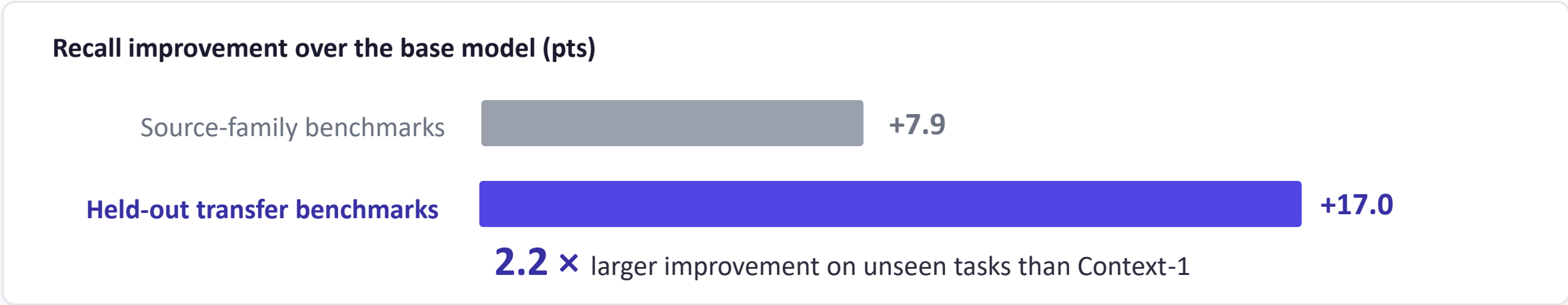
Most of the behavior lives in the interface, not in the weights. The harness does the remembering; RL only has to learn the decisions.

P. Jiang, et al., "Harness-1," 2026. 4,352 = 899 SFT + 3,453 RL.

Transfer to Unseen Environments

The Biggest Gains Appear Where It Never Trained

P. Jiang, et al., "Harness-1: A Stateful Harness for Training Long-Horizon Search Agents," 2026.



It didn't memorize domains — it learned a **reusable search workflow**: plug in your corpus, retriever, and verifier, and the trained policy operates the same interface.

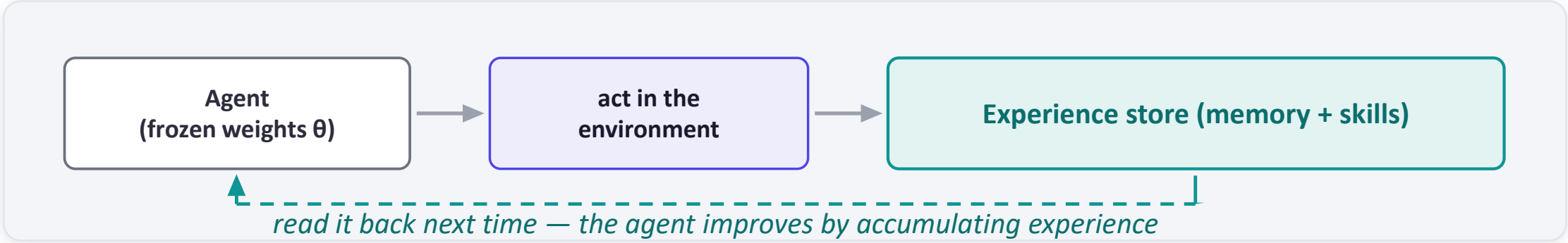
The Lesson Learned from Harness-1

Don't just train a bigger brain on a thin interface. Shape the interface itself — move the bookkeeping out, and the learned search behavior transfers to new tasks and environments.

small model `20B` + **stateful harness** `working memory` = **frontier search** `0.730 recall`

Experience I/O: Memory & Skills

Self-evolving without Touching the Weights



Memory

what the agent remembers

- Facts, past episodes, and self-reflections
- An external store the agent reads & writes
- Grows continually across tasks

Skills

what the agent can reuse

- Reusable procedures & workflows
- Distilled once from experience, reused often
- Accumulate into a growing skill library

Framing: "Rethinking Memory Mechanisms of Foundation Agents in the Second Half: A Survey," 2026.

A Taxonomy of Agentic Memory: How Agents Store & Reuse Experience

Working / short-term

The context window: what the agent is actively attending to right now.

Long-term stores

Read / write an external memory the agent manages itself.

e.g. **MemGPT, Mem0**

Episodic & reflective

Store outcomes and self-reflections to improve later attempts.

e.g. **Reflexion, Generative Agents**

Semantic / structured

Organize memory as a graph, tree, or database for retrieval.

Parametric / hybrid

Fold memory into the weights, or mix parametric + external.

Test-time curation

Decide on the fly what to keep, and how important it is.

e.g. **Harness-1**

Categories follow "Rethinking Memory Mechanisms of Foundation Agents in the Second Half: A Survey," 2026. (Original schematic.)

Memory as a Learnable Skill: Curate The Memory — Don't Just Store It

Memory operations are decisions

What to write, keep, evict, and retrieve are choices the agent makes — so they can be learned, not hand-coded.

- write: what is worth remembering?
- evict: what can be forgotten?
- retrieve: what to surface for this step?

Test-time curation

The agent decides during the task what enters memory and at what importance — keeping it small and high-signal.

- Bounded memory → fits the context budget
- Importance tags rank what survives
- Self-editing: memory the agent rewrites

We have already seen this

Harness-1's **curated set with importance tags** is exactly a learned, test-time-curated memory: the policy chooses what to `curate`, and importance eviction is a **memory-management skill trained by RL**.

Connects Harness-1's learned curation to the memory-as-skill view.

Agent Skills & Skill Libraries: Reusable Know-how the Agent Accumulates

A skill = a reusable procedure the agent distills once and reuses many times. Skills accumulate into a **growing library** the agent can search and apply — capability without retraining.

Voyager

An LLM agent in Minecraft writes executable code skills and stores them in a self-growing library it reuses for harder tasks.

Wang et al., 2023

Agent Workflow Memory

Induces reusable workflows from past trajectories, then applies them to new tasks — memory of how, not just what.

Wang et al., 2024

Claude Skills

Procedural skill files loaded into context on demand — exactly the 'Skills' slice of the context budget we saw earlier.

Anthropic, 2025

Full circle. On the context slide, **Skills** was a slice of the prompt budget — here is what fills it: distilled, reusable procedures.

Examples representative; framing follows the memory & skills survey, 2026.

Outline

- LLM Agents: A General Introduction
- Early Agents: LLM External Tools
- LLM Search Agent
- LLM Agent Memory 
- Some Recent Research on Agents

A Multidimensional View of Agent Memory Research

- **Substrates**
  - External memory
  - Internal memory
- **Cognitive Mechanism**
  - Sensory memory
  - working memory
  - episodic memory
  - semantic memory
  - procedural memory
- **Subjects**
  - User-centric
  - Agent- centric
- **Operational Architectures**
  - Single-agent
  - Multi-agent
- **Learning Policy**
  - Prompt-based
  - Fine-tuning
  - Reinforcement Learning
- **Evaluation**
  - Accuracy-based
  - Similarity-based
  - LLM-as-a-Judge

Wei-Chieh Huang, et al, "Rethinking Memory Mechanisms of Foundation Agents in the Second Half: A Survey", ArXiv:2602.06052

Memory Substrate

Human: Plan a 2-day Tokyo trip (budget $1.5k). Prefer trains over flights. Like food & museums.

Agent: Got it. I'll retrieve your preferences + recent Japan travel notes, then draft an itinerary.

Calling Memory for the historical information..

Internal (Parametric, etc..)

Memory stored in model weight, state, or KV cache

Parametric priors: transit & costs; Latent state: current constraints + intermediate reasoning; KV cache: recent conversation context

External (Non-parametric, etc…)

Memory stored in vector store, structural store, text-record, or hierarchical store

Preference: trains; avoid rush-hour transfers; budget style: mid-range. Past episode: Osaka-2024 (you liked Umeda night view + local food).

Agent: Using your profile and saved information, I suggest a trip for Day 1: Shinjuku → Meiji Jingu → Harajuku → Asakusa (by JR/Metro).

Human: Looks good! Can we add a seafood dinner and avoid too many transfers?

Based on the human feedback…

Internal (Parametric, etc…)

Model weight, latent-state, and KV cache are adjusted.

Update working state + cache. Small weight shift toward rail-first plans.

External (Non-Parametric, etc…)

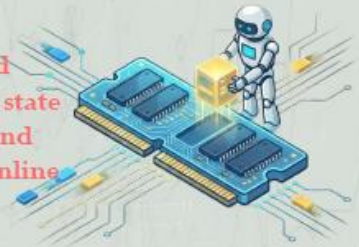
Storage and database are updated.

write preference ("prefer trains; fewer transfers") + log episode (Tokyo-2025 itinerary accepted)

Memory Cognitive Mechanism

Short-term Memory

Temporarily maintains and manipulates task-relevant state for immediate reasoning and control within a limited online budget.



Working

Holds and updates task-relevant context and intermediate states for immediate reasoning and planning.

Sensory

Briefly buffers raw perceptual inputs (e.g., visual/audio embeddings) so attention can select what to pass onward.

Long-term Memory

Persistently stores experiences, knowledge, and skills across interactions so they can be retrieved and reused over extended time horizons.

Episodic

Stores contextualized experience records (what/where/when and outcomes) for later retrieval and reuse.

Semantic

Stores stable abstract knowledge, including facts, concepts, and schemas, usable across tasks beyond specific episodes.

Procedural

Stores reusable skills, routines, and workflows that enable consistent tool use and action execution.

Memory Subject

User-Centric Memory

Stores persistent user facts (goals, preferences, constraints, and interaction history)

Agent-Centric Memory

Stores and consolidates the agent's own experiences (trajectories, outcomes, and learned heuristics or skills)

Memory Management in Dialogues

Keep user-relevant context consistent across sessions.

Long-Horizon Tasks

Store key states to support long executions.

Persistent User Simulation

Preserve user profiles to simulate long-term behavior.

Domain-Specific Long-Tail Solutions

Remember rare fixes and heuristics.

Long-Term Personalization

Remember preferences to personalize over time.

Cross-Task Knowledge Transfer

Reuse distilled knowledge across tasks.

Privacy-Preserving Memory

Protect stored user data with access or retention control.

Strategy and Skill Learning

Accumulate reusable skills and policies.

Adapted from Wei-Chieh Huang, et al, "Rethinking Memory Mechanisms of Foundation Agents in the Second Half: A Survey", ArXiv:2602.06052

Memory Substrates: External vs. Internal Memory

- **External memory**: Store the knowledge, info, and experience **outside the agent model's parameter/state**
  - Types: **vector index**, **text-record**, **structural store**, and **hierarchical store**
  - Adv: A clear separation between LLM's internal parameters and the external knowledge
    - Enable scalable, easy-to-update, cross-session retention of knowledge and interaction history
    - Can preserve past info in its original form, helping reduce hallucinations or knowledge cutoff
  - Drawbacks: Retrieval can be unreliable; costs of storage and indexing, need transformation
- **Internal memory***:* Store **directly within the model's architecture** (in *parameters*/*working states*)
  - Knowledge/experience embedded in parameters (*pre-training*, *post-training*, or *parm editing*)
  - Weight updates by *continual learning*, *model editing*, and *distillation*
  - **Latent-State:** the environment holds not only parameters but also intermediate states in memory
  - Use **KV Cache:** Cache per-layer attention keys and values from previous tokens during decoding
    - Compress the KV cache: e.g., balance "heavy hitters" with recently generated tokens, identify consistent attention patterns, retain only the most informative vectors

Memory Cognitive Mechanisms

- Cognitive psychology distinguishes multiple interacting systems that explain how information is perceived, maintained, and reused
- Five atomic systems: **Sensory**, **working**, **episodic**, **semantic**, and **procedural memory**

	Short-Term Memory	
Sensory Memory	**What is perceived**. Brief retention of recent visual, audio, or other sensory inputs before further processing	**Keep the last 2–5 sec of audio and video frames** (or recent sensor embeddings) to smooth perception and handle brief occlusion
Working Memory	**What is currently handled**. Temporary holding and manipulation of current information.	**An in-progress reasoning state (chain of thought):** "goal: refine the survey; earlier sections set the framing; the next revision should preserve framing consistency."

	Long-Term Memory	
Episodic Memory	**What happened**. Contextual record of specific experiences	**A past interaction log:** "last time you preferred a 2-page summary; the previous plan failed due to missing API keys," stored with its time and situational context.
Semantic Memory	**What is known**. Conceptual and factual knowledge about the world	**A knowledge base:** entities or facts (e.g., project info, preferences, definitions) retrieved by query and checked for reliability.
Procedural Memory	**How to act**. Skills and action patterns	**A reusable workflow or tool skill:** "search → read → extract → cite," or "debug with sanitizer," invoked as a routine.

Memory Operation Mechanism refers to the procedures an agent uses to construct, organize, access, and maintain memory over long-horizon interaction. **These operations jointly determine what historical information becomes accessible within the limited context window and.** In multi-agent systems, memory operations must also respect the *memory architecture, routing and access policies, and conflict and isolation controls*, so that agents can coordinate cross agent read and write while reducing redundancy, inconsistency, and information leakage.

Memory Operation Mechanism

Operation Mechanism

Storage & Index

Storage organizes information using vector, structured, or text formats to ensure efficient retrieval.

Update & Refresh

Agents dynamically revise or refresh memory entries to adapt to new information.

Single-Agent

Loading & Retrieval

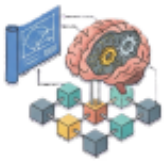
Retrieval mechanisms filter and rank relevant memories to inject into agent's current context.

Compression & Summarization

Compression condenses detailed interaction histories into compact abstracts to regulate memory growth.

Forgetting & Retention

Forgetting removes obsolete data while retention preserves high-utility knowledge for long-term reasoning



Memory architecture

It defines where memory resides and who can read or write it, covering private, shared workspace, hybrid, and orchestrated setups. These choices trade-off reuse, isolation, and controller logic.

Multi-Agent

Memory routing

It governs how memories are selected for each role, via orchestrator-based routing, agent-initiated routing, or memory-driven routing. The routing rule decides what gets retrieved and injected into each agent's context.

Memory isolation and conflicts

It limits leakage, redundancy, and contradictions when multiple agents update memory in parallel. Typical mechanisms include explicit memory editing and constraint feedback dual memory methods to keep the shared state consistent.

Memory Update and Consolidation

Agent integrates newly generated information to keep long-term knowledge consistent and up to date.

Memory Subjects: User-Centric vs. Agent-Centric

- **Memory subjects** characterize who the memory is primarily modeling and serving for
- **User-centric memory:** An abstraction of ***user-specific facts and preferences***
  - Ex. Memory management in dialogues; persistent user simulation; long-term personalization; privacy-preserving memory.
- **Agent-centric memory:** ***distilled knowledge, skills, and operational task priors;*** supports long-context, long-horizon, and long-running tasks across real-world environments.
  - **Ex. Long-Horizon Tasks (**key intermediate states); **Domain-Specific Long-Tail Solutions (**retention of the rare insights and knowledge); **Cross-Task Knowledge Transfer (**task-agnostic cognitive knowledge from diverse interactions); **Strategy and Skill Learning (**Retain environment-grounded procedural memories)
- **Working memory, procedural memory, and sensory memory** are predominantly **agent-centric**, reflecting their roles in supporting many real-world downstream tasks
- **Semantic and episodic memory** appear in **both user- and agent-centric settings**
  - For users, they encode stable preferences and interaction histories
  - For agents, they support world modeling and experience accumulation.

Memory Learning Policy

Learning policy refers to how an agent *learns* to manage memory, **what** to store, **when** to store it, **how** to represent it, **when** to retrieve or discard it, and **where** to store or retrieve, rather than relying on fixed, hand-crafted heuristics. Such policies are typically optimized from data or feedback (e.g., supervised signals, reinforcement learning, or self-improvement)

Prompting

Designing structured to guide *when* and *how* the agent updates and uses memory, without changing model parameters.

Fine-Tuning

Updating model parameters with curated data to internalize effective memory behaviors and improve robustness and consistency across tasks.

Reinforcement Learning

Learning memory management through trial and error to increase or balance performance by optimizing long-horizon rewards

Learning policy refers to how an agent *learns* to manage memory, **what** to store, **when** to store it, **how** to represent it, **when** to retrieve or discard it, and **where** to store or retrieve, rather than relying on fixed, hand-crafted heuristics. Such policies are typically optimized from data or feedback (e.g., supervised signals, reinforcement learning, or self-improvement)

Memory Learning Policy: Prompt-based, Fine-Tuning vs. RL

- **Prompt-based Memory Learning:** Parameterizes the memory policy as natural language prompts.
  - **Static Prompt-based Control:** Human-designed rules, invariant during execution
    - Design targets: *Static memory OS and organization*; *single-agent vs. multi-agent systems*
  - **Dynamic prompt-based control:** Adaptable at test time based on experience and feedback
    - Methods on ***memory usage policies***; ***memory representations*** **(**but lack explicit credit assignment, limiting their capacity for long-term policy optimization compared to fine-tuning and RL-based)
- **Fine-Tuning: Parameterized Memory Policies**
  - **Policy Internalized into parameters:** Parameterized policy stabilization and boundary control; parameterized policy efficiency and retrieval refinement
- **Reinforcement Learning for Memory Policies:** Optimized through interaction and reward feedback
  - **Step-Level Memory Decisions:** based on the immediate or short-horizon impact on task reward
  - **Trajectory-Level Memory Representation:** View memory as part of the agent's Markov state, whose quality is assessed through downstream decision performance
  - **Cross-Episode and Multi-Agent Memory:** Distill higher-level decision-relevant knowledge; extending across agents or representation spaces

Applications of the Foundation Agent Memory System

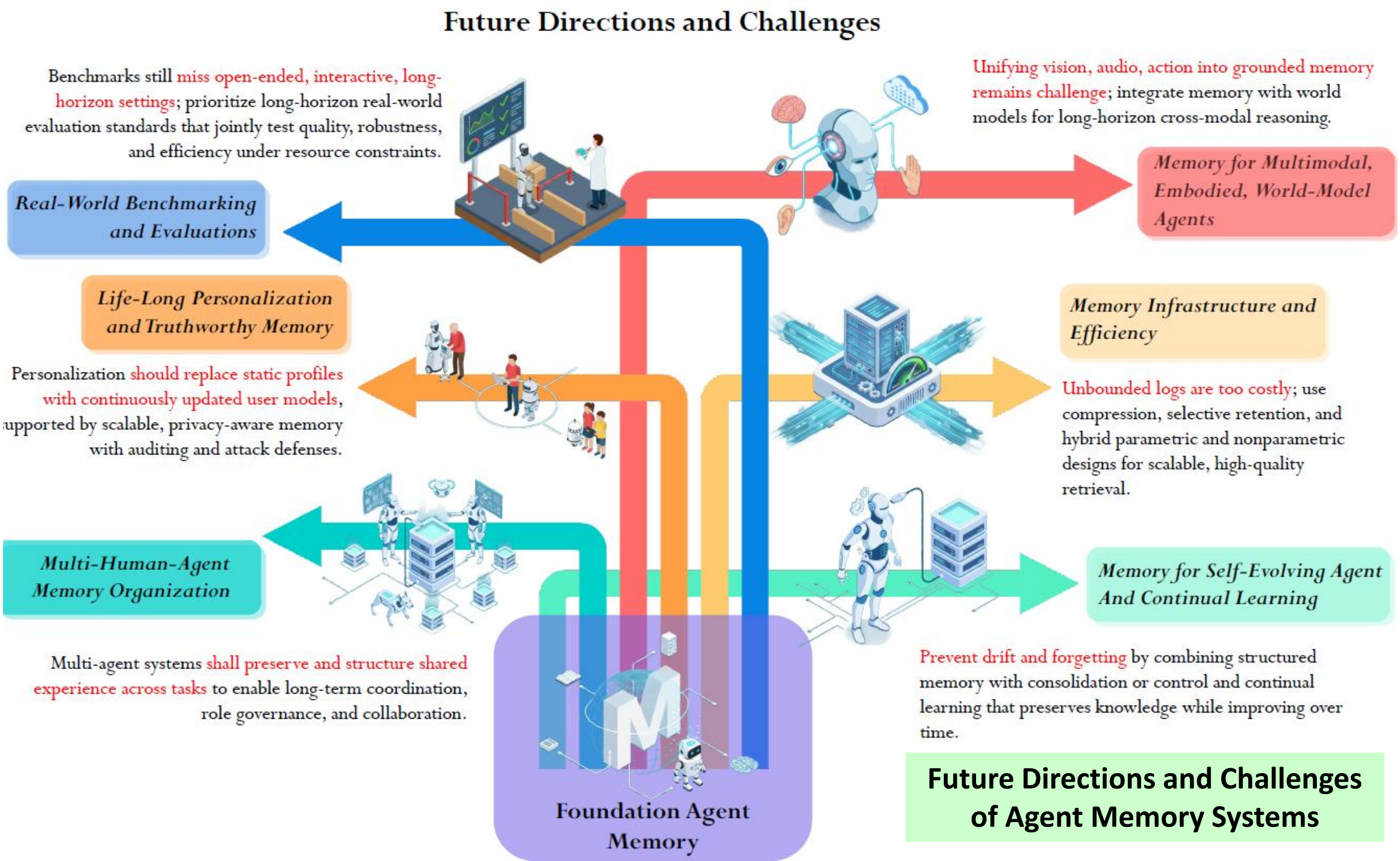
Memory transforms LLMs into dynamic, persistent agents, representing a fundamental shift in recent research.

When implemented in complex real-world scenarios, agentic memory has emerged not merely as a storage utility, but as the cognitive substrate that enables continuity, learning, and personalization, bridging an agent's past experiences with its future actions.

Recent work has broadly investigated memory-enabled capabilities in LLM agents where the ways of storing, operating, and managing memory vary significantly.

We summarize recent representative works across education, scientific research, gaming and simulation, robotics, healthcare, dialogue systems, software engineering, and workflow automation.

Future Directions and Challenges



Future Directions and Challenges of Agent Memory Systems

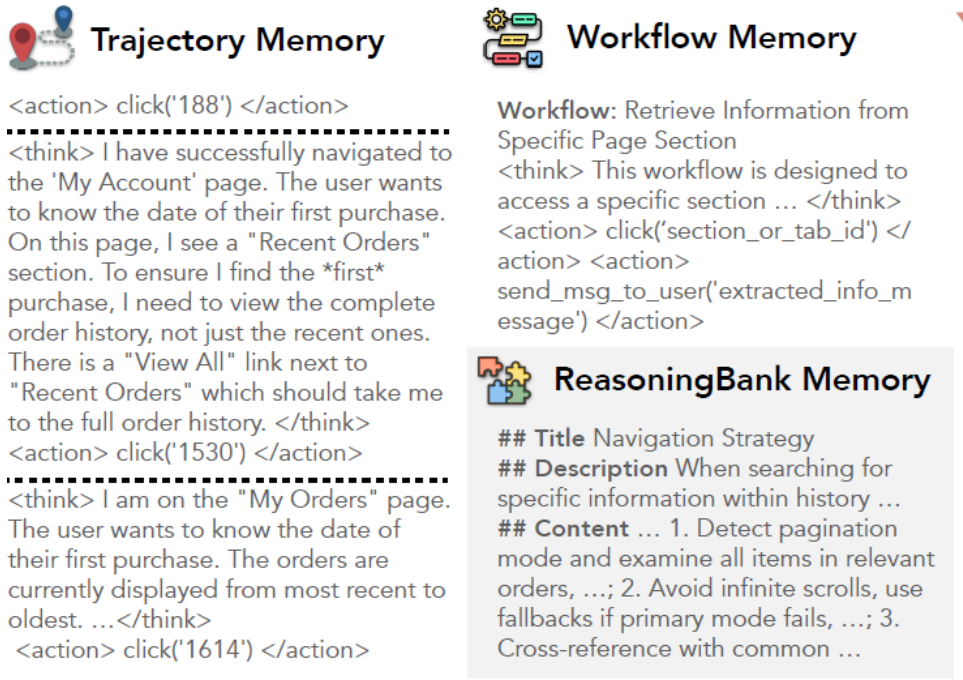
Outline

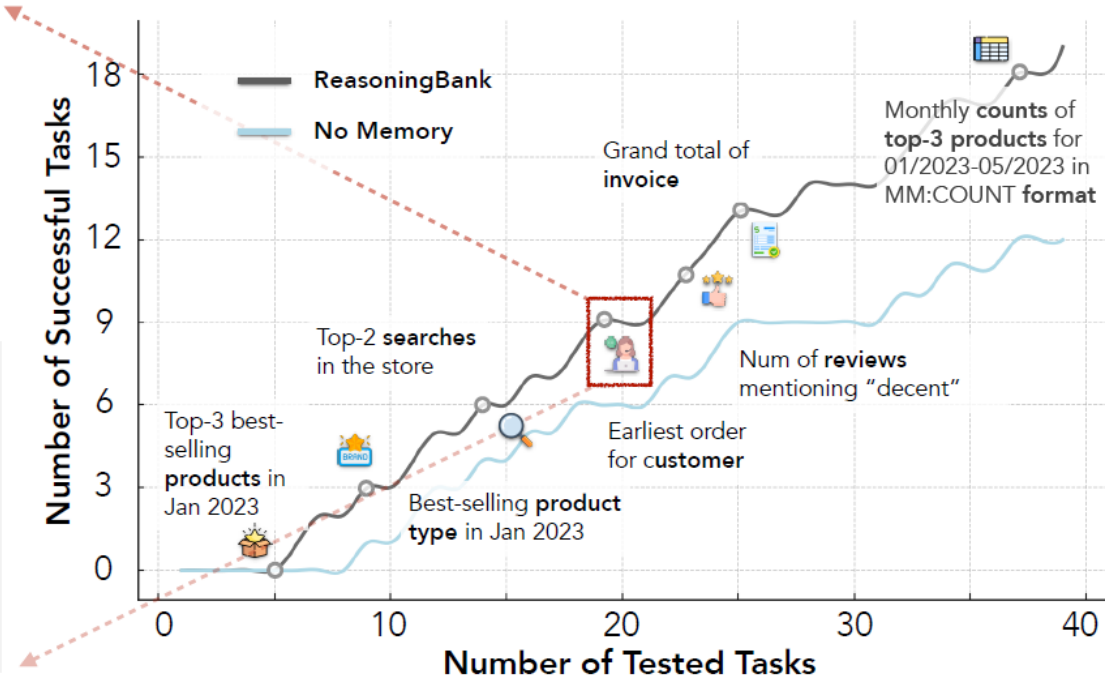
- LLM Agents: A General Introduction
- Early Agents: LLM External Tools
- LLM Search Agent
- LLM Agent Memory
- Some Recent Research on Agents

ReasoningBank: Scaling Agent Self-Evolving with Reasoning Memory

Siru Ouyang, et al., "ReasoningBank: Scaling Agent Self-Evolving with Reasoning Memory" ArXiv: 2509.25140

- Learning from past experiences as memory
- Developing self-evolving agent systems

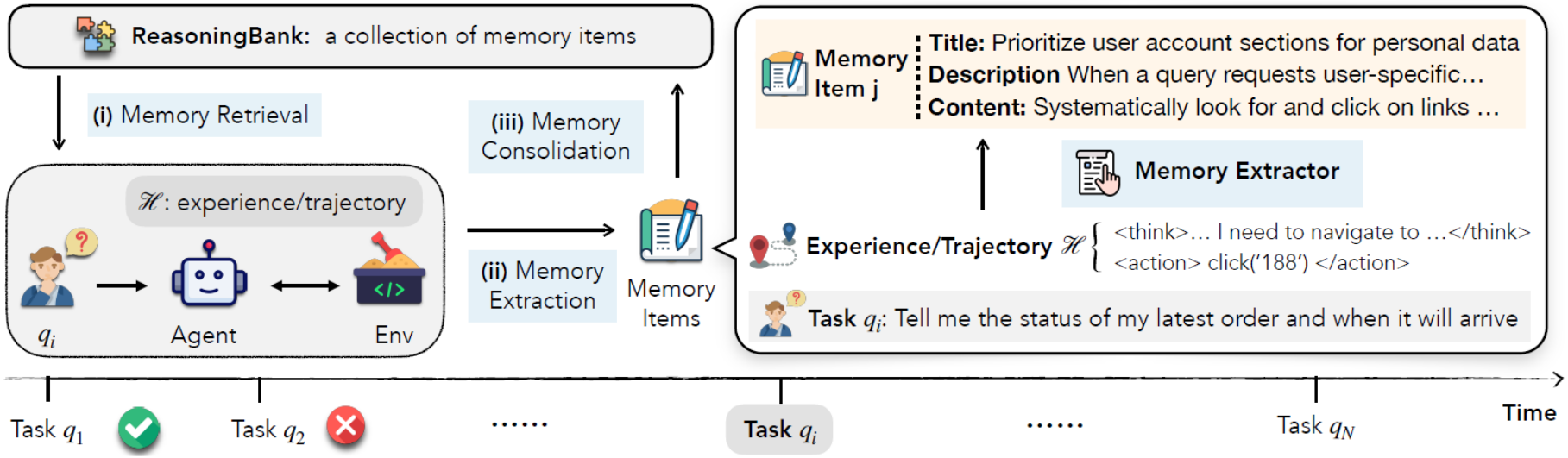




- A key limitation of LLMs on continuous streams of tasks: failure to learn from the accumulated interaction history, forcing them to discard valuable insights and repeat past errors
- ReasoningBank, a memory framework, distills generalizable reasoning strategies from an agent's self-judged successful and failed experiences (store high-level strategies and reasoning hints)
- At test time, an agent retrieves relevant memories from ReasoningBank to inform its interaction and then integrates new learnings back, enabling it to become more capable over time
- MaTTS (memory-aware test-time scaling) accelerates and diversifies this learning process by scaling up the agent's interaction experience

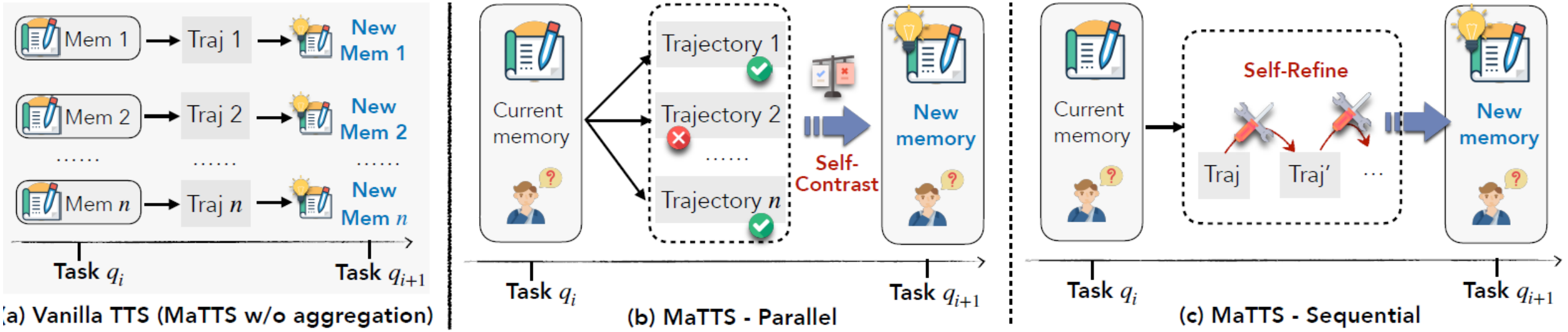
Overview of ReasoningBank

Integration of ReasoningBank with Agents: recall effective insights, avoid previously observed pitfalls, and adapt more robustly to unseen queries via (i) *memory retrieval*, (ii) *memory construction*, and (iii) *memory consolidation*



- Experiences are distilled into **structured memory items** with a title, description, and content
  - *Title:* a concise identifier summarizing the core strategy or reasoning pattern
  - *Description:* a brief one-sentence summary of the memory item
  - *Content*: The distilled reasoning steps, decision rationales, or operational insights extracted from the past
- For each new task, the agent retrieves relevant items to interact with the environment and constructs new ones from both successful and failed trajectories
- These items are then consolidated into ReasoningBank, forming a closed-loop memory process

MaTTS: Memory-aware Test-Time Scaling



Memory-aware Test-Time Scaling: Translate more experiences into greater Improvements

- Vanilla TTS (direct combination): No aggregation ,no leverage of inherent contrastive signal
- MaTTS (parallel): Generate multiple trajectories for the same query under the guidance of retrieved memory items: Identify consistent reasoning patterns while filtering out spurious solutions
- MaTTS (sequential): Iteratively refine its reasoning *within a single trajectory* after the initial completion, following the principle of *self-refinement*. During this process, the intermediate notes generated in self-refinement are also used as valuable signals for memory, since they capture reasoning attempts, corrections, and insights that may not appear in the final solution.

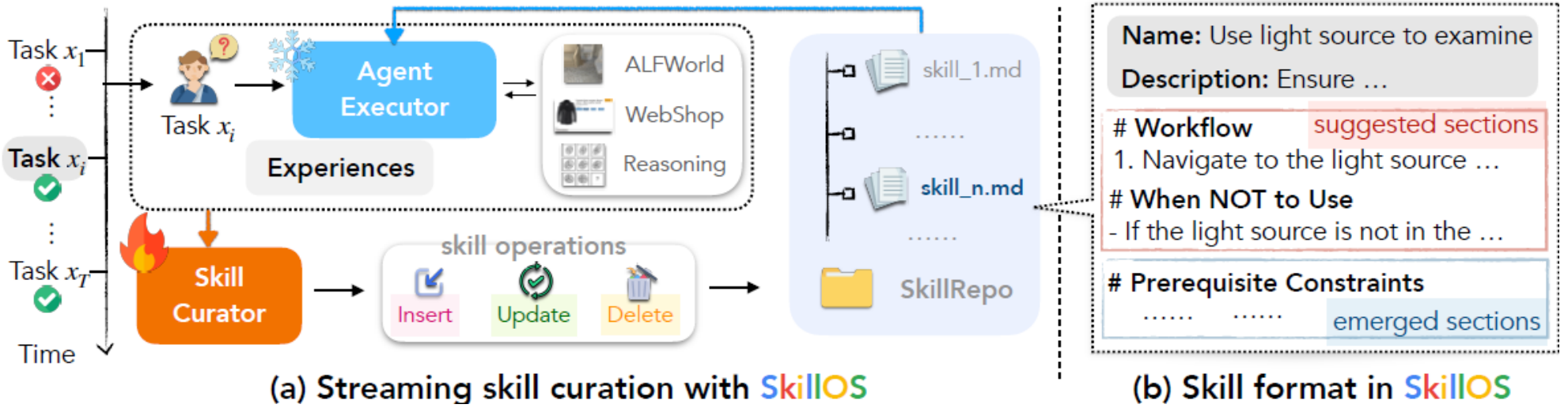
Experiment: ReasoningBank on Web Arena Benchmark

Models	Shopping (187)		Admin (182)		Gitlab (180)		Reddit (106)		Multi (29)		Overall (684)	
	SR	*Step*	*SR*	*Step*	*SR*	*Step*	*SR*	*Step*	*SR*	*Step*	*SR*	*Step*
Gemini-2.5-flash												
No Memory	39.0	8.2	44.5	9.5	33.9	13.3	55.7	6.7	10.3	10.0	40.5	9.7
Synapse	40.6	7.0	45.1	9.1	35.6	13.0	59.4	6.5	10.3	10.5	42.1	9.2
AWM	44.4	7.0	46.7	8.8	37.2	13.2	62.3	6.1	3.4	**7.7**	44.1	9.0
REASONINGBANK	**49.7**	**6.1**	**51.1**	**8.2**	**40.6**	**12.3**	**67.0**	**5.6**	**13.8**	8.8	**48.8**	**8.3**
Gemini-2.5-pro												
No Memory	45.5	7.6	51.1	8.7	35.0	11.6	71.7	6.0	6.9	8.8	46.7	8.8
Synapse	46.5	6.6	52.2	8.9	38.3	11.3	68.9	5.9	6.9	9.0	47.7	8.5
AWM	48.1	6.4	49.3	9.8	40.0	11.2	68.9	6.4	3.4	9.3	47.6	8.7
REASONINGBANK	**51.9**	**6.0**	**56.6**	**7.7**	**44.4**	**9.8**	**80.2**	**5.1**	**13.8**	**8.2**	**53.9**	**7.4**
Claude-3.7-sonnet												
No Memory	38.5	6.1	49.5	8.4	36.7	10.6	53.8	5.5	0.0	11.6	41.7	8.0
Synapse	39.6	5.8	50.5	8.5	38.0	10.0	53.8	6.1	0.0	11.8	42.6	7.9
AWM	39.6	7.2	47.8	9.3	34.6	10.9	52.8	7.0	0.0	12.4	40.8	8.9
REASONINGBANK	**44.9**	**5.6**	**53.3**	**7.6**	**41.1**	**9.5**	**57.5**	**5.2**	**3.4**	**10.5**	**46.3**	**7.3**

Success rate (SR ↑)
# of steps (Step ↓)
3 backbone LLMs

- ReasoningBank consistently outperforms baselines across LLM backbones on all datasets

SkillOS: Learning Skill Curation for Self-Evolving Agents



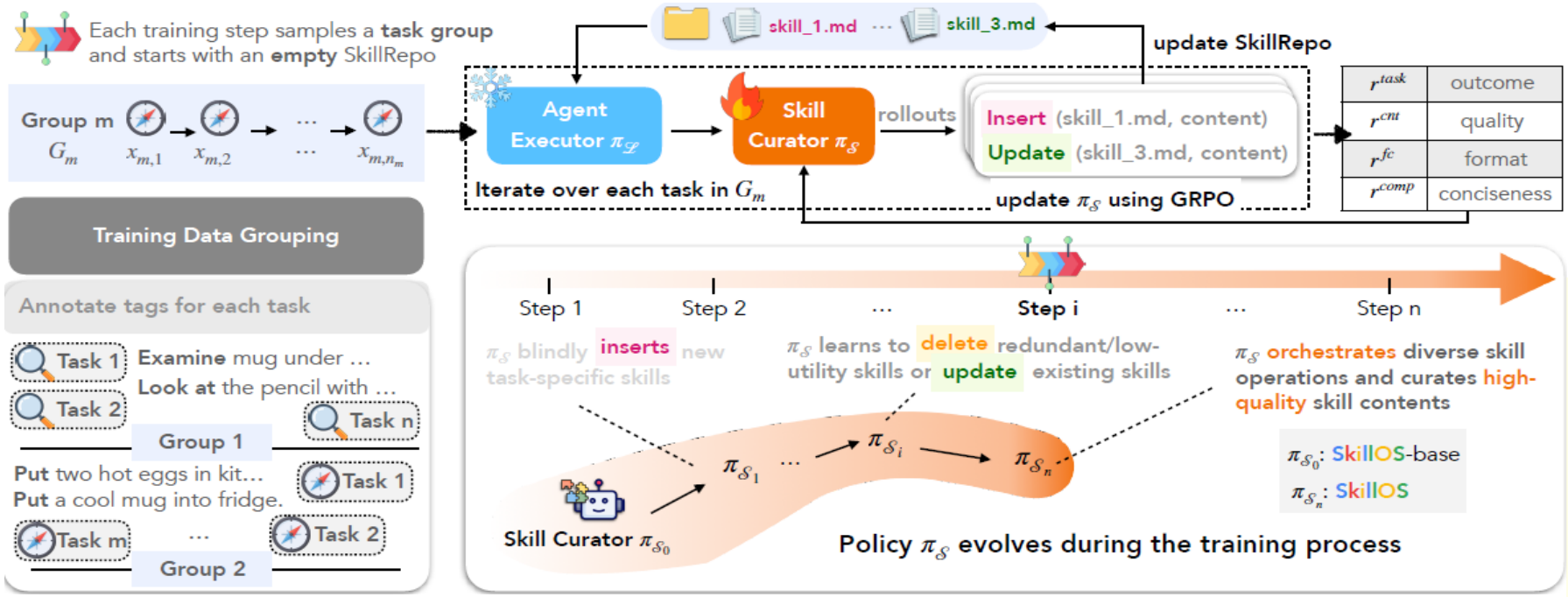
(a) Streaming skill curation with SkillOS

(b) Skill format in SkillOS

Siru Ouyang, Jun Yan, Yanfei Chen, Rujun Han, Zifeng Wang, Bhavana Dalvi Mishra, Rui Meng, Chun-Liang Li, Yizhu Jiao, Kaiwen Zha, Maohao Shen, Vishy Tirumalashetty, George Lee, Jiawei Han, Tomas Pfister, Chen-Yu Lee, "SkillOS: Learning Skill Curation for Self-Evolving Agents", arXiv:2605.06614

- Traditional LLM-based agents are often one-off problem solvers that do not learn from past experiences, limiting their long-term adaptability and performance in streaming or sequential task settings.
- SkillOS: Enable agents to self-evolve by accumulating, refining, and reusing procedural knowledge (skills) from experience, thus improving future task performance
- SkillOS pairs a frozen ***Agent Executor*** with a trainable ***Skill Curator***. The executor retrieves relevant skills from SkillRepo to act; the curator edits the repo (insert/update/delete) based on the resulting experiences
- **Skill Representation:** Each skill is stored as a Markdown file with: (1) **YAML Frontmatter:** contains the skill name and a concise, actionable description; (2) **Markdown Body:** provides instructions, workflows, constraints, and reusable heuristics.

SkillOS: Learning Skill Curation for Self-Evolving Agents



SkillOS training pipeline:
Each step samples a group of related tasks and initializes an empty SkillRepo. πS is optimized with composite rewards, enabling self-evolution.

Action Guidelines
1. Analyze the agent trajectory and its result. Identify what went well and what didn't.
2. If the trajectory is correct, extract reusable knowledge or skills. If it is incorrect, identify the failure point and extract skills that can help fix the issue.
3. Compare the extracted skills with past skills. Determine whether to insert a new skill, update an existing skill, or delete an existing skill using the following tools.

- **Experience-Driven RL Training:** The skill curator is trained using RL, with feedback signals derived from the executor's performance on future related tasks.
- **Grouped Task Streams:** Training instances are constructed as groups of related tasks, so skills learned from earlier tasks are evaluated by their impact on later tasks in the group
- **Composite Reward Function:** Task outcome reward; function call reward; compression reward; content quality reward

SkillOS: Performance Study

Methods	Curator π_S	Pick (35)	Look (13)	Clean (27)	Heat (16)	Cool (25)	Pick2 (24)	Avg. SR (140)	Steps
Executor $\pi_\mathcal{L}$: Qwen3-8B									
No Memory	None	78.1 1.6	46.2 7.7	33.3 13.4	37.5 10.8	29.3 6.1	47.2 6.4	47.9 1.2	21.1
ReasoningBank	❄ Qwen3-8B	83.8 0.0	48.7 7.2	49.4 16.2	39.6 4.4	41.3 8.5	54.2 8.8	55.7 3.1	20.1
MemP	❄ Qwen3-8B	80.0 5.7	43.6 4.4	24.7 4.3	33.3 3.6	38.7 6.1	48.6 6.4	49.7 0.7	21.0
SkillOS-base	❄ Qwen3-8B	79.0 8.7	41.0 4.4	45.7 4.3	37.5 9.5	38.7 4.0	55.6 2.1	53.1 2.5	20.4
SkillOS-gemini	❄ Gemini-2.5-Pro	77.1 6.0	53.8 6.1	37.0 6.4	37.5 9.5	36.0 3.2	50.0 6.7	50.7 3.6	20.8
SkillOS	🔥 Qwen3-8B	**85.7** 3.3	**56.4** 7.7	**54.3** 8.6	**43.8** 9.5	**46.7** 2.3	**62.5** 6.4	**61.2** 4.6	**18.9**
Executor $\pi_\mathcal{L}$: Qwen3-32B									
No Memory	None	80.0 2.9	69.2 0.0	45.6 7.7	37.5 16.5	42.7 6.1	43.1 2.4	54.5 2.5	20.3
ReasoningBank	❄ Qwen3-8B	86.7 3.0	71.8 5.4	50.6 6.3	45.8 13.3	52.0 8.9	51.4 5.1	61.4 2.5	18.7
MemP	❄ Qwen3-8B	80.0 2.9	76.9 0.0	44.4 7.4	37.5 10.8	42.7 2.3	47.2 6.4	55.7 3.7	20.0
SkillOS-base	❄ Qwen3-8B	82.9 2.9	69.2 11.8	48.1 2.1	50.0 9.7	48.0 14.4	52.8 11.0	59.8 3.0	19.2
SkillOS-gemini	❄ Gemini-2.5-Pro	**97.1** 3.0	76.9 5.4	55.6 6.0	43.8 11.3	40.0 5.7	54.2 4.9	63.6 4.2	18.1
SkillOS	🔥 Qwen3-8B	91.4 3.3	**76.9** 4.4	**59.3** 8.6	**56.3** 12.5	**57.3** 10.1	**62.5** 4.2	**68.6** 5.7	**17.3**
Executor $\pi_\mathcal{L}$: Gemini-2.5-pro									
No Memory	None	90.5 3.2	66.7 5.1	48.1 10.2	39.6 17.1	68 7.4	68.1 3.8	66.4 2.0	17.7
ReasoningBank	❄ Qwen3-8B	91.4 3.4	61.5 4.1	63.0 9.3	39.6 10.3	70.7 3.2	76.4 8.5	71.4 2.9	16.0
MemP	❄ Qwen3-8B	95.2 2.1	74.4 6.8	61.7 7.6	56.3 12.4	76.0 6.2	68.1 8.5	74.3 3.4	15.2
SkillOS-base	❄ Qwen3-8B	91.4 1.6	69.2 7.7	56.8 5.7	54.2 13.7	72.0 4.0	66.7 11.0	70.7 3.0	16.3
SkillOS-gemini	❄ Gemini-2.5-Pro	94.3 5.7	69.2 0.0	77.8 5.7	75.0 16.5	80.0 12.2	66.7 2.4	79.3 2.6	14.9
SkillOS	🔥 Qwen3-8B	**95.2** 2.9	71.8 7.7	74.1 13.0	72.9 10.1	77.3 6.1	77.8 10.0	**80.2** 3.1	**14.8**

Experiment on ALFWorld benchmark. Success rate (SR ↑) and the number of steps (Steps ↓) are reported on 6 subsets with 3 different frozen executors.

Further performance study shows:

Effectiveness: SkillOS outperforms both memory-free and memory-based baselines in success rate and efficiency across multiple benchmarks (e.g., ALFWorld, WebShop, mathematical reasoning tasks).

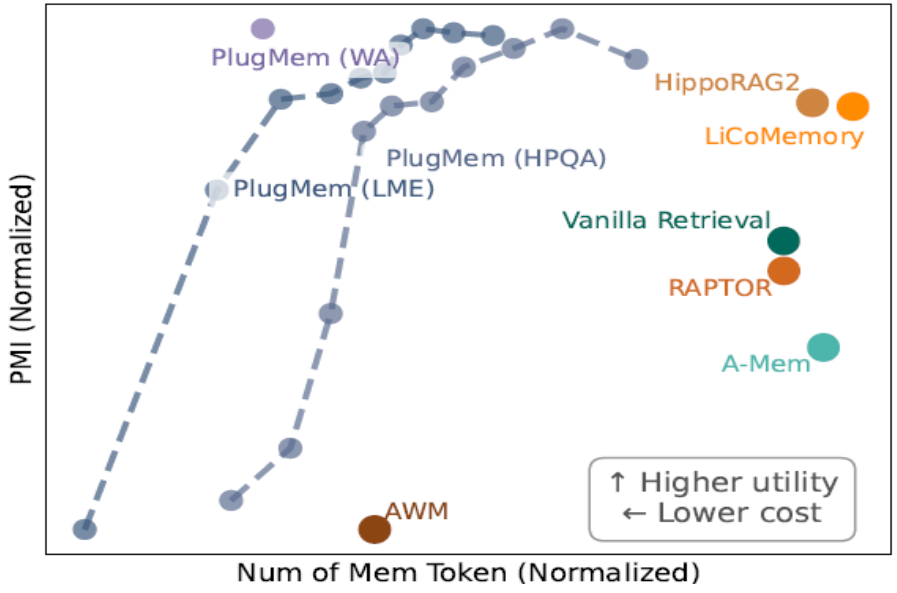
Efficiency: Achieves higher task success with fewer interaction steps, indicating more targeted and actionable skill use.

Generalization: The trained skill curator transfers well across different executors and task domains, showing modularity and robustness.

Skill Evolution: Over time, SkillRepo evolves to contain more structured, higher-level meta-skills and actionable strategies, not just verbatim task trajectories.

PlugMem: A Task-Agnostic Plugin Memory Module for LLM Agents

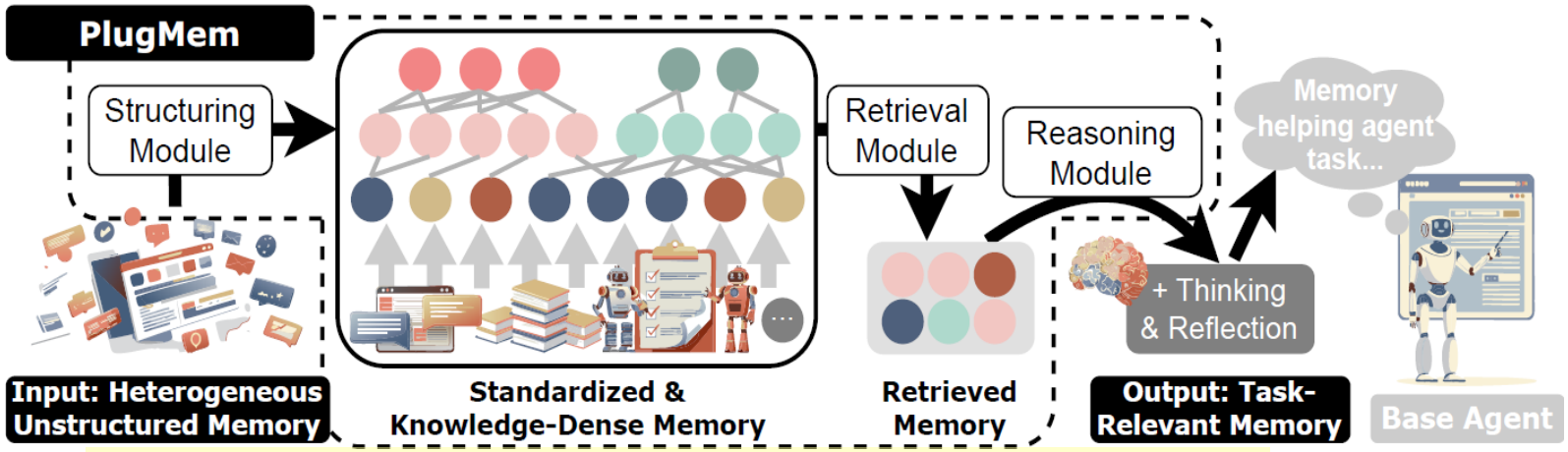
- Existing memory design: Task-specific (non-transferable) or task-agnostic but less effective
  - Observation: Decision-relevant info is concentrated as abstract knowledge
- PLUGMEM: A task-agnostic plugin memory module that can be attached to arbitrary LLM agents without task-specific redesign
- Representation: Structure episodic memories into a compact, extensible knowledge-centric memory graph that explicitly represents propositional and prescriptive knowledge
  - Efficient memory retrieval and reasoning over task-relevant knowledge, rather than verbose raw trajectories
  - Treat knowledge as the unit of memory access and organization instead of entities or text chunks
- Evaluation: Across three heterogeneous benchmarks
  - PLUGMEM consistently outperforms task-agnostic baselines and exceeds task-specific memory designs, while achieving the highest info density under a unified information-theoretic analysis.



Ke Yang, Zixi Chen, Xuan He, Jize Jiang, Michel Galley, Chenglong Wang, Jianfeng Gao, Jiawei Han, ChengXiang Zhai, "PlugMem: A Task-Agnostic Plugin Memory Module for LLM Agents", in ICML, July 2026

PlugMem Transforms Raw Episodic Memory into Structured, Knowledge-dense Representations

- Effective agent memory should actively transform raw *episodic memory* into *structured, knowledge-dense representations*
  - **Episodic memory** (detailed records of experience) vs. *knowledge-level memory* (**semantic memory** (knowing that; factual propositions) & **procedural memory** (knowing how; action-oriented prescriptions))
- PlugMem operates on knowledge units (i.e., propositions and prescriptions) which form the fundamental units of memory access and manipulation
  - vs. GraphRAG (operating on entities and relations)



PLUGMEM performs memory-to-knowledge abstraction & supports the unified management of multiple key memory types across agentic tasks

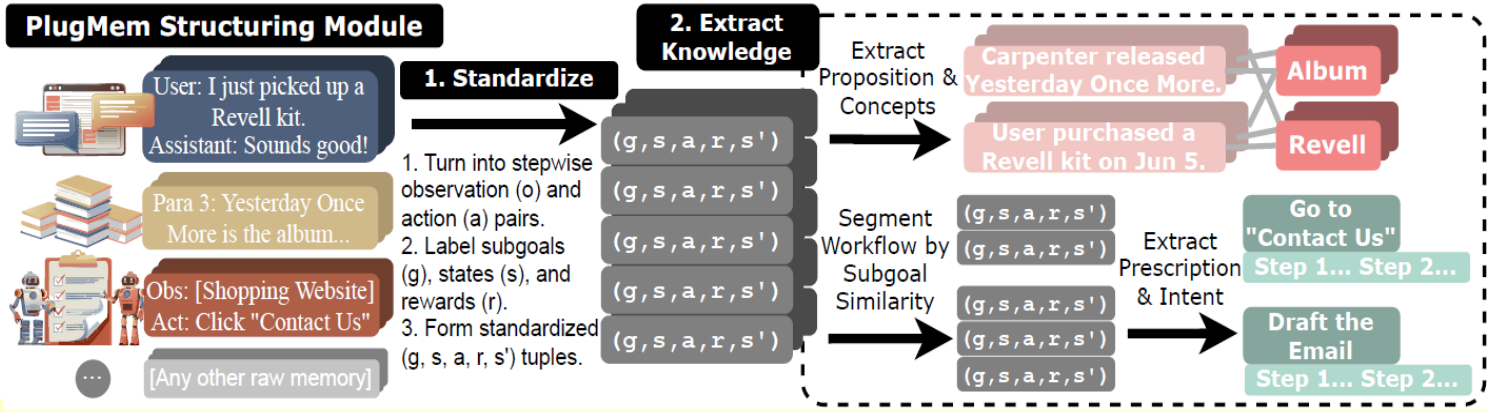
PlugMem consists of (1) a ***structuring module*** that standardizes heterogeneous raw memories and extracts propositional and prescriptive knowledge through hierarchical abstraction, organizing them into a memory graph; (2) a ***retrieval module*** that selects task-relevant subgraphs; and (3) a ***reasoning module*** that further adapts and compresses retrieved knowledge for the base agent

The Structuring Module of PlugMem

- Structuring module: Standardize heterog. episodic memories and induce propositional and prescriptive knowledge
  - Retain episodic traces as verifiable evidence

Methods:

- Standardize heterogeneous interaction traces into a **unified episodic representation**

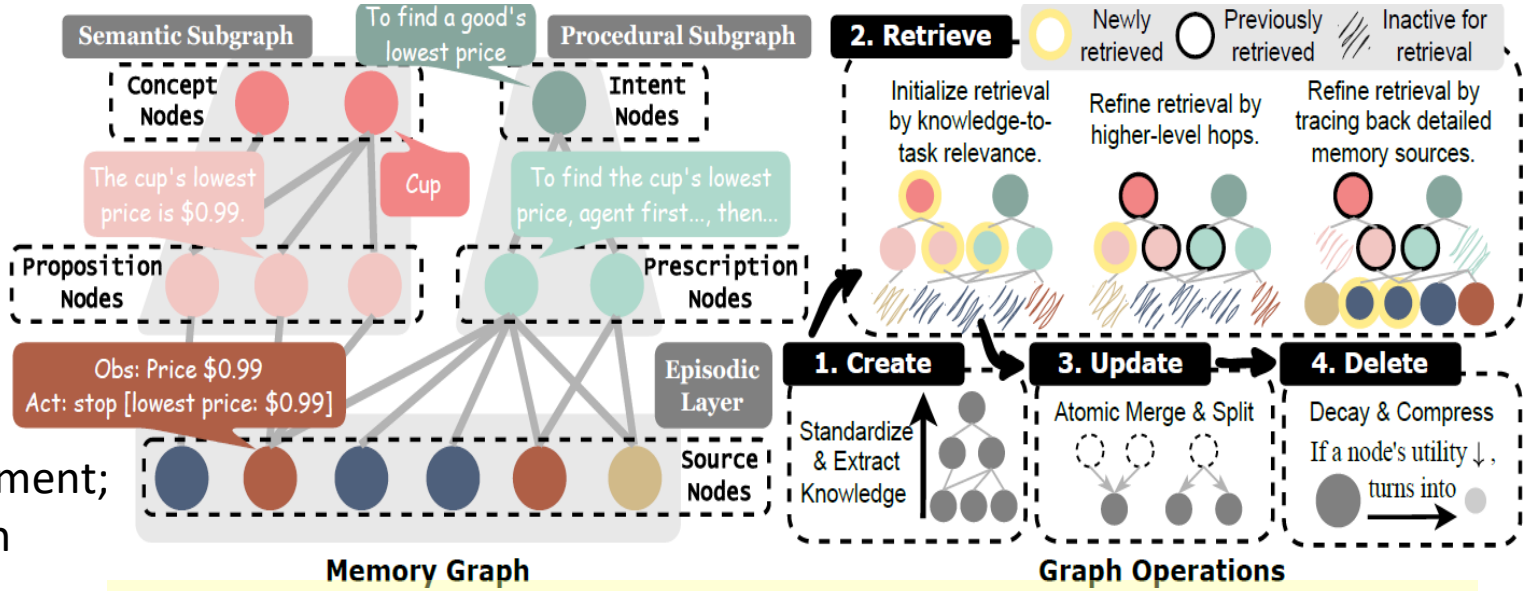


The structuring module transforms heterog. memory into a formalized knowledge-dense memory graph: From **(o: observation)** to **(g: subgoal, s:state, a:action, r:reward, s')**

  - Rep. a raw interaction trace as a sequence of observ.-action pairs: (o, a), then standardized as $e_t = (o_t, s_t, a_t, r_t, g_t)$
  - Aggregating all standardized steps yields an episodic memory sequence $[e_t]_1^T$
- Induce propositional (semantic) & prescriptive knowledge that can be independently indexed and reused across tasks
  - Use an LLM to extract a set of atomic propositions that describe salient facts implied by the interaction
  - Each proposition is accompanied by a set of associated concepts, which serve as semantic tags for indexing
    - Ex. "*Tam Sventon, known in Swedish as Ture Sventon, is a fictional private detective based in Stockholm.*"
      - → {*Tam Sventon, fictional private detective, Stockholm*}
  - To ensure extraction quality, use coreference resolution, proposition deduplication, and length control
  - The extracted propositions and concepts are stored in a semantic graph GS

The Retrieval Module of PlugMem

- **Procedural Memory**: Extract reusable action strategies from episodic trajectories to support future decision making
  - Segment the trajectory into coherent sub-trajectories by detecting boundaries
  - For each trajectory segment, using LLM induce a compact (*intent*, *prescription*) pair
  - *Intent:* the objective pursued within the segment; *prescription:* an environment agnostic action workflow that captures the key steps



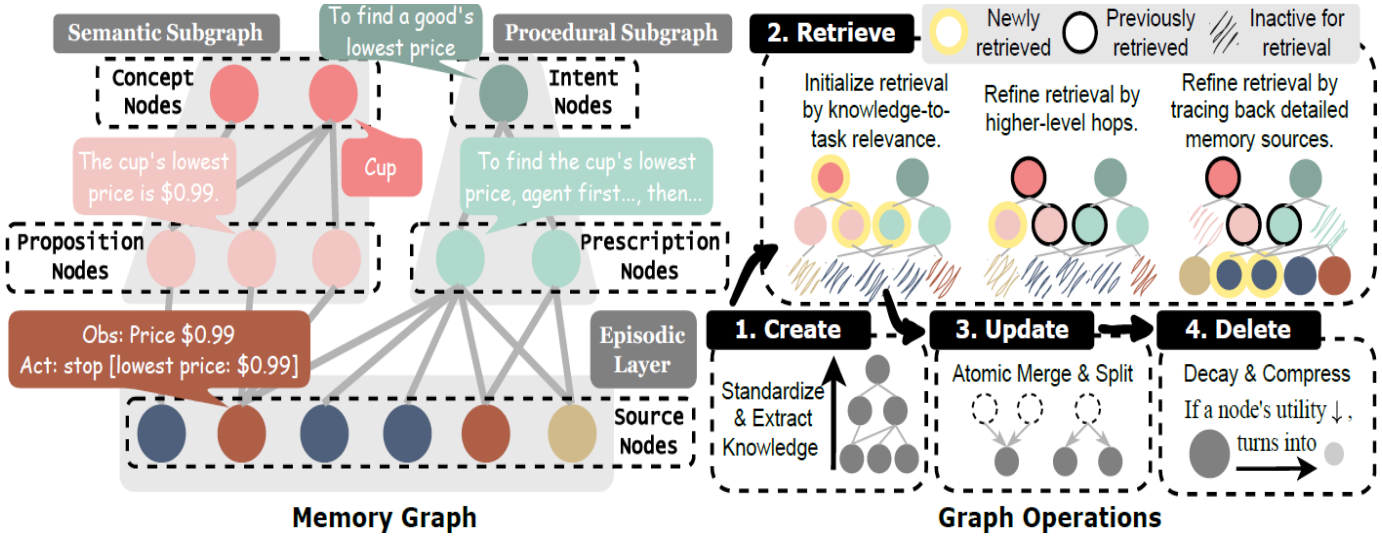
Knowledge-centric memory graph design and the standard graph operations

Ex. of *prescription:* "To identify the lowest price of an item, search for the item using the search bar, sort the results by price and verify the minimum across variants."

- The Structuring stage constructs an episodic graph G^E, a semantic graph G^S, and a procedural graph G^P
  - Both G^S and G^P maintain explicit provenance links to G^E, enabling verifiable grounding info.
- Given a task description or query Q, an **LLM-based retriever** first determines which memory types to emphasize: *episodic, semantic*, or *procedural*.
- Retrieval begins by encoding Q into an embedding q and scoring it against all low-level nodes (i.e., proposition or prescription nodes) to initialize a candidate set C_0

PlugMem: The Reasoning Module and Overall Operations

- **Reasoning module**: Transform retrieved memory into immediately actionable guidance for the playing agent
  - Retrieved memory may contain multiple overlapping or verbose descriptions of past interactions that are individually relevant but collectively redundant for the current decision



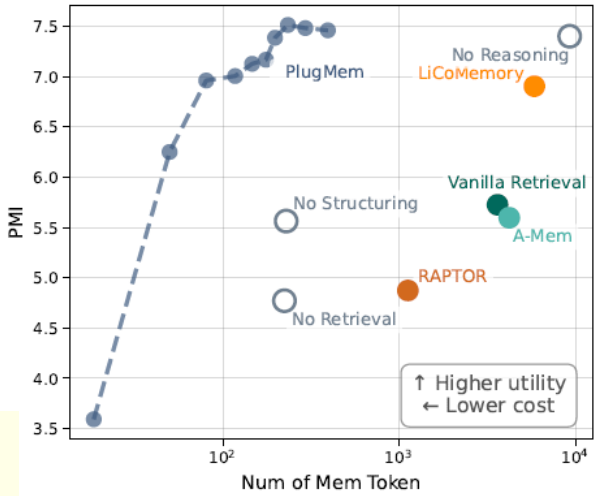
- The reasoning module leverages the LLM to aggregate and condense such info into a compact, task-aligned representation, distilling the shared signal across messages into a single actionable summary

- **Summary**: PLUGMEM standardizes episodic memory, extracts semantic and procedural knowledge, organizes them into structured memory graphs, and enables retrieval and reasoning over stored experience to support downstream decision making

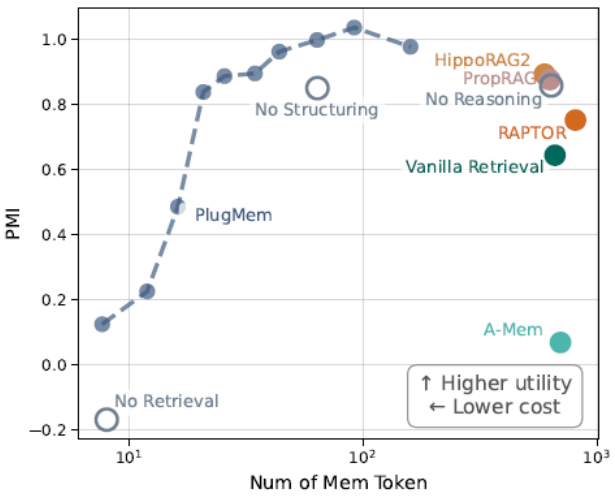
- At the system level, PLUGMEM supports 4 **basic memory graph operations**: i) **create**, which inserts newly observed episodic experience into structured memory, ii) **retrieve**, which retrieves relevant semantic, procedural, or episodic memory given a task or query, iii) **update**, which revises existing memory entries when new evidence becomes available, and iv) **delete**, which removes obsolete or low-utility memory.

PlugMem: Performance Study

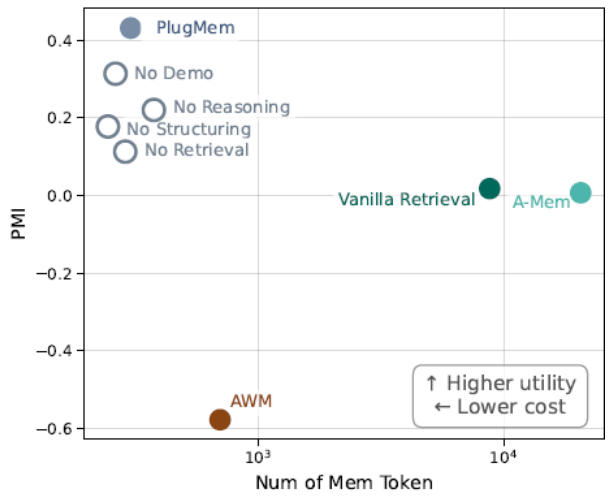
PLUGMEM consistently achieves a more favorable utility-cost trade-off, dominating prior approaches by providing higher decision-relevant utility under smaller memory budgets across benchmarks



LongMemEval



HotpotQA



WebArena

Results on LongMemEval. #TokAvg. is the average length of memory tokens. Experiments use NV-Embed-v2 (abbreviated as NVE) as the embedding model for retrieval, and Qwen2.5-32B (Q32)/gpt-4o (4o) as base LLMs for structuring and reasoning.

Method	Emb	LLM	Acc.	#Tok$_{Avg.}$	Info. Density
Vanilla Baseline					
No Context	-	4o	8.8	-	-
All Context	-	4o	63.0	115K	4.8e-5
Task-Agnostic					
Vanilla Retrieval	NVE	4o	65.2	3612.6	1.6e-3
A-Mem†	NVE	4o	60.4	4211.4	1.3e-3
Task-Specific					
RAPTOR†	NVE	4o	53.0	1042.2	4.7e-3
Zep*	BGE-m3	4o	71.2	1600	-
LiCoMemory†	NVE	4o	74.1	5846.0	1.2e-3
Ours					
PLUGMEM	NVE	Q32 + 4o	**82.8**	**233.9**	**3.2e-2**

Results on HotPotQA

Mem. Method	Emb	LLM	EM	F1	#Tok$_{Avg.}$	Info. Density
Vanilla Baseline						
No Context	-	Q32	22.1	31.0	-	-
Gold Context	-	Q32	69.2	82.1	86.5	1.6e-1
Task-Agnostic						
Vanilla Retrieval	NVE	Q32	51.7	62.7	659.2	1.2e-2
A-Mem	NVE	Q32	43.8	53.6	695.6	1.2e-2
Task-Specific						
GraphRAG*	NVE	L70	55.2	68.6	-	-
RAPTOR	NVE	Q32	56.7	69.7	806.3	1.1e-2
PropRAG	NVE	Q32	57.8	72.1	626.1	1.9e-2
HippoRAG2	NVE	Q32	60.0	73.3	595.1	1.9e-2
Ours						
PLUGMEM	NVE	Q32	**61.4**	**74.1**	**81.6**	**1.4e-1**

Ablation Study on LongMemEva

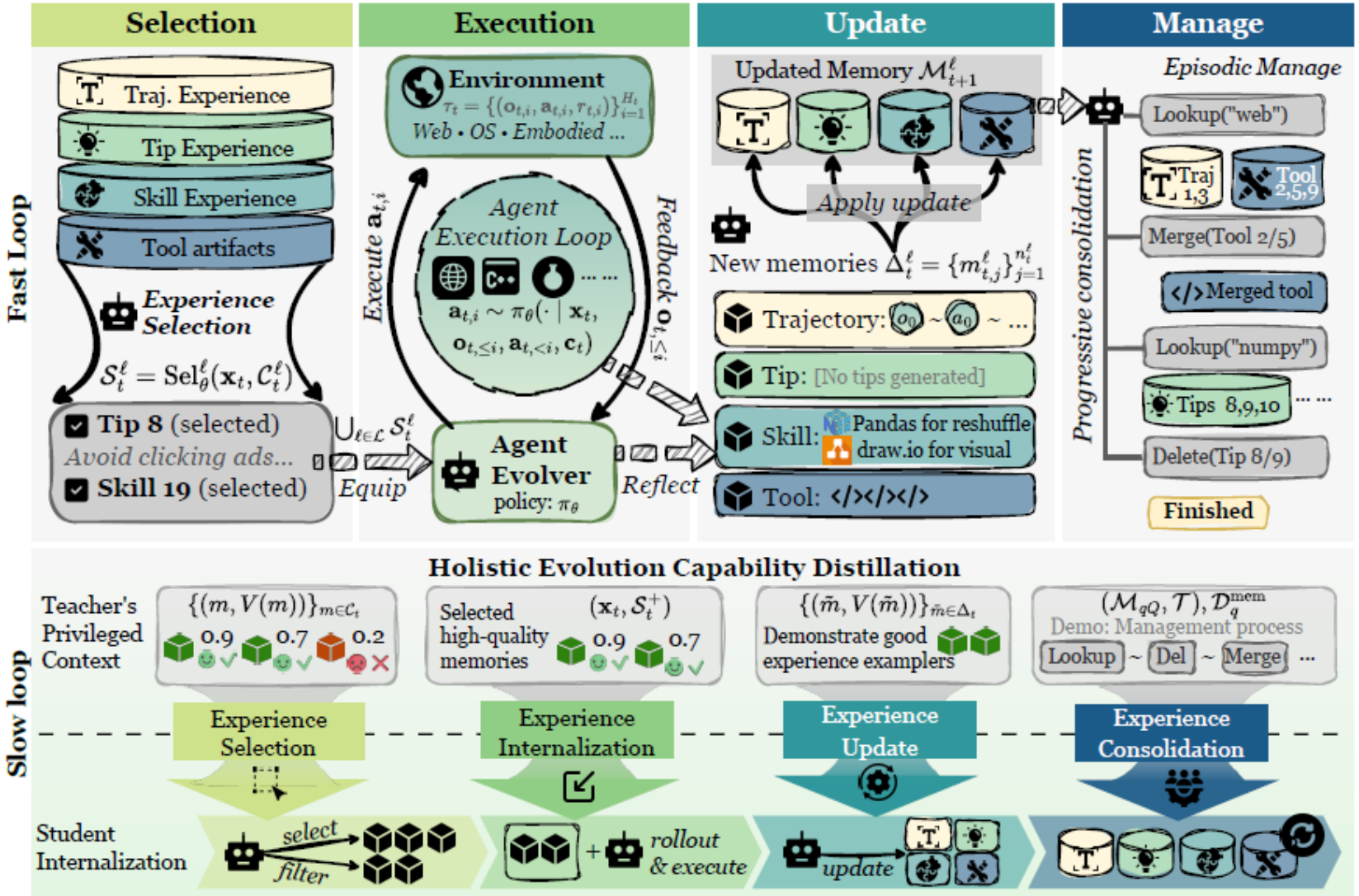
Method	Emb	LLM	Acc.	#Tok$_{Avg.}$	Info. Density
PLUGMEM	NVE	Q32 + 4o	**82.8**	233.9	**3.2e-2**
No Structuring	NVE	Q32 + 4o	63.6	228.8	2.4e-2
No Retrieval	-	Q32 + 4o	55.8	**223.6**	2.1e-2
No Reasoning	NVE	Q32	81.6	9267.9	8.0e-4

Ablation Study on HotPotQA

Method	Emb	LLM	EM	F1	#Tok$_{Avg.}$	Info. Density
PLUGMEM	NVE	Q32	**61.4**	**74.1**	**81.6**	**1.4e-1**
No Structuring	NVE	Q32	51.4	62.0	116.7	6.8e-2
No Retrieval	-	Q32	20.0	24.3	8.0[1]	-3.8e-1
No Reasoning	NVE	Q32	59.3	71.8	635.1	1.7e-2

OPD-Evolver: Cultivating Holistic Agent Evolver via On-Policy Distillation

- OPD-Evolver, a slow-fast co-evolution framework that cultivates an agent evolver through on-policy self-distillation
- Fast loop: interact with a 4-level memory hierarchy to read, use, write & maintain experience for rapid test-time evolution
- Slow loop: Outcome-calibrated memory attribution and privileged hindsight distill the 4 abilities into the deployable policy



(Top) The fast loop lets the agent interact with environments and a four-level memory hierarchy; (Down) the slow loop converts outcome-calibrated hindsight into on-policy self-distillation signals

Guibin Zhang, Xun Xu, Yanwei Yue, Zikun Su, Wangchunshu Zhou, Xiaobin Hu, Shuicheng Yan, "**OPD-Evolver**: Cultivating Holistic Agent Evolver via On-Policy Distillation", ArXiv: 2606.17628

OPD-Evolver: Cultivating Holistic Agent Evolver via On-Policy Distillation

- Task: Systematically transforms interaction history/feedback into persistent future improvements
  - Unlike **memory-augmented agents (**store trajectories, reflections, tips, or lessons and inject them into later prompts); **skill-augmented agents** (distill experience into reusable strategies, tools, or procedures); or **optimize only one fragment** (e.g., retrieving experience)
- A qualified agent evolver should have 4 inter-coupled capabilities:

  ❶ **experience selection** identifies useful memories from a growing and noisy repository;

  ❷ **experience-grounded execution** converts selected experience into effective multi-turn actions;

  ❸ **experience writing** extracts reusable knowledge from new trajectories and feedback; and

  ❹ **experience management** scores, consolidates, updates, and retires memories over time.
- **Fast evolution loop**: operates over a 4-level memory substrate of trajectories, tips, skills, and tools: it selects task-relevant mem before execution, acts with them, then writes and periodically maintains mem from trajectories, rewards, and feedback. Convert into supervision for what should be selected, used, written, and preserved.
- **Slow evolution loop**: On-policy self-distillation: Observes attribution-enriched evidence unavailable at deployment (candidate memory value, trajectory snippets, future utility of written memories, and repository-level diagnostics), enabling OPD-Evolver to acquire transferable lifecycle-level competence for self-improvement.

OPD-Evolver: Performance

LLM	Method	LifelongAgentBench		MemoryArena		AMA-Bench			InterCode		
		DB	OS	Math	Physics	CI	SU	SA	Bash	CTF	SQL
Qwen3.5-397B-A17B		86.00	63.00	3.98	4.65	57.21	58.73	51.41	51.34	56.00	62.74
Step-3.5-Flash (196B)		81.00	58.00	1.98	2.33	49.50	46.99	48.88	44.20	48.00	59.87
Qwen3-4B	No Memory	65.00	36.50	2.40	2.33	24.83	27.67	29.73	30.36	26.00	38.85
	ExpeL	62.50	34.50	1.98	1.74	26.17	24.88	28.49	30.80	33.00	34.08
	AWM	63.00	36.00	1.69	3.49	28.36	30.14	33.49	27.68	33.00	38.54
	Cheatsheet	69.00	38.50	2.97	1.74	25.34	30.91	30.37	33.04	25.00	37.90
	Memp	73.00	41.00	4.66	0.00	32.38	28.44	31.81	31.70	27.00	43.31
	ReasoningBank	70.00	38.00	3.81	1.16	27.35	29.68	30.97	31.25	25.00	43.95
	EvolveR	71.50	46.50	1.27	2.91	26.51	24.42	30.01	33.93	31.00	44.90
	MemEvolve	72.00	44.00	1.84	2.33	29.53	31.53	32.73	33.93	29.00	44.59
	OPD-Evolver-4B	74.00	49.50	5.51	4.07	32.89	34.93	34.90	36.16	34.00	45.86
Qwen3.5-9B	No Memory	75.50	52.50	5.51	5.23	40.10	47.14	45.63	41.52	44.00	55.73
	ExpeL	74.00	46.50	8.19	8.72	41.11	43.74	44.67	40.18	50.00	52.23
	AWM	72.50	44.00	7.34	11.05	43.96	48.38	48.00	39.73	49.00	51.59
	Cheatsheet	79.50	52.50	6.64	5.23	40.77	49.30	46.19	41.96	47.00	56.05
	Memp	82.00	56.00	4.80	6.40	45.81	47.14	46.59	44.20	50.00	58.28
	ReasoningBank	80.50	55.00	4.94	6.98	42.28	48.38	47.04	43.75	45.00	57.96
	EvolveR	82.50	59.50	6.07	7.56	41.78	42.81	46.11	44.64	52.00	60.51
	MemEvolve	81.00	61.00	4.24	9.88	44.30	49.77	47.20	45.98	53.00	61.15
	OPD-Evolver-9B	84.50	65.00	10.88	11.63	47.32	53.94	52.92	49.55	57.00	64.01

Task success metric across self-evolving agent benchmarks. For AMA-Bench, CI: Causal Inference, SU: State Updating, SA: State Updating.

Method	MiniHack			InterCode		
	Room	Maze	KeyRoom	Bash	CTF	SQL
Vanilla	80.39	19.61	0.00	55.73	44.00	41.25
SFT	82.35	17.65	0.00	59.87	49.00	44.64
GRPO	100.00	23.53	3.92	63.69	58.00	47.77
Skill0	94.12	25.49	3.92	62.10	54.00	47.32
MemRL	96.08	19.61	1.96	61.15	50.00	44.20
Complementary RL	96.88	20.85	5.16	63.20	55.00	48.10
OPD-Evolver	98.04	27.45	9.80	64.01	59.00	49.55

Comparison with training-based agent improvement methods

Variant	Bash	CTF	SQL
OPD-Evolver-4B	36.16	34.00	45.86
w/o Slow Evolution	32.14	28.00	39.17
w/o Memory Attribution	31.20	26.69	38.50
w/o Selection	35.27	32.00	42.04
w/o Writing Distill.	34.38	29.00	41.08
w/o Maintenance	35.30	30.10	43.51

Ablation study on InterCode (Bash/CTF/SQL) with OPD-Evolver-4B. “Writing Distill.” denotes the exclusion of self-distilling experience writing capability

References (I)

- Aadharsh Aadhithya, Sachin Kumar, and KP Soman, “Enhancing Long-Term Memory using Hierarchical Aggregate Tree for Retrieval Augmented Generation”, arXiv:2406.06124 (2024)
- Petr Anokhin, Nikita Semenov, Artyom Sorokin, Dmitry Evseev, Mikhail Burtsev, and Evgeny Burnaev, “AriGraph: Learning Knowledge Graph World Models with Episodic Memory for LLM Agents”, arXiv:2407.04363 (2024)
- Anthropic, 2026, “How Claude remembers your project”, https://code.claude.com/docs/en/memory (accessed 2026-04-26)
- A. Asai, et al., *“Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection,”* ICLR 2024
- DeepSeek-AI, *“DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via RL,”* arXiv 2025
- Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su, “From RAG to Memory: Non-Parametric Continual Learning for Large Language Models, in ICML’2025, July 2025
- Zexue He, Yu Wang, Churan Zhi, Yuanzhe Hu, Tzu-Ping Chen, Lang Yin, Ze Chen, Tong Arthur Wu, Siru Ouyang, Zihan Wang, Jiaxin Pei, Julian McAuley, Yejin Choi, and Alex Pentland, “MemoryArena: Benchmarking Agent Memory in Interdependent Multi-Session Agentic Tasks”, arxiv:2602.16313 (2026)
- Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo, “HiAgent: Hierarchical Working Memory Management for Solving Long-Horizon Agent Tasks with Large Language Model”, in ACL 2025, July 2025
- Wei-Chieh Huang, et al, "Rethinking Memory Mechanisms of Foundation Agents in the Second Half: A Survey", ArXiv:2602.06052
- P. Jiang, et al., *“DeepRetrieval: Hacking Real Search Engines and Retrievers with LLMs via RL,”* COLM 2025
- P. Jiang, et al., *“s3: You Don’t Need That Much Data to Train a Search Agent via RL,”* EMNLP 2025
- P. Jiang, et al., *“Harness-1: A Stateful Harness for Training Long-Horizon Search Agents,”* arXiv 2026
- P. Jiang, et al., *“Adaptation of Agentic AI: A Survey of Post-Training, Memory, and Skills,”* arXiv 2026
- Z. Jiang, et al., *“Active Retrieval Augmented Generation (FLARE),”* EMNLP 2023
- B. Jin, et al., *“Search-R1: Training LLMs to Reason and Leverage Search Engines with RL,”* COLM 2025
- Zixuan Ke, Yifei Ming, Austin Xu, Ryan Chin, Xuan-Phi Nguyen, Prathyusha Jwalapuram, Jiayu Wang, Semih Yavuz, Caiming Xiong, and Shafiq Joty, “Mas-orchestra: Understanding and improving multiagent reasoning through holistic orchestration and controlled benchmarks”, arXiv:2601.14652 (2026)

References (II)

- Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn, “Meta-Harness: End-to-End Optimization of Model Harnesses”, arxiv:2603.28052
- Xiang Lei, Qin Li, and Min Zhang, “D-smart: Enhancing LLM dialogue consistency via dynamic structured memory and reasoning tree,” arXiv:2510.13363 (2025)
- X. Li, et al., *“Search-o1: Agentic Search-Enhanced Large Reasoning Models,”* EMNLP 2025
- Zaijing Li, Yuquan Xie, Rui Shao, Gongwei Chen, Dongmei Jiang, and Liqiang Nie, “Optimus-1: Hybrid Multimodal Memory Empowered Agents Excel in Long-Horizon Tasks”, NeurIPS’2024, Dec. 2024
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candes, and Tatsunori Hashimoto, “s1: Simple test-time scaling”, arXiv:2501.19393 (2025)
- OpenAI, 2026. Harness engineering: leveraging codex in an agent-first world, openai.com/index/harness-engineering (accessed: 2026-04-26)
- L. Ouyang, et al., *“Training Language Models to Follow Instructions with Human Feedback,”* NeurIPS 2022
- Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, Tomas Pfister, “ReasoningBank: Scaling Agent Self-Evolving with Reasoning Memory”, in ICLR 2026
- Siru Ouyang, Jun Yan, Yanfei Chen, Rujun Han, Zifeng Wang, Bhavana Dalvi Mishra, Rui Meng, Chun-Liang Li, Yizhu Jiao, Kaiwen Zha, Maohao Shen, Vishy Tirumalashetty, George Lee, Jiawei Han, Tomas Pfister, Chen-Yu Lee, "SkillOS: Learning Skill Curation for Self-Evolving Agents", arXiv:2605.06614
- C. Packer, et al., *“MemGPT: Towards LLMs as Operating Systems,”* arXiv 2023
- J. S. Park, et al., *“Generative Agents: Interactive Simulacra of Human Behavior,”* UIST 2023
- O. Press, et al., *“Measuring and Narrowing the Compositionality Gap (Self-Ask),”* Findings of EMNLP 2023
- Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef, “Zep: a temporal knowledge graph architecture for agent memory”, arXiv:2501.13956 (2025)
- Alireza Rezazadeh, Zichao Li, Wei Wei, and Yujia Bao, “From Isolated Conversations to Hierarchical Schemas: Dynamic Tree Memory Representation for LLMs”, in ICLR’2025, April 2025

References (II)

- J. Schulman, et al., *"Proximal Policy Optimization Algorithms,"* arXiv 2017
- Z. Shao, et al., *"DeepSeekMath: Pushing the Limits of Mathematical Reasoning (GRPO),"* arXiv 2024
- N. Shinn, et al., *"Reflexion: Language Agents with Verbal Reinforcement Learning,"* NeurIPS 2023
- R. S. Sutton, A. G. Barto, *"Reinforcement Learning: An Introduction,"* MIT Press, 2018
- 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, "In Prospect and Retrospect: Reflective Memory Management for Long-term Personalized Dialogue Agents", in ACL'2025
- Xiangru Tang, Tianyu Hu, Muyang Ye, Yanjun Shao, Xunjian Yin, Siru Ouyang, Wangchunshu Zhou, Pan Lu, Zhuosheng Zhang, Yilun Zhao, Arman Cohan, and Mark Gerstein, "ChemAgent: Self-updating Memories in Large Language Models Improves Chemical Reasoning", in ICLR 2025
- H. Trivedi, et al., *"Interleaving Retrieval with Chain-of-Thought Reasoning (IRCoT),"* ACL 2023
- 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", in Transactions on Machine Learning Research (2024)
- Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig, "Agent Workflow Memory", in ICML 2025
- Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, Madian Khabsa, Han Fang, Yashar Mehdad, Sharan Narang, Kshitiz Malik, Angela Fan, Shruti Bhosale, Sergey Edunov, Mike Lewis, Sinong Wang, and Hao Ma, "Effective Long-Context Scaling of Foundation Models", in NAACL' 2024
- Ke Yang, Zixi Chen, Xuan He, Jize Jiang, Michel Galley, Chenglong Wang, Jianfeng Gao, Jiawei Han, ChengXiang Zhai, "PlugMem: A Task-Agnostic Plugin Memory Module for LLM Agents", in ICML 2026
- Ruozhen Yang, Yucheng Jiang, Yueqi Jiang, Priyanka Kargupta, Yunyi Zhang, Jiawei Han, "Grounding Agent Memory in Contextual Intent", in ACL 2026
- S. Yao, et al., *"ReAct: Synergizing Reasoning and Acting in Language Models,"* ICLR 2023
- Guibin Zhang, Muxin Fu, Kun Wang, Guancheng Wan, Miao Yu, and Shuicheng Yan, "G-Memory: Tracing Hierarchical Memory for Multi-Agent Systems", in NeurIPS 2025
- Guibin Zhang, Xun Xu, Yanwei Yue, Zikun Su, Wangchunshu Zhou, Xiaobin Hu, Shuicheng Yan, "**OPD-Evolver**: Cultivating Holistic Agent Evolver via On-Policy Distillation", ArXiv: 2606.17628
- Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. "Memskill: Learning and evolving memory skills for self-evolving agents", arXiv:2602.02474 (2026)

Course Coverage

- A Brief Introduction to Large Language Models
- Retrieval and Retrieval Augment Generation (RAG)
- Reasoning with Structures for LLMs
- LLM Agents and Agent Memory Technology
- Look Forward to the Future and Q&A