Multi-Agent Transactive Memory

To Eun Kim[1*] **Xuhong He**[1*] **Dishank Jain**[1*]
Ambuj Agrawal[1] **Negar Arabzadeh**[2] **Fernando Diaz**[1]
[1]Carnegie Mellon University [2]University of California, Berkeley

Abstract

The decentralized deployment of LLM agents with diverse capabilities across diverse tasks motivates infrastructure for knowledge sharing across heterogeneous agent populations. Just as search engines index human-generated artifacts to support human problem solving, retrieval systems can organize agent-generated artifacts for reuse across agent populations. We extend retrieval-augmented generation—which demonstrates the value of human-authored artifacts to individual agents—to retrieval of agent-generated artifacts supporting a population of agents. In particular, agent trajectories encode reusable procedural knowledge, yet these artifacts are typically discarded after a single use or retained only by the producing agent, forcing newly instantiated agents to repeatedly rediscover existing solutions. We propose Multi-Agent Transactive Memory (MATM), a framework for population-level storage and retrieval of agent-generated trajectories, where *producer agents* contribute trajectories to a shared repository and *consumer agents* retrieve them to improve task execution. We focus on interactive environments (ALFWorld and WebArena), where trajectories are long and encode especially rich procedural structure. Our experiments demonstrate that retrieving trajectories from MATM improves downstream task performance and reduces interaction steps without coordination or joint training. These results position MATM as a design pattern for population-level experience sharing in open agent ecosystems.

1 Introduction

As heterogeneous LLM agents are deployed across increasingly diverse domains, research on individual agent design must be complemented by methods for supporting decentralized populations of agents. The need for population-level infrastructure has motivated protocols to support agent-tool interaction (Model Context Protocol, 2026) as well as inter-agent communication (A2A Protocol, 2026). Beyond standards, tools such as search engines are beginning to be optimized for agents (Zamani et al., 2022; Salemi and Zamani, 2024b). Although retrieval-augmented generation (RAG) demonstrates the value of human-authored artifacts to individual agents, infrastructure for knowledge sharing amongst agents offers a compelling alternative. Just as search engines index human-generated artifacts to support human problem solving, retrieval systems can organize agent-generated artifacts for reuse across agent populations (Figure 1).

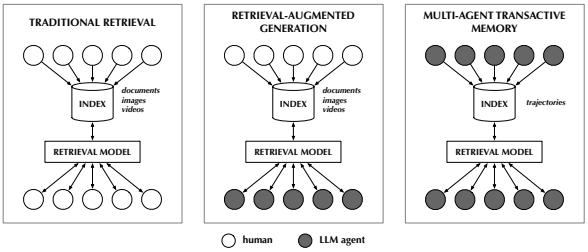

Figure 1: Multi-Agent Transactive Memory (MATM). Traditional search serves humans retrieving human-authored documents. RAG extends this to agents retrieving from human-generated corpora. MATM takes the next step by letting agents retrieve agent-generated artifacts such as interaction trajectories, which are atypical documents that differ fundamentally from human-written text. MATM can continually grow while serving a distributed population of agents.

Artifact sharing and reuse are essential for enabling scalable, efficient, and continually improving agent populations. As agents operate across environments, they produce a number of intermediate artifacts, which contain rich procedural knowledge, such as action-observation trajectories (Muennighoff et al., 2025). Yet these artifacts are typically discarded after a single use or retained

[*]Denotes equal contribution.
[*]https://github.com/kimdanny/matm

only by the producing agent (Zheng et al., 2024). The ability to efficiently reuse learned behaviors and continually acquire new knowledge or experience becomes critical for scalability and long-term performance (Wang et al., 2025b; Liang et al., 2026; Shi et al., 2025). In contrast with RAG, agent-generated artifacts can be more suitable for agent consumption compared to human-authored documents (Chen et al., 2026). The need for population-level reuse is further amplified by practical considerations. Many modern agents rely on inference-time scaling and generate a number of intermediate artifacts, incurring substantial computational cost (Kaplan et al., 2020; Yao et al., 2023; Wu et al., 2024; Welleck et al., 2024). As a result, reusing those artifacts can reduce costs for reasoning and exploration (Ahmed et al., 2026).

Existing approaches to artifact reuse are insufficient for heterogeneous agent ecosystems. Prior work on reasoning or thought reuse (Zheng et al., 2024; Ouyang et al., 2026; Ahmed et al., 2026) improves cost-efficiency and effectiveness within individual agents, but reuse remains limited to the original artifact-producer; despite substantial overlap in the tasks agents solve, interaction trajectories are typically discarded after a single use (Zheng et al., 2024; Zhao et al., 2024), causing newly instantiated agents to repeatedly rediscover solutions that already exist elsewhere in the ecosystem. Related paradigms such as transfer learning (Konidaris and Barto, 2007; Brunskill and Li, 2013) and knowledge distillation (Li et al., 2025; Kang et al., 2026) require alignment between source and target domains and often demand additional training, making them impractical for diverse, dynamically instantiated populations of heterogeneous agents. Centralized multi-agent coordination methods (Dang et al., 2025) further assume *cooperative settings* and shared protocols, constraining their applicability in open ecosystems (Tran et al., 2025) where agents can freely join at any time. Indeed, based on analysis of Moltbook, Li et al. (2026) identify shared social memory as a missing prerequisite for the development of agent societies.

To address this gap, we propose Multi-Agent Transactive Memory (MATM), a framework for population-level storage and retrieval of agent-generated artifacts, based on the concept of transactive memory (Wegner, 1987), in which human groups coordinate by distributing knowledge across individuals by using shared mechanisms for locating and retrieving relevant information. Similarly, MATM maintains a shared repository to which agents can contribute artifacts produced during their own task execution (*producer agents*) and from which agents can retrieve procedural knowledge to improve their own task effectiveness and efficiency (*consumer agents*). Roles are not mutually exclusive: an agent may produce trajectories in one context and consume them in another.

This producer-consumer structure induces a two-sided marketplace for agent-generated procedural knowledge, with clear attribution between retrieved artifacts and their sources. As more agents interact with the repository, MATM grows organically, accumulating a corpus across an increasingly diverse set of tasks and environments. Operating as a specialized retrieval system over agent-generated artifacts, MATM further enables retrieval functions that go beyond generic similarity search, including agent-specific personalization, producer trust modeling, and periodic update of retriever as the population evolves.

We empirically demonstrate the effectiveness of MATM in interactive environments (ALFWorld (Shridhar et al., 2021) and WebArena (Zhou et al., 2024)). We first show that agents consistently benefit from a simple single-stage retrieval pipeline: retrieving relevant trajectories from a MATM repository populated by diverse agents not only improves downstream task performance without requiring additional coordination or joint training, but also improves task efficiency as measured by a reduced number of interaction steps. We further introduce an efficient yet powerful learning-to-rank (LTR)-based trajectory reranking stage. With simple featurization of trajectory information, reranking yields better retrieval quality, leading to improved task effectiveness and greater step efficiency. Moreover, we find that retrieval benefit extends to both weaker and stronger agents, generalizes across tasks, and continues to improve as the repository grows. Taken together, our results demonstrate that MATM provides a scalable mechanism for population-level experience reuse, enabling agents to leverage collective trajectories rather than repeatedly rediscovering solutions in isolation.

2 Background (Appendix A)

Memory has long played a role in the development of AI agents. Existing approaches can be understood as memory over various sources of data. *Memory of training data* provides agents

with access to knowledge explicitly or implicitly stored during optimization. Explicit methods include nearest-neighbor algorithms (Cover and Hart, 1967; Khandelwal et al., 2020) or case-based reasoning (Kolodner, 1992; Das et al., 2021) or implicit behaviors (Carlini et al., 2023). *Memory over experience data* provides agents with access to traces of their own interactions. Historically, methods reflecting memory of experience data include early cognitive architectures like SOAR (Laird et al., 1987), reinforcement learning (Lin, 1992), and neural networks (Weston et al., 2015). In the context of LLM agents, recent extensions treat an agent's own interaction history as retrievable context, giving rise to memory-augmented generation where past conversations or execution traces guide future behavior (Shinn et al., 2023; Majumder et al., 2024; Zheng et al., 2024). Agents generate rich intermediate artifacts during problem solving, including action-observation trajectories, thinking traces, plans, workflows, and reusable code analogous to options in reinforcement learning (Garcia et al., 2019; Veeriah et al., 2021). At the trajectory level, Buffer of Thoughts (Yang et al., 2024) and Retrieval of Thought (Ahmed et al., 2026) retrieve reasoning templates as in-context guidance, while Zheng et al. (2024) and Zhao et al. (2024) reuse action-observation trajectories for downstream decision-making. Beyond trajectories, works such as CLIN (Majumder et al., 2024), Voyager (Wang et al., 2024), AWM (Wang et al., 2025b), MaestroMotif (Klissarov et al., 2025), ASI (Wang et al., 2025a), ReasoningBank (Ouyang et al., 2026), and T3 (Arabzadeh et al., 2026) extract and reuse more abstract artifacts such as causal abstractions, workflows, skills, and executable code. Agent artifacts can further serve as distillation signals to transfer competence across models (Yang et al., 2025; Li et al., 2025; Kang et al., 2026). *Memory of external data* provides agent with access to shared artifact repositories and is represented by retrieval-augmented generation (RAG) (Lewis et al., 2020), which enhances language models by conditioning generation on retrieved external context (Fan et al., 2024).

3 Multi-Agent Transactive Memory

In existing memory systems, experience data is typically reused only by the same or homogeneous agent(s) that produced it, leaving valuable experience isolated and forcing less-experienced agents to rediscover existing solutions. In contrast, we propose a population-level memory of experience data, providing a collective memory for a population of agents. Rather than treating memory as private to each agent, we consider it a shared, structured resource that heterogeneous agents can both contribute to and retrieve from. This shifts artifact reuse from an individual optimization mechanism to a collective knowledge infrastructure, enabling continual learning and cross-agent transfer, reducing redundant exploration, and supporting cumulative capability growth at the population level.

We consider a population of n LLM agents $\mathcal{A} = \{A_i\}_{i=1}^{n}$ each potentially pursuing heterogeneous goals and operating across one or more environments $\mathcal{E} = \{E_i\}_{i=1}^{m}$. A task is specified by a description $x \in \mathcal{X}$, which corresponds to goal specification or initial state for an agent. Given a task description x in environment E_i, an LLM agent A_j performs a series of interleaved turns with the environment to solve the task. During this process, we can record a variable-length trajectory $\mathcal{T}_{E_i,A_j} = (\tau_t)_{t=1}^{H}$, where each step τ_t represents a unit of interaction. For example, in a web navigation environment, each τ_t corresponds to an action-observation pair (e.g., a click action and the resulting HTML observation) in the interaction sequence. For simplicity, we denote this agent-generated trajectory as $\mathcal{T}$.

As the agent population $\mathcal{A}$ operates across environments $\mathcal{E}$, these trajectories accumulate into a rich collection of intermediate artifacts. We denote the population-level artifact repository as $\mathcal{D} = \{\mathcal{T}\}$ and refer to this incrementally growing shared memory as Multi-Agent Transactive Memory (MATM). Within this framework, we refer to agents that contribute trajectories to $\mathcal{D}$ as *producer agents* and those that retrieve from it to aid their own task-solving as *consumer agents*, where these roles are not mutually exclusive. Our goal is to study how retrieval from this population-level memory can be optimized to improve outcomes for the population of consumer agents.

Although we focus on raw trajectories, this does not preclude higher-level abstractions such as skills or induced policies. We operate with trajectories as the lowest-level and most universally available outputs produced by agents across environments, and therefore provide a natural foundation for studying indexing and retrieval in MATM while still allowing higher-level abstractions such as skill induction (Klissarov et al., 2025) to be built on top. More-

over, retrieval over interactive trajectories is itself non-trivial. Prompt-like artifacts such as `SKILL.md` files can be indexed with standard RAG techniques (Liang et al., 2026) or further transformed into more retrieval-friendly forms (Arabzadeh et al., 2026), but state-conditioned retrieval over action-observation histories has received much less attention and is the setting we study.

3.1 Transactive Memory Indexing & Retrieval

For action-observation trajectories, we adopt a state-conditioned key-value indexing scheme in which recent interaction history serves as the retrieval key and the subsequent interaction segment as the stored value. This allows consumer agents to retrieve continued guidance conditioned on their current state rather than only on the original task instruction.

Given a window size l, for each interaction step t we define the key $\mathbf{e}_{\text{key}}^{(t)} = f(x, \tau_{t-l+1}, \ldots, \tau_t)$ and the associated value as the next l steps, $(\tau_t, \ldots, \tau_{t+l-1})$ which serves as the document d that an agent retrieves at inference time, where f is a shared embedding function, and τ_i contains both an observation and an action.

Given a task description x and MATM memory $\mathcal{D}$, a trajectory retriever $\mathcal{R}$ forms a search query q, following the process described above and returns a ranked list $\pi = \mathcal{R}(x, \mathcal{D}, K)$ of candidate trajectory chunks, where higher-ranked chunks are predicted to be more relevant for the current task and state. The trajectory retriever $\mathcal{R}$ may be instantiated as a dense retriever, or a cascaded retrieval pipeline combining an initial retriever with a reranker.

Although the embedding model f can in principle be tuned to better support artifact retrieval, we instead explore a simpler and underexamined approach that aligns retrieval results with consumer-agent preferences using lightweight learning-to-rank (LTR) rerankers (Cao et al., 2007).

3.2 Learning To Rank Trajectories (LTRT)

Learning to rank pipelines consist of a retrieval stage known as candidate generation, followed by a feature-based ranking stage that re-orders the retrieved candidates.

A feature map ϕ is designed to capture multiple complementary aspects of trajectory usefulness. Let $\phi(q, d) \in \mathbb{R}^z$ be a feature map that extracts features by inspecting the query q and document key d. Let $g_\theta : \mathbb{R}^z \mapsto \mathbb{R}$ be a parameterized scoring function, where larger outputs indicate greater predicted helpfulness of a document d value for task q.

In MATM, we define features in six categories: (i) producer agent metadata (e.g.,relevant benchmark scores); (ii) consumer agent metadata (e.g., agent ID); (iii) first-stage retrieval features (e.g., retrieval scores); (iv) query features (e.g., query length); (v) trajectory features (e.g., trajectory length); and (vi) query-trajectory interaction features (e.g., query-trajectory embedding similarity). Two of these categories carry particular conceptual weight. Producer-agent metadata is designed to enable a form of *trust modeling*, allowing the reranker to learn which producers' trajectories are reliable for a given context. Consumer-agent metadata is designed to enable *personalization* of retrieval to the individual consumer that has joined the MATM framework, since the same trajectory may be more or less useful depending on the consumer's capabilities.

Training g_θ requires supervision over which retrieved trajectories actually help. Rather than treating relevance as semantic similarity, we label trajectory chunks by their *marginal utility* (Salemi and Zamani, 2024a): a chunk is helpful to the extent that injecting it into the consumer agent improves task outcome relative to running the same agent with no retrieval. The concrete procedure for collecting these labels is intertwined with how the memory itself is built, and we describe both jointly in Section 4.1 and 4.2.

4 Experimental Setup

We instantiate MATM in two interactive benchmarks: ALFWorld (Shridhar et al., 2021), a text-based household-task environment, and WebArena (Zhou et al., 2024), a web navigation-based task environment. Each benchmark yields its own MATM index, populated by trajectories from 35 to 37 producer agents and consumed by 34 consumer agents (full population list in Appendix C).

For ALFWorld, we use the official train and test split, using 3553 episodes from the official training set, and evaluating on all 274 official test episodes (Appendix D). For WebArena, which ships without a standard train/test partition, we construct a custom split that preserves the distribution of task intents, yielding 724 training and 88 test episodes (Appendix E). In both benchmarks, all MATM construction and LTRT training is performed strictly on the training partition, so the test set remains un-

touched throughout the MATM corpus construction phase. As a result, the test set may contain questions whose task type or environment configuration overlaps with those seen during MATM construction (e.g., similar map layouts in ALFWorld or shared website domains in WebArena), but no test question is itself solved by a producer agent and inserted into the corpus.

4.1 Transactive Memory Construction

To construct MATM emerging as a trajectory storage of a population of agents producing and consuming trajectories, we expand the MATM corpus through two phases. *Pre-population* initializes the index from existing trajectory sources, and *incremental update* grows it as the producer and consumer agent population processes new training questions and contributes *successful* trajectories back to the shared memory. Both phases operate exclusively over the training partition.

Pre-Population. The pre-population phase seeds an initial index $\mathcal{D}_0$ with publicly available trajectories. For ALFWorld, we collect trajectories from a trained seq2seq model released by the benchmark authors, supplemented by trajectories generated by running Qwen3-32B and GPT-OSS 20B on the training set. For WebArena, we collect publicly available trajectories produced by GPT-4-Turbo, GPT-4-Turbo-Preview, and Claude-3.5-Sonnet from the official benchmark runs, again supplemented by Qwen3-32B and GPT-OSS 20B trajectories generated on the training set. In both cases, the collected trajectories are segmented into document chunks, encoded with the shared embedding function f, and inserted into the dense index $\mathcal{D}_0$, yielding 85,615 and 8,547 chunks for ALFWorld and WebArena respectively.

Incremental Update. After pre-population, MATM grows incrementally as the agent population operates over a stream of new training questions. This phase serves a dual purpose. It enriches the index with trajectories from a diverse set of producer agents, and it simultaneously creates the supervision signals needed to train LTRT rerankers.

The training questions are organized into partitions $\{\mathcal{X}_p\}_{p=1}^{P}$ processed sequentially. Within each partition, every question is assigned to a producer agent via a deterministic allocation function $\sigma(x, \mathcal{A}, p)$ that ensures balanced coverage across task categories and agents (Appendix F). For each assigned pair (x, A_n), the agent first attempts x without retrieval to obtain a baseline trajectory $\mathcal{T}_{\text{base}}$ and score s_{base}, which serves as the reference point for downstream marginal-utility comparisons. We then sample T branching points $\{t_1, \ldots, t_T\}$ randomly from the steps of $\mathcal{T}_{\text{base}}$. Following Chang et al. (2015), at each branching point t, we *roll in* to the corresponding prefix $h_t = (\tau_1, \ldots, \tau_t)$ and retrieve the top-K chunks from the current index most similar to x combined with h_t. We then *roll out* $|\mathcal{I}|$ one-shot trajectory-augmented generations from h_t, one per selected rank $j \in \mathcal{I} \subseteq \{1, \ldots, K\}$, scoring each resulting trajectory $\mathcal{T}_t^{(j)}$ as $s_t^{(j)}$.

This loop produces two outputs simultaneously. Any trajectory meeting a quality threshold θ, including the baseline, is added to a trajectory buffer $\mathcal{B}_p$. After all questions in $\mathcal{X}_p$ have been processed, every trajectory in $\mathcal{B}_p$ is segmented, embedded with f, and added to the index, yielding $\mathcal{D}_p$.

After all partitions are processed, the final MATM corpus contains 86,833 chunks for ALFWorld and 20,102 chunks for WebArena (Appendix G).[1] The full algorithm is given in Appendix H.

4.2 LTRT Dataset & Reranker Training

The incremental construction procedure yields a labeled training dataset $\mathcal{S} = \{(q, d, \ell)\}$ for the LTRT reranker. For each retrieved chunk $d_t^{(j)}$ evaluated at branching point t, we record the tuple $(q_t, d_t^{(j)}, \ell)$ with label $\ell = s_t^{(j)} - s_{\text{base}}$, capturing the chunk's marginal utility relative to the no-retrieval baseline. With $Q = \sum_p |\mathcal{X}_p|$ training questions, T branching points per question, and $|\mathcal{I}|$ ranks evaluated per branching point, the resulting dataset contains $Q \times T \times |\mathcal{I}|$ labeled tuples. In our experiments, we used $T = 2$ for ALFWorld, $T = 1$ for WebArena. We sample rank positions $\mathcal{I} = \{1, 5, 10, 15, 20\}$ for both benchmarks, exposing the LTRT model to candidates across the full retrieval depth while avoiding the cost of generating all twenty labeled episodes.

We compute 44 features per (q, d) pair, spanning the six categories introduced in §3.2. The full feature list is provided in Appendix I. We train three reranker families spanning common LTR paradigms: a pointwise feed-forward network (FFN), pairwise LambdaMART (Wu et al., 2010),

[1] All trajectories are available at https://huggingface.co/datasets/toeunkim/matm-trajectories.

and pairwise SVMRank (Joachims, 2006). 20% of the training set was used for the validation set for LTRT training.

4.3 Inference-Time Configuration & Baselines

Across both environments we use the E5-Base embedding model (Wang et al., 2022) as the shared embedding function f. Trajectory chunks span $l = 5$ action-observation steps under the key-value scheme. At inference time, a cascaded retrieval pipeline first retrieves the top 20 candidate trajectory chunks, after which an LTRT reranker selects the final top-1 chunk. The retrieval budget is therefore 1: the working agent conditions on a single retrieved trajectory unit per retrieval call.

Model Setups. We compare three configurations: a vanilla LLM without retrieval, MATM with single-stage dense retrieval only, and MATM with an LTRT reranker (LLM prompts in Appendix M). **RetrievalPlanner.** Each consumer agent is equipped with a *RetrievalPlanner* LLM that decides, at each interaction step, whether to issue a retrieval call against MATM. This allows agents to call on shared memory selectively rather than on every step, which is important because indiscriminate retrieval can dilute the agent's context with irrelevant guidance. **Metrics.** We evaluate both task performance and efficiency. Task performance is measured by downstream success rate (SR) and efficiency by the number of interaction steps per episode (# steps). To jointly capture both dimensions, we adopt return-paired preference (RPP) (Diaz, 2026), which measures the Pareto-dominance of trajectories between a candidate model and a fixed baseline (Appendix B). Since consumer agents operate at population scale, unless specified, all reported metrics reflect average performance across consumer models, referred to as consumer population welfare.

5 Results

We organize results around five research questions: whether MATM augmentation improves the downstream effectiveness and efficiency of consumer agents (§5.1), whether a learned reranking model can further boost retrieval quality (§5.2), whether MATM retrieval benefit is exclusive to certain model groups or distributed across the population (§5.3), whether MATM generalizes across task types (§5.4), and how consumer population performance scales with memory size (§5.5).

5.1 MATM-Augmentation improves effectiveness and efficiency

Table 1 summarizes results for ALFWorld and WebArena under no-retrieval and single-stage retrieval from MATM. Across both benchmarks, retrieval from the shared repository consistently improves task outcomes.

On ALFWorld, success rate increases from 47% to 55% (+8.0%p), while average steps per episode decrease from 11.77 to 11.18. The RPP score rises from -0.16 to -0.05, indicating that the retrieval-augmented population more frequently Pareto-dominates the no-retrieval baseline in terms of the joint success-efficiency. On WebArena, success rate improves from 18% to 20% (+2%p), with average steps falling from 22.0 to 20.3 and RPP turning positive at 0.03. The improvement is more modest than ALFWorld, possibly due to WebArena's longer task horizons and greater sensitivity to early-step errors. Together, these results show that a shared repository of heterogeneous agent trajectories improves consumer population welfare along both effectiveness and efficiency dimensions.

5.2 Learning to Rank Trajectories further improves MATM participants' welfare

Single-stage retrieval selects trajectories by embedding similarity alone. We next ask whether a learned reranker trained to predict downstream utility can improve over this baseline. We experiment with three reranker configurations: a feed-forward network (FFN), LambdaMART, and SVMRank.

On ALFWorld, all three rerankers improve over single-stage retrieval, and SVMRank achieves the strongest results across all metrics: success rate reaches 64.3% (+9.2%p over single-stage, +17.2%p over no-retrieval), average steps fall to 10.35, and RPP rises to 0.15. On WebArena, reranker effectiveness is more moderate. FFN matches the success rate of single-stage retrieval at 20.5% while achieving the lowest step count (19.91) and highest RPP (0.04) among all methods, making it the most effective reranker for that environment. LambdaMART, by contrast, reverts success rate to the no-retrieval level on WebArena, suggesting that the features it relies on are better calibrated to ALFWorld's task structure than WebArena's.

Method	ALFWorld			WebArena		
	SR↑	# steps↓	RPP↑	SR↑	# steps↓	RPP↑
no retrieval	0.4708	11.7664	-0.1586	0.1818	21.9886	-0.0511
single-stage	0.5511	11.1796	-0.0451	**0.2045**	20.2614	0.0284
rerank-FFN	0.5861	11.0336	-0.0046	**0.2045**	**19.9091**	**0.0398**
rerank-LambdaMART	0.5715	10.8474	0.0608	0.1818	20.4205	-0.0341
rerank-SVMRank	**0.6431**	**10.3453**	**0.1474**	**0.2045**	20.2841	0.0170

Table 1: Evaluation of MATM-augmented agents in interactive environments. Success Rate (SR) and number of steps (# steps) are used for measuring the effectiveness and efficiency. Values are average of the five runs of randomized task-model allocation.

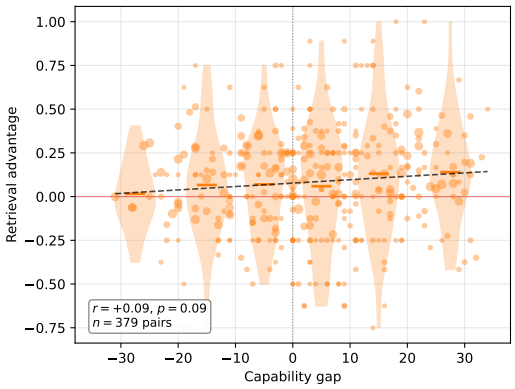

Figure 2: Retrieval Advantage vs. Producer-Consumer Capability Gap on ALFWorld with SVMRank reranking. Each point is one producer–consumer pair.

Retrieval Scope	SR ↑	# steps ↓	RPP ↑
ALFWorld (274)			
full	**0.6460**	**10.2300**	**0.0566**
same task type	0.6314	10.4635	0.0055
cross task type	0.5985	10.7445	-0.0620
WebArena (47)			
full	**0.2979**	19.0426	**0.1383**
same task type	0.2766	19.1915	0.0000
cross task type	0.1915	**19.0000**	-0.1383

Table 2: MATM retrieval scope ablation results across three candidate pool restrictions. Number of tasks in parentheses.

5.3 MATM benefits are distributed across the agent population (Appendix K)

While the previous sections establish that MATM improves average consumer welfare, they leave open whether that benefit is concentrated among particular producer-consumer pairings or distributed across the population. To answer this, we measure two quantities for each (producer, consumer) pair: the *retrieval advantage*, defined as the gain in consumer success rate when retrieving from that producer relative to its no-retrieval baseline, and the *capability gap*, defined as the difference between the two agents' aggregated benchmark scores (AAI (Artificial Analysis, 2026)). Higher the capability gap value, the producer is more capable than the consumer.

Figure 2 shows that consumers benefit from retrieval regardless of whether the producer is weaker, comparable, or stronger than themselves, indicating MATM's value cannot be reduced to a single strong producer. Although the correlation between capability gap and retrieval advantage shows a slight positive trend — suggesting that stronger producers may yield marginally higher benefit — it remains small and insignificant across both benchmarks, indicating that retrieval utility is not primarily driven by producer-to-consumer competence transfer. Finally, reranking lifts the entire retrieval advantage distribution, confirming the finding in §5.2: on ALFWorld, SVMRank roughly doubles the mean retrieval advantage, and the same direction of effect appears on WebArena under FFN reranking (Appendix K).

5.4 MATM offers cross-task generalization

We study how well MATM generalizes across tasks by varying the retrieval scope. We evaluate three conditions: (i) *full* retrieval places no restriction on the candidate pool; (ii) *same-task* retrieval limits candidates to trajectories from the same task type as the query; and (iii) *cross-task* retrieval limits candidates exclusively to trajectories from different task types (Appendix L).

Table 2 shows that full retrieval achieves the highest SR and RPP in both environments, confirming that unrestricted candidate diversity is beneficial. Two findings point to genuine cross-task generalization. First, even under cross-task retrieval, ALFWorld SR reaches 59.9%, which remains well above the no-retrieval baseline of 47.1% from Table 1. This shows that trajectories from structurally different task types still carry transferable utility. Second, the effectiveness and efficiency gap between full and same-task retrieval in both environments suggests that restricting the candidate pool

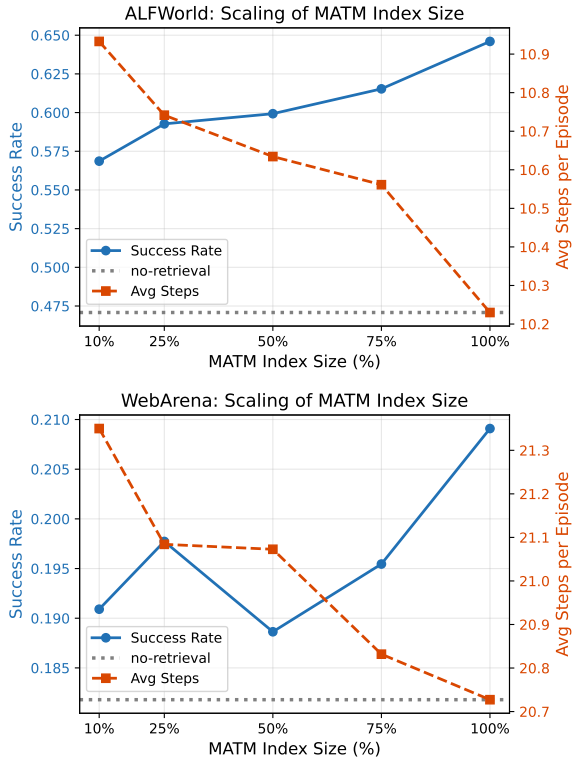

Figure 3: MATM memory scaling curves for ALFWorld (top) and WebArena (bottom). Success rate (left axis) and average steps per episode (right axis) as a function of index size. The dotted line marks SR of the no-retrieval baseline. Results are averaged over five runs with different random seeds.

to same-type trajectories is itself a source of degradation, excluding useful candidates that happen to cross task boundaries. However, for both benchmarks, same-task retrieval outperforms cross-task retrieval, indicating that task-type alignment still carries a meaningful relevance signal.

5.5 MATM scales with memory size

We study how downstream effectiveness and efficiency change as MATM grows in size. We construct nested memory subsets at 10%, 25%, 50%, 75%, and 100% of the full index, with each subset preserving producer model composition and benchmark coverage to isolate the effect of memory size from shifts in data distribution. Each subset is constructed with five different random seeds and results are averaged across runs.

Figure 3 shows the scaling curves for both environments. ALFWorld exhibits monotonic improvement in both success rate and step efficiency with index size, confirming that larger memory consistently benefits the agent. On WebArena, step efficiency also decreases monotonically, consistent with ALFWorld. Success rate, however, exhibits a non-monotonic pattern: it dips at the 50% index before recovering sharply to 20.9% at full scale, the strongest result across all index sizes and the clearest margin above the no-retrieval baseline of 18.2%. We hypothesize that at intermediate scales, the index is large enough to surface plausible but ultimately unhelpful trajectories, yet not diverse enough to reliably include high-quality matches. At full scale, sufficient coverage overcomes this noise, restoring and exceeding the gains observed at smaller index sizes.

6 Discussion

Our results support the hypothesis that transactive memory provides a viable architecture for population-level agent memory. The results in §5.1 and §5.2 confirm that MATM with learned reranking consistently improves consumer agent welfare in both effectiveness and efficiency.

Our feature importance analysis (Appendix J) suggests that predictive signals for reranking extend beyond retrieval-level similarity to include producer agent metadata such as benchmark scores that represents the producer agents' capability. This reframes trajectory selection as, in part, a problem of producer trust modeling: the reranker learns to prefer trajectories from agents whose competence profiles predict downstream utility for the consumer. However, feature importance concentrates differently across environments — ALFWorld relies heavily on a small set of producer features while WebArena distributes importance more evenly — which likely explains why no single reranker dominates both, and reinforces the need for retrieval systems that adapt to task structure rather than a fixed ranking policy.

The capability-gap analysis in §5.3 further shows that retrieval benefit is broadly distributed across the population rather than driven by transfer from stronger to weaker agents. This opens a natural future direction of per-consumer (group) personalization of retrieval policy since the same shared repository may be optimally exploited differently by agents with different competence profiles and task preferences.

The retrieval scope experiments in §5.4 provide direct evidence for why population-level memory is necessary. Trajectories from different task types carry transferable utility, and restricting the candidate pool to same-type trajectories degrades perfor-

mance relative to unrestricted retrieval. This means that useful trajectories are not confined within task boundaries. They encode reusable patterns of interaction that generalize across tasks. An agent-specific or task-specific memory, by design, would exclude these candidates. MATM's value lies in making them accessible.

Our scaling experiments in §5.5 demonstrate that the value of MATM improves as the repository grows, suggesting that incentives or remuneration for producer contributions will be an important challenge for such platforms.

7 Conclusion

We introduced MATM, a shared population-level memory where heterogeneous agents contribute and retrieve trajectories to improve task performance. Retrieval improves effectiveness and efficiency across the agent population regardless of capability, with reranking further amplifying gains. Retrieved trajectories generalize across task boundaries and performance scales with memory size, suggesting shared artifact storage as a promising substrate for collective and continual intelligence among distributed agents.

Limitations and Future Work

Our experiments cover two interactive benchmarks (ALFWorld and WebArena) and a 34-model consumer population. While no single study can cover every environment or every model in a rapidly evolving landscape, this restricts the scope of our empirical claims. Our experiments and rerankers are also trained and evaluated within the same benchmark, so cross-benchmark reranker generalization remains untested. Also, the LTR dataset uses sparse rank sampling at positions $\mathcal{I} = \{1, 5, 10, 15, 20\}$, which does not fully cover the label distribution at all rank positions. This was a practical choice given experiment budget, and we found it to yield strong LTR learning performance with a favorable cost-quality tradeoff. Finally, our work focuses entirely on the consumer side of MATM. Because MATM is fundamentally a two-sided market, evaluating producer-side welfare is equally important. Future work could draw on attribution fairness in RAG (Kim and Diaz, 2025) or marketplace evaluation frameworks (Kim et al., 2026) to address this gap. Relatedly, the current framework does not account for adversarial producers who may contribute malicious trajectories, potentially placing consumer agents at risk.

Acknowledgments

This work was supported by NSF grant 2402874. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

References

A2A Protocol. 2026. Agent2agent (a2a) protocol. https://a2a-protocol.org. Accessed: 2026-05-26.

Ammar Ahmed, Azal Ahmad Khan, Ayaan Ahmad, Sheng Di, Zirui Liu, and Ali Anwar. 2026. Retrieval-of-thought: Efficient reasoning via reusing thoughts. In *The Fourteenth International Conference on Learning Representations*.

Negar Arabzadeh, Wenjie Ma, Sewon Min, and Matei Zaharia. 2026. Rag over thinking traces can improve reasoning tasks. *arXiv preprint arXiv:2605.03344*.

Artificial Analysis. 2026. Artificial analysis. https://artificialanalysis.ai. Accessed: 2026-05-25.

Emma Brunskill and Lihong Li. 2013. Sample complexity of multi-task reinforcement learning. In *Proceedings of the Twenty-Nineth Conference on Uncertainty in Artificial Intelligence (UAI-13)*. Association for Uncertainty in Artificial Intelligence.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2023. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, and John Langford. 2015. Learning to search better than your teacher. In *International Conference on Machine Learning*, pages 2058–2066. PMLR.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Jimmy Lin, Akari Asai, and Victor Zhong. 2026. Agentir: Reasoning-aware retrival for deep research agents. *arXiv preprint arXiv:2603.04384*.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Ruo Yu Tao, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2018. Textworld: A learning environment for text-based games. *CoRR*, abs/1806.11532.

T. Cover and P. Hart. 1967. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

Yufan Dang, Chen Qian, Xueheng Luo, Jingru Fan, Zihao Xie, Ruijie Shi, Weize Chen, Cheng Yang, Xiaoyin Che, Ye Tian, Xuantang Xiong, Lei Han, Zhiyuan Liu, and Maosong Sun. 2025. Multi-agent collaboration via evolving orchestration. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew McCallum. 2021. Case-based reasoning for natural language queries over knowledge bases. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9594–9611. Association for Computational Linguistics.

Fernando Diaz. 2026. Offline preference-based trajectory evaluation. *arXiv preprint arXiv:2606.17541*.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pages 6491–6501.

Francisco M. Garcia, Bruno Castro da Silva, and Philip S. Thomas. 2019. A compression-inspired framework for macro discovery. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19, page 1973–1975. International Foundation for Autonomous Agents and Multiagent Systems.

Thorsten Joachims. 2006. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226.

Minki Kang, Jongwon Jeong, Seanie Lee, Jaewoong Cho, and Sung Ju Hwang. 2026. Distilling LLM agent into small models with retrieval and code tools. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through Memorization: Nearest Neighbor Language Models. In *International Conference on Learning Representations (ICLR)*.

To Eun Kim and Fernando Diaz. 2025. Towards fair rag: On the impact of fair ranking in retrieval-augmented generation. In *Proceedings of the 2025 International ACM SIGIR Conference on Innovative Concepts and Theories in Information Retrieval (ICTIR)*, ICTIR '25, page 33–43. Association for Computing Machinery.

To Eun Kim, Alireza Salemi, Andrew Drozdov, Fernando Diaz, and Hamed Zamani. 2024. Retrieval-enhanced machine learning: Synthesis and opportunities. *arXiv preprint arXiv:2407.12982*.

To Eun Kim, Alireza Salemi, Hamed Zamani, and Fernando Diaz. 2026. Evaluation of agents under simulated ai marketplace dynamics. *arXiv preprint arXiv:2604.14256*.

Martin Klissarov, Mikael Henaff, Roberta Raileanu, Shagun Sodhani, Pascal Vincent, Amy Zhang, Pierre-Luc Bacon, Doina Precup, Marlos C. Machado, and Pierluca D'Oro. 2025. Maestromotif: Skill design from artificial intelligence feedback. In *The Thirteenth International Conference on Learning Representations*.

Janet L. Kolodner. 1992. An introduction to case-based reasoning. *Artificial Intelligence Review*, 6(1):3–34.

George Konidaris and Andrew Barto. 2007. Building portable options: skill transfer in reinforcement learning. In *Proceedings of the 20th international joint conference on Artifical intelligence*, IJCAI'07, pages 895–900. Morgan Kaufmann Publishers Inc.

John E. Laird, Allen Newell, and Paul S. Rosenbloom. 1987. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33(1):1–64.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Ming Li, Xirui Li, and Tianyi Zhou. 2026. Does Socialization Emerge in AI Agent Society? A Case Study of Moltbook. *arXiv preprint*. ArXiv:2602.14299 [cs].

Yang Li, Youssef Emad, Karthik Padthe, Jack Lanchantin, Weizhe Yuan, Thao Nguyen, Jason E Weston, Shang-Wen Li, Dong Wang, Ilia Kulikov, and Xian Li. 2025. Naturalthoughts: Selecting and distilling reasoning traces for general reasoning tasks. In *NeurIPS 2025 Workshop on Efficient Reasoning*.

Yuan Liang, Ruobin Zhong, Haoming Xu, Chen Jiang, Yi Zhong, Runnan Fang, Jia-Chen Gu, Shumin Deng, Yunzhi Yao, Mengru Wang, and 1 others. 2026. Skillnet: Create, evaluate, and connect ai skills. *arXiv preprint arXiv:2603.04448*.

Long-Ji Lin. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3):293–321.

Bodhisattwa Prasad Majumder, Bhavana Dalvi Mishra, Peter Jansen, Oyvind Tafjord, Niket Tandon, Li Zhang, Chris Callison-Burch, and Peter Clark. 2024. CLIN: A continually learning language agent for rapid task adaptation and generalization. In *First Conference on Language Modeling*.

Model Context Protocol. 2026. Model context protocol. https://modelcontextprotocol.io. Accessed: 2026-05-26.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candes, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20275–20321. Association for Computational Linguistics.

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, and Tomas Pfister. 2026. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *The Fourteenth International Conference on Learning Representations*.

Alireza Salemi and Hamed Zamani. 2024a. Evaluating retrieval quality in retrieval-augmented generation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 2395–2400. Association for Computing Machinery.

Alireza Salemi and Hamed Zamani. 2024b. Towards a search engine for machines: Unified ranking for multiple retrieval-augmented large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 741–751. Association for Computing Machinery.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, and Hao Wang. 2025. Continual learning of large language models: A comprehensive survey. *ACM Computing Surveys*, 58(5):1–42.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. {ALFW}orld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*.

Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D. Nguyen. 2025. Multi-Agent Collaboration Mechanisms: A Survey of LLMs. ArXiv:2501.06322 [cs].

Vivek Veeriah, Tom Zahavy, Matteo Hessel, Zhongwen Xu, Junhyuk Oh, Iurii Kemaev, Hado P van Hasselt, David Silver, and Satinder Singh. 2021. Discovery of options via meta-learned subgoals. *Advances in Neural Information Processing Systems*, 34:29861–29873.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*.

Jiayu Wang, Yifei Ming, Zixuan Ke, Shafiq Joty, Aws Albarghouthi, and Frederic Sala. 2026. SkillOrchestra: Learning to Route Agents via Skill Transfer. *arXiv preprint*. ArXiv:2602.19672 [cs].

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*.

Zora Zhiruo Wang, Apurva Gandhi, Graham Neubig, and Daniel Fried. 2025a. Inducing programmatic skills for agentic tasks. In *Second Conference on Language Modeling*.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. 2025b. Agent workflow memory.

Daniel M Wegner. 1987. Transactive memory: A contemporary analysis of the group mind. In *Theories of group behavior*, pages 185–208. Springer.

Sean Welleck, Amanda Bertsch, Matthew Finlayson, Hailey Schoelkopf, Alex Xie, Graham Neubig, Ilia Kulikov, and Zaid Harchaoui. 2024. From decoding to meta-generation: Inference-time algorithms for large language models. *Transactions on Machine Learning Research*. Survey Certification.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Qiang Wu, Christopher JC Burges, Krysta M Svore, and Jianfeng Gao. 2010. Adapting boosting for information retrieval measures. *Information Retrieval*, 13:254–270.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2024. Scaling inference computation: Compute-optimal inference for problem-solving with language models. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E. Gonzalez, and Bin CUI. 2024. Buffer of thoughts: Thought-augmented reasoning with large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Minkai Xu, Joseph E. Gonzalez, Bin CUI, and Shuicheng YAN. 2025. Supercorrect: Advancing small LLM reasoning with thought template distillation and self-correction. In *The Thirteenth International Conference on Learning Representations*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Hamed Zamani, Fernando Diaz, Mostafa Dehghani, Donald Metzler, and Michael Bendersky. 2022. Retrieval-enhanced machine learning. In *Proceedings of the 45th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

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.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. 2024. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *The Twelfth International Conference on Learning Representations*.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*.

A Unabridged Related Work

A.1 Memory-Augmented Agents

Retrieval-augmented generation (RAG) (Lewis et al., 2020), an instance of Retrieval-Enhanced Machine Learning (Zamani et al., 2022; Kim et al., 2024), enhances language models by conditioning generation on retrieved external context, most commonly human-authored documents such as web pages or knowledge bases (Fan et al., 2024). Recent extensions of this paradigm treat an agent's own interaction history as retrievable context, giving rise to memory-augmented generation, where past conversations or execution traces are indexed and reused to guide future behavior (Shinn et al., 2023; Majumder et al., 2024; Zheng et al., 2024).

MATM fits within this retrieval- and memory-augmented paradigm but differs in both content and scope: instead of retrieving web documents or single-agent's local history, MATM indexes agent-generated trajectories and treats memory as a population-level resource shared across agents, where agents can contribute to and retrieve from the shared repository.

A.2 Reuse of Agent Artifacts

Modern reasoning agents, particularly those employing high inference-time scaling, generate rich intermediate artifacts during problem solving. These include low-level action-observation trajectories and thinking traces, as well as higher-level plans, strategies, workflows, and reusable code which are anologous to the notion of options in reinforcement learning (Garcia et al., 2019; Veeriah et al., 2021). These *agent artifacts* can be leveraged to improve inference efficiency, generalization, and continual adaptation.

At the trajectory level, prior work has explored reusing reasoning or action-observation trajectories as in-context guidance. Buffer of Thoughts (Yang et al., 2024) maintains and retrieves reasoning templates to guide new problem instances, while Retrieval of Thought (Ahmed et al., 2026) constructs thought templates on the fly by retrieving prior reasoning trajectories. For action-observation trajectories, Zheng et al. (2024) and Zhao et al. (2024) reuse environment interaction trajectories as in-context examples to improve downstream decision-making.

Beyond the trajectory level, several works extract and reuse more abstract artifacts such as plans, strategies, workflows, and skills. CLIN (Majumder et al., 2024) stores textual causal abstractions to support continual improvement. Agent Workflow Memory (Wang et al., 2025b) distills reusable workflows from web interaction trajectories, and MaestroMotif (Klissarov et al., 2025) induces reusable skills via reinforcement learning. ReasoningBank (Ouyang et al., 2026) retrieves strategy-level reasoning patterns to guide problem solving. Arabzadeh et al. (2026) transform math reasoning thinking trajectories into higher level structures and use them as retrievable objects. Applied to a programming domain, Wang et al. (2025a) enable agents to induce, verify, and reuse program-based skills on the fly in web-based tasks,

while Voyager (Wang et al., 2024) maintains a growing library of executable code for open-ended task execution. In a concurrent work, SkillNet (Liang et al., 2026) assembles a collection of skills contributed by multiple agents, framing skill accumulation as a system-design problem. Collectively, these works frame artifact reuse as a mechanism for accumulating reusable competence over time.

Agent artifacts can also serve as supervision signals for model distillation. SuperCorrect (Yang et al., 2025) extracts thought templates from a teacher model to guide smaller models during reasoning. Related approaches similarly distill structured reasoning artifacts to transfer competence across models (Li et al., 2025; Kang et al., 2026). In these settings, artifacts function not only as inference-time memory but also as compressed representations of reasoning expertise.

While these systems reuse different types of agent artifacts, such artifacts are typically reused only by the same or homogeneous agent(s) that produced them, with less consideration of emerging society of agents (Li et al., 2026; Wang et al., 2026). As a result, valuable experience remains isolated, and newly instantiated agents repeatedly rediscover solutions that already existed elsewhere in the other systems.

In contrast, MATM proposes a *population-level* artifact repository. Rather than treating artifacts as private, per-agent memory, we model them as shared, structured resources that heterogeneous agents can both contribute to and retrieve from. This shifts artifact reuse from an individual optimization mechanism to a collective knowledge infrastructure, enabling continual learning and cross-agent transfer, reducing redundant exploration, and supporting cumulative capability growth at the ecosystem level. The most closely related concurrent work, SkillNet (Liang et al., 2026), also assembles a collection of skills across heterogeneous agents. However, it primarily addresses system design and does not evaluate the benefit of retrieval for consumer agents in a population setting, nor does it provide in-depth analysis of artifact repository search.

B Return-Paired Preference (Diaz, 2026)

Given a set of n tasks and two agents—a control or baseline agent A and a treatment agent A'—we have a set of n trajectories for each agent. For a task, we say that A' is preferred to A if A' is successful and A is not or, if both are successful, A' reaches the success in fewer steps; similarly, A is preferred to A' if A is successful while A' is not or if it faster to success if both are successful. In all other situations, we say that there is no preference between A' and A for that task. If the value of the preference is 1 when A' is preferred, -1 when A is preferred, and 0 if there is no preference, then the return-paired preference metric for agent A' is the **mean** preference value over all n tasks compared with all other agents.

Model	Producer	Consumer
trained-seq2seq	A	
openai/gpt-4-turbo	W	
openai/gpt-4-turbo-preview	W	
anthropic/claude-3.5-sonnet	W	
openai/gpt-oss-20b	AW	AW
openai/gpt-oss-120b	AW	AW
openai/gpt-5.4	AW	AW
openai/gpt-5.4-nano	AW	AW
anthropic/claude-3-haiku	AW	AW
anthropic/claude-sonnet-4.6	AW	AW
anthropic/claude-opus-4	AW	AW
anthropic/claude-opus-4.6	AW	AW
google/gemini-2.5-flash-lite	AW	AW
google/gemini-2.5-flash	AW	AW
google/gemini-2.5-pro	AW	AW
google/gemma-4-31b-it	AW	AW
meta-llama/llama-3.1-8b-instruct	AW	AW
meta-llama/llama-3.2-3b-instruct	AW	AW
meta-llama/llama-3.3-70b-instruct	AW	AW
meta-llama/llama-4-maverick	AW	AW
deepseek/deepseek-r1	AW	AW
deepseek/deepseek-r1-0528	AW	AW
deepseek/deepseek-chat-v3.1	AW	AW
deepseek/deepseek-v3.2	AW	AW
qwen/qwen3-32b	AW	AW
qwen/qwen3-235b-a22b	AW	AW
qwen/qwen3-max	AW	AW
qwen/qwen3.5-flash-02-23	AW	AW
x-ai/grok-3-mini	AW	AW
x-ai/grok-4-fast	AW	AW
x-ai/grok-4.1-fast	AW	AW
x-ai/grok-4.20	AW	AW
z-ai/glm-4.5	AW	AW
z-ai/glm-5	AW	AW
minimax/minimax-m1	AW	AW
minimax/minimax-m2	AW	AW
mistralai/ministral-14b-2512	AW	AW
mistralai/mistral-large-2512	AW	AW

Table 3: Producer and consumer agents across ALFWorld (A) and WebArena (W). AW indicates the model serves in that role for both environments.

C Population of LLM Agents

Table 3 shows the list of agents used as a population in each benchmark.

D ALFWorld Description

ALFWorld (Shridhar et al., 2021) contains interactive TextWorld environments (Côté et al., 2018) that parallel embodied worlds in the ALFRED dataset (Shridhar et al., 2020). The aligned environments allow agents to reason and learn high-level policies in an abstract space before solving embodied tasks through low-level actuation. ALFWorld translates complex household tasks such as finding, cleaning, heating, or placing objects into textual observations and actions, allowing researchers to train and evaluate agents using natural language rather than raw visual input. The dataset consists of 3553 tasks for training and a heldout test set of 274 tasks. The tasks are grouped into 6 task types: Pick & Place, Examine in Light, Clean & Place, Heat & Place, Cool & Place, Pick Two & Place. Within each task category there is significant variation: the embodied environment includes 120 rooms (30 kitchens, 30 bedrooms, 30 bathrooms, 30 living rooms), each dynamically populated with a set of portable objects (e.g., apple, mug), and static receptacles (e.g., microwave, fridge). For interaction, TextWorld environments allow 9 high-level actions such as 'open', 'heat', etc. For our experiments, we use the 3553 training episodes for populating MATM memory, and a sampled representative subset of 355 episodes for generating data for training LTRT rerankers. We use the heldout 274 episodes as the test set.

E WebArena Description

WebArena (Zhou et al., 2024) is a standalone, self-hostable web environment for building autonomous agents. WebArena creates websites from five popular categories (Ecommerce platforms, Social Forums, Maps, Content Management Systems and Collaborative Development Platforms for software development) with functionality and data mimicking their real-world equivalents. The dataset consists of 812 examples consisting of high-level natural language instructions that require interaction with the WebArena environment to solve. The dataset was created by curating realistic intents to carry out complex and creative tasks within WebArena. Annotators were guided to spend a few minutes exploring the websites to familiarize themselves with the websites' content and functionalities. Then the annotators are tasked with intent formulation. At the end, 241 intents were curated and 812 tasks were created with different instantiations of these intents. Figure 4 shows the distribution of intents accross different sites.

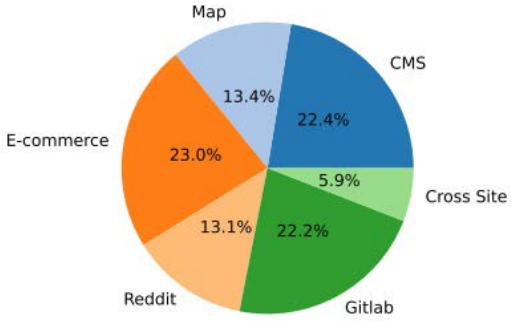

Figure 4: The intent distribution across different websites for WebArena

For our experiments, the test set comprised of 88 tasks sampled from the total 812 tasks while maintaining the distribution of intents. We treated the leftover 724 tasks as training data for populating MATM memory, and used a subset of 58 episodes for generating LTRT training data. In the end, our test set consisted of 88 tasks.

F Task Allocation Function

To ensure that every model in the population produces trajectories across all task categories, we employ a *task-type-aware stratified round-robin* assignment with an offset. Given a partition $\mathcal{X}_p$, questions are first grouped into buckets by task type (e.g., Algebra, Geometry, Precalculus for mathematical problem solving, or task categories for interactive benchmarks), and each bucket is sorted in a deterministic order. Within each bucket, questions are assigned to agents by cycling through the ordered population $A_1, A_2, \ldots, A_N$ in round-robin fashion, starting at an offset $o \in \{0, \ldots, N-1\}$. The offset shifts the starting agent but does not change the bucket composition, ensuring that different offset values produce complementary assignments across agents.

G MATM Index Statistics

Table 4 shows the size of MATM index across environments.

H Incremental Construction of MATM & LTRT Dataset

Algorithm 1 describes formal procedure of incremental construction of MATM index and LTRT

Algorithm 1 Incremental MATM & LTRT Dataset Construction

Require: Agent population $\mathcal{A}$, question partitions $\mathcal{X}_1, \ldots, \mathcal{X}_P$, pre-warmed index $\mathcal{D}_0$, retrieval depth K, rank positions $\mathcal{I} \subseteq \{1, \ldots, K\}$, branching points per question T, allocation σ, evaluator EVAL, quality threshold θ, embedding function f

Ensure: Updated MATM index $\mathcal{D}_P$ and LTRT dataset $\mathcal{S}$

$\mathcal{S} \leftarrow \emptyset$ ▷ LTRT Dataset
for $p = 1, \ldots, P$ **do** ▷ Process each partition
  $\mathcal{B}_p \leftarrow \emptyset$ ▷ Trajectory buffer
  for each question $x \in \mathcal{X}_p$ **do**
    $A_n \leftarrow \sigma(x, \mathcal{A}, p)$ ▷ Assign agent
    $(\mathcal{T}_{\text{base}}, \hat{y}_{\text{base}}) \leftarrow A_n(x)$ ▷ Baseline trajectory without retrieval
    $s_{\text{base}} \leftarrow \text{EVAL}(\hat{y}_{\text{base}}, x)$ ▷ Reference score for marginal utility
    if $s_{\text{base}} \geq \theta$ **then**
      $\mathcal{B}_p \leftarrow \mathcal{B}_p \cup \{\mathcal{T}_{\text{base}}\}$
    end if
    Sample $\{t_1, \ldots, t_T\} \subseteq \{1, \ldots, |\mathcal{T}_{\text{base}}|\}$ uniformly at random ▷ Branching points
    for $t \in \{t_1, \ldots, t_T\}$ **do** ▷ Roll-in to step t of $\mathcal{T}_{\text{base}}$
      $h_t \leftarrow (\tau_1, \ldots, \tau_t)$ from $\mathcal{T}_{\text{base}}$
      $q_t, (d_t^{(1)}, \ldots, d_t^{(K)}) \leftarrow \text{RETRIEVE}(x, h_t, \mathcal{D}_{p-1}, K)$
      for $j \in \mathcal{I}$ **do** ▷ Roll-out: one-shot augmented generation per rank
        $(\mathcal{T}_t^{(j)}, \hat{y}_t^{(j)}) \leftarrow A_n(x \mid h_t, d_t^{(j)})$
        $s_t^{(j)} \leftarrow \text{EVAL}(\hat{y}_t^{(j)}, x)$
        if $s_t^{(j)} \geq \theta$ **then**
          $\mathcal{B}_p \leftarrow \mathcal{B}_p \cup \{\mathcal{T}_t^{(j)}\}$ ▷ Add successful trajectory
        end if
        $\mathcal{S} \leftarrow \mathcal{S} \cup \{(q_t, d_t^{(j)}, s_t^{(j)} - s_{\text{base}})\}$ ▷ Marginal utility label
      end for
    end for
  end for
  $\mathcal{D}_p \leftarrow \text{INDEXUPDATE}(\mathcal{D}_{p-1},\ \mathcal{B}_p,\ f)$ ▷ Chunk, embed, add
end for

dataset for trajectory reranker training.

I Learning-To-Rank Features

Table 5 shows the complete list of learning-to-rank features used across environments.

J Feature Importance Test of the Trained Learning-To-Rank Trajectories (LTRT) Model

Table 6 shows the top ten most important features for both benchmarks.

K Extended Results of Section §5.3

K.1 Formalism

We formalize the analysis of §5.3 as follows. With some abuse of notation, let $\mathcal{P}$ denote the set of producer agents and $\mathcal{C}$ the set of consumer agents, with $\mathcal{X}$ the set of evaluation tasks. For a consumer $c \in \mathcal{C}$, let $\mu_0(c) \in [0, 1]$ denote its average final score on $\mathcal{X}$ without retrieval, and let $\mu_r(p, c) \in [0, 1]$ denote its average final score on episodes where producer $p \in \mathcal{P}$ appeared among the retrieved source models. The *retrieval advantage* of the pair (p, c) is

$$\mu_\alpha(p, c) = \mu_r(p, c) - \mu_0(c),$$

the gain in consumer success rate attributable to retrieving from producer p relative to that consumer's own no-retrieval baseline.

Each agent $i \in \mathcal{P} \cup \mathcal{C}$ has a standalone capability $\kappa(i)$, measured by its aggregated Artificial Analysis Intelligence Index score (Artificial Analysis, 2026). The *capability gap* of a producer-consumer pair is

$$\kappa_\alpha(p, c) = \kappa(p) - \kappa(c),$$

which is positive when the producer is stronger than the consumer in standalone capability, zero when

Producer Agents	# chunks	%
trained-seq2seq	67,570	77.82%
Qwen3-32B	11,264	12.97%
GPT-OSS 20B	6,781	7.81%
DeepSeek-V3.2	85	0.10%
MiniMax-M1	68	0.08%
Qwen3.5-Flash	68	0.08%
Claude Sonnet 4.6	66	0.08%
GLM-4.5	63	0.07%
DeepSeek-R1	59	0.07%
Claude Opus 4	54	0.06%
MiniMax-M2	54	0.06%
Gemini 2.5 Flash	52	0.06%
DeepSeek-Chat-V3.1	52	0.06%
Qwen3-Max	50	0.06%
Grok-4-Fast	45	0.05%
Gemini 2.5 Pro	41	0.05%
Grok-4.1-Fast	39	0.04%
Gemini 2.5 Flash Lite	36	0.04%
GLM-5	35	0.04%
Mistral Large 2512	34	0.04%
Gemma-4-31B	33	0.04%
Claude Opus 4.6	32	0.04%
Ministral-14B	30	0.03%
DeepSeek-R1-0528	30	0.03%
Qwen3-235B-A22B	29	0.03%
Grok-4.20	27	0.03%
Llama-3.3-70B	24	0.03%
GPT-OSS 120B	23	0.03%
Llama-4-Maverick	23	0.03%
Grok-3-Mini	21	0.02%
Claude 3 Haiku	17	0.02%
GPT-5.4	12	0.01%
GPT-5.4-Nano	9	0.01%
Llama-3.1-8B	7	0.01%
Total (34)	**86,833**	**100%**

(a) ALFWorld

Producer Agents	# chunks	(%)
GPT-4-Turbo	2,222	11.05%
GPT-4-Turbo-Preview	2,054	10.22%
Claude 3.5 Sonnet	1,673	8.32%
Qwen3-32B	1,500	7.46%
GPT-OSS 20B	1,098	5.46%
DeepSeek-R1-0528	1,040	5.17%
Claude Opus 4	982	4.89%
MiniMax-M1	804	4.00%
Claude Sonnet 4.6	801	3.98%
Claude Opus 4.6	725	3.61%
DeepSeek-R1	674	3.35%
Gemini 2.5 Flash	673	3.35%
GLM-5	577	2.87%
Llama-4-Maverick	442	2.20%
Grok-4.1-Fast	437	2.17%
Grok-4.20	431	2.14%
Qwen3.5-Flash	411	2.04%
DeepSeek-V3.2	404	2.01%
GPT-5.4	388	1.93%
GLM-4.5	342	1.70%
GPT-5.4-Nano	306	1.52%
MiniMax-M2	289	1.44%
Grok-4-Fast	262	1.30%
Qwen3-Max	246	1.22%
Qwen3-235B-A22B	224	1.11%
Gemini 2.5 Pro	217	1.08%
Grok-3-Mini	203	1.01%
GPT-OSS 120B	158	0.79%
DeepSeek-Chat-V3.1	139	0.69%
Gemma-4-31B	118	0.59%
Mistral Large 2512	116	0.58%
GPT-4-1106-Preview	89	0.44%
Llama-3.1-8B	22	0.11%
Llama-3.3-70B	20	0.10%
Gemini 2.5 Flash Lite	11	0.05%
Claude 3 Haiku	4	0.02%
Total (36)	**20,102**	**100%**

(b) WebArena

Table 4: MATM index statistics across benchmarks.

they are matched, and negative when the producer is weaker.

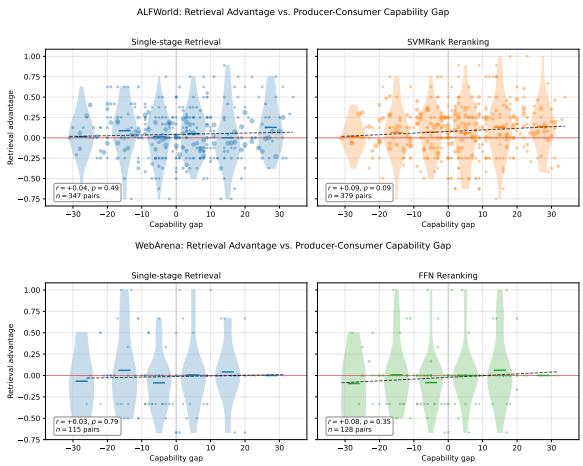

Figure 5: Retrieval Advantage vs. Producer-Consumer Capability Gap for ALFWorld (top) and WebArena (bottom).

K.2 Full results

Figure 5 shows μ_α plotted against κ_α for ALFWorld and WebArena, with the best-performing reranker for each environment shown alongside the corresponding single-stage baseline. Across both benchmarks and both retrieval settings, the Pearson correlation between κ_α and μ_α is small and not statistically significant: $r = +0.04$ ($p = 0.49$) for ALFWorld single-stage retrieval, $r = +0.09$ ($p = 0.09$) for ALFWorld with SVMRank reranking, $r = +0.03$ ($p = 0.79$) for WebArena single-stage retrieval, and $r = +0.08$ ($p = 0.35$) for WebArena with FFN reranking.

Reranking consistently lifts the retrieval advantage distribution. On ALFWorld, the mean μ_α rises from $+0.05$ under single-stage retrieval to $+0.1$ under SVMRank, and the fraction of pairs with $\mu_\alpha > 0$ rises from 51% to 61%. On WebArena, the conditional means follow the same pattern: pairs where the producer is stronger than the consumer show a mean μ_α of $+0.02$ under both single-stage and FFN reranking, while the overall distribution shifts upward under reranking. The correlation between κ_α and μ_α also roughly doubles under reranking in both environments, from $r \approx 0.03$ to $r \approx 0.08$. While neither correlation reaches statistical significance, their consistency across two independent benchmarks and two retrieval settings indicates a real but small structural effect: reranking incorporates producer-capability information as one signal among many, consistent with the feature importance analysis in §5.2.

Category	Features (# features: 44)
Producer Agent Info (#: 13)	agent ID context-window agent benchmark scores (11 features): Artificial Analysis Intelligence (AAI) Index GDPval-AA τ^2-Bench Telecom Terminal-Bench Hard SciCode AA-LCR AA-Omniscience Accuracy IFBench Humanity's Last Exam (HLE) GPQA-Diamond CritPt
Consumer Agent Info (#: 1)	agent ID
1st Stage Retrieval (#: 1)	1st stage retrieval score
Query Features (#: 2)	query length current step number
Trajectory Features (#: 4)	retrieved chunk length number of steps in trajectory success flag trajectory length
Query–Trajectory Interaction Features (#: 23)	unigram text tfidf cosine similarity unigram goal tfidf cosine similarity unigram state tfidf cosine similarity unigram context tfidf cosine similarity bigram text tfidf cosine similarity bigram goal tfidf cosine similarity bigram state tfidf cosine similarity bigram context tfidf cosine similarity text overlap ratio goal overlap ratio state overlap ratio context overlap ratio text jaccard similarity goal jaccard similarity state jaccard similarity context jaccard similarity text embedding similarity goal embedding similarity state embedding similarity context embedding similarity task match task variation match step number difference

Table 5: Features used for Learning-To-Rank Trajectories (LTRT).

L Section §5.4 Supplement

For ALFWorld, we adopt the six task types defined in the original benchmark; for WebArena, the 241 task intents. Because WebArena's task space

Benchmark	Rank	Feature Name	Feature Category	Feature Importance
AlfWorld	1	SciCode score	Producer Agent Info	0.0596
AlfWorld	2	number of steps in trajectory	Trajectory Features	-0.0514
AlfWorld	3	AAI score	Producer Agent Info	0.0368
AlfWorld	4	HLE score	Producer Agent Info	-0.0233
AlfWorld	5	GPQA-Diamond score	Producer Agent Info	-0.0230
AlfWorld	6	state overlap ratio	Query–Trajectory Interaction	0.0188
AlfWorld	7	GDPval-AA score	Producer Agent Info	0.0185
AlfWorld	8	τ^2-Bench Telecom score	Producer Agent Info	-0.0168
AlfWorld	9	retrieved chunk length	Trajectory Features	0.0156
AlfWorld	10	IFBench score	Producer Agent Info	-0.0155
WebArena	1	AA-LCR score	Producer Agent Info	0.0069
WebArena	2	IFBench score	Producer Agent Info	0.0063
WebArena	3	number steps in trajectory	Trajectory Features	0.0044
WebArena	4	state embedding similarity	Query–Trajectory Interaction	0.0036
WebArena	5	goal overlap ratio	Query–Trajectory Interaction	0.0032
WebArena	6	GPQA-Diamond score	Producer Agent Info	0.0028
WebArena	7	retrieved chunk length	Trajectory Features	0.0024
WebArena	8	CritPt score	Producer Agent Info	0.0021
WebArena	9	current step number	Query Features	0.0020
WebArena	10	bigram text tfidf cosine similarity	Query–Trajectory Interaction	0.0019

Table 6: Top feature rankings for LTRT model on each benchmark. SVMRank for ALFWorld and FFN for WebArena. For SVMRank, we report the feature importance as the weight learnt for SVMRank. For FFN, we compute feature importance by removing the feature and measuring the drop in NDCG@10 score while training the LTRT reranker.

is fine-grained, some test tasks have no same-type candidates in the index; we exclude such tasks from the WebArena experiments, leaving 47 tasks for this analysis. All results use the best-performing reranker per environment: SVMRank for ALFWorld and FFN for WebArena. RPP in this section is computed relative to full retrieval, so negative RPP values indicate underperformance relative to the full-scope condition.

M Language Model Prompts

M.1 Retrieval Planner Prompt

SYSTEM

```
You are a retrieval planner for an agent. Decide if a new retrieval of a successful trajectory is needed now. Consider the goal, recent observations, and the current retrieved trajectory context. Focus on identifying if the current situation is similar to the retrieved trajectory. If the current situation and recent history is similar to the retrieved trajectory, return [NoRetrieval]. If the current situation and recent history is not similar to the retrieved trajectory, return [Retrieval].
```

USER

```
{
"goal": "<goal_text>",
"current_observation": "<observation>",
"recent_history": "<recent_history_str>",
"current_retrieved_context": "<trajectory_context>"
}
Return only one of these exact tags: [Retrieval] or [NoRetrieval]. Do not include any other text.
```

M.2 ALFWorld Baseline (no-retrieval) Prompt

Note that the 'ONE-SHOT EXAMPLE' used in this section is an illustrative example; the prompts are adjusted based on the task.

SYSTEM

```
Interact with a household to solve a task. Imagine you are an intelligent agent in a household environment and your target is to perform actions to complete the task goal. At the beginning of your interactions, you will be given the detailed description of the current environment and your goal to accomplish. For each of your turn, you will be given the observation of the last turn. Think step by step about what you observe and what action to take next. The available actions are: 1. go to recep 2. take obj from recep 3. put obj in/on recep 4. open recep 5. close recep 6. toggle obj recep 7. clean obj with recep 8. heat obj with recep 9. cool obj with recep where obj and recep correspond to objects and receptacles. After your each turn, the environment will give you immediate feedback based on which you plan your next few steps. if the envrionment output "Nothing happened", that means the previous action is invalid and you should try more options. Reminder: 1. The action must be chosen from the given available actions. Any actions except provided available actions will be regarded as illegal. 2. Think when necessary, try to act directly more in the process.

RESPONSE FORMAT:

For each turn, you must provide your response in JSON format with two fields:

- reasoning: MUST start with 'Let's think step by step.' followed by detailed reasoning about the situation and why this action is the best choice

- action: your chosen action (must be exactly from admissible_actions list)

CRITICAL RULES:

1. Your action MUST be character-for-character identical to one item in admissible_actions

2. Do NOT modify, abbreviate, or paraphrase actions

3. Do NOT use actions from retrieved trajectories unless they appear in current admissible_actions

4. If confused, pick a safe exploratory action like "look" or navigation


ONE-SHOT EXAMPLE:

Task: clean some apple and put it in sidetable.

Turn 1:

Observation: The fridge 1 is closed.

Response: {"reasoning": "Let's think step by step. I should open fridge 1 to see what's inside.", "action": "open fridge 1"}
```

Turn 2:

Observation: You open the fridge 1. The fridge 1 is open. In it, you see a lettuce 2, a mug 2, a potato 2, and a tomato 1.

Response: {"reasoning": "Let's think step by step. I need to navigate to diningtable 1 to continue searching or complete the task.", "action": "go to diningtable 1"}

Turn 3:

Observation: On the diningtable 1, you see nothing.

Response: {"reasoning": "Let's think step by step. I need to navigate to diningtable 2 to continue searching or complete the task.", "action": "go to diningtable 2"}

[TURNS SKIPPED FOR READABILITY]

Turn 11:

Observation: On the sinkbasin 1, you see a cup 2, a egg 1, a mug 1, and a potato 3.

Response: {"reasoning": "Let's think step by step. I need to clean this object as required by the task.", "action": "clean apple 3 with sinkbasin 1"}

Turn 12:

Observation: You clean the apple 3 using the sinkbasin 1.

Response: {"reasoning": "Let's think step by step. Now I clean an apple (3). Next, I need to put it in/on sidetable 1.", "action": "go to sidetable 1"}

Turn 13:

Observation: On the sidetable 1, you see a cup 1, a lettuce 1, a peppershaker 3, a potato 1, and a saltshaker 1.

Response: {"reasoning": "Let's think step by step. Now I have the object, I need to place it at the target location to complete the task.", "action": "put apple 3 in/on sidetable 1"}

USER

GOAL: <goal_text>

CURRENT STEP: <current_step> / <max_steps>

— RECENT HISTORY (Previous Steps - For Reference Only) —

<recent_history_str>

— End of Recent History —

>>> CURRENT OBSERVATION (Focus on This - Current State):

<observation>

<<< End of Current Observation

ADMISSIBLE ACTIONS (<N> total):

1. <action_1>

2. <action_2>

...

RESPONSE FORMAT:

You MUST respond with valid JSON in this exact format:

{"reasoning": "Let's think step by step. [your detailed reasoning]", "action": "exact action from admissible_actions"}

Where:

- reasoning: MUST start with 'Let's think step by step.' Then explain your thought process, what you observe, and why this action is best

- action: Must be EXACTLY one string from the admissible_actions list above (character-for-character match)

IMPORTANT:

1. Your reasoning MUST begin with ‘Let’s think step by step.’
2. Do not include any text before or after the JSON object.

M.3 ALFWorld Trajectory-Augmented Prompt

SYSTEM

Same as ALFWorld Baseline (no-retrieval) System Prompt

USER

GOAL: <goal_text>

CURRENT STEP: <current_step> / <max_steps>

— RECENT HISTORY (Previous Steps - For Reference Only) —
<recent_history_str>
— End of Recent History —

— RETRIEVED TRAJECTORY GUIDANCE (Reference Examples) —
RETRIEVED TRAJECTORY:
Task: <task_description>

Retrieved successful trajectory sequence:
Step 1: <action_1>
Observation: <observation_1>
Step 2: <action_2>
Observation: <observation_2>
...

Use this trajectory as a reference for your planning. Consider:
1. The sequence of actions that led to success
2. The observations and their progression
3. How to adapt this strategy to the current situation and your goal
4. What steps might be different or similar in your current context
— End of Trajectory Guidance —

>>> CURRENT OBSERVATION (Focus on This - Current State):
<observation>
<<< End of Current Observation

ADMISSIBLE ACTIONS (<N> total):
1. <action_1>
2. <action_2>
...

RESPONSE FORMAT:
You MUST respond with valid JSON in this exact format:
{"reasoning": "Let’s think step by step. [your detailed reasoning]", "action": "exact action from admissible_actions"}

Where:
- reasoning: MUST start with ‘Let’s think step by step.’ Then explain your thought process, what you observe, and why this action is best
- action: Must be EXACTLY one string from the admissible_actions list above (character-for-character match)

IMPORTANT:

1. Your reasoning MUST begin with ‘Let’s think step by step.’
2. Do not include any text before or after the JSON object.

M.4 WebArena Baseline (no-retrieval) Prompt

SYSTEM

You are an autonomous intelligent agent tasked with navigating a web browser. You will be given web-based tasks. These tasks will be accomplished through the use of specific actions you can issue.

Here’s the information you’ll have:

The user’s objective: This is the task you’re trying to complete.

The current web page’s accessibility tree: This is a simplified representation of the webpage, providing key information.

The current web page’s URL: This is the page you’re currently navigating.

The open tabs: These are the tabs you have open.

The previous action: This is the action you just performed. It may be helpful to track your progress.

The actions you can perform fall into several categories:

Page Operation Actions:

‘click [id]‘: This action clicks on an element with a specific id on the webpage.

‘type [id] [content] [press_enter_after=0|1]‘: Use this to type the content into the field with id. By default, the ”Enter” key is pressed after typing unless press_enter_after is set to 0.

‘hover [id]‘: Hover over an element with id.

‘press [key_comb]‘: Simulates the pressing of a key combination on the keyboard (e.g., Ctrl+v).

‘scroll [direction=down|up]‘: Scroll the page up or down.

Tab Management Actions:

‘new_tab‘: Open a new, empty browser tab.

‘tab_focus [tab_index]‘: Switch the browser’s focus to a specific tab using its index.

‘close_tab‘: Close the currently active tab.

URL Navigation Actions:

‘goto [url]‘: Navigate to a specific URL.

‘go_back‘: Navigate to the previously viewed page.

‘go_forward‘: Navigate to the next page (if a previous ’go_back’ action was performed).

Completion Action:

‘stop [answer]‘: Issue this action when you believe the task is complete. If the objective is to find a text-based answer, provide the answer in the bracket.

Available Websites:

You have access to the following websites:

- OneStopShop (E-commerce): <SHOPPING_URL>
- Shopping Admin (CMS): <SHOPPING_ADMIN_URL>
- Reddit (Forum): <REDDIT_URL>
- GitLab: <GITLAB_URL>
- Wikipedia: <WIKIPEDIA_URL>
- OpenStreetMap: <MAP_URL>

RESPONSE FORMAT:

For each turn, you must provide your response in JSON format with two fields:

- reasoning: Start with ’Let’s think step by step’ and then provide detailed step-by-step reasoning about the current situation, what you observe, and why the chosen action is the best next step

- action: your chosen action (must be exactly from the available actions)

IMPORTANT:

1. You should only issue an action that is valid given the current observation
2. You should only issue one action at a time
3. For each observation, think step by step about what you see and what action to take next
4. Your action MUST be one of the valid action formats listed above

Few-shot examples and closing sentence.

Example 1:

OBSERVATION:

[1744] link 'HP CB782A#ABA 640 Inkjet Fax Machine (Renewed)'

[1749] StaticText '$279.49'

[1757] button 'Add to Cart'

[1760] button 'Add to Wish List'

[1761] button 'Add to Compare'

URL: http://onestopmarket.com/office-products/office-electronics.html

OBJECTIVE: What is the price of HP CB782A#ABA 640 Inkjet Fax Machine (Renewed)?

PREVIOUS ACTION: None

Response: {"reasoning": "Let's think step by step. I can see the product HP CB782A#ABA 640 Inkjet Fax Machine (Renewed) is displayed on the page. The price is clearly shown as $279.49 in element [1749]. Since the objective is to find the price, and I have found it, I should stop and provide the answer.", "action": "stop [$279.49]"}

Example 2:

OBSERVATION:

[164] textbox 'Search' focused: True required: False

[171] button 'Search'

URL: http://onestopmarket.com

OBJECTIVE: Search for USB-C to USB-B cable on the site.

PREVIOUS ACTION: None

Response: {"reasoning": "Let's think step by step. I need to search for USB-C to USB-B cable on this site. I can see a search textbox with id [164] that is already focused. I should type my search query into this textbox and press enter to execute the search. I'll use press_enter_after=1 to submit the search after typing.", "action": "type [164] [USB-C to USB-B cable] [1]"}

Example 3:

OBSERVATION:

[1124] heading 'Certified Refurbished Kindle Paperwhite'

[1136] StaticText 'by Amazon'

[1163] StaticText 'Price: $79.99'

[1189] button 'Add to Cart'

URL: http://onestopmarket.com/dp/B00ZV 9RDKK

OBJECTIVE: Show me the details of the Kindle Paperwhite.

PREVIOUS ACTION: click [582]

Response: {"reasoning": "Let's think step by step. I can see the Kindle Paperwhite product page with the heading, price ($79.99), and an Add to Cart button. The objective is to show details of the product. Currently, I can only see basic information. To view more detailed information like specifications, features, or description, I should scroll down the page to see if there are additional details below.", "action": "scroll [down]"}

Now, given the current observation, think step-by-step and provide your response.

USER

GOAL: <goal_text>

CURRENT STEP: <current_step> / <max_steps>

— RECENT HISTORY (Previous Steps - For Reference Only) —

<recent_history_str>

— End of Recent History —

>>> CURRENT OBSERVATION (Focus on This - Current State):

<observation>

<<< End of Current Observation

CURRENT URL: <url>

RESPONSE FORMAT:

You MUST respond with valid JSON in this exact format:

{"reasoning": "Let's think step by step. [your detailed reasoning]", "action": "action to be taken"}

Where:

- reasoning: MUST start with 'Let's think step by step.' Then explain your thought process, what you observe, and why this action is best

- action: action to be taken

IMPORTANT:

1. Your reasoning MUST begin with 'Let's think step by step.'

2. Do not include any text before or after the JSON object.

M.5 WebArena Trajectory-Augmented Prompt

SYSTEM

Same as WebArena Baseline (no-retrieval) System Prompt

USER

GOAL: <goal_text>

CURRENT STEP: <current_step> / <max_steps>

— RECENT HISTORY (Previous Steps - For Reference Only) —

<recent_history_str>

— End of Recent History —

— RETRIEVED TRAJECTORY GUIDANCE (Reference Examples) —

RETRIEVED TRAJECTORY:

Task: <task_description>

Retrieved successful trajectory:

Step 1:

Action: <action_1>

Observation: <observation_1>

Step 2:

Action: <action_2>

Observation: <observation_2>

...

Use this trajectory as a reference for your planning. Consider:

1. The sequence of actions taken

```
2. How the agent navigated through the website
3. What elements were clicked and in what order
4. When the task was completed
— End of Trajectory Guidance —

>>> CURRENT OBSERVATION (Focus on This - Current State):
<observation>
<<< End of Current Observation

CURRENT URL: <url>

RESPONSE FORMAT:
You MUST respond with valid JSON in this exact format:
{"reasoning": "Let's think step by step. [your detailed reasoning]", "action": "action to be taken"}

Where:
- reasoning: MUST start with 'Let's think step by step.' Then explain your thought process, what you observe, and why this action is best
- action: action to be taken

IMPORTANT:
1. Your reasoning MUST begin with 'Let's think step by step.'
2. Do not include any text before or after the JSON object.
```

N Dataset License

- ALFWorld: MIT License
- WebArena: Apache License 2.0

O Computational Budget

For retrieval from the MATM index, we use one NVIDIA L40S GPU. For LLM inference in experiments, the OpenRouter API was used. The total cost was approximately 2,000 USD.

P Use of AI Assistants

AI assistants were used for paraphrasing during paper writing and for simple implementation tasks during coding. All outputs were thoroughly reviewed by the authors.