# EVOKE: ELICITING WORLD KNOWLEDGE IN AGENTS FOR TRANSFERABLE DECISION-MAKING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Large language models (LLMs) are increasingly deployed as agents for multi-step decision-making, yet transfer poorly to unseen environments. World-model methods address this by training agents to predict future observations, at the cost of additional training and errors that compound when predictions are used for planning. However, for LLM agents operating in digital environments, much of this world knowledge is already internalized during pretraining, which shifts the problem from acquiring it to eliciting it. We argue that typical post-training provides little pressure for such elicitation, since supervision under a single goal at each visited state inadvertently drives policies to rely on superficial contextual habits. We introduce **EVOKE**, a post-training method that supplies this pressure through goal diversity at fixed states. Motivated by theory showing that an agent competent across diverse goals must encode a world model recoverable from its action preferences, EVOKE holds the environment state and interaction history fixed and ranks the same candidate actions under alternative goals, forcing action preferences to change, so that a policy relying on contextual habits or single-goal correlations cannot order them correctly. This implicitly elicits the policy's pretrained world knowledge to inform decisions. We evaluate EVOKE across diverse tasks in three backbones, demonstrating improved task performance, unseen environment generalization, and data efficiency. We further conduct controlled analyses to better understand what drives these gains. These findings offer a new perspective on eliciting internalized world knowledge for transferable action through direct decision supervision.

## 1 INTRODUCTION

Large language models (LLMs) are increasingly post-trained as agents to make multi-step decisions in interactive environments (Yao et al., 2023; Liu et al., 2026b; 2025). These agents often perform well where they were trained, yet their performance drops considerably in environments they have not seen (Zhang et al., 2026b). What they learn stays tied to the tasks and environments encountered during training and rarely transfers to new ones.

One way to improve transferability is to let the agent anticipate the consequences of an action before choosing it. The effect of an action often follows rules that hold across environments (e.g., placing an order ends a purchase on any shopping site), so decisions that account for these effects are more likely to carry over. World-model methods pursue this idea by training agents to predict the consequences of their actions, in the form of future observations in Figure 1a (Zhang et al., 2026a; Lu et al., 2026a; Wang et al., 2026d; Liu et al., 2026a). However, learning these consequences through prediction comes with two costs. (1) It adds a prediction objective that makes training more expensive, and much of what the agent is asked to predict is irrelevant to the decision at hand (Grimm et al., 2020). (2) When predicted futures are used for planning, their errors compound over successive steps (Zhou et al., 2025; Liu et al., 2026a).

Yet do LLM agents need to learn these consequences through prediction? Such learning is important in physical environments, where dynamics such as contact and motion are hard to capture without grounding (Assran et al., 2025). LLM agents, however, mostly act in digital environments such as websites and search engines, where the effects of actions fall largely within the world knowledge that LLMs acquire through pretraining (Brown et al., 2020). Indeed, recent work finds that pretrained



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

models, given only a few examples, can already predict how text-based environments respond to many actions (Li et al., 2026a). This suggests that LLM agents may not need to rely entirely on a world model to predict, since they already know much of the world knowledge from pretraining.

*This shifts the problem from acquiring this knowledge to eliciting it, ensuring the model actually leverages what it already knows to make decisions.* However, typical training paradigms inadvertently limit this elicitation by driving the policy toward surface-level behavior matching. When a visited state is tied to a training goal, the policy can easily fit the training signal just by memorizing the habitual next action for that context (Figure 1b). While these superficial habits work within the training distribution, they quickly break down in new environments, reflecting how single-goal supervision provides little pressure to use pretrained knowledge at decision time.

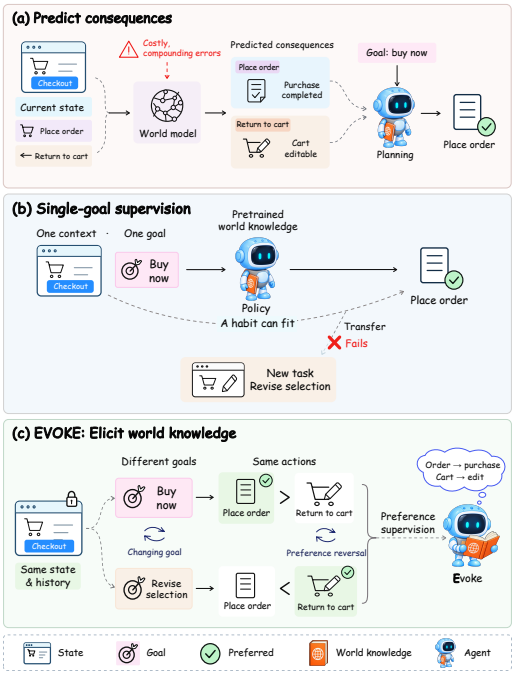



Figure 1: **From prediction to preference.** (a) World-model approaches predict action consequences before choosing an action. (b) Under single-goal supervision, a policy can fit habits that fail to transfer. (c) EVOKE instead supervises action preferences at a fixed state and history under different goals.

We introduce EVOKE, a post-training method that supplies this pressure through goal diversity at fixed states. Richens et al. (2025) motivate this choice, showing that an agent competent across a sufficiently rich set of multi-step goals must contain a world model recoverable from its goal-conditioned behavior. Targeting this competence, EVOKE holds the environment state and interaction history fixed while introducing alternative goals (Figure 1c), thereby evaluating the exact same set of candidate actions against different objectives. Crucially, this isolates the goal's effect and often forces action preferences to flip. Whenever the preferred actions differ across these alternative goals, a policy that relies on contextual habits from standard supervision or on correlations learned under a single fixed goal cannot rank them correctly. This encourages the policy to elicit its pretrained world knowledge of action consequences into its decision-making.

In practice, EVOKE collects states from the policy's own rollouts, poses alternative goals at these states, and trains the policy to rank candidate actions under each goal. This process is repeated over rounds in the manner of DAgger (Ross et al., 2011). Across household tasks, web navigation, and search-based QA, EVOKE demonstrates transferability, yielding improved performance, unseen environment generalization and data efficiency. To isolate the mechanisms driving these gains, we conduct extensive controlled analyses. Ultimately, we hope these findings offer a new perspective on eliciting internalized world knowledge for transferable action. Taken together, our work makes the following contributions:

1. We revisit how LLM agents can benefit from world knowledge, framing the problem as eliciting knowledge already present in pretrained models rather than acquiring it through additional prediction objectives.
2. We introduce EVOKE, which explores goal diversity at fixed states as a source of decision-level supervision. By ranking the same candidate actions under different goals, it encourages the policy to bring the knowledge of action consequences acquired in pretraining into its decisions.
3. We evaluate EVOKE on text-based household tasks, web navigation, and search-based QA across three backbones, observing gains in task performance, generalization to unseen environments, and data efficiency. We further conduct controlled analyses to better understand what drives these gains.

## 2 Related Work

**World models for LLM agents.** A line of work equips LLM agents with knowledge of action consequences by learning to predict them. Some methods build an explicit model for lookahead: WALL-E combines pretrained knowledge with symbolic rules, ITP learns textual dynamics for adaptive lookahead, and MemWM adds memory (Zhou et al., 2025; Liu et al., 2026a; Wang et al., 2026c). Others train the policy with an auxiliary prediction signal: IWM learns next-state prediction before imitation learning, PaW jointly optimizes observation prediction and policy learning, EnvRL adds forward and inverse dynamics, and Role-Agent and RWML derive rewards from predicted–observed state agreement (Zhang et al., 2026a; Lu et al., 2026a; Wang et al., 2026d;b; Yu et al., 2026). In embodied settings, V-JEPA 2's action-conditioned extension predicts latent representations for robotic planning (Assran et al., 2025). In a complementary knowledge-based approach, WKM trains a separate world knowledge model to guide planning with task-level and state-level knowledge (Qiao et al., 2024). From Word to World finds that pretrained LLMs, given a few demonstrations, already predict next states well in structured text-based environments (Li et al., 2026a). EVOKE builds on this observation and elicits the consequence knowledge already present in the policy, without a prediction objective or a separate model.

**Policy optimization for LLM agents.** LLM agents are commonly improved from their own interaction. Reinforcement learning optimizes the policy with task rewards, using group-relative estimates at the trajectory or step level (Shao et al., 2024; Feng et al., 2026), and distillation-based methods add denser guidance from teacher signals (Lu et al., 2026b). ETO learns from preferences between successful and failed trajectories (Song et al., 2024), and DAgger aggregates labels at learner-visited states (Ross et al., 2011). These methods mainly differ in how the training signal is computed. EVOKE focuses on a complementary aspect: the goals under which each visited state is supervised. When a state is supervised under a single goal, the policy can fit the signal through contextual habits (Section 1); EVOKE supplies goal diversity at fixed states to add the pressure that single-goal supervision leaves out.

**Learning across goals.** Goal-conditioned reinforcement learning shares experience across goals. Universal value functions generalize over goals (Schaul et al., 2015), successor features separate environment dynamics from task-specific rewards to enable transfer (Barreto et al., 2017), and Hindsight Experience Replay relabels trajectories with achieved goals so that failures still provide learning signals (Andrychowicz et al., 2017). Hindsight Supervised Learning brings relabeling to language-agent trajectories by constructing demonstrations for achieved goals (Li et al., 2026b). Richens et al. (2025) show that an agent competent across a sufficiently rich set of goals must contain a world model recoverable from its goal-conditioned behavior. EVOKE applies goal diversity to a single decision: it holds the state and history fixed, assesses the same candidate actions under alternative goals, and trains on the resulting action preferences rather than on relabeled trajectories alone.

## 3 EVOKE

EVOKE trains a language-model policy $\pi_\theta$ that selects the next action from the input $x_g = (g, h, \mathcal{A}(s))$, consisting of a goal $g$, the visible interaction history $h$, and the actions $\mathcal{A}(s)$ available in the current state $s$. Training repeats a loop of state collection, goal intervention, action assessment, and contrastive ranking (Figure 2); at deployment, EVOKE is a standard policy.

### 3.1 Goal Interventions at Fixed States

Changing the goal does not change how the environment responds to an action, but it changes which responses matter. We write this as

$$T_g(s' \mid s, a) = T(s' \mid s, a), \qquad J_g(s, h, a) = \sum_{s'} T(s' \mid s, a)\, u_g(h, s, a, s'), \tag{1}$$

where $T$ is the transition shared by all goals and $u_g$ scores how useful a consequence is for goal $g$. Equation 1 separates what an action does, which is common across goals, from what it is worth, which is specific to each goal.

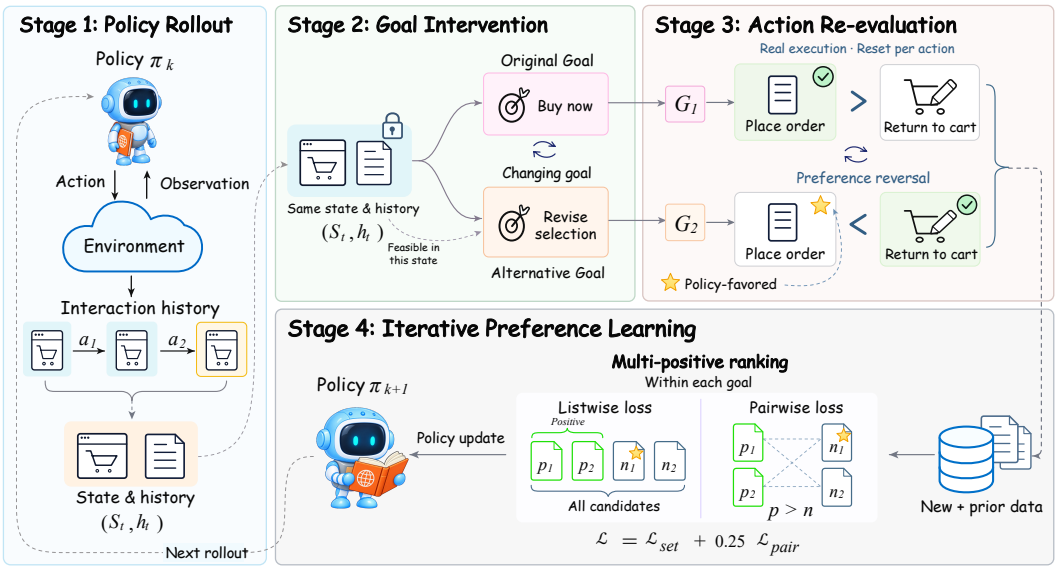



Figure 2: **The EVOKE training loop.** At collected states, actions are executed and assessed under alternative goals with the state, history, and available actions fixed. The policy learns by contrastive ranking on aggregated preferences; actions can switch between positive and competing across goals.

Single-goal supervision leaves this structure unused: when each visited context $(s, h)$ is labeled under its original goal only, the policy can fit the labels by associating familiar contexts with habitual next actions. A goal intervention holds $s$, $h$, and $\mathcal{A}(s)$ fixed and replaces $g$ with an alternative goal $g'$. In Figure 2, the agent is at a checkout page: under the goal of buying now, it should place the order, whereas under the goal of revising the selection, it should return to the cart. Whenever the preferred actions differ, no mapping from the context alone ranks the candidates correctly under both goals. Ranking correctly then requires conditioning on the goal, most directly by relating what each action does, which is shared across goals, to what the current goal requires. Goal interventions thus pressure the policy to use its knowledge of action consequences, rather than leaving it merely available.

### 3.2 Constructing Goal-Conditioned Action Preferences

**Collecting decision states.** We initialize the policy, denoted $\pi_{\text{boot}}$, by supervised fine-tuning on successful demonstrations decomposed into per-step input–action pairs. The policy then interacts with training environments, and we record the goal and action–observation history at each step, so that the same state can be revisited. We sample decision states where the policy makes progress as well as where it takes detours, repeats actions, or heads toward failure, so supervision concentrates on situations the policy actually encounters.

**Goal interventions.** For each collected state, we keep the original goal and add alternative goals achievable from the same state. An LLM annotator proposes candidates; each is instantiated in the environment together with its success condition and retained if it is compatible with the scene and not yet satisfied. The state, visible history, and available actions stay identical across goals.

**Grounded action assessment.** Candidate actions are drawn from the shared action set $\mathcal{A}(s)$ (Appendix A), and each is executed from the same state, with a short follow-up interaction when one step does not reveal its effect. Given the goal, the history, and the executed outcomes, the annotator labels actions that advance the goal as positives $P$ and the others as competitors $N$, leaving actions with insufficient evidence unlabeled. Because the annotator judges observed outcomes rather than predicting them, the labels reflect how the environment actually responds.

**Policy-favored hard negatives.** We fill each competitor set first with actions that the policy favors but the annotator judges worse: the policy's greedy action when it is not a positive, followed by other high-probability non-positive actions, and then actions of other types for coverage. These are the

mistakes the policy is most likely to make, and thus the most informative negatives (Robinson et al., 2020). For a goal of putting a clean cup on the table while holding a dirty cup, going to the sink is a positive, whereas placing the dirty cup directly on the table, which the policy readily proposes, is a hard negative.

### 3.3 Learning through Contrastive Ranking

We score an action $a$ by the mean token log-probability of its completion $c(a)$,

$$z_\theta(a \mid x_g) = \frac{1}{|c(a)|} \sum_{t=1}^{|c(a)|} \log \pi_\theta\big(c_t(a) \mid x_g, c_{<t}(a)\big), \tag{2}$$

and write $r_a = z_\theta(a \mid x_g)/\tau$ with temperature $\tau$. For a group with positives $P$ and competitors $N$, the loss is $\mathcal{L} = \mathcal{L}_{\text{set}} + \lambda \mathcal{L}_{\text{pair}}$, where $\lambda$ weights the pairwise term and

$$\mathcal{L}_{\text{set}} = -\log \frac{\sum_{p \in P} e^{r_p}}{\sum_{a \in P \cup N} e^{r_a}}, \qquad \mathcal{L}_{\text{pair}} = -\frac{1}{|P||N|} \sum_{p \in P} \sum_{n \in N} \log \sigma(r_p - r_n), \tag{3}$$

and $\sigma$ is the logistic function. The listwise term $\mathcal{L}_{\text{set}}$ raises the mass of the positive set as a whole, so several actions can be correct in one context; the pairwise term separates every positive from every competitor. Supervised fine-tuning on positives raises preferred actions but does not explicitly push down the policy's own tempting mistakes, whereas both ranking terms push these hard negatives down. The loss is computed within each goal; the relation across goals comes from contexts that share a state but differ in their positives. Executed outcomes and annotator judgments build labels only and never enter the policy input. We update LoRA adapters (Hu et al., 2021) with $\tau = 1$ and $\lambda = 0.25$; further details are in Appendix A.

### 3.4 Iterative Aggregation

An updated policy reaches new states and makes new mistakes. Each round therefore repeats state collection, goal intervention, action assessment, and hard-negative selection with the current policy, and trains on the aggregate of all rounds, following dataset aggregation in imitation learning (Ross et al., 2011) (per-round data statistics in Appendix Table 6). This keeps the hard negatives aligned with the errors of the current policy. At deployment, the policy acts from the goal, history, and available actions alone, with no world-model module, no inference-time planning, and no annotator.

## 4 Experiments

We evaluate whether goal-conditioned decision supervision improves complete task execution in unseen environments, then examine which training choices account for the gains. The main comparison covers three model backbones. Section 5 separately studies supervision design and training budget using a shared 3B initialization.

### 4.1 Setup and Evaluation Protocol

**Benchmarks and metrics.** All scores are percentages, and EVOKE results are averaged over training seeds. **ALFWorld** reports success on Pick, Look, Clean, Heat, Cool, and Pick2 (Shridhar et al., 2020); Avg is their unweighted mean. EVOKE uses 140 Seen games in the main table and 134 Unseen games separately, with frozen parameters and a 50-action limit. **Search-based QA** reports answer accuracy on NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), PopQA (Mallen et al., 2023), HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023); Avg equally weights these seven datasets. **WebShop** (Yao et al., 2022) reports mean task score (Score) and task success rate (Succ.).

**Baselines.** We compare with the three groups in Tables 1–3: prompting (Vanilla, ReAct (Yao et al., 2023), Skill-Prompt); policy optimization and distillation (GRPO (Shao et al., 2024), Skill-GRPO, OPSD (Zhao et al., 2026), GRPO+OPSD, Skill-SD (Wang et al., 2026a), RLSD (Yang et al., 2026a), SDAR (Lu et al., 2026b), OPID (Yang et al., 2026b), PCSD (Lv et al., 2026), GRSD (Zheng et al.,

Table 1: ALFWorld and WebShop (%).

| Method | ALFWorld Pick | Look | Clean | Heat | Cool | Pick2 | Avg | WebShop Score | Succ. |
|---|---|---|---|---|---|---|---|---|---|
| *Qwen2.5-3B-Instruct* | | | | | | | | | |
| *Prompting* | | | | | | | | | |
| Vanilla | 44.4 | 11.1 | 6.2 | 15.4 | 28.6 | 12.5 | 19.7 | 6.7 | 0.8 |
| Skill-Prompt* | 51.7 | 66.7 | 48.4 | 0.0 | 4.3 | 10.0 | 30.2 | 0.2 | 0.8 |
| *Policy optimization and distillation* | | | | | | | | | |
| OPSD | 48.8 | 41.7 | 16.7 | 0.0 | 15.8 | 16.7 | 23.3 | 11.3 | 3.1 |
| GRPO | 91.2 | 62.5 | 96.2 | 61.9 | 65.0 | 47.4 | 70.7 | 79.8 | 63.3 |
| Skill-GRPO | 88.9 | 71.4 | 58.8 | 70.6 | 40.7 | 29.2 | 59.9 | 77.3 | 60.9 |
| Skill-GRPO* | 94.3 | 57.1 | **100.0** | 66.7 | 73.1 | 57.1 | 74.7 | 76.3 | 66.4 |
| GRPO+OPSD | **100.0** | **82.4** | 85.7 | 75.0 | 70.0 | 60.0 | 78.9 | 77.8 | 66.4 |
| Skill-SD | 88.2 | 50.0 | 96.2 | 52.4 | 65.0 | 57.9 | 68.3 | 75.9 | 64.0 |
| RLSD | 87.9 | 75.0 | 90.9 | 75.0 | 73.1 | 68.4 | 78.4 | 84.4 | 66.4 |
| SDAR | 97.1 | 62.5 | **100.0** | 61.9 | 75.0 | 84.2 | 80.1 | 85.0 | 68.0 |
| **EVOKE (Ours)** | 91.4 | 76.9 | 96.3 | **100.0** | **92.0** | **91.7** | **91.4** | **87.4** | **82.8** |
| *Qwen2.5-7B-Instruct* | | | | | | | | | |
| *Prompting* | | | | | | | | | |
| Vanilla | 36.1 | 22.2 | 3.1 | 0.0 | 0.0 | 0.0 | 10.2 | 5.9 | 1.6 |
| Skill-Prompt* | 51.7 | 50.0 | 32.3 | 5.3 | 4.3 | 0.0 | 23.9 | 1.7 | 0.8 |
| *Policy optimization and distillation* | | | | | | | | | |
| OPSD | 50.0 | 60.0 | 22.7 | 21.4 | 17.6 | 9.5 | 30.2 | 4.5 | 2.3 |
| GRPO | 91.2 | 87.5 | 96.2 | 81.0 | 65.0 | 57.9 | 79.8 | 80.9 | 72.6 |
| Skill-GRPO | 88.5 | 66.7 | 65.2 | 61.1 | 57.7 | 73.1 | 68.7 | 80.4 | 71.9 |
| Skill-GRPO* | **100.0** | 83.3 | 96.4 | 83.3 | 75.0 | 78.9 | 86.2 | 87.0 | 81.2 |
| GRPO+OPSD | 91.4 | 61.5 | **100.0** | 87.5 | 76.5 | 52.2 | 78.2 | 86.8 | 76.5 |
| Skill-SD | 93.9 | 93.8 | 90.9 | **100.0** | 69.2 | 68.4 | 86.0 | 86.1 | 76.5 |
| RLSD | **100.0** | 87.5 | 92.3 | 58.8 | 80.0 | 65.2 | 80.6 | 87.4 | 77.3 |
| SDAR | 94.7 | 75.0 | **100.0** | 86.7 | 68.2 | 78.9 | 83.9 | 89.4 | 82.8 |
| *World-model and dynamics methods* | | | | | | | | | |
| IWM | 90.6 | 42.9 | 85.2 | 88.2 | 84.2 | 76.9 | 78.0 | 69.5 | 56.2 |
| $ITP_R$ | 94.3 | 76.0 | 88.9 | 87.5 | 53.8 | 91.7 | 82.0 | - | 60.2 |
| MemWM | 70.8 | 88.9 | 45.2 | 47.8 | 71.4 | 17.7 | 57.0 | 46.9 | - |
| GRPO+PaW | 90.4 | 80.7 | 86.8 | 82.9 | 76.5 | 67.3 | 80.8 | 84.5 | 70.5 |
| GIGPO+PaW | 98.2 | 85.6 | 98.6 | 84.5 | 91.5 | 84.3 | 90.5 | 87.6 | 76.7 |
| EnvRL-GRPO | 92.6 | 77.5 | 92.9 | 79.7 | 77.4 | 71.1 | 81.9 | 81.8 | 68.6 |
| EnvRL-GiGPO | 98.3 | 92.2 | 98.9 | 93.8 | 93.3 | 94.6 | 95.2 | 88.4 | 76.3 |
| **EVOKE (Ours)** | 94.3 | **100.0** | 96.3 | 93.8 | **96.0** | **95.8** | **96.0** | **90.6** | **85.9** |
| *Qwen3-1.7B* | | | | | | | | | |
| *Prompting* | | | | | | | | | |
| Vanilla | 25.0 | 22.2 | 3.1 | 0.0 | 21.4 | 4.2 | 12.7 | 46.5 | 4.7 |
| Skill-Prompt* | 10.3 | 50.0 | 16.1 | 0.0 | 0.0 | 5.0 | 13.6 | 23.0 | 2.3 |
| *Policy optimization and distillation* | | | | | | | | | |
| OPSD | 26.3 | 33.3 | 9.1 | 0.0 | 4.5 | 5.3 | 13.1 | 47.4 | 9.3 |
| GRPO | 71.1 | 41.7 | 36.4 | 40.0 | 31.8 | 31.6 | 42.1 | 67.3 | 38.3 |
| Skill-GRPO | 27.6 | 54.5 | 22.7 | 27.3 | 0.0 | 19.2 | 25.2 | 73.4 | 46.1 |
| Skill-GRPO* | 31.4 | 42.9 | 51.9 | 8.3 | 11.5 | 7.1 | 25.5 | 80.4 | 50.0 |
| GRPO+OPSD | 38.2 | 50.0 | 30.8 | 28.6 | 30.0 | 21.1 | 33.1 | 70.7 | 38.3 |
| Skill-SD | 52.9 | 37.5 | 69.2 | 42.9 | 60.0 | 36.8 | 49.9 | 81.8 | 53.9 |
| RLSD | 50.0 | 37.5 | 61.5 | 19.0 | 50.0 | 21.1 | 39.9 | 74.0 | 50.8 |
| SDAR | 73.5 | 25.0 | 76.9 | 33.3 | 40.0 | 36.8 | 47.6 | 76.8 | 58.6 |
| **EVOKE (Ours)** | **94.3** | **84.6** | **96.3** | **87.5** | **92.0** | **91.7** | **91.1** | **88.4** | **80.5** |

Table 2: Search-based QA accuracy (%).

| Method | Search-based QA NQ | Triv | Pop | Hotp | 2Wk | MuS | Bam | Avg |
|---|---|---|---|---|---|---|---|---|
| *Qwen2.5-3B-Instruct* | | | | | | | | |
| *Prompting* | | | | | | | | |
| Vanilla | 24.6 | 48.1 | 31.0 | 26.3 | 25.3 | 7.2 | 59.7 | 31.7 |
| Skill-Prompt* | 23.7 | 46.2 | 30.6 | 24.4 | 22.1 | 7.5 | 12.5 | 23.9 |
| *Policy optimization and distillation* | | | | | | | | |
| OPSD | 0.1 | 0.1 | 0.1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| GRPO | 39.3 | 60.6 | 41.1 | 37.4 | 34.6 | 15.4 | 26.4 | 36.4 |
| Skill-GRPO | 43.5 | 58.8 | 43.0 | 36.8 | 32.2 | 11.7 | 12.5 | 34.1 |
| Skill-GRPO* | 44.3 | 59.6 | 44.3 | 39.0 | 36.1 | 14.5 | 14.9 | 36.1 |
| GRPO+OPSD | 44.9 | 61.2 | 45.2 | 40.4 | 38.5 | 16.0 | 66.1 | 44.6 |
| Skill-SD | 44.4 | 60.4 | 44.0 | 39.5 | 40.4 | 15.4 | 64.9 | 44.1 |
| RLSD | 41.5 | 58.6 | 42.3 | 40.4 | 40.2 | **16.8** | **66.9** | 43.8 |
| SDAR | 44.8 | 58.1 | 44.3 | 38.6 | 36.2 | 15.7 | 66.1 | 43.4 |
| *World-model and dynamics methods* | | | | | | | | |
| GRPO+PaW | 45.8 | 61.2 | **47.5** | 39.4 | 40.1 | 14.4 | 65.2 | 44.8 |
| GIGPO+PaW | **46.2** | **61.8** | 46.7 | 37.6 | 38.0 | 13.9 | 64.9 | 44.2 |
| **EVOKE (Ours)** | 46.1 | **61.8** | 45.2 | **41.5** | **41.0** | 15.3 | 66.1 | **45.3** |
| *Qwen2.5-7B-Instruct* | | | | | | | | |
| *Prompting* | | | | | | | | |
| Vanilla | 25.2 | 50.8 | 29.5 | 29.0 | 29.0 | 10.4 | 63.7 | 33.9 |
| Skill-Prompt* | 30.9 | 52.1 | 32.7 | 32.7 | 27.9 | 12.7 | 66.1 | 36.4 |
| *Policy optimization and distillation* | | | | | | | | |
| OPSD | 8.8 | 8.6 | 17.5 | 2.5 | 4.2 | 0.5 | 1.2 | 6.2 |
| GRPO | 45.1 | 63.7 | 44.0 | 43.6 | 43.2 | 16.8 | 37.6 | 42.0 |
| Skill-GRPO | 45.2 | 63.7 | 45.7 | 43.1 | 43.3 | 19.6 | 21.4 | 40.3 |
| Skill-GRPO* | 44.8 | 63.0 | 45.1 | 43.7 | 43.7 | 20.5 | 71.4 | 47.5 |
| GRPO+OPSD | 47.3 | 64.5 | 46.9 | 43.8 | 39.3 | 18.0 | 69.4 | 47.0 |
| Skill-SD | 47.1 | 64.5 | 47.8 | 44.2 | 42.1 | 20.2 | 69.0 | 47.8 |
| RLSD | 46.8 | 63.0 | 44.4 | **45.5** | 48.9 | **21.5** | **73.0** | 49.0 |
| SDAR | 46.3 | 63.5 | 48.2 | 43.8 | 48.4 | 19.6 | **73.0** | 49.0 |
| *World-model and dynamics methods* | | | | | | | | |
| GRPO+PaW | **48.9** | 64.9 | **48.5** | 44.9 | 45.1 | 18.9 | 70.1 | 48.8 |
| GIGPO+PaW | 46.5 | **66.0** | 47.2 | 42.2 | 42.8 | 18.6 | 69.5 | 47.5 |
| **EVOKE (Ours)** | 46.8 | 65.5 | 47.8 | 44.5 | **49.4** | 20.0 | 70.5 | **49.2** |
| *Qwen3-1.7B* | | | | | | | | |
| *Prompting* | | | | | | | | |
| Vanilla | 29.4 | 46.9 | 37.0 | 23.5 | 19.6 | 6.4 | 10.5 | 24.8 |
| Skill-Prompt* | 29.4 | 46.5 | 36.2 | 22.9 | 20.8 | 4.3 | 10.1 | 24.3 |
| *Policy optimization and distillation* | | | | | | | | |
| OPSD | 4.2 | 8.3 | 4.6 | 6.6 | 15.3 | 0.7 | 1.2 | 5.8 |
| GRPO | 40.0 | 58.9 | 43.5 | 35.4 | 30.3 | 12.0 | 65.7 | 40.8 |
| Skill-GRPO | 39.2 | 58.6 | 43.9 | 35.2 | 28.2 | 11.5 | 66.1 | 40.4 |
| Skill-GRPO* | 38.0 | 58.4 | 43.9 | 36.3 | 29.0 | 12.5 | **66.9** | 40.7 |
| GRPO+OPSD | 40.7 | 58.9 | 45.0 | 37.0 | 34.6 | 13.3 | 65.7 | 42.2 |
| Skill-SD | 39.1 | 57.5 | **45.4** | 34.8 | 34.1 | 10.7 | 64.1 | 40.8 |
| RLSD | 38.6 | 57.3 | 43.0 | 34.5 | 34.1 | 11.5 | 65.3 | 40.6 |
| SDAR | 39.7 | 58.9 | 45.3 | 35.9 | 35.5 | 12.6 | 65.3 | 41.9 |
| **EVOKE (Ours)** | **44.5** | **60.4** | 43.3 | **41.1** | **39.6** | **14.2** | 64.1 | **43.9** |

The **best** and second-best results are highlighted. All ALFWorld results in this table are on seen games (unseen: Table 3). "-" denotes unreported results. * denotes evaluation with skills.

2026), AHEAD (Jin et al., 2026)); and world-model and dynamics methods (IWM (Zhang et al., 2026a), PaW (Lu et al., 2026a), EnvRL (Wang et al., 2026d), ITP (Liu et al., 2026a), MemWM (Wang et al., 2026c)).

### 4.2 Main Results

Tables 1 and 2 compare methods on Qwen2.5-3B/7B-Instruct (Qwen Team et al., 2025) and Qwen3-1.7B (Yang et al., 2025) backbones. EVOKE achieves the best average on every benchmark and backbone, including all world-model and dynamics methods on 7B, and remains the best on unseen ALFWorld games (Table 3). The margins are largest on ALFWorld and WebShop, which require multi-step interaction, and on the smallest backbone, Qwen3-1.7B. On search-based QA, where each question involves only a few retrieval decisions and the baselines are closer, EVOKE still leads the strongest baseline on each backbone. Section 5 analyzes where the gains come from.

## 5 Ablations and Analysis

We examine where the gains of EVOKE come from and how they arise. All analyses use Qwen2.5-3B on ALFWorld. Every variant starts from the same walkthrough-initialized policy $\pi_{\text{boot}}$ and, unless noted, uses the same preference data and training schedule as EVOKE. Each checkpoint is evaluated

Table 3: ALFWorld Unseen results (%). The **best** and second-best results are highlighted within each backbone.

| Method | Pick | Look | Clean | Heat | Cool | Pick2 | Avg |
|---|---|---|---|---|---|---|---|
| *Qwen2.5-3B-Instruct* | | | | | | | |
| *Prompting* | | | | | | | |
| ReAct | 17.4 | 6.7 | 8.8 | 7.4 | 9.1 | 0.0 | 8.2 |
| *Policy optimization and distillation* | | | | | | | |
| GRPO | 73.9 | 60.0 | 82.4 | 59.3 | 72.7 | 76.9 | 70.9 |
| SDAR | 71.0 | 77.5 | 71.8 | 73.9 | 73.2 | 57.6 | 70.8 |
| OPID | 78.3 | **86.7** | 82.4 | 77.8 | 77.3 | 69.2 | 78.6 |
| PCSD | 95.1 | 80.9 | 89.6 | 81.5 | 92.4 | 75.9 | 85.9 |
| AHEAD | 83.3 | 77.8 | 77.4 | 87.0 | 76.2 | **94.1** | 82.6 |
| **EVOKE (Ours)** | **95.8** | 72.2 | **93.6** | **95.7** | **95.2** | **94.1** | **91.1** |
| *Qwen2.5-7B-Instruct* | | | | | | | |
| *Policy optimization and distillation* | | | | | | | |
| GRPO | 62.5 | 38.9 | 83.9 | 65.2 | 85.7 | 52.9 | 64.9 |
| AHEAD | 79.2 | **83.3** | 83.9 | 82.6 | 85.7 | **88.2** | 83.8 |
| *World-model and dynamics methods* | | | | | | | |
| IWM | - | - | - | - | - | - | 70.3 |
| EnvRL-GRPO | - | - | - | - | - | - | 70.2 |
| EnvRL-GiGPO | - | - | - | - | - | - | 81.8 |
| **EVOKE (Ours)** | **95.8** | 77.8 | **96.8** | **100.0** | **100.0** | **88.2** | **93.1** |
| *Qwen3-1.7B* | | | | | | | |
| *Policy optimization and distillation* | | | | | | | |
| GRPO | 79.2 | 66.7 | 77.4 | 87.0 | 71.4 | 64.7 | 74.4 |
| SDAR | 75.0 | 72.2 | 77.4 | 87.0 | 76.2 | 76.5 | 77.4 |
| GRSD | 83.3 | **77.9** | 77.4 | 91.3 | **85.7** | 88.2 | 84.0 |
| AHEAD | 66.7 | 44.4 | 51.6 | 73.9 | 57.1 | 58.8 | 58.8 |
| **EVOKE (Ours)** | **87.5** | 55.6 | **93.6** | **95.7** | 81.0 | **94.1** | **84.6** |

on the 140 seen and 134 unseen games with greedy, free-form action generation, and we report micro success averaged over training seeds (Appendix B).

### 5.1 DO THE GAINS COME FROM GOAL INTERVENTIONS?

*Source only* in Table 4 trains with the same ranking objective as EVOKE but only on the original goals, removing every alternative-goal context. Unseen success drops from 91.8% to 82.8%, and the policy needs 40.7% more actions per game. This is not a matter of training longer: *Source replay* repeats the original-goal data until it matches the updates of EVOKE, and still reaches only 85.9% while taking 34.4% more actions. The gain persists under a fixed supervision budget and across paired training seeds (Appendix C.1).

Table 4: **Decision-supervision ablations.** Micro success (%) and average unseen steps, averaged over training seeds. All variants start from $\pi_{\text{boot}}$. The **best** and second-best results are highlighted.

| Method | Alt. goals | Objective | Seen ↑ | Unseen ↑ | Avg Steps ↓ |
|---|---|---|---|---|---|
| $\pi_{\text{boot}}$ | — | Walkthrough SFT | 69.3 | 60.4 | 25.9 |
| Source only | No | Ranking | 82.9 | 82.8 | 16.0 |
| Source replay | No | Ranking | 90.4 | 85.9 | 15.3 |
| Positive SFT | Yes | SFT | 91.4 | 84.3 | 15.0 |
| EVOKE | Yes | Ranking | **92.1** | **91.8** | **11.4** |

### 5.2 IS MORE GOAL DATA ENOUGH?

Goal interventions also multiply the number of goals the policy is trained on, so the gain could simply reflect more and more diverse training tasks. *Positive SFT* receives exactly the same goal contexts, positive actions, and updates as EVOKE, but imitates the positives instead of ranking them. It nearly matches EVOKE on seen games but falls 7.5 points behind on unseen games. More goal data with imitation fits the training environments but transfers much less, which is the surface-level behavior matching discussed in Section 1. Imitation raises the positives without directly pushing down the policy's habitual choices; ranking places these choices below the action each goal calls for, and transfers best when the competitors are the actions the policy itself favors (Appendix C.1).

The advantage holds at every amount of training data (Figure 3; nested subsets of 20–100% of the labeled states, three training orders each). EVOKE outperforms SFT in all 15 paired runs; with 60%

of the states, it already exceeds SFT trained on all of them. Ranking thus extracts more from each labeled state.

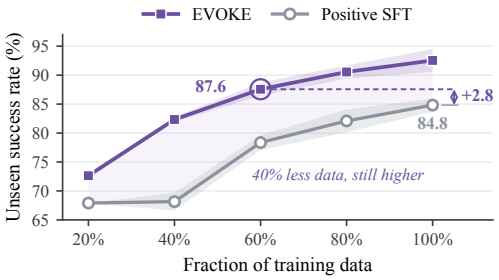



Figure 3: **Data efficiency.** EVOKE outperforms SFT at every budget. Unseen success vs. fraction of labeled states; mean ± s.d. over three training orders.

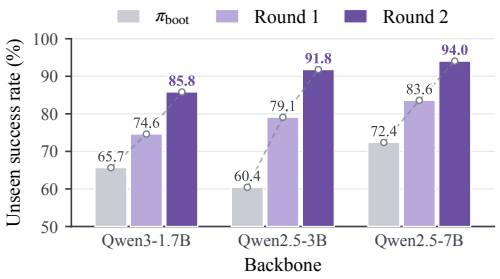



Figure 4: **Iterative improvement.** Every round improves every backbone. Unseen success across training rounds for three backbones.

### 5.3 DOES ITERATION KEEP IMPROVING THE POLICY?

Each round lets the updated policy reach new states and expose new mistakes, which become new hard negatives. Figure 4 follows three backbones through two rounds of aggregation. Unseen success rises by 20.1–31.3 points on every backbone, with a clear gain in every round (Appendix Table 9). On Qwen2.5-3B, the two rounds together supervise about 9.1% as many decisions as the demonstrations behind $\pi_{\text{boot}}$ (Appendix Table 6).

### 5.4 IS THE KNOWLEDGE ALREADY THERE?

If EVOKE elicits rather than adds knowledge, the pretrained model should already encode what its actions do. We feed each model a goal, a history, and a candidate action, extract its hidden state before the action is executed, and train a linear probe to predict five consequences observed after execution: whether the target object is held, lies in the receptacle named by the action, or is hot, clean, or cool. Probes are trained on training games, with layer and regularization selected on held-out training games, and tested on unseen games (Appendix D.1).

Figure 5a shows that consequences are almost perfectly decodable in every model, including the original Qwen2.5-3B backbone before any ALFWorld training. This is not a surface cue: on transitions where the same action text leads to different outcomes, a probe on the action text is at chance, while the model representations remain above 97.0%, against 65.0–68.0% for the same

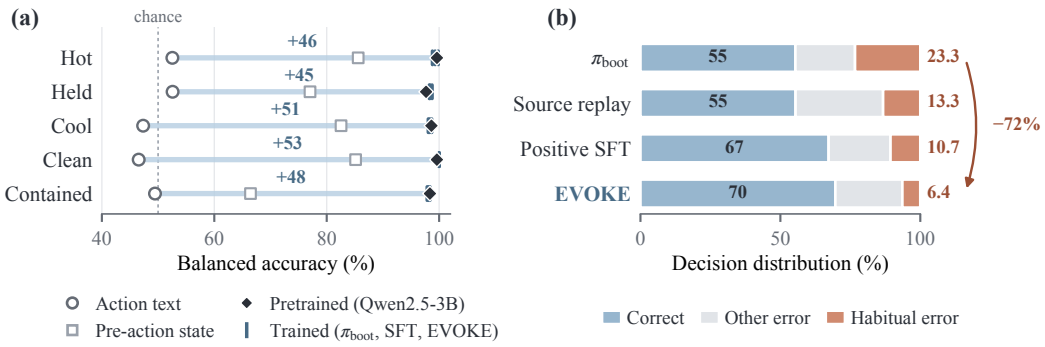



Figure 5: **From knowledge to goal-directed decisions on unseen games.** Consequences are decodable before any ALFWorld training, and EVOKE makes the fewest habitual errors. (a) Balanced accuracy of linear probes that predict an action's consequence from hidden states before the action is executed, on actions whose text leads to different outcomes; Qwen2.5-3B is the backbone before any ALFWorld training. (b) Outcome of each decision on 812 goal pairs that require different actions. A *habitual error* chooses the action that is correct for the other goal of the pair.

architecture with random weights (Appendix Table 15). Yet $\pi_{\text{boot}}$, whose representations are equally decodable, succeeds on only 60.4% of unseen games, and EVOKE raises this to 91.8% (Table 4). The backbone already represents what its actions do; what remains is whether the policy uses this to decide.

### 5.5 DOES EVOKE TURN THIS KNOWLEDGE INTO GOAL-DIRECTED DECISIONS?

A policy relying on contextual habits cannot choose correctly when the same context calls for different actions under different goals (Section 3.1). We test this directly: from the 134 unseen games, we construct 812 goal pairs in which two goals share the same state, history, and available actions but have disjoint sets of progress-making actions. Each model scores all available actions under both goals, and a pair counts as correct only if the top-ranked action is a positive under each goal; random choice succeeds on 0.3% of pairs. We also count *habitual errors*: decisions that choose the action correct for the other goal of the pair (Appendix D.2).

Figure 5b makes the habit visible. $\pi_{\text{boot}}$ solves only 19.5% of pairs, and more than half of its errors are habitual: it follows the context rather than the goal. Training on the intervened data more than doubles joint accuracy, and ranking suppresses the habit further. EVOKE makes the fewest habitual errors (6.4% of decisions), 4.3 points fewer than Positive SFT (95% CI [3.0, 5.6]). Ranking alone already helps, since Source replay reduces habitual errors even without alternative goals, and combined with goal interventions it reduces them the most. EVOKE thus decides by the goal rather than by habit.

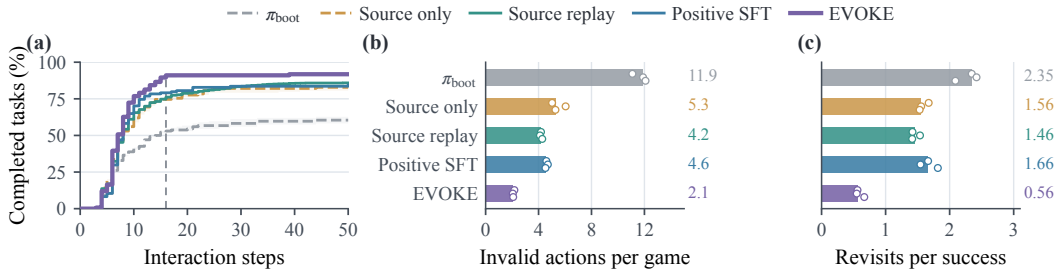



Figure 6: **Execution on unseen games.** EVOKE succeeds earlier and wastes fewer actions. (a) Cumulative success over interaction steps. (b) Invalid actions per game. (c) Revisits to an already visited location per successful game.

### 5.6 BETTER DECISIONS OR MORE TRIAL AND ERROR?

With free-form actions and a 50-step budget, higher success could come from persistence rather than better decisions. Figure 6 separates the two. EVOKE solves 91.0% of unseen games within 20 actions, versus at most 80.6% for the other trained variants, and needs 24.4–28.9% fewer actions per game (Table 4). The speed-up is not merely a consequence of failing less: on the 70 games that all five variants solve, EVOKE still takes the fewest actions. It also makes less than half as many invalid actions as Positive SFT and about a third as many revisits as the other trained variants. EVOKE thus reaches the goal along more direct paths instead of trying more (examples in Appendix Table 13).

## 6 CONCLUSION

We presented EVOKE, which elicits the world knowledge of pretrained LLM agents for transferable decisions by ranking actions under different goals at fixed states, against the policy's own tempting mistakes. Across household tasks, web navigation, and search-based QA with three backbones, EVOKE improves task performance and generalization to unseen environments. Our analyses trace the gains to goal interventions and ranking, and indicate that the pretrained backbone already represents what its actions do, and that EVOKE makes the policy decide by the goal rather than by habit. For LLM agents, eliciting world knowledge through decision supervision offers a practical alternative to acquiring it through prediction.

### AI USE STATEMENT

Large language models (LLMs) were used to aid in the writing and polishing of the manuscript. Within the method, an LLM assists in data construction: it proposes alternative goals and assesses executed candidate actions to build preference labels (Section 3.2 and Appendix A). The annotator is not used at deployment. All research ideas, methods, and conclusions were developed by the authors. The authors bear full responsibility for the content of the manuscript, including any text generated or polished by an LLM, and have ensured that it complies with ethical guidelines and does not involve plagiarism or scientific misconduct.

### REPRODUCIBILITY STATEMENT

We have made every effort to ensure that our results are reproducible. Section 3 and Appendix A describe preference construction, the training objective, model architecture, and hyperparameters; Appendix B describes the evaluation protocol, statistical aggregation, and data-efficiency subsets; Appendix D.1 and Appendix D.2 detail the diagnostic analyses. All benchmarks used in our experiments are publicly available. We will release the source code, trained checkpoints, and experiment scripts to support independent verification and extension of this work.

## REFERENCES

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/453fadbd8a1a3af50a9df4df899537b5-Paper.pdf.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado Van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *Advances in Neural Information Processing Systems*, 38:46375–46408, 2026.

Christopher Grimm, André Barreto, Satinder Singh, and David Silver. The value equivalence principle for model-based reinforcement learning. *Advances in neural information processing systems*, 33: 5541–5552, 2020.

John Hewitt and Percy Liang. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)*, pp. 2733–2743, 2019.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In Donia Scott, Nuria Bel, and Chengqing Zong (eds.), *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.580. URL https://aclanthology.org/2020.coling-main.580/.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Xiaolong Jin, Dingmin Wang, Vijay Lingam, and Varun Kumar. Ahead: Adaptive hindsight with environment-augmented distillation for agentic rl. *arXiv preprint arXiv:2608.24114*, 2026.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Yixia Li, Hongru Wang, Jiahao Qiu, Zhenfei Yin, Dongdong Zhang, Cheng Qian, Zeping Li, Xiaoteng Ma, Guanhua Chen, and Heng Ji. From word to world: Can large language models be implicit text-based world models? In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8084–8111, San Diego, California, United States, July 2026a. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.366. URL https://aclanthology.org/2026.acl-long.366/.

Zichao Li, Gang Wu, Jack Wang, Ruiyi Zhang, Wanrong Zhu, Ryan Rossi, Vlad Morariu, and Jihyung Kil. Spinning straw into gold: Relabeling llm agent trajectories in hindsight for successful demonstrations. In *International Conference on Learning Representations*, volume 2026, pp. 148186–148209, 2026b.

Youwei Liu, Jian Wang, Hanlin Wang, Beichen Guo, and Wenjie Li. Imagine-then-plan: Agent learning from adaptive lookahead with world models. *arXiv preprint arXiv:2601.08955*, 2026a.

Zhipeng Liu, Xuefeng Bai, Kehai Chen, Xinyang Chen, Xiucheng Li, Yang Xiang, Jin Liu, Hong-Dong Li, Yaowei Wang, Liqiang Nie, and Min Zhang. A survey on the feedback mechanism of llm-based ai agents. In James Kwok (ed.), *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pp. 10582–10592. International Joint Conferences on Artificial Intelligence Organization, 8 2025. doi: 10.24963/ijcai.2025/1175. URL https://doi.org/10.24963/ijcai.2025/1175. Survey Track.

Zichen Liu, Anya Sims, Keyu Duan, Changyu Chen, Simon Yu, Xiangxin Zhou, Haotian Xu, Shaopan Xiong, Bo Liu, Chenmien Tan, Weixun Wang, Hao Zhu, Weiyan Shi, Diyi Yang, Michael Qizhe Shieh, Yee Whye Teh, Wee Sun Lee, and Min Lin. Gem: A gym for generalist llms. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 110681–110700, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/b3956812189001f673d29c626bc897ae-Paper-Conference.pdf.

Ning Lu, Baijiong Lin, Shengcai Liu, Jiahao Wu, Haoze Lv, Yanbin Wei, Lingting Zhu, Shengju Qian, Xin Wang, Ying-Cong Chen, et al. Policy and world modeling co-training for language agents. *arXiv preprint arXiv:2606.02388*, 2026a.

Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, Jinyang Wu, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, et al. Self-distilled agentic reinforcement learning. *arXiv preprint arXiv:2605.15155*, 2026b.

Chunji Lv, Yangguang Wei, Junlin Liu, Yang Gao, Ming Liu, Xinming Wang, Jinyang Wu, Guoren Wang, and Changsheng Li. Pcsd: Persistent consistency for self-distillation in agentic reinforcement learning. *arXiv preprint arXiv:2608.01837*, 2026.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric

memories. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9802–9822, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.546. URL https://aclanthology.org/2023.acl-long.546/.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.378. URL https://aclanthology.org/2023.findings-emnlp.378/.

Shuofei Qiao, Runnan Fang, Ningyu Zhang, Yuqi Zhu, Xiang Chen, Shumin Deng, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Agent planning with world knowledge model. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=j6kJSS9O6I.

Qwen Team, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025. URL https://arxiv.org/abs/2412.15115.

Jonathan Richens, Tom Everitt, and David Abel. General agents need world models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 51659–51687. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/richens25a.html.

Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples. *arXiv preprint arXiv:2010.04592*, 2020.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/ross11a.html.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1312–1320, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/schaul15.html.

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

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization of LLM agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7584–7600, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.409. URL https://aclanthology.org/2024.acl-long.409/.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475. URL https://aclanthology.org/2022.tacl-1.31/.

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

Xucong Wang, Ziyu Ma, Shidong Yang, Tongwen Huang, Pengkun Wang, Yong Wang, and Xiangxiang Chu. Role-agent: Bootstrapping llm agents via dual-role evolution. *arXiv preprint arXiv:2606.10917*, 2026b.

Yujun Wang, Tao Zhang, Jinhe Bi, Wenxuan Ye, Boliang Liu, Sikuan Yan, Shuning Wang, Xuebing Zhou, Sören Pirk, Hinrich Schütze, et al. Memwm: Memory-augmented text-based world model. *arXiv preprint arXiv:2608.07107*, 2026c.

Zhitong Wang, Songze Li, Hao Peng, Shuzheng Si, Yi Wang, Maosong Sun, and Juanzi Li. Envrl: Learn from environment dynamics in agentic reinforcement learning. *arXiv preprint arXiv:2606.17680*, 2026d.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Chenxu Yang, Chuanyu Qin, Qingyi Si, Minghui Chen, Naibin Gu, Dingyu Yao, Zheng Lin, Weiping Wang, Jiaqi Wang, and Nan Duan. Self-distilled rlvr. *arXiv preprint arXiv:2604.03128*, 2026a.

Shuo Yang, Jinyang Wu, Zhengxi Lu, Yuhao Shen, Fan Zhang, Lang Feng, Shuai Zhang, Haoran Luo, Zheng Lian, Zhengqi Wen, et al. Opid: On-policy skill distillation for agentic reinforcement learning. *arXiv preprint arXiv:2606.26790*, 2026b.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 20744–20757. Curran Associates, Inc., 2022. doi: 10.52202/068431-1508. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/82ad13ec01f9fe44c01cb91814fd7b8c-Paper-Conference.pdf.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Xiao Yu, Baolin Peng, Ruize Xu, Yelong Shen, Pengcheng He, Suman Nath, Nikhil Singh, Jiangfeng Gao, and Zhou Yu. Reinforcement world model learning for llm-based agents. *arXiv preprint arXiv:2602.05842*, 2026.

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, Jian Xie, Yuxuan Sun, Boyu Gou, Qi Qi, Zihang Meng, Jianwei Yang, Ning Zhang, Xian Li, Ashish Shah, Dat Huynh, Hengduo Li, Zi Yang, Xuefei Cao, Lawrence Keunho Jang, Shuyan Zhou, Jiacheng Zhu, Huan Sun, Jason E Weston, Yu Su, and Yifan Wu. Agent learning via early experience. In *Forty-third International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=N3dXUHY5dD.

Minxing Zhang, Yi Yang, Roy Xie, Bhuwan Dhingra, Shuyan Zhou, and Jian Pei. Generalizability of large language model-based agents: A comprehensive survey. *ACM Comput. Surv.*, 58(10), April 2026b. ISSN 0360-0300. doi: 10.1145/3794858. URL https://doi.org/10.1145/3794858.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

Binbin Zheng, Zijun Xie, Guanqun Zhao, Enlei Gong, Xing Ma, Xiaoliang Fu, and Zeyu Chen. Group-reflective self-distillation for agentic reinforcement learning. *arXiv preprint arXiv:2607.28076*, 2026.

Siyu Zhou, Tianyi Zhou, Yijun Yang, Guodong Long, Deheng Ye, Jing Jiang, and Chengqi Zhang. Wall-e: World alignment by neurosymbolic learning improves world model-based llm agents. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 65352–65391. Curran Associates, Inc., 2025. doi: 10.52202/085713-2192. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/5e772a13ccba5255331240dcd99aa38b-Paper-Conference.pdf.

# A Implementation Details of Evoke

This section follows the order of Section 3.

**Policy input and demonstration initialization.** The policy input consists of the goal, the visible action–observation history, and the list of available actions; the output is the next action as text. $\pi_{\mathrm{boot}}$ is obtained by supervised fine-tuning on 16,247 per-step examples from successful demonstrations in 2,700 training games, for one epoch with learning rate $2\times10^{-4}$, batch size 2, gradient accumulation 8, warmup ratio 0.03, and maximum length 1,280 tokens. LoRA adapters of rank 16 and scaling 16 are attached to the attention query, key, value, and output projections and to the feed-forward gate, up, and down projections.

**Decision-state collection.** The policy runs in training games, and each step records the goal and the visible history, so that the state can be revisited. States are sampled along the trajectories, covering both steps that make progress and steps that precede detours, repeated actions, or failure. The first round yields 344 states from 193 games; after aggregation, the dataset contains 767 states from 312 games (Table 5). The two rounds supervise 1,482 decisions, about 9.1% of the 16,247 demonstration steps used for $\pi_{\mathrm{boot}}$, and raise unseen success from 60.4% to 91.8% (Table 6).

**Goal interventions.** Given the scene and the original goal, the LLM annotator (Qwen2.5-72B-Instruct) proposes alternative goals. Each proposal is instantiated in the environment together with its success condition. A goal is kept if it is compatible with the scene, not yet satisfied, and achievable from the current state. The original task text is removed from the visible history, so the policy sees only the goal of the current context, while the environment state, history, and available actions are shared by all goals of a state.

**Candidate actions and annotation.** For each state–goal context, candidates are drawn from $\mathcal{A}(s)$: the policy's highest-scoring actions, further available actions, and annotator suggestions that match an available action. Each candidate is executed once from the same state, and its resulting observation is recorded, followed by a short continuation when a single step does not reveal the effect of the action. The annotator receives the goal, the history before the action, and every candidate paired with its executed outcome, and classifies each candidate as advancing the goal, worse, or uncertain. Advancing actions form $P$, worse actions form the competitor pool, and uncertain actions are discarded. The annotator thus serves as a labeler of observed outcomes rather than a planner: each label rests on an executed transition and a goal whose success condition is defined by the environment.

**Preference groups and hard negatives.** Each group contains up to four positives and four competitors. Competitors are selected in order of priority: the policy's greedy action when it is not a positive, then non-positive actions with the highest mean log-probability under the policy, and then actions of other types for coverage. A context with more than four positives is split into several groups that share the same competitors, and each group is weighted by the inverse of the number of groups in its context. In the final dataset, every group has four competitors and on average 1.6 positives.

**Optimization.** Action scores average the token log-probabilities of the action completion, including its end-of-turn token. The ranking loss uses $\tau = 1$ and $\lambda = 0.25$. Each round starts a new AdamW optimizer with zero weight decay, continues the LoRA adapter from the previous round, and accumulates gradients over eight groups; the backbone stays frozen throughout. The final checkpoint of each round is used. Table 5 lists the schedule. The positive-SFT variant in Section 5 uses the same groups, weights, and schedule, and minimizes $\mathcal{L}_{\mathrm{SFT}} = -|P|^{-1}\sum_{p\in P} z_\theta(p \mid x_g)$.

Table 5: **Training schedule of the analyses in Section 5.** Round 2 trains on the aggregate of both rounds.

| Round | States | Contexts | Groups | Games | Epochs | Learning rate | Updates |
|---|---|---|---|---|---|---|---|
| 1 | 344 | 687 | 706 | 193 | 2 | $10^{-5}$ | 178 |
| 2 (aggregate) | 767 | 1,482 | 1,524 | 312 | 1 | $5\times10^{-6}$ | 191 |

Table 6: **Supervision and success.** $\pi_{\text{boot}}$ imitates one action per demonstration step; EVOKE ranks the candidate actions of each context. Micro success (%).

| | Supervised decisions | % of $\pi_{\text{boot}}$ | Games | Seen | Unseen |
|---|---|---|---|---|---|
| $\pi_{\text{boot}}$ | 16,247 | 100.0 | 2,700 | 69.3 | 60.4 |
| + Round 1 | 687 | 4.2 | 193 | 88.6 | 79.1 |
| + Round 2 (aggregate) | 1,482 | 9.1 | 312 | 92.1 | 91.8 |

**Other environments.** The same pipeline applies to WebShop and search-based QA; only the form of goals, states, and actions changes. In WebShop, for example, a goal can be a purchase instruction, a state the current page with its history, and a candidate action a search query or a click. In search-based QA, a goal can be a question, a state the retrieval history so far, and a candidate action a new query or an answer; an alternative goal is then another question the same retrieval history can lead to, such as one about an entity already retrieved. In both environments, the annotator labels the executed outcomes, i.e., the next page or the retrieved documents.

# B EVALUATION PROTOCOL

**Evaluation.** ALFWorld Seen and Unseen contain 140 and 134 games, with 35/13/27/16/25/24 and 24/18/31/23/21/17 games in Pick/Look/Clean/Heat/Cool/Pick2 order. At each step, the policy receives the goal, the full visible history, and the list of available actions, and generates the next action greedily as free text; no legal-action masking or output repair is applied, and the output is passed to the environment as is. An episode succeeds if the environment reports task completion within 50 actions.

**Aggregation.** Micro success is the fraction of successful games, and macro success is the unweighted mean over the six task types. Each checkpoint is evaluated with three seeds and summarized by the median, and all reported results are averaged over training seeds. The data-efficiency curves average three training orders per point. Confidence intervals resample games with replacement 10,000 times, pooling training seeds and keeping all observations of a game together.

## B.1 DATA-EFFICIENCY PROTOCOL

Subsets contain 20, 40, 60, 80, or 100% of the labeled states and are nested, so each smaller subset is contained in every larger one. States are stratified by collection round and task type, and every selected state keeps all of its goal contexts and preference groups. Both objectives use the same subsets, the same two-round schedule as the full model (Table 5), and the same three training orders, so the number of updates scales with the amount of data. Table 7 lists the subset sizes and Table 8 the results.

Table 7: **Data subsets.** States and goal contexts after aggregation.

| Data | 20% | 40% | 60% | 80% | 100% |
|---|---|---|---|---|---|
| States | 154 | 307 | 460 | 613 | 767 |
| Contexts | 301 | 592 | 889 | 1,185 | 1,482 |

Table 8: **Data efficiency.** Micro success (%), mean ± s.d. over three training orders; all budgets, including 100%, are retrained on nested subsets, independently of Table 4. Δ: Unseen EVOKE minus SFT.

| | Seen | | Unseen | | |
|---|---|---|---|---|---|
| Data | EVOKE | SFT | EVOKE | SFT | Δ |
| 20% | 77.4±1.6 | 75.7±0.7 | 72.6±0.4 | 67.9±0.0 | +4.7 |
| 40% | 82.4±1.5 | 78.6±0.0 | 82.3±0.4 | 68.2±1.6 | +14.2 |
| 60% | 89.3±0.7 | 84.3±0.7 | 87.6±1.1 | 78.4±1.3 | +9.2 |
| 80% | 92.1±1.9 | 85.7±0.0 | 90.5±1.1 | 82.1±2.0 | +8.5 |
| 100% | 93.6±1.4 | 89.5±0.8 | 92.5±2.0 | 84.8±1.1 | +7.7 |

# C ADDITIONAL RESULTS

Table 9: **Iterative training.** Seen / Unseen micro success (%) after each round.

| Backbone | $\pi_{\text{boot}}$ | Round 1 | Round 2 |
|---|---|---|---|
| 1.7B | 80.0 / 65.7 | 87.1 / 74.6 | 92.1 / 85.8 |
| 3B | 69.3 / 60.4 | 88.6 / 79.1 | 92.1 / 91.8 |
| 7B | 81.4 / 72.4 | 87.1 / 83.6 | 95.7 / 94.0 |

Table 10: **Taskwise results of the decision-supervision ablations.** Success (%) on Seen and Unseen games; averaged over training seeds.

| Method | Pick | Look | Clean | Heat | Cool | Pick2 | Micro | Macro |
|---|---|---|---|---|---|---|---|---|
| **Seen** | | | | | | | | |
| $\pi_{\text{boot}}$ | 74.3 | 61.5 | 77.8 | 62.5 | 60.0 | 70.8 | 69.3 | 67.8 |
| Source only | 91.4 | 84.6 | 81.5 | 81.3 | 68.0 | 87.5 | 82.9 | 82.4 |
| Source replay | 95.4 | 95.9 | 90.9 | 92.5 | 73.9 | 95.3 | 90.4 | 90.6 |
| Positive SFT | 97.1 | 84.6 | 81.5 | 87.5 | 96.0 | 95.8 | 91.4 | 90.4 |
| EVOKE | 91.4 | 76.9 | 96.3 | 100.0 | 92.0 | 91.7 | 92.1 | 91.4 |
| **Unseen** | | | | | | | | |
| $\pi_{\text{boot}}$ | 50.0 | 33.3 | 64.5 | 78.3 | 71.4 | 58.8 | 60.4 | 59.4 |
| Source only | 91.7 | 88.9 | 74.2 | 87.0 | 71.4 | 88.2 | 82.8 | 83.6 |
| Source replay | 88.9 | 85.2 | 88.8 | 86.1 | 80.0 | 83.9 | 85.9 | 85.5 |
| Positive SFT | 79.2 | 77.8 | 83.9 | 87.0 | 95.2 | 82.4 | 84.3 | 84.2 |
| EVOKE | 95.8 | 72.2 | 93.6 | 95.7 | 95.2 | 94.1 | 91.8 | 91.1 |

## C.1 ADDITIONAL ABLATIONS

Both comparisons start from $\pi_{\text{boot}}$ and follow the protocol of Section 5.

**Alternative goals under a fixed budget.** Two variants share the same 600 policy-visited states, 300 and 600 contexts in the two rounds, the ranking objective and competitor selection of EVOKE, and 76 + 75 updates. They differ only in whether half of the contexts replace the original goal with an alternative goal while holding the state and history fixed (Table 11). Across paired training seeds, the 95% confidence interval of the unseen-success difference between EVOKE and Source replay is [1.3, 9.5].

**Competitors in the ranking.** With the data, positives, and objective of EVOKE fixed, the competitors ranked below the positives are replaced by the positives of the other goals posed at the same state (filled with random actions), by random valid actions, or by the valid actions least likely under $\pi_{\text{boot}}$ (Table 12).

Table 11: **Alternative goals under a fixed budget.** Micro success (%).

| Goals | Seen | Unseen |
|---|---|---|
| Original goals only | 84.6 | 78.8 |
| Half alternative goals | **86.1** | **84.5** |

Table 12: **Competitors in the ranking.** Micro success (%) and average unseen steps.

| Ranked against | Seen | Unseen | Steps |
|---|---|---|---|
| Policy-favored actions (EVOKE) | **92.1** | **91.8** | **11.4** |
| Other goals' positives | 85.7 | 79.9 | 16.7 |
| Random actions | 78.6 | 74.6 | 18.2 |
| Least likely actions | 72.1 | 67.2 | 21.8 |

Table 13: **Examples of goal-directed execution** from paired unseen trajectories; parentheses give action indices.

| Goal | EVOKE | Positive SFT |
|---|---|---|
| Clean a bowl; place in cabinet | Takes the bowl (3), cleans it at the sink (5), and places it in the cabinet (7). | Repeatedly visits countertops; reaches the 50-action limit. |
| Put a hot cup in cabinet | Takes the cup out of the cabinet (6), heats it in the microwave (8), and returns it to the cabinet (10). | Revisits a countertop and repeats ineffective moves; reaches the 50-action limit. |
| Put two keychains in safe | Places the first keychain (5), finds and takes a second keychain (14), and places it in the same safe (16). | Shares actions 1–5, then revisits locations and picks up a watch (21); reaches the 50-action limit. |

# D Diagnostic Details

## D.1 Consequence Probes

Probe data come from 120 training games (probe training), 36 further training games (layer and regularization selection), and all 134 unseen games (test), with task types stratified and no game shared across splits. In each game, we take up to eight states along a trajectory and, at each state, up to six take, put, or move actions, including actions that fail, and execute them in the simulator to obtain the post-action labels. Besides the trained policies, we probe the original Qwen2.5-3B-Instruct without any adapter. The input contains the goal, the history before the action, and the candidate action, without the list of available actions or any post-action information. We extract the hidden state at the last action token from layers 9, 18, 27, and 36 and fit class-balanced logistic regression with L2 strength in $\{10^{-4}, 10^{-3}, 10^{-2}\}$, selecting the layer and strength on the selection games. The action-text baseline fits the same classifier to unigram and bigram features of the action, and the pre-action baseline predicts that the label is unchanged by the action. For hot, clean, and cool, probe data come from the heat, clean, and cool task families of the same game splits, using heat, clean, and cool actions from states along reference trajectories and states next to the relevant appliance. Labels are read from the simulator after execution; the test set contains 1,110 held/contained transitions from 134 unseen games and 2,102 hot/clean/cool transitions from 75 unseen games. "Same action" restricts the test set to actions whose text appears with different outcomes in the same game; Figure 5a reports this subset, and Table 14 gives both.

Table 14: **Consequence probes on unseen games.** Balanced accuracy (%) on all test transitions and on the same-action subset.

| | Hot | | Held | | Cool | | Clean | | Contained | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Input** | **All** | **Same** | **All** | **Same** | **All** | **Same** | **All** | **Same** | **All** | **Same** |
| Action text | 64.9 | 52.5 | 68.5 | 52.6 | 54.4 | 47.3 | 56.4 | 46.6 | 70.1 | 49.5 |
| Pre-action state | 86.9 | 85.6 | 72.7 | 77.0 | 83.4 | 82.6 | 85.1 | 85.1 | 60.4 | 66.4 |
| Qwen2.5-3B | 99.5 | 99.6 | 99.1 | 97.7 | 98.8 | 98.6 | 99.5 | 99.6 | 99.3 | 98.4 |
| $\pi_{\text{boot}}$ | 99.4 | 98.8 | 99.5 | 98.8 | 98.4 | 98.1 | 99.8 | 100.0 | 99.3 | 98.4 |
| Positive SFT | 99.7 | 99.8 | 99.1 | 98.3 | 98.4 | 98.3 | 99.8 | 100.0 | 99.1 | 98.1 |
| EVOKE | 99.6 | 99.3 | 99.4 | 98.4 | 98.5 | 98.7 | 99.8 | 100.0 | 98.9 | 97.9 |

**Probe controls.** Three controls use the held and contained data and the same probe protocol (Table 15). A model with the Qwen2.5-3B architecture and random weights tests whether decodability requires pretraining. A control task (Hewitt & Liang, 2019) assigns each action type (verb, object class, and receptacle class) a fixed random label with the base rate of the real label, so that accuracy on it reflects what the probe can memorize. A history-text baseline fits the classifier to unigram and bigram features of the goal, history, and action.

Table 15: **Probe controls on unseen games.** Balanced accuracy (%).

| | Held | | Contained | |
|---|---|---|---|---|
| **Input** | **All** | **Same** | **All** | **Same** |
| Action text | 68.5 | 52.6 | 70.1 | 49.5 |
| History text | 80.9 | 77.1 | 75.7 | 71.3 |
| Random weights | 66.1 | 65.0 | 69.2 | 68.0 |
| Qwen2.5-3B | 99.1 | 97.7 | 99.3 | 98.4 |
| Qwen2.5-3B, control task | 60.5 | 56.9 | 66.7 | 60.2 |

### D.2 Goal-Conditioned Action Choice

We take states at seven evenly spaced positions along reference trajectories of the 134 unseen games and pair each state with up to three alternative goals. A pair is kept if both goals are achievable and not yet satisfied, the state, history, and available actions are identical under both goals, and every available action is labeled as advancing each goal or not. This yields 986 pairs from 109 games; the 812 pairs whose two goals have disjoint positive sets form the evaluation set of Figure 5b. Each model scores every available action by its mean token log-probability, and its top-ranked action is taken as its choice. A decision counts as a habitual error when the top-ranked action belongs to the positive set of the other goal in the pair; the rate is computed over both decisions of every pair. Relative to $\pi_{\mathrm{boot}}$, source replay, and positive SFT, EVOKE improves joint accuracy by 27.3 [23.6, 31.2], 22.4 [18.9, 26.0], and 3.2 [0.2, 6.2] points and reduces habitual errors by 16.9 [15.0, 18.8], 6.9 [4.8, 9.0], and 4.3 [3.0, 5.6] points, respectively (95% game-level bootstrap intervals). Choosing uniformly among the available actions solves 0.3% of pairs. Goal pairs and task success measure different abilities. A goal pair isolates one decision, the same state, history, and available actions under two goals, whereas success on unseen games also depends on execution, such as issuing valid actions and not revisiting locations (Figure 6). Source replay executes far better than $\pi_{\mathrm{boot}}$ (4.2 vs. 11.9 invalid actions per game) while leaving its per-decision accuracy on goal pairs unchanged (55.4%, Figure 5b); Positive SFT chooses by the goal more often but makes more invalid actions and revisits than Source replay. EVOKE improves both.