Learning Hierarchical Procedural Memory for LLM Agents through Bayesian Selection and Contrastive Refinement

Saman Forouzandeh
School of Engineering, Royal Melbourne Institute of Technology University
Melbourne, Australia
saman.forouzandeh@rmit.edu.au

Wei Peng
School of Engineering, Royal Melbourne Institute of Technology University
Melbourne, Australia
wei.peng3@rmit.edu.au

Parham Moradi
College of Arts, Business, Law Education, and Information Technology, Victoria University
Melbourne, Australia
parham.moradi@vu.edu.au

Xinghuo Yu
School of Engineering, Royal Melbourne Institute of Technology University
Melbourne, Australia
xinghuo.yu@rmit.edu.au

Mahdi Jalili
School of Engineering, Royal Melbourne Institute of Technology University
Melbourne, Australia
mahdi.jalili@rmit.edu.au

ABSTRACT

We present **MACLA**, a framework that decouples reasoning from learning by maintaining a frozen large language model (LLM) while performing all adaptation in an external hierarchical procedural memory. MACLA extracts reusable procedures from trajectories, tracks reliability via Bayesian posteriors, selects actions through expected-utility scoring, and refines procedures by contrasting successes vs. failures. Across four benchmarks (ALFWorld, WebShop, TravelPlanner, InterCodeSQL), MACLA achieves **78.1% average performance**, outperforming all baselines. On ALFWorld unseen tasks, MACLA reaches **90.3%** with **+3.1% positive generalization**. The system constructs memory in **56 seconds** (2,800× faster than the state-of-the-art LLM parameter-training baseline), compresses **2,851 trajectories into 187 procedures** (15:1). Experimental results demonstrate that structured external memory with Bayesian selection and constrastive refinement enable sample-efficient, interpretable and continually improving agents without LLM parameter updates. **Code is publicly available at** https://github.com/S-Forouzandeh/MACLA-LLM-Agents-AAMAS-Conference.

KEYWORDS

Memory-augmented agents; Procedural memory; Bayesian decision making; Contrastive learning; LLM agents

ACM Reference Format:
Saman Forouzandeh, Wei Peng, Parham Moradi, Xinghuo Yu, and Mahdi Jalili. 2026. Learning Hierarchical Procedural Memory for LLM Agents through Bayesian Selection and Contrastive Refinement. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 9 pages. https://doi.org/10.65109/FKYO8341

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/FKYO8341

Large language model (LLM) agents can solve complex, interactive tasks such as web shopping [25] and embodied AI housekeeping [9], by transforming natural-language instructions into sequences of environment actions [26]. In these settings, agents navigate step-by-step through partially observable environments to pursue subgoals and ultimately complete the task [9, 22]. The resulting *trajectory* is the ordered record of an episode's interaction, typically written as (T, A, O, R), where T represents a task to complete, A are actions, O stand for observations for the outcome of corresponding actions, and R records step-level outcomes or rewards. Trajectories thus capture the full decision process, not merely terminal success or failure, and provide dense supervision for how an agent progresses through a task [16, 22]. When a new task arrives, the agent synthesizes an appropriate trajectory (that is, a step-by-step plan and its execution) to achieve the goal in the current context, deciding which information to gather, which tools to invoke, and which subroutines to chain in order to achieve completion [25, 26].

Early LLM agents used prompt-based planning [26] and self-critique [15], but lack persistent *"how-to"* procedures — when tasks are similar but not identical, agents must re-plan from scratch, increasing cost and latency. Fine-tuning approaches [2, 27, 29] adapt agents via supervised learning or RLHF, but typically treat entire trajectories as single units weighted by terminal success/failure, neglecting rich intermediate steps. In practice, failed trajectories often contain correct substeps (e.g., "successfully navigating and retrieving an egg, but failing to boil it" [16]), while successful ones may include suboptimal actions that accidentally cancel out. Recent work [22] addresses this via step-level rewards, but requires repeated policy training on densely-labeled data, incurring substantial computational cost.

We introduce **MACLA** (Memory-Augmented Contrastive Learning Agent), a framework that disentangles *reasoning* from *learning* through the coupling a frozen LLM and a structured external procedural memory (Figure 1). Unlike fine-tuning approaches where reasoning and adaptation are entangled within billions of parameters, MACLA fixes the LLM as a stable semantic reasoner responsible for trajectory segmentation, abstraction, and action generation.

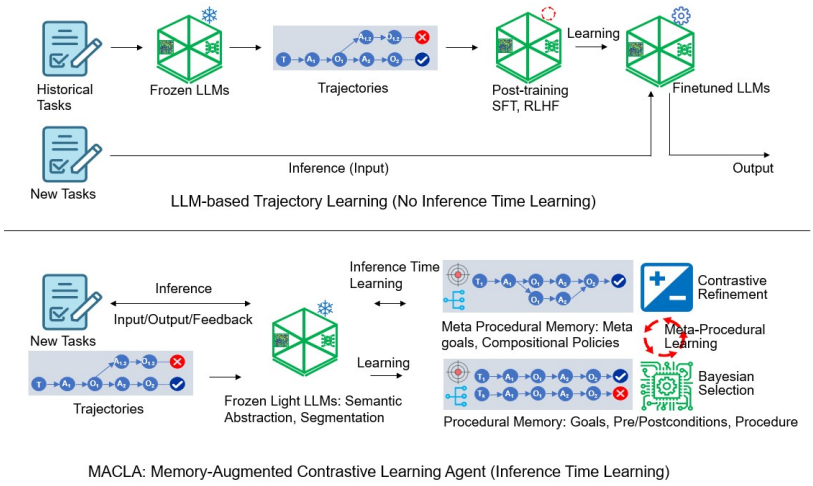

Figure 1: Comparison between existing LLM-based trajectory learning (top) and the proposed memory-augmented contrastive learning agent (MACLA, bottom). Existing methods train trajectories (T, A, O, R) (Task, Action, Observation, Reward) into LLM parameters through post-training (finetuning and/or RLHF), whereas MACLA constructs procedural and meta-procedural memory externally through frozen LLM abstraction, segmentation, Bayesian selection, and contrastive refinement. Memories are learned during memory construction. Besides learning during memory construction, MACLA enables inference-time learning in which outputs are verified in the task environment, with feedback used for contrastive refinement on the retrieved memories. Meta-procedural learning enables the composition policy to be learned among procedures.

All learning occurs externally through explicit, interpretable memory operations - maintaining human-readable procedures, updating Bayesian posteriors, and refining preconditions through contrastive analysis. MACLA operates through three core mechanisms:

(1) **Bayesian procedure selection:** Maintains Beta posteriors $\text{Beta}(\alpha_i, \beta_i)$ over procedure success rates and ranks candidates via expected-utility scoring that balances contextual relevance, success probability, failure risk, and information gain, providing principled exploration-exploitation.
(2) **Contrastive refinement:** Compares successful and failed execution contexts to tighten preconditions, repair action sequences, and refine postconditions once procedures accumulate sufficient evidence (i.e., $\geq$ a threshold), progressively improving procedure quality through memory edits rather than gradient updates.
(3) **Meta-procedural learning:** Composes frequently co-occurring procedures into hierarchical "playbooks" with conditional control policies (continue, skip, repeat, abort) for long-horizon tasks, enabling strategic reuse beyond atomic skills.

This architecture yields sample-efficient, interpretable agents with human-readable procedural knowledge, closed-form utility computation, and minimal LLM usage. Specifically, this work contributes:

- **Online procedural memory adaptation:** Continual updates to procedural and meta-procedural memory during and after episodes, enabling adaptation without weight updates, compared with offline LLM post-training approaches [17, 22, 29] that remain static at inference.
- **Reasoning/learning decoupling:** A frozen LLM for parsing and abstraction with all improvements occurring in an external, structured procedural memory, avoiding the computational cost and catastrophic forgetting risks of parameter fine-tuning.
- **Bayesian uncertainty-aware selection:** A principled procedure selection module that maintains Beta posteriors over success rates with closed-form expected utility objectives balancing relevance, success probability, failure risk and information gain.
- **Contrastive procedural refinement:** An algorithm leveraging paired successes and failures to tighten preconditions, repair action schemas, and refine postconditions of stored procedures without requiring expert demonstrations.
- **Hierarchical meta-procedural composition:** Automatic discovery and maintenance of conditional *playbooks* with control policies (skip, repeat, abort) for long-horizon tasks, enabling compositional generalization.

We evaluate MACLA across four benchmarks (ALFWorld [9], WebShop [25], TravelPlanner [21], InterCodeSQL [24]), achieving **78.1% average performance** — the highest among all methods, including those using models 10× larger (later in Table 1). On ALFWorld [16], MACLA reaches 87.2% on seen and 90.3% on unseen tasks, with a positive generalization gap (+3.1%) indicating compositional transfer rather than overfitting. The system achieves this with only 0.016 GPU-hours for one-time memory construction — 2,800× faster than the state-of-the-art LLM parameter-training baseline [22], which requires 44.8 GPU-hours of iterative training

— while simultaneously producing human-interpretable procedural knowledge.

1 RELATED WORKS

LLM agents have advanced rapidly in reasoning and decision-making, enabling multi-step interaction in embodied and web-based environments. Early frameworks such as ReAct[26] and Reflexion[14] integrate reasoning and acting within the same loop, while trajectory-tuning methods [2, 22] fine-tune models using expert demonstrations. However, fine-tuning is computationally expensive, requires offline data collection and training cycles, and does not support true online adaptation at inference time. To overcome this issue, a line of research augments LLM agents with memory for continuous reasoning. Memory is a foundational component of language agents, supporting competence across multiple timescales from transient working context to persistent long-term knowledge [6, 8, 31]. Research on memory for LLM agents can be usefully organized along two directions: where memory resides and what is stored. Along the first direction, some methods such as MemGPT [11] and MemoryBank [32], use buffer-based systems to store conversational or episodic traces and retrieve them with embedding search and simple heuristics. Some others, such as HiAgent[5], A-Mem[23], MemAgent [28] use hierarchical designs to separate working buffers from episodic and long-term stores to relieve context pressure and improve persistence. Recently, SAGE [7] used reflective multi-agent controllers to curate these stores while controlling growth. The second direction concerns what is stored. Many systems retain free-form text snippets such as notes, summaries, or dialogue chunks; these are easy to write but suffer from retrieval drift and weak compositionality as repositories scale [11, 32]. More structured artifacts appear as tuples and key–value frames (e.g., tool logs or entity/event graphs), which aid filtering but still lack executable semantics for reuse. A growing line of work targets skills and procedures: agents capture reusable action patterns, tool workflows, and instruction-like steps across related tasks [3, 19, 20]. Memp [4] advances this view by treating procedural memory as a first-class object and studying its construction, retrieval, and update across domains. However, several key limitations remain; (1) it represents know-how largely as monolithic text (scripts or full trajectories) with heuristic retrieval and simple updates; (2) it lacks uncertainty-aware selection or principled exploration-exploitation balance, preventing reason about reliability or risk of retrieved memory; and (3) it lacks a mechanism to refine procedures from paired successes and failures or abstract recurring patterns into meta-procedural compositions. Comparatively, we represent experience as structured, hierarchical procedures with explicit preconditions, action schemas, and postconditions, enabling interpretable reuse and safe composition and direct schema edits when evidence warrants change. The proposed approach enables the system to continuously adapt and improve.

2 PROPOSED METHOD

The key components of MACLA are described in detail below.

3.1 LLM-based Procedural Abstraction.

The first stage transforms raw episodic trajectories into structured, reusable procedural knowledge. Given a trajectory $\tau = \{(o_t, a_t, r_t)\}_{t=0}^{T}$ consisting of textual observations o_t, primitive actions a_t, and rewards r_t, the frozen LLM $\mathcal{L}_\theta$ receives the full trajectory and identifies semantically coherent segments that correspond to meaningful sub-tasks:

$$\mathcal{S}eg = \mathcal{L}_\theta\big(\text{Prompt}_{\text{segment}}(\tau)\big) = \{(t_k^{\text{start}}, t_k^{\text{end}}, d_k)\}_{k=1}^{K}, \tag{1}$$

where each segment k spans time steps $[t_k^{\text{start}}, t_k^{\text{end}}]$ and is summarized by a description d_k. For each segment, MACLA constructs a structured procedure $\text{Proc}_k = \langle \mathcal{G}_k, \Psi_k, \pi_k, \Phi_k \rangle$, where $\mathcal{G}_k$ is a natural-language goal, Ψ_k are precondition patterns inferred from the observations before the segment, π_k is an abstracted action sequence, and Φ_k are postcondition patterns extracted from the final observations. This decomposition produces interpretable "how-to" skills that can be invoked whenever their preconditions are met. To support retrieval and merging, each procedure is embedded into a semantic vector space using an encoder ϕ, $\mathbf{e}_k = \phi([\mathcal{G}_k; \Psi_k; \Phi_k]) \in \mathbb{R}^d$. When a new procedure is created, it is compared to existing ones via cosine similarity, $i^* = \arg\max_i \text{sim}(\mathbf{e}_k, \mathbf{e}_i)$. If $\text{sim}(\mathbf{e}_k, \mathbf{e}_{i^*}) > \theta_{\text{dup}}$, the new procedure is merged into the existing one by expanding its condition sets; otherwise, a new entry is added. This process yields a continually growing procedural library $\mathbb{M}_{\text{proc}} = \{(\text{Proc}_i, \mathbf{e}_i, \alpha_i, \beta_i)\}_{i=1}^{N_p}$ that forms the foundation for later Bayesian selection and refinement.

3.2 Bayesian Reliability and Utility Selection.

Given the procedural library, the agent must decide which procedure to execute for the current observation. Each procedure Proc_i maintains a Beta posterior over its success probability $\rho_i \in [0, 1]$:

$$p(\rho_i | \mathcal{D}_i) = \text{Beta}(\rho_i; \alpha_i, \beta_i) \tag{2}$$

where α_i and β_i accumulate successful and failed executions from history $\mathcal{D}_i$. The posterior mean $\mathbb{E}[\rho_i] = \alpha_i/(\alpha_i + \beta_i)$ estimates current reliability, while the variance $\text{Var}[\rho_i] = \frac{\alpha_i \beta_i}{(\alpha_i+\beta_i)^2(\alpha_i+\beta_i+1)}$ quantifies epistemic uncertainty. For each candidate, we compute expected utility by integrating over the Beta posterior. Given utility $U(\rho \mid o_t, i) = \text{Rel}_i(o_t) \cdot \rho \cdot R_{\max} - \text{Risk}_i(o_t) \cdot (1-\rho) \cdot C_{\text{fail}} + \lambda_{\text{info}} \cdot I(\rho)$, the expected utility is:

$$\text{EU}(\text{Proc}_i \mid o_t) = \int_0^1 U(\rho \mid o_t, i)\, \text{Beta}(\rho; \alpha_i, \beta_i)\, d\rho \tag{3}$$

Exploiting $\mathbb{E}_{\text{Beta}(\alpha,\beta)}[\rho] = \frac{\alpha}{\alpha+\beta}$ and $\mathbb{E}[1-\rho] = \frac{\beta}{\alpha+\beta}$, this simplifies to:

$$\text{EU}(\text{Proc}_i \mid o_t) = \underbrace{\text{Rel}_i(o_t)\, \frac{\alpha_i}{\alpha_i + \beta_i}\, R_{\max}}_{\text{expected reward}} - \underbrace{\text{Risk}_i(o_t)\, \frac{\beta_i}{\alpha_i + \beta_i}\, C_{\text{fail}}}_{\text{failure cost}} + \underbrace{\lambda_{\text{info}}\, H[\text{Beta}(\alpha_i, \beta_i)]}_{\text{information gain}} \tag{4}$$

where $\text{Rel}_i(o_t) = \cos(\phi(o_t), \mathbf{e}_i)$ is contextual similarity, $\text{Risk}_i(o_t)$ is the fraction of past failures with similar contexts, and $H[\cdot]$ is differential entropy encouraging exploration. The selected procedure is:

$$\text{Proc}_t^* = \arg \max_{\text{Proc}_i \in C_t} \text{EU}(\text{Proc}_i | o_t) \tag{5}$$

subject to confidence threshold θ_{conf}. If $\max_i \text{EU}(\text{Proc}_i | o_t) < \theta_{\text{conf}}$, the agent falls back to zero-shot LLM reasoning. This Bayesian selection mechanism balances exploitation (high $\frac{\alpha}{\alpha+\beta}$ procedures),

risk aversion (avoiding contexts similar to past failures), and exploration (high entropy procedures). The expected utility formulation naturally handles the explore-exploit tradeoff: early in learning, high entropy dominates selection, while after sufficient evidence accumulates, expected reward becomes the primary driver.

3.3 Contrastive Refinement of Procedures.

As experience accumulates, procedures with both successful and failed instances are subjected to contrastive refinement to improve their accuracy and robustness. For a procedure Proc_i with sets of successful and failed contexts $\mathcal{S}_i$ and $\mathcal{F}_i$, the LLM performs discriminative comparison, $\mathcal{D}_i = \text{ContrastiveExtract}(\mathcal{S}_i, \mathcal{F}_i)$, identifying differences in three dimensions: (i) precondition patterns ($\Delta\Psi_i^+$ and $\Delta\Psi_i^-$) that distinguish successful from failed initial contexts, (ii) action discrepancies ($\Delta\pi_i$) revealing missing or misordered actions, and (iii) postcondition mismatches ($\Delta\Phi_i$) that capture incomplete goal states. These discriminators drive explicit refinement operations

$$\begin{aligned} \Psi_i &\leftarrow \Psi_i \cup \Delta\Psi_i^+ \cup \{\neg\psi \mid \psi \in \Delta\Psi_i^-\}, \\ \pi_i &\leftarrow \text{Merge}(\pi_i, \Delta\pi_i), \\ \Phi_i &\leftarrow \Phi_i \cup \Delta\Phi_i. \end{aligned} \tag{6}$$

When distinct execution modes are detected, the procedure is specialized into separate variants with inherited reliability priors. This process progressively tightens applicability conditions and action precision, yielding interpretable improvements purely through memory edits rather than gradient updates.

3.4 Meta-procedural Composition.

To extend reasoning beyond atomic skills, MACLA automatically discovers and learns meta-procedures that are structured compositions of procedures that capture recurrent long-horizon strategies. When a sequence of procedures $\langle \text{Proc}_{i_1}, \ldots, \text{Proc}_{i_m} \rangle$ repeatedly leads to success under a common high-level goal, the agent abstracts it as $\text{MP}_j = \langle \mathcal{G}_j^{\text{meta}}, \Psi_j^{\text{meta}}, \{\text{Proc}_{i_1}, \ldots, \text{Proc}_{i_m}\}, \Theta_j \rangle$. Here, Θ_j denotes a lightweight control policy governing conditional transitions among sub-procedures based on the current observation and execution context, $\Theta_j(o_t, \text{index}) \in \{\text{continue}, \text{skip}, \text{repeat}, \text{abort}\}$. This policy is distilled by analyzing successful traces, where the LLM identifies observation patterns that triggered each branch—for example, repeating when postconditions are unmet, skipping when preconditions already hold, or aborting when failures recur. Each meta-procedure maintains its own Beta success posterior $p(\sigma_j|\mathcal{D}_j) = \text{Beta}(\alpha_j, \beta_j)$ and is refined periodically to add new branches, reorder sub-procedures, or prune redundant ones. Through these hierarchical compositions, MACLA acquires flexible "playbooks" that encapsulate extended strategies with conditional logic.

3.5 Ontological Semantic Grounding.

To enable cross-context generalization (e.g., procedures learned on "mug" applying to "cup"), MACLA constructs a lightweight **ontological semantic index** during offline memory construction. We extract the k_{vocab} most frequent words from task descriptions and actions, then cluster semantically similar words using Sentence-Transformer embeddings [12] to form an implicit domain ontology:

$$C_w = \{w' \mid \text{sim}(\phi(w), \phi(w')) > \theta_{sim}\} \tag{7}$$

where each cluster C_w represents a semantic category (e.g., $C_{\text{container}} = \{\text{mug}, \text{cup}, \text{glass}\}$). During retrieval, observations are mapped to these ontological categories, allowing procedures to match across lexically different but semantically equivalent contexts. This ontological grounding enables domain-adaptive generalization without requiring explicit knowledge engineering.

3.6 System Efficiency and Memory Management.

To ensure practical scalability, MACLA employs efficient retrieval, bounded growth, and strict control over LLM usage. All procedures and meta-procedures are embedded in an approximate nearest-neighbor index supporting sublinear retrieval ($O(\log N_p)$) for semantic search. The episode buffer stores at most $N_b = 1000$ steps, providing local context for LLM prompts and post-episode updates. Each procedure maintains a failure index limited to $K_{\text{fail}} = 15$ entries, managed through success-based removal, redundancy-aware eviction, and temporal decay, ensuring that memory remains concise and informative. To prevent memory saturation, procedures and meta-procedures are periodically pruned using a multi-factor utility score that balances reliability, usage frequency, and temporal relevance:

$$U(\text{Proc}_i) = \lambda_r \cdot \frac{\alpha_i}{\alpha_i + \beta_i} + \lambda_f \cdot \frac{n_i}{N_{\text{total}}} + \lambda_t \cdot e^{-(t_{\text{current}} - t_i^{\text{last}})/\tau} \tag{8}$$

where $\frac{\alpha_i}{\alpha_i+\beta_i}$ is the Bayesian success rate (reliability), n_i is the execution count of procedure i, N_{total} is the total invocations across all procedures in the current episode window, t_{current} is the current episode index, t_i^{last} is the episode when i was last used, and τ is the temporal decay constant.

The weighting coefficients λ_r=0.5, λ_f=0.3, and λ_t=0.2 reflect the relative importance of each factor: reliability receives the highest weight (0.5) as it directly predicts future success; frequency receives moderate weight (0.3) to favor well-tested procedures while avoiding over-retention of obsolete frequently-used skills; recency receives the lowest weight (0.2) to provide soft temporal decay without aggressive forgetting. These values were determined through grid search over $\{0.3, 0.4, 0.5, 0.6\} \times \{0.2, 0.3, 0.4\} \times \{0.1, 0.2, 0.3\}$ on ALFWorld validation, with the constraint $\lambda_r + \lambda_f + \lambda_t = 1.0$ for interpretability. The selected configuration (0.5, 0.3, 0.2) yielded the best balance between retaining high-quality procedures (>0.7 success rate) and pruning low-utility entries (<0.4 success rate), as validated later in Figure 4. Entries with the lowest utility are removed while ensuring diversity across goal clusters through stratified sampling. These operations keep the total memory footprint below 4 MB for hundreds of procedures.

Finally, MACLA limits LLM usage to a fixed budget of API calls per episode to cover segmentation, abstraction, and occasional refinement, while all retrieval, Bayesian scoring, and updates are symbolic or vectorized. As a result, per-step runtime remains effectively constant and inference cost does not scale with experience. This memory-first design ensures that MACLA remains efficient, interpretable, and deployable for continual learning across long interaction horizons.The theoretical foundations are provided in Appendix **??**.

3.7 Algorithm.

At runtime, MACLA executes a new task by coupling frozen semantic reasoning with memory-driven decision making. The agent receives an initial observation o_0 (and, optionally, an instruction string) and embeds it as $\mathbf{h}_0 = \phi(o_0)$. This embedding queries the semantic index of the external memory to retrieve a compact candidate set consisting of procedures $\{\text{Proc}_i\}$ and meta-procedures

$\{\text{MP}_j\}$ whose embeddings are most similar to $\mathbf{h}_0$. Retrieval is approximate nearest neighbor over the concatenated descriptors of goals, preconditions, and postconditions, which keeps lookup sublinear in memory size.

Given the candidate set, MACLA ranks each item with a Bayesian expected-utility score that trades off contextual relevance, estimated success, risk, and information gain under the procedure's Beta posterior. The highest-scoring item above a confidence threshold is selected; otherwise the agent falls back to zero-shot LLM reasoning for that step, logs the outcome, and continues. If a meta-procedure is chosen, execution proceeds hierarchically under its composition policy $\Theta_j(o_t, \text{index}) \in \{\text{continue}, \text{skip}, \text{repeat}, \text{abort}\}$ until completion or abort; if an atomic procedure is chosen, the agent checks preconditions Ψ_i against o_t, invokes the action sketch π_i via the frozen LLM's action formatter, and verifies postconditions Φ_i to certify completion. In both cases the outcome updates (α, β) and appends the initial context to the corresponding success or failure set for later analysis.

After each execution, the agent re-embeds the new observation and repeats retrieval and selection until the task is solved or a horizon is reached. When a procedure accumulates both successes and failures, a contrastive pass is triggered: the LLM proposes discriminators that tighten Ψ_i, repair π_i, and refine Φ_i, or if distinct modes are detected, specializes the procedure into variants that inherit prior counts. When successful episodes repeatedly traverse a small set of procedures in a stable order, the agent abstracts a meta-procedure with its own success posterior and a lightweight Θ_j distilled from divergence points across traces. Throughout, memory remains bounded by pruning with a utility that blends reliability, frequency, and recency, and the LLM-call budget is capped, as retrieval, scoring, and updates are vectorized operations. The complete runtime procedure is outlined in Algorithm 1.

Algorithm 1 MACLA Runtime Procedure with Function Descriptions

Require: observation o_0, memory $\mathbb{M}$ (procedures, meta-procedures, indices), horizon H

$\mathbf{h} \leftarrow \phi(o_0)$ ▷ Embed observation
$C \leftarrow \textsc{RetrieveCandidates}(\mathbf{h}, \mathbb{M})$ ▷ Top-k ANN search
while not $\textsc{Terminal}$ and $t < H$ **do**
  for all $c \in C$ **do**
    $\text{EU}[c] \leftarrow \textsc{ExpectedUtility}(c, o_t, \mathbb{M})$ ▷ Compute Eq. 4
  end for
  $c^\star \leftarrow \arg\max_{c \in C} \text{EU}[c]$
  if $\text{EU}[c^\star] < \theta_{\text{conf}}$ **then**
    $(o_{t+1}, y) \leftarrow \textsc{ZeroShotStep}(o_t)$ ▷ LLM generates action directly
  else if $c^\star$ is MP_j **then**
    $(o_{t+1}, y) \leftarrow \textsc{ExecuteMeta}(\text{MP}_j, \Theta_j, o_t)$ ▷ Run with control policy
    $(\alpha_j, \beta_j) \leftarrow \textsc{UpdateBeta}((\alpha_j, \beta_j), y)$ ▷ $\alpha \leftarrow \alpha + y$, $\beta \leftarrow \beta + (1-y)$
  else ▷ $c^\star$ is atomic Proc_i
    if $\textsc{CheckPre}(\Psi_i, o_t)$ **then** ▷ Verify preconditions match o_t
      $(o_{t+1}, y) \leftarrow \textsc{ExecuteProc}(\pi_i, o_t)$ ▷ Instantiate & execute π_i
      $y \leftarrow y \wedge \textsc{CheckPost}(\Phi_i, o_{t+1})$ ▷ Verify postconditions in o_{t+1}
    else
      $(o_{t+1}, y) \leftarrow \textsc{ZeroShotStep}(o_t)$ ▷ Preconditions failed, fallback
    end if
    $(\alpha_i, \beta_i) \leftarrow \textsc{UpdateBeta}((\alpha_i, \beta_i), y)$
    $\textsc{RecordContext}(\mathcal{S}_i, \mathcal{F}_i, o_t, y)$ ▷ Add to success/fail sets
  end if
  if $\textsc{RefineTrigger}(c^\star)$ **then** ▷ If $|\mathcal{S}|, |\mathcal{F}| \geq 3$
    $\textsc{ContrastiveRefine}(c^\star)$ ▷ LLM compares $\mathcal{S}$ vs. $\mathcal{F}$ (§2)
  end if
  $\mathbf{h} \leftarrow \phi(o_{t+1})$; $C \leftarrow \textsc{RetrieveCandidates}(\mathbf{h}, \mathbb{M})$; $t \leftarrow t + 1$
end while
if $\textsc{EligibleForMeta}(\text{trace})$ **then** ▷ If ≥3 procs in stable order
  $\textsc{ExtractOrRefineMeta}(\text{trace}, \mathbb{M})$ ▷ Create/update meta-proc
end if
$\textsc{PruneAndMaintain}(\mathbb{M})$ ▷ Remove low-utility via Eq. 8

3 EXPERIMENTS

We evaluate MACLA on four challenging interactive agent benchmarks spanning diverse domains. All experiments use consistent hyperparameters across tasks to demonstrate generalization without task-specific tuning.

Memory Architecture: Episode buffer N_{buffer}=1000 (stores recent observations and actions for temporal context provision during action generation); procedural memory N_{proc}=200 (capacity for extracted reusable skills); meta-procedural memory N_{meta}=50 (capacity for hierarchical procedure compositions). Critically, MACLA does not store raw trajectories. Instead, the LLM segments each episode into coherent sub-tasks and extracts structured procedures (Section 2). Duplicate detection with similarity threshold θ_{dup}=0.85 prevents redundant storage. Through this process, the 2,851 ALFWorld training trajectories compress into approximately 187 unique procedures—demonstrating efficient knowledge distillation from experience.

Bayesian Selection. Information gain weight λ_{info}=0.1, failure cost C_{fail}=0.5. These parameters balance exploration (trying uncertain procedures to reduce epistemic uncertainty) with exploitation (selecting high-posterior reliable procedures).

Contrastive Refinement. Minimum contexts n^s_{min}=n^f_{min}=3. Refinement activates only when a procedure has accumulated at least 3 successes and 3 failures, ensuring sufficient statistical evidence for discriminative pattern extraction.

LLM Configuration. Llama-2-7B [18] via Ollama with 4-bit quantization and temperature T=0.7. The LLM parameters remain frozen throughout all experiments—learning occurs exclusively through external memory updates.

Benchmarks and Dataset Statistic: ALFWorld [16] (2,851 train, 274 test) is a text-based embodied environment with six household tasks (e.g., retrieval, placement). We follow the standard train/validation-seen/validation-unseen split, where test trajectories feature novel object-location configurations. WebShop [25] (1,624 train, 200 test) simulates e-commerce search over 12,087 products, requiring agents to follow natural-language instructions via multi-step navigation and filtering. TravelPlanner [21] (1,000 train, 180 validation, 45 test) involves multi-day itinerary planning under hard constraints (budget, dates) and soft preferences (cuisine, attractions). Evaluation uses Common Sense (CS) and Hard

Constraint (HC) scores. InterCodeSQL [24] benchmarks interactive text-to-SQL generation over diverse schemas, requiring correct handling of schema relationships and varying query difficulty.

4.1 Experimental Results and Analysis.

Table 1 compares MACLA against state-of-the-art baselines across all benchmarks. We organize baselines into three paradigms: *prompt-based* methods using in-context learning, *outcome refinement* approaches optimizing trajectory-level rewards, and *process refinement* methods refining step-level generation. MACLA achieves the highest average performance (78.1%) while using a 7B parameter model, demonstrating that domain-agnostic procedural memory with Bayesian selection and contrastive refinement enables competitive performance without task-specific engineering.

In Table 1, MACLA achieves state-of-the-art results on TravelPlanner (83.3 CS) and ALFWorld-Unseen (90.3%), outperforming methods that rely on models 10× larger. Its strong performance across all benchmarks demonstrates cross-domain generalization, while the positive generalization gap on ALFWorld (+3.1 points for unseen vs. seen) indicates robust compositional transfer rather than memorization.

4.2 Ablation Study.

To understand the contribution of each component in MACLA, we conduct an ablation study by systematically removing key modules. Table 2 reports results on ALFWorld (seen and unseen splits), evaluating: (1) Bayesian procedural selection (Section 2), (2) contrastive learning from failed trajectories (Section 2), (3) meta-procedural composition (Section 2), and (4) ontological semantic grounding (Section 2). Removing Bayesian selection leads to the largest degradation (−7.7 seen, −9.1 unseen), highlighting its role in effective exploration. Meta-procedural composition is essential for compositional generalization, with a sharp drop on unseen tasks (−11.9). Contrastive learning and ontological clustering provide smaller but consistent improvements (−3.5/−4.6 and −4.3/−6.2 respectively). Overall, all four components contribute synergistically to MACLA's robustness.

4.3 Computational and Memory Efficiency Analysis.

MACLA's efficiency comes from three design choices: (1) the frozen LLM eliminates gradient updates, (2) external memory construction is trivially parallelizable, and (3) learned procedures amortize LLM costs across episodes. Table 3 summarizes training costs.

MACLA builds memory in 56s (2,800× faster than IPR [22]) by extracting reusable procedures with a frozen LLM, instead of iterative parameter updates. For new tasks, IPR post-trains for 44.8 GPU-hrs, whereas MACLA ingests new trajectories in seconds. Memory construction scales nearly linearly with resources.

4.4 Memory Capacity and Performance Saturation.

Figure 2 reveals logarithmic performance growth across three capacity regimes: **(1) Undercapacity (25–50):** Sharp degradation (64.1% unseen at 25) due to insufficient task coverage, forcing frequent zero-shot fallback. Low posterior (0.61) indicates pruning removes procedures before adequate validation. **(2) Optimal (100–200):** Rapid improvement (85.6%→90.3% unseen), capturing core reusable procedures. The system extracts 187 unique procedures from 2,851 training trajectories (15:1 compression), leaving 13 of 200 slots unused—indicating automatic discovery of task-space boundaries. **(3) Overcapacity (300):** Performance declines (-0.2% unseen) despite more slots, as redundant variants introduce retrieval noise. The posterior plateau at 0.79 confirms saturation. This bounded growth (3.6 MB footprint) contrasts with neural approaches requiring unbounded parameter expansion, demonstrating ALFWorld's task space has finite complexity discoverable through procedural abstraction.

4.5 Bayesian Posterior Evolution and Convergence.

Figure 3 demonstrates uncertainty-aware learning through Bayesian posterior evolution. Panel (a) shows diverging α trajectories reflecting the explore-exploit tradeoff: general-purpose procedures (Navigate) accumulate evidence fastest through frequent invocation, while specialized procedures (Heat/Cool) converge slower but maintain high posteriors when applicable. Panel (b) reveals all top procedures stabilize above the 0.75 reliability threshold within 50 test episodes, with posterior variance decreasing as evidence accumulates:

$$\mathrm{Var}[\rho] = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)} \xrightarrow{\alpha+\beta\to\infty} 0 \tag{9}$$

By episode 50, $\alpha+\beta>30$ for all procedures, yielding standard deviations < 0.05—demonstrating principled uncertainty quantification. This self-reinforcing cycle ensures memory quality: poor procedures accumulate failures (high β), receive low utility scores, and are pruned before reaching high evidence totals.

4.6 Memory Pruning Characteristics.

Figure 4 validates MACLA's self-regulating pruning mechanism. Panel (a) shows clear distributional separation: 73% of pruned procedures have success rates below 0.5 (primarily spurious extractions from failed exploration trajectories), while 81% of retained procedures exceed 0.7. The utility-based criterion effectively discriminates signal from noise:

$$U(p) = 0.5\cdot\frac{\alpha}{\alpha+\beta} + 0.3\cdot\min\left(1, \frac{\text{count}}{10}\right) + 0.2\cdot\left(1 - \frac{\text{age}}{\text{max\_age}}\right) \tag{10}$$

Panel (b) reveals 68% of pruned procedures are both young (<40 trajectories old) and rarely used (<5 invocations)—the system identifies unpromising candidates early rather than wasting execution budget. Critically, the top-right quadrant is empty: *no high-quality procedures* (>0.7 success, >10 uses) are pruned, confirming conservative retention. This automatic quality control explains why performance plateaus at 187 procedures (mean posterior 0.79) without manual curation.

4.7 Task-Specific Memory Effectiveness.

Figure 5 explains SQL underperformance through three metrics. **Low reuse (51%):** SQL queries are schema-specific, e.g., `customers.age` does not apply to `employees.experience`. ALFWorld generalizes via placeholders (`<object>`), but SQL column names vary unpredictably. **Low reliability (64%):** Schema mismatches, join complexity, and edge cases accumulate failures (β counts), suppressing posteriors. **Minimal composition (18%):** SQL queries are atomic (2-3 actions), too short for meta-procedures. ALFWorld tasks naturally decompose into multi-step sub-procedures. MACLA excels when tasks have: (1) reusable actions, (2) hierarchical structure, and (3) consistent semantics — SQL violates all three.

4 CONCLUSION

We presented MACLA, a framework that decouples reasoning from learning by maintaining a frozen LLM and performing all adaptation

Table 1: Performance comparison across four agent benchmarks. Baseline results are from [22] and [4]. All metrics report average reward or quality score (0–100 scale, higher is better). Best results per column in bold.

Method	WebShop	InterCodeSQL	TravelPlanner	ALFWorld		Avg.
				Seen	*Unseen*	
Prompt-based Methods						
GPT-4 [1]	63.2	38.5	71.9	42.9	38.1	50.9
GPT-3.5-Turbo [10]	62.4	37.8	–	7.9	10.5	29.7
Llama-2-7B [18]	17.9	4.0	–	0.0	0.0	5.5
Outcome Refinement Methods						
Llama-2-7B + SFT [2]	60.2	54.9	–	60.0	67.2	60.6
Llama-2-7B + RFT-PPO [13]	64.2	52.4	–	22.1	29.1	42.0
Llama-2-7B + RFT-CR [30]	63.6	56.3	–	62.9	66.4	62.3
Llama-2-7B + ETO [17]	67.4	57.2	–	68.6	72.4	66.4
Process Refinement Methods						
Llama-2-7B + Step-PPO [22]	64.0	60.2	–	65.7	69.4	64.8
Llama-2-7B + IPR [22]	**71.3**	**61.3**	–	70.3	74.7	69.4
Claude-3.5-Sonnet† [4]	–	–	65.5	82.5	74.7	74.2
Qwen2.5-72B† [4]	–	–	63.8	85.7	77.2	75.6
Llama-2-7B + MACLA	70.2	59.3	**83.3**	**87.2**	**90.3**	**78.1**

†Substantially larger models (Claude-3.5: proprietary, Qwen2.5: 72B vs. 7B parameters). TravelPlanner reports Commonsense (CS) score; other benchmarks report task completion reward.

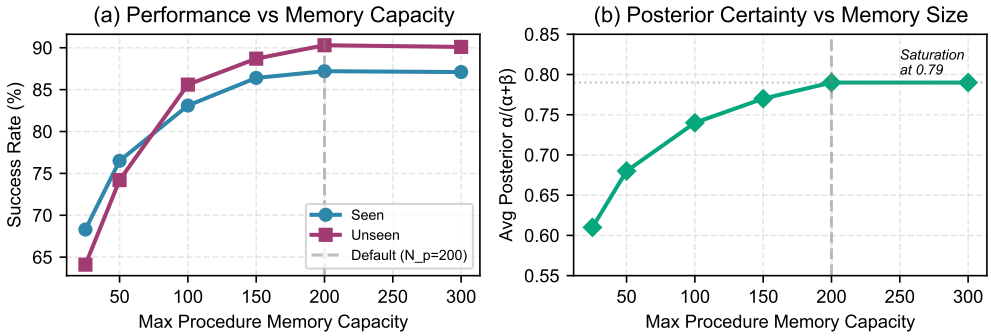

Figure 2: Ablation study varying maximum procedural memory capacity. (a) Success rate on ALFWorld seen/unseen splits saturates beyond 150 procedures, with diminishing returns from 150→200 (+1.6% unseen) and slight decline at 300 (-0.2%). (b) Average Bayesian posterior $\frac{\alpha}{\alpha+\beta}$ plateaus at 0.79, showing extra capacity adds redundancy rather than quality.

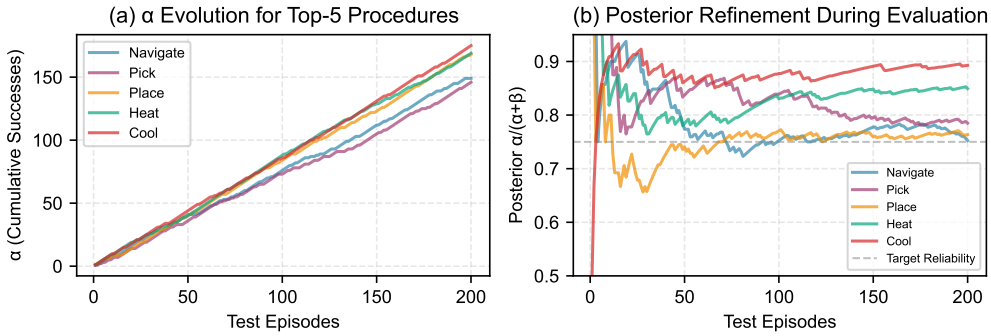

Figure 3: Bayesian learning dynamics for top-5 procedures during 200 test episodes. (a) Cumulative success count α grows at different rates: Navigate (blue) reaches 150+ invocations, while task-specific procedures (Heat/Cool, green/red) accumulate evidence more slowly due to limited applicability. (b) Posterior success rates $\frac{\alpha}{\alpha+\beta}$ converge above 0.75 within 50 episodes, with variance decreasing as $O(1/(\alpha+\beta))$.

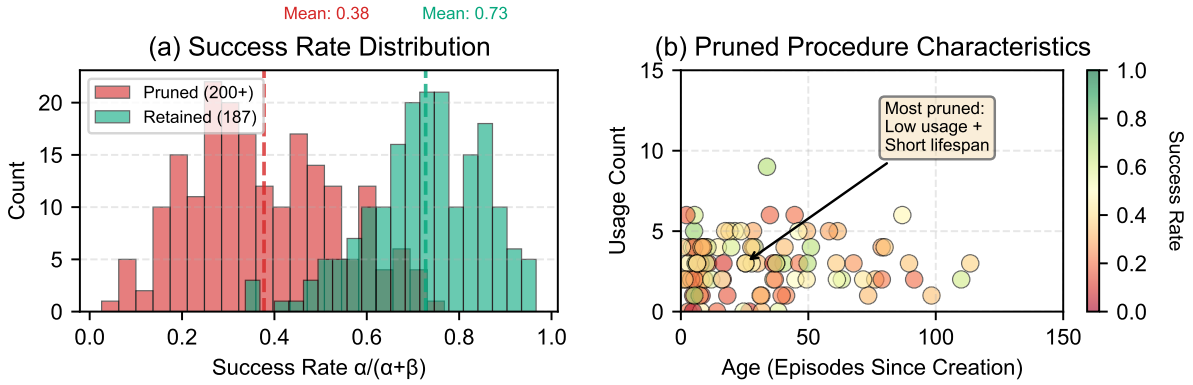

Figure 4: Analysis of 200+ pruned procedures during ALFWorld training. (a) Bimodal success rate distribution: pruned procedures (red, mean 0.42) separate cleanly from retained procedures (green, mean 0.79), validating utility-based retention. (b) Scatter plot shows pruned procedures cluster in bottom-left (young + rarely used), with no high-quality procedures (>0.7 success, >10 uses) pruned.

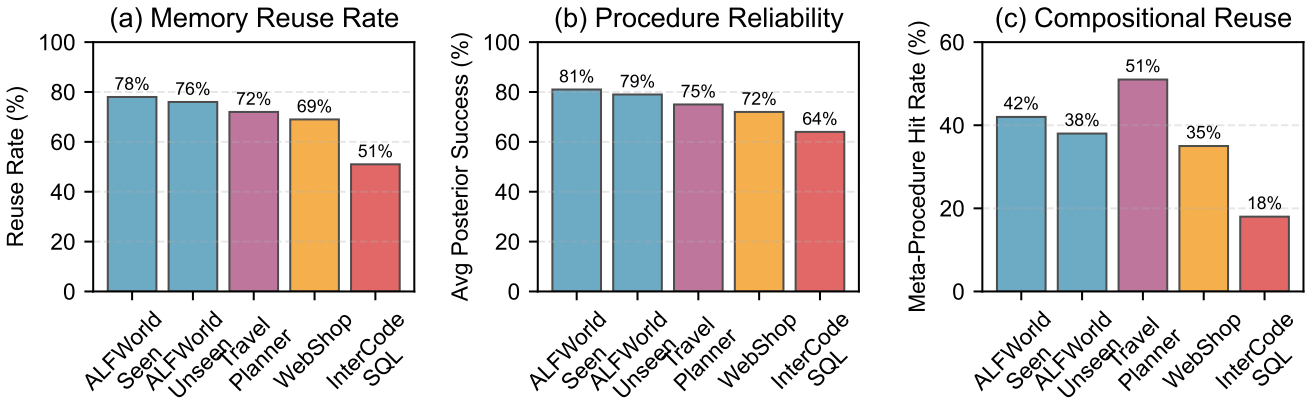

Figure 5: Cross-domain analysis. (a) Memory reuse: 51% (SQL) to 78% (ALFWorld). (b) Procedure reliability: 64% (SQL) to 81% (ALFWorld). (c) Meta-procedure usage: 18% (SQL) to 51% (TravelPlanner).

Table 3: Efficiency comparison. MACLA avoids iterative training, yielding 99.96% less training compute while maintaining competitive performance.

Method	Training (GPU-hrs)	WebShop	ALFWorld Unseen
IPR [22]	44.8	71.3	74.7
SFT [2]	8.0	60.2	67.2
ETO [17]	20.0	67.4	72.4
MACLA	**0.016**	**70.2**	**90.3**
Speedup vs IPR	**2,800×**	–	**+15.6 pts**

Training cost: IPR = 5.6h on 8×A100 (44.8 GPU-hrs); MACLA = 56s on 1×RTX 3090 (0.016 GPU-hrs), representing a 2,800× reduction. MACLA's frozen-LLM architecture eliminates iterative parameter training while achieving superior generalization on unseen tasks (+15.6 points on ALFWorld-Unseen vs IPR).

Table 2: Ablation study on ALFWorld with Llama-2-7B backbone. Each component is removed in turn to assess its contribution. Results are success rates (0–100).

Config.	Bayes.	Contr.	Meta	Ontol.	Seen	Unseen
Full MACLA	✓	✓	✓	✓	**87.1**	**90.3**
w/o Bayesian	✗	✓	✓	✓	79.4	81.2
w/o Contrast.	✓	✗	✓	✓	83.6	85.7
w/o Meta	✓	✓	✗	✓	81.2	78.4
w/o Ontology	✓	✓	✓	✗	82.8	84.1

Bayes.: probabilistic selection (Sec. 2); Contr.: success/failure refinement (Sec. 2); Meta: hierarchical composition (Sec. 2); Ontol.: semantic clustering (Sec. 2).

in an external hierarchical procedural memory through Bayesian selection, contrastive refinement, and meta-procedural composition. MACLA achieves 78.1% average performance across four benchmarks using only a 7B model, with state-of-the-art results on ALFWorld (87.2% seen; 90.3% unseen) and TravelPlanner (83.3%). The system compresses 2,851 ALFWorld training trajectories into 187 reusable procedures through semantic abstraction and duplicate detection, demonstrating efficient knowledge distillation.

REFERENCES

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. 2023. Fireact: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915* (2023).

[3] Ming Chen, Yifei Li, Yao Yang, et al. 2024. AutoManual: Constructing Instruction Manuals by LLM Agents via Interactive Environmental Learning. *arXiv preprint arXiv:2405.16247* (2024).

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

[5] Mingyuan Hu, Tianhong Chen, Qian Chen, et al. 2024. HiAgent: Hierarchical Working Memory Management for Long-Horizon Agent Tasks. In *CVPR Workshops*.

[6] Zhen Li, Shijie Song, Chen Xi, et al. 2025. MEMOS: A Memory OS for AI Systems. In *Proceedings of the Web Conference (Companion)*.

[7] Xuechen Liang, Meiling Tao, Yinghui Xia, et al. 2025. SAGE: Self-evolving Agents with Reflective and Memory-augmented Abilities. *Neurocomputing* 647 (2025), 130470.

[8] Bowen Liu, Xuan Li, Jing Zhang, et al. 2025. Advances and Challenges in Foundation Agents: From Brain-inspired Intelligence to Evolutionary, Collaborative, and Safe Systems. (2025). arXiv:2504.01990 [cs.AI] https://arxiv.org/abs/2504.01990

[9] Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. 2025. AgentBoard: an analytical evaluation board of multi-turn LLM agents. In *Proceedings of the 38th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS'24)*. Curran Associates Inc., Red Hook, NY, USA, Article 2365, 38 pages.

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

[11] Charles Packer, Vivian Fang, Shishir_G Patil, Kevin Lin, Sarah Wooders, and Joseph_E Gonzalez. 2023. MemGPT: Towards LLMs as Operating Systems. (2023).

[12] Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084* (2019).

[13] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[14] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. *arXiv preprint arXiv:2303.11366* (2023). https://arxiv.org/abs/2303.11366

[15] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems* 36 (2023), 8634–8652.

[16] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *ICLR*. https://alfworld.github.io/

[17] Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. 2024. Trial and Error: Exploration-Based Trajectory Optimization of LLM Agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 7584–7600. https://doi.org/10.18653/v1/2024.acl-long.409

[18] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[19] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An Open-Ended Embodied Agent with Large Language Models. *arXiv preprint arXiv:2305.16291* (2023). https://arxiv.org/abs/2305.16291

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

[21] Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. 2024. Travelplanner: A benchmark for real-world planning with language agents. *arXiv preprint arXiv:2402.01622* (2024).

[22] Weimin Xiong, Yifan Song, Xiutian Zhao, Wenhao Wu, Xun Wang, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. 2024. Watch Every Step! LLM Agent Learning via Iterative Step-level Process Refinement. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, Miami, Florida, USA, 1556–1572. https://doi.org/10.18653/v1/2024.emnlp-main.93

[23] Wujiang Xu, Kai Mei, Hang Gao, Juntao Tan, Zujie Liang, and Yongfeng Zhang. 2025. A-MEM: Agentic Memory for LLM Agents. (2025). arXiv:2502.12110 [cs.CL] https://arxiv.org/abs/2502.12110

[24] John Yang, Akshara Prabhakar, Karthik Narasimhan, and Shunyu Yao. 2023. InterCode: standardizing and benchmarking interactive coding with execution feedback. In *Proceedings of the 37th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '23)*. Curran Associates Inc., Red Hook, NY, USA, Article 1035, 29 pages.

[25] Shunyu Yao et al. 2022. WebShop: Towards Scalable Real-World Web Interaction with LLM Agents. In *NeurIPS Datasets and Benchmarks*. https://arxiv.org/abs/2207.09486

[26] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

[27] Da Yin, Faeze Brahman, Abhilasha Ravichander, Khyathi Chandu, Kai-Wei Chang, Yejin Choi, and Bill Yuchen Lin. 2024. Agent Lumos: Unified and Modular Training for Open-Source Language Agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 12380–12403. https://doi.org/10.18653/v1/2024.acl-long.670

[28] Hang Yu, Tianyu Chen, Jiale Feng, et al. 2025. MemAgent: Reshaping Long-Context LLM with Multi-Conv RL-based Memory Agent. *arXiv preprint arXiv:2507.02259* (2025).

[29] Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2024. AgentTuning: Enabling Generalized Agent Abilities for LLMs. In *Findings of the Association for Computational Linguistics: ACL 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 3053–3077. https://doi.org/10.18653/v1/2024.findings-acl.181

[30] Yifan Zhang, Jingqin Yang, Yang Yuan, and Andrew Chi-Chih Yao. 2023. Cumulative reasoning with large language models. *arXiv preprint arXiv:2308.04371* (2023).

[31] Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2025. A Survey on the Memory Mechanism of Large Language Model-based Agents. *ACM Trans. Inf. Syst.* 43, 6, Article 155 (Sept. 2025), 47 pages. https://doi.org/10.1145/3748302

[32] Wei Zhong, Liang Guo, Qian Gao, Haotian Ye, and Yuxin Wang. 2024. MemoryBank: Enhancing Large Language Models with Long-term Memory. In *AAAI*, Vol. 38. 19724–19731.