STATE2STATE: Environment-Derived Mid-Training for LLM Agents

Anonymous ACL submission

Abstract

Training LLM agents commonly relies on supervised fine-tuning from expert trajectories or online reinforcement learning over human-specified tasks with handcrafted verifiers. Though effective, both remain bottlenecked by externally specified tasks and supervision signals, limiting the scalability and diversity of agent training. We study an *environment learning* paradigm in which agents acquire interaction and manipulation capabilities solely through environment interaction, without externally specified tasks. We propose STATE2STATE, an environment-derived mid-training method that converts explored environment states into training objectives, challenging agents to reach a specified target state. By deriving tasks from environment exploration and verifying success through rule-based state matching, STATE2STATE provides scalable and verifiable training objectives without expert supervision or manual task design. Experiments on ALFWorld and ScienceWorld show that STATE2STATE improves agent performance as a standalone environment-learning stage in most settings. As initialization for downstream RL, it further improves final performance and learning efficiency, with promising evidence of cross-environment generalization.

1 Introduction

Existing methods for training LLM agents mainly follow two paradigms: supervised fine-tuning (SFT) from expert trajectories and reinforcement learning with verifiable rewards (RLVR). SFT trains agents to imitate trajectories collected from human experts or stronger teacher models (Zeng et al., 2024; Wang et al., 2025a), but such trajectories are costly to obtain, inherit the capability limits of their sources, and constrain learning to the coverage of a limited task distribution. RLVR reduces the reliance on expert demonstrations by allowing agents to learn through interaction (Wang et al., 2025c; Feng et al., 2025; Jin et al., 2025), but it shifts the bottleneck to task and reward construction: effective training still requires externally specified goals of suitable difficulty and manually designed verifiers for judging success.

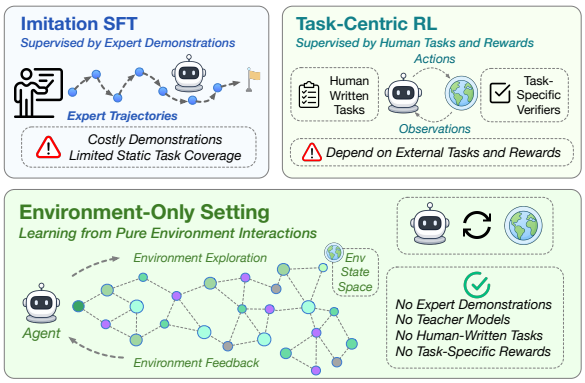


Figure 1: Comparison of supervision sources for LLM agent training. SFT relies on expert trajectories, and task-centric RL relies on human-specified tasks with task-specific verifiers. In contrast, the environment-only setting studied in this work derives learning signals purely from environment itself.

Recent task generation methods (Hu et al., 2025; Shi et al., 2025) attempt to alleviate this bottleneck by expanding the supply of training tasks. For example, recent works (Zhai et al., 2025; Lin et al., 2025) use stronger teacher models to collect interaction trajectories, synthesize task instructions, and construct domain-specific verifiers. While these methods improve the scalability of task acquisition, they still largely treat agent training as learning to solve externally formulated tasks. As a result, the diversity and depth of training objectives remain tied to human-task distributions, teacher-model capabilities, or domain-specific task construction pipelines.

Viewed through the lens of supervision signals, SFT relies on expert action trajectories, while RLVR relies on externally specified goals and verifiers. However, both paradigms remain centered

on human-defined task distributions and external supervision sources. As illustrated in Figure 1, this raises a natural question beyond action-level or goal-level supervision: *Can agents learn useful environment perception and manipulation capabilities directly from environment-only supervision?* In this work, we study this question through an *environment learning* paradigm. Environment learning considers a minimal training setting with only the agent and the environment, where supervision is derived from the environment itself, without external teachers or human-authored tasks. By first learning to understand and manipulate the environment, agents can acquire primitive environment skill priors that later serve as a foundation for RL on human-specified tasks, where these capabilities are aligned with human intent. This progression makes environment learning naturally suited for agent mid-training (Tu et al., 2025), which aims to develop useful capabilities by intermediate training before optimizing specific downstream tasks.

Under this setting, we propose STATE2STATE, a simple instantiation of environment learning that turns reachable environment states into verifiable mid-training objectives. As shown in Figure 2, STATE2STATE begins by exploring the environment with an exploration policy to collect reproducible reachable states. It then filters and samples learnable target states to improve diversity and reproducibility, and pairs each target with a corresponding initial configuration to construct a state-reaching task. In each task, given the initial observation and the target observation, the agent must reason and act through environment interactions to transform the current state into the target state. By sampling from explored states rather than predefined static tasks, STATE2STATE derives task diversity directly from the reachable environment state space, exposing agents to environment dynamics beyond human-designed task distributions. Task success can be verified by matching the resulting state against the target state, yielding rule-based rewards without task-specific test cases. We train agents on these objectives using GRPO-based reinforcement learning (Shao et al., 2024) with dynamic sampling (Yu et al., 2025), which collects informative rollouts for effective and stable training. Overall, STATE2STATE turns environment exploration into verifiable RL objectives, allowing agents to learn from the environment itself without expert demonstrations, teacher models, or human-authored task instructions, while requiring only reproducible environment states and a rule-based state-matching verifier. The resulting environment-aware policy serves as a stronger starting point for subsequent human-task RL, which further aligns the environment skill priors with human intent and achieves better task completion performance.

We evaluate STATE2STATE on ALFWorld (Shridhar et al., 2021), ScienceWorld (Wang et al., 2022), covering household and scientific environments, and further conduct an extension to the more complex MobileWorld (Kong et al., 2025) GUI environment. Across these environments, STATE2STATE improves agent performance before task-specific RL in most settings, showing that verifiable state-reaching objectives can induce useful environment understanding and manipulation capabilities. Serving as a stronger initialization for subsequent RL on human-specified tasks, STATE2STATE further improves final task performance and learning efficiency of downstream RL, demonstrating its effectiveness as an environment-centric mid-training stage. We also observe positive transfer from ScienceWorld to ALFWorld: models mid-trained with STATE2STATE generalize better than those trained on standard ScienceWorld tasks, suggesting that environment-derived objectives potentially support stronger cross-environment generalization. Finally, STATE2STATE yields a standalone improvement on the GUI tasks of MobileWorld, providing initial evidence that the method can extend to more complex GUI interaction environments where collecting human-specified training tasks and reliable verifiers is costly and difficult.

In summary, our main contributions are:

- We study *environment learning* as a mid-training paradigm for LLM agents, where agents learn environment perception and manipulation capabilities from direct agent-environment interaction without external teachers or manually designed tasks.
- We introduce STATE2STATE, which turns explored reachable states into verifiable state-reaching objectives and trains agents with state-matching rewards, enabling reinforcement learning from the environment itself.
- We evaluate STATE2STATE on ALFWorld and ScienceWorld, demonstrating gains before task-specific RL in most settings, better initialization for subsequent human-task RL, and promising cross-environment generalization.

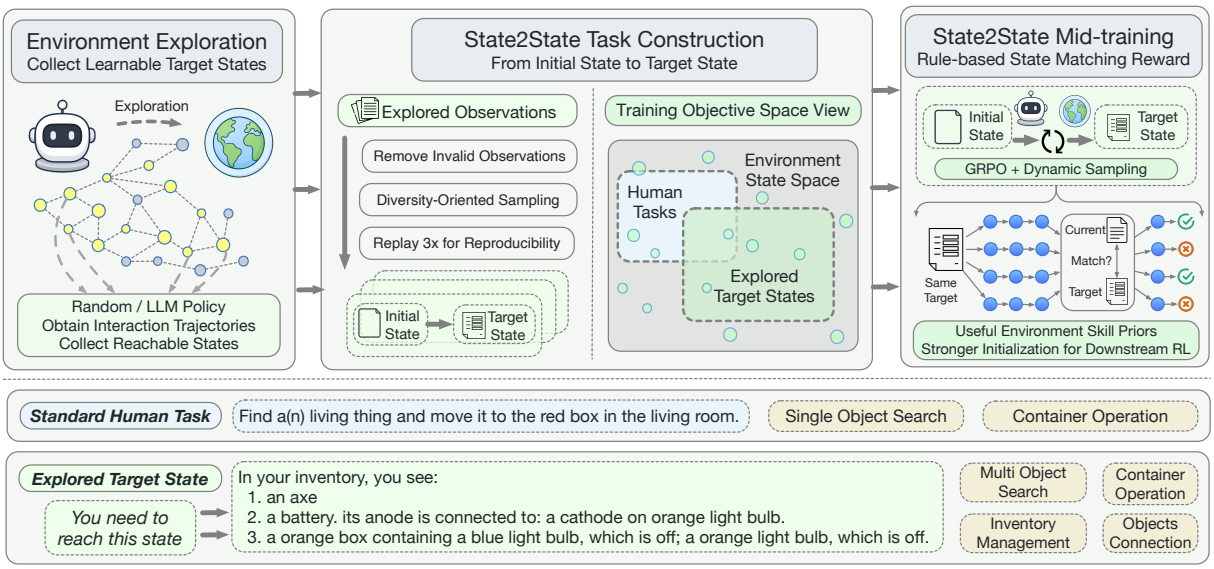


Figure 2: Overview of STATE2STATE, an environment-derived mid-training method. STATE2STATE converts explored environment states into reproducible training objectives and trains agents with rule-based state-matching rewards. By sampling targets from explorations, STATE2STATE aims to expose agents to a different and potentially broader set of environment states, not limited by human-task distributions.

2 Related Work

LLM Agent Training. Recent work has improved LLM agents through supervised fine-tuning and reinforcement learning on task-oriented interaction data. Early studies (Zeng et al., 2024; Chen et al., 2023) fine-tune LLMs on expert trajectories to elicit tool use and decision making. More recent efforts (Li et al., 2025; Team, 2025a; Su et al., 2025; Wang et al., 2025a) further scale this paradigm by automatically constructing task instructions and corresponding trajectories. In parallel, agentic RL (Qi et al., 2025; Wang et al., 2025c) optimizes agents through online interactions and verifiable rewards. Other studies further refine long-horizon behavior with more fine-grained optimization signals (Feng et al., 2025; Zhang et al., 2025b). Despite their effectiveness, these methods largely rely on predefined tasks, expert trajectories or manually designed rewards.

Agentic Mid-Training. Agentic mid-training introduces intermediate objectives before final task training. Prior work (Tu et al., 2025) studies general mid-training for LLMs, and this idea has been extended to agent training by introducing intermediate objectives including world modeling (Wu et al., 2026; Chen et al., 2025a; Yu et al., 2026), reflection and action evaluation (Liu et al., 2026; Xia et al., 2025; Chen et al., 2025b; Yuan et al., 2025) and domain-specific skill-prior learning (Zeng et al., 2026; Yang et al., 2025). As a representative, Agent Early Experience (Zhang et al., 2025a) develops two variants for constructing world-modeling and self-reflection data from reward-free interactions respectively. Unlike these approaches, which mostly turn environment experience into auxiliary supervision for prediction or reflection, STATE2STATE turns explored states into end-to-end comprehensive state-reaching RL objectives.

Learning from Environment Exploration. A closely related direction leverages environment exploration to generate agent training signals. Recent efforts (Zhai et al., 2025; Lin et al., 2025; Wang et al., 2025b) collect autonomous interaction experience and transform the resulting trajectories into tasks, demonstrations or world knowledge, often with additional teacher annotation. Hindsight-based methods (Andrychowicz et al., 2017; Ding, 2026; Pourcel et al., 2025) use achieved outcomes or failed trajectories as alternative goals or supervision. These works demonstrate the value of learning from exploration, but they often require model-based annotation to ensure quality, and their generated objectives remain largely guided by human-task distributions. In contrast, STATE2STATE enables environment learning without external teachers or human-task distributions.

3 Method

3.1 Overview

We present STATE2STATE, an environment-centric mid-training method that turns reachable environment states into verifiable RL objectives for LLM

agents. The core idea is to derive both the task and reward from the environment itself. Instead of learning from expert trajectories or externally specified tasks, STATE2STATE asks the agent to transform an initial environment state into a target state previously explored. Task success is verified by comparing the resulting observation with the target observation, enabling environment-derived reward signals. This formulation applies to environments where reachable states can be reproduced through initialization or replay, and where observations provide a reliable proxy for verifying task-relevant state equivalence.

As shown in Figure 2, STATE2STATE proceeds in three stages. First, an exploration policy interacts with the environment and collects reachable observations, forming a pool of candidate target states. Second, we filter, sample, and replay these observations to construct reproducible state-reaching tasks, each specified by an initial environment configuration and a target observation. Third, the agent is trained with RL to reach the target observation through environment interaction. The resulting policy is then used as initialization for downstream RL on standard human-specified tasks.

3.2 Environment Exploration

The first stage of STATE2STATE collects reachable states through environment exploration. The exploration policy can in principle be an LLM policy, a heuristic policy, or a random policy over current valid actions. While stronger LLM explorers may produce more purposeful trajectories, they incur high inference cost and latency. Moreover, they may introduce task-oriented priors that bias exploration toward states similar to human tasks and therefore reduce state diversity. This is misaligned with our goal: we expect exploration to uncover diverse states from the environment, not limited by human task distributions.

Therefore, we deliberately adopt random exploration as a simple, efficient and scalable strategy for state collection. At each step, the explorer samples an action from the valid action set derived from the environment and executes it. When useful, the sampling distribution can be lightly adjusted with environment-specific priors, such as favoring navigation actions in early steps to enhance exploration breadth and manipulation actions in later steps to prioritize interaction depth. Since no LLM inference is required, random exploration can be highly parallelized and scaled primarily with environment interaction, enabling the construction of large candidate pools of target states.

3.3 STATE2STATE Task Construction

We construct STATE2STATE tasks from trajectories collected during environment exploration. In many interactive environments, the full underlying state is inaccessible or environment-specific, while the agent observes the environment through textual or structured observations. We therefore use observations as an operational representation of state: a target state is specified by a reachable observation, and task success is verified by matching the current observation against the target.

Given the explored observation pool, we first filter out invalid or uninformative observations, such as error messages and meaningless transitions. We then sample target observations with a diversity-oriented strategy that favors broad coverage of unique reachable observations while limiting excessive repetition of the same instance. To ensure that targets define reproducible objectives, we replay the corresponding exploration trajectory from the same initial configuration for three times and retain only observations that can be consistently reached.

Each retained target observation is paired with its source environment configuration to form a STATE2STATE task. At training time, the environment is reset to this initial configuration, and the agent is asked to reach the target observation through interaction. Formally, let $o(s)$ denote the observation exposed at environment state s, and let $\mathcal{O}_{\text{exp}}$ be the set of reachable observations collected during exploration. A STATE2STATE task is specified by an initial state s_0 and a target observation $o^\star \in \mathcal{O}_{\text{exp}}$; starting from s_0, the policy succeeds once it reaches a state s_t with $\text{match}(o(s_t), o^\star) = 1$.

The target observation also naturally serves as a verifier: after each step, we compare the current observation with the target using an environment-specific matching function. In our textual environments, this is instantiated as normalized exact match, yielding the reward:

$$r_t = \begin{cases} 1, & \text{match}(o(s_t), o^\star), \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Unlike standard human tasks, where objectives are given as natural-language instructions and verified by task-specific rewards, STATE2STATE derives both objectives and verifiers from reachable environment states. Solving these tasks requires

the agent to infer environment dynamics, plan over long-horizon interactions, and manipulate the environment toward the target observation, encouraging the agent to acquire useful environment skill priors for downstream human tasks.

3.4 STATE2STATE Mid-Training

We optimize STATE2STATE tasks with the GRPO algorithm augmented with dynamic sampling (Yu et al., 2025). For each target observation, we sample multiple rollouts from the same initial configuration and optimize them with the GRPO algorithm, which compares successful and failed attempts to provide contrastive rewards. Additionally, we apply dynamic sampling by discarding rollout groups with identical rewards and continuing to sample new groups until the required batch size is reached. This increases the density of informative groups and adaptively focuses training on targets near the current policy frontier, allowing the model to effectively learn from environment-derived targets that naturally vary in difficulty.

Finally, STATE2STATE serves as an environment-centric mid-training stage before downstream RL on human-specified tasks. After environment learning, we continue training the policy on human-specified tasks using reinforcement learning, where objectives are natural-language instructions and rewards are given by task-specific verifiers. This design separates environment capability acquisition from human-intent alignment: STATE2STATE first provides scalable environment-grounded skill priors, while downstream RL adapts these priors to solve human-specified tasks.

4 Experiments

To demonstrate the effectiveness of STATE2STATE, we conduct experiments to examine whether STATE2STATE can further improve downstream agent RL beyond direct task-specific training.

4.1 Benchmarks and Evaluation

We evaluate STATE2STATE on two interactive environments: ScienceWorld (Wang et al., 2022) and ALFWorld (Shridhar et al., 2021). ScienceWorld focuses on science-oriented environments, where agents must gather observations, perform procedural manipulation, and execute long-horizon action sequences to complete a scientific experiment. ALFWorld focuses on household navigation and manipulation in textual embodied environments. For both benchmarks, we report task success rates on in-distribution (*ID*) and out-of-distribution (*OOD*) partitions. For ScienceWorld, we randomly sample 400 tasks from each partition due to large volume and for ALFWorld, we use the full set. For stable evaluation, we set temperature to 0.1 during model generation. We include more details in Appendix A.

4.2 Experiment Setup

We view STATE2STATE as a mid-training stage rather than a compute-matched replacement for downstream RL. Therefore, the main comparison tests whether environment-derived mid-training provides useful initialization before task-specific RL. We conduct experiments with two models of different scales, Qwen3-4B and Qwen3-8B (Team, 2025b), under several training settings. STATE2STATE trains the model solely on environment-derived STATE2STATE tasks, which evaluates the transferable interaction capabilities induced by environment learning alone. RL-only(denoted as RL) directly optimizes the model on downstream human tasks without mid-training. STATE2STATE + RL is our full pipeline, where the model first learns environment perception and manipulation, and then continues training on downstream human tasks. Comparing this setting with RL-only isolates the effect of STATE2STATE as a mid-training stage for downstream agent RL.

For additional reference, we also include two baselines based on expert demonstrations: distillation SFT and agent early experience (Zhang et al., 2025a), described in Appendix D. For SFT, we collect 200 high-quality successful trajectories from Qwen-Plus using rejection sampling. Additionally, we report the performance of several competent LLMs including GPT-5.2 (Singh et al., 2025), Claude 4.5 Haiku (Anthropic, 2025), DeepSeek V4 Flash (DeepSeek-AI, 2026) and Qwen-Plus (Team, 2025b). For ScienceWorld, the prompting agents are tested on 200 samples each split to reduce cost.

4.3 Implementation Details

We use a random policy to explore both environments. For ScienceWorld, which involves larger maps, we adopt a two-stage sampling strategy: early exploration favors movement and observation actions, while later exploration favors scientific and state-changing actions. We collect exploration episodes in parallel with hundreds of steps per episode. For filtering, we filter invalid actions, ambiguous commands, error messages, and other

Model	Method	Human-Specified Task Training	ALFWorld			ScienceWorld		
			ID	OOD	Average	ID	OOD	Average
Prompting-based	GPT-5.2	✗	75.71	78.36	77.04	50.50	47.50	49.00
	Claude 4.5 Haiku	✗	79.29	81.34	80.32	48.00	48.00	48.00
	Qwen-Plus	✗	80.00	85.07	82.54	44.50	45.50	45.00
	DeepSeek V4 Flash	✗	84.29	87.31	85.80	44.50	39.00	41.75
Qwen3-4B	Base	✗	35.71	31.34	33.53	25.75	21.50	23.63
	STATE2STATE (Ours)	✗	43.57	46.27	44.92	24.50	17.50	21.00
	Agent early experience	✓	67.14	76.87	72.01	41.00	41.00	41.00
	Distillation SFT	✓	64.29	67.91	66.10	48.00	44.75	46.38
	RL	✓	86.43	85.82	86.13	51.00	48.25	49.63
	STATE2STATE + RL (Ours)	✓	**90.71**	**93.28**	**92.00**	**59.75**	**51.25**	**55.50**
Qwen3-8B	Base	✗	74.29	76.12	75.21	28.50	34.50	31.50
	STATE2STATE (Ours)	✗	70.00	84.33	77.17	31.75	34.50	33.13
	Agent early experience	✓	70.00	72.39	71.20	43.25	44.00	43.63
	Distillation SFT	✓	83.57	88.81	86.19	47.00	49.75	48.38
	RL	✓	91.43	93.28	92.36	56.00	48.25	52.13
	STATE2STATE + RL (Ours)	✓	**97.14**	**97.76**	**97.45**	**59.50**	**52.50**	**56.00**

Table 1: Main results on ALFWorld and ScienceWorld, reported as task success rates. Human-Specified Task Training indicates whether the model is further trained on standard human-specified tasks in our experiments. STATE2STATE generally improves base models as a standalone environment-learning stage, and the full STATE2STATE + RL pipeline achieves the best performance across both benchmarks and model scales.

uninformative observations, and then sample diverse targets by prioritizing unique observations and limiting max repetition number to 5 for each observation. Task success is verified by observation matching, and we use textual normalized exact match for both environments. We include more details about our constructed dataset and several representative examples in Appendix B.

We train STATE2STATE with GRPO augmented with dynamic sampling for 80 steps and continue downstream training with the best-performing checkpoint on the state-reaching validation set. We present more details in Appendix E.

4.4 Main Results

Standalone environment learning. Table 1 reports the main results on ALFWorld and ScienceWorld. As a standalone environment-learning stage, STATE2STATE improves performance in most settings, with consistent gains on ALFWorld and additional gains for Qwen3-8B on ScienceWorld. This shows that environment-centric STATE2STATE can induce useful environment cognition and manipulation capabilities even before training on human-specified tasks. The only exception is Qwen3-4B on ScienceWorld with slight degradation, possibly due to the sparse and challenging exact state-reaching objectives for smaller models. Nevertheless, the same mid-training stage substantially improves downstream RL, indicating that the learned environment priors are better realized after human-task alignment.

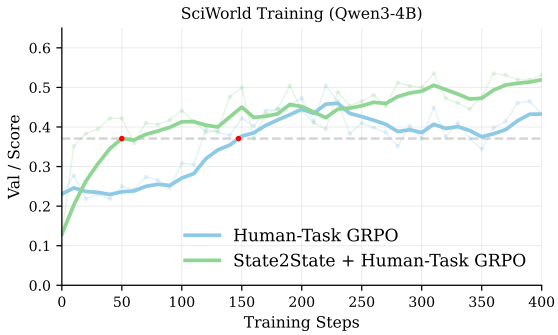


Figure 3: Training curves for Qwen3-4B on ScienceWorld validation set. Ours achieves stronger final performance and higher learning efficiency.

Stronger initialization for downstream RL. The full STATE2STATE + RL pipeline achieves the best performance across both environments and model scales. Its gains over direct RL appear not only on ALFWorld, where RL is already strong, but also on the more challenging ScienceWorld benchmark. Together with consistent improvements on both ID and OOD splits, these results suggest that STATE2STATE provides useful environment-level capabilities rather than merely fitting a specific task distribution. The resulting models also surpass the strong prompting LLMs in these environments.

Improved learning efficiency. Figure 3 further shows that STATE2STATE improves not only final performance but also downstream RL efficiency. On ScienceWorld with Qwen3-4B, STATE2STATE + Human-Task GRPO reaches a relatively high score by around step 50, comparable to the score

Method	ID	OOD	Avg.
Base	25.75	21.50	23.63
SFT	48.00	44.75	46.38
S2S + SFT	47.25	48.75	48.00
SFT + S2S	55.75	52.50	54.13
SFT + RL	65.00	51.75	58.38
S2S + SFT + RL	60.75	60.25	60.50
SFT + S2S + RL	**70.75**	**61.00**	**65.88**

Table 2: Ordering of SFT and STATE2STATE, with STATE2STATE denoted as S2S. SFT + S2S + RL performs best, showing that S2S is most effective as a mid-training stage placed after SFT and before RL.

reached by Human-Task GRPO at around step 150. This indicates that environment-level mid-training provides useful skill priors that can accelerate subsequent learning on human tasks.

Latent benefits released by human-task RL. The full pipeline yields more consistent gains over direct RL than STATE2STATE-only training over the base model. Especially for Qwen3-8B, adding STATE2STATE before RL brings larger improvements than applying STATE2STATE alone on both ALFWorld (+5.09 vs. +1.96) and ScienceWorld (+3.87 vs. +1.63), suggesting that STATE2STATE learns reusable environment-level capabilities whose benefits are only partially reflected by state-reaching performance alone. Downstream RL can further align these capabilities with user-facing task goals, thereby converting latent environment manipulation skills into stronger task-solving performance.

5 Analysis

We conduct the following analyses to better understand the mechanisms of STATE2STATE, with experiments on ScienceWorld with Qwen3-4B model unless otherwise specified.

5.1 Ordering of SFT and STATE2STATE

We analyze how STATE2STATE interacts with commonly used SFT method using the same dataset for the distillation SFT baseline. When applying STATE2STATE on the stronger SFT model, we observe more consistent and stable performance gains and extend the STATE2STATE training to 150 steps. As shown in Table 2, applying STATE2STATE after SFT substantially improves over SFT, showing that STATE2STATE objectives derived from environment explorations still provide complementary benefits beyond SFT.

The benefit becomes clearer when followed by downstream RL. SFT + STATE2STATE + RL achieves the best overall and OOD performance, indicating that STATE2STATE remains effective on top of SFT, benefiting from a stronger initial policy. This suggests that expert demonstrations and environment learning are complementary: while SFT imitates human-task trajectories, STATE2STATE trains on state-reaching objectives sampled from a broader reachable environment space. The training order is also important: applying STATE2STATE before SFT brings limited gains, likely because subsequent imitation learning can overwrite part of the exploratory environment priors. Therefore, STATE2STATE is best used after SFT and before final task-specific RL, where it can improve the SFT policy with environment-grounded skill priors.

Method	ID	OOD	Avg.
Base	25.75	21.50	23.63
GRPO	51.00	48.25	49.63
STATE2STATE + GRPO	59.75	51.25	55.50
GiGPO	60.25	49.25	54.75
STATE2STATE + GiGPO	**61.75**	**53.75**	**57.75**

Table 3: Analysis of different RL backbone algorithms on ScienceWorld. STATE2STATE improves both GRPO and GiGPO, showing that its benefit is complementary to the downstream optimizer.

5.2 Integration with Different RL Backbones

We examine whether STATE2STATE remains effective with a stronger downstream RL backbone algorithm. We use GiGPO (Feng et al., 2025), an RL algorithm with fine-grained rewards designed for agentic RL, and compare each backbone with and without STATE2STATE mid-training.

As shown in Table 3, STATE2STATE + GRPO reaches 55.50 average, outperforming GiGPO alone, showing that environment learning with a standard RL backbone can already match or exceed a stronger RL algorithm without STATE2STATE. When combined with GiGPO, STATE2STATE further improves the performance on both ID and OOD partitions. These results indicate that STATE2STATE is complementary to the choice of RL algorithm and can provide consistent benefits even with a stronger RL optimizer.

5.3 Effect of Exploration Strategy

The environment exploration stage can instantiate different exploration policies for collecting target states. We conduct experiments with two choices: LLM exploration and random exploration. For LLM exploration, we use Qwen-Plus as the strong

Method	Exploration	ID	OOD	Avg.
Base	–	25.75	21.50	23.63
RL	–	51.00	48.25	49.63
STATE2STATE + RL	LLM	54.75	50.50	52.63
STATE2STATE + RL	Random	**59.75**	**51.25**	**55.50**

Table 4: Analysis of exploration strategies on ScienceWorld. Random exploration yields the strongest STATE2STATE + RL result, showing that low-cost state discovery can provide effective training targets.

Mid Training	ID	OOD	Avg.
None	86.43	85.82	86.13
ScienceWorld RL	86.43	87.31	86.87
ScienceWorld STATE2STATE	**88.57**	**90.30**	**89.44**

Table 5: Cross-environment transfer results from ScienceWorld to ALFWorld, showing that STATE2STATE as mid-training transfers better than human tasks. Before training on ALFWorld standard tasks, we conduct mid-training on ScienceWorld with STATE2STATE tasks and human-specified tasks separately.

LLM explorer. As Table 4 shows, both variants improve over direct RL, showing that STATE2STATE can benefit from different sources of states.

Random exploration performs better compared with LLM explorers, and we attribute this to the greater depth and diversity of random exploration in the current environments. Based on our observations, since LLM explorers tend to carry priors toward manually designed tasks, they may visit a narrower region of the reachable state space, while also requiring higher inference cost and latency. In contrast, random exploration provides a cheaper and broader way to collect diverse target states.

5.4 Cross-Environment Generalization

To examine the generalization of STATE2STATE, we evaluate cross-environment transfer from ScienceWorld to ALFWorld. Specifically, we first conduct STATE2STATE mid-training on ScienceWorld and then perform downstream RL on ALFWorld. We also compare against a variant that uses ScienceWorld human-task RL before ALFWorld RL, with the same amount of mid-training steps.

As shown in Table 5, ScienceWorld STATE2STATE improves performance over the baseline without mid-training, indicating that STATE2STATE can provide positive transfer across environments with related dynamics. In contrast, RL on ScienceWorld human tasks shows limited capability transfer. Since this baseline uses the same steps of mid-training, the stronger transfer of our method suggests that the STATE2STATE objective, rather than additional cross-environment RL alone, is critical for learning generalizable environment capabilities.

Overall, these results suggest that STATE2STATE supports both in-domain adaptation and potentially promising cross-environment transfer.

Method	Training Steps	Score
MAI-UI-8B	0	0.275
STATE2STATE	80	**0.308**

Table 6: Results on the GUI-only subset of MobileWorld. STATE2STATE improves MAI-UI-8B in a mobile GUI environment without downstream RL on human-specified tasks.

5.5 Extension to Mobile GUI Environments

To examine whether STATE2STATE can extend beyond text-based environments, we conduct an additional experiment on MobileWorld (Kong et al., 2025), a more complex mobile GUI benchmark with long-horizon and cross-application tasks. We evaluate on its GUI-only subset, containing 117 complex tasks. MobileWorld provides a suitable testbed for evaluating standalone STATE2STATE: its benchmark tasks are limited without available official training data, and scaling human-specified task training with reliable verifiers is costly. We therefore evaluate STATE2STATE without downstream RL on human-specified tasks, directly testing whether environment-derived objectives can improve interaction capabilities in a more complex GUI environment. More details are in Appendix F.

As shown in Table 6, STATE2STATE improves MAI-UI-8B from 0.275 to 0.308 after 80 training steps. This result suggests that STATE2STATE can provide useful environment-learning signals in more complex GUI environments, where collecting human-specified training tasks is difficult.

6 Conclusion

We present STATE2STATE, an environment-derived mid-training method that converts explored environment states into verifiable state-reaching objectives. By deriving both objectives and rewards from the environment, STATE2STATE enables agents to acquire environment skill priors without expert supervision or human-specified tasks. Across several environments, STATE2STATE standalone generally improves agent performance, strengthens downstream human-task RL, and shows promising cross-environment generalization.

Limitations

Our experiments provide an initial study of environment learning for LLM agents, prioritizing controlled online RL comparisons over exhaustive scaling. We evaluate STATE2STATE on Qwen3-4B and Qwen3-8B, which are strong enough to exhibit meaningful interactive learning but do not cover frontier-scale models. Larger models may interact with STATE2STATE differently due to stronger priors, longer-horizon planning, or better exploration. While the state-reaching objectives and rule-based verification should remain applicable, systematic scaling analysis is needed to understand how gains evolve with model size and training compute.

Our evaluation also focuses on representative environments with reproducible states and verifiable success conditions. ALFWorld and ScienceWorld test household and scientific procedural interaction, and our results suggest benefits for both in-domain adaptation and cross-environment transfer. However, environment learning may benefit from richer reachable state spaces. Extending STATE2STATE to more visually grounded, web-based, software-engineering, and real-device control environments would better assess the generality of the learned manipulation capabilities. We view this as a natural extension, since the core method only requires reproducible states and a state-matching verifier.

References

Marcin Andrychowicz, Dwight Crow, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight experience replay. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5048–5058.

Anthropic. 2025. System card: Claude haiku 4.5.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. 2023. Fireact: Toward language agent fine-tuning. *CoRR*, abs/2310.05915.

Shiqi Chen, Tongyao Zhu, Zian Wang, Jinghan Zhang, Kangrui Wang, Siyang Gao, Teng Xiao, Yee Whye Teh, Junxian He, and Manling Li. 2025a. Internalizing world models via self-play finetuning for agentic RL. *CoRR*, abs/2510.15047.

Yihan Chen, Benfeng Xu, Xiaorui Wang, Yongdong Zhang, and Zhendong Mao. 2025b. Training llm-based agents with synthetic self-reflected trajectories and partial masking. *CoRR*, abs/2505.20023.

DeepSeek-AI. 2026. Deepseek-v4: Towards highly efficient million-token context intelligence.

Liang Ding. 2026. Agenther: Hindsight experience replay for LLM agent trajectory relabeling. *CoRR*, abs/2603.21357.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. 2025. Group-in-group policy optimization for LLM agent training. *CoRR*, abs/2505.10978.

Mengkang Hu, Pu Zhao, Can Xu, Qingfeng Sun, Jian-Guang Lou, Qingwei Lin, Ping Luo, and Saravan Rajmohan. 2025. Agentgen: Enhancing planning abilities for large language model based agent via environment and task generation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, V.1, KDD 2025, Toronto, ON, Canada, August 3-7, 2025*, pages 496–507. ACM.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *CoRR*, abs/2503.09516.

Quyu Kong, Xu Zhang, Zhenyu Yang, Nolan Gao, Chen Liu, Panrong Tong, Chenglin Cai, Hanzhang Zhou, Jianan Zhang, Liangyu Chen, Zhidan Liu, Steven Hoi, and Yue Wang. 2025. Mobileworld: Benchmarking autonomous mobile agents in agent-user interactive and mcp-augmented environments. *CoRR*, abs/2512.19432.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Baixuan Li, Bo Zhang, Dingchu Zhang, Fei Huang, Guangyu Li, Guoxin Chen, Huifeng Yin, Jialong Wu, Jingren Zhou, Kuan Li, Liangcai Su, Litu Ou, Liwen Zhang, Pengjun Xie, Rui Ye, Wenbiao Yin, Xinmiao Yu, Xinyu Wang, Xixi Wu, and 36 others. 2025. Tongyi deepresearch technical report. *CoRR*, abs/2510.24701.

Musen Lin, Minghao Liu, Taoran Lu, Lichen Yuan, Yiwei Liu, Haonan Xu, Yu Miao, Yuhao Chao, and Zhaojian Li. 2025. Gui-rewalk: Massive data generation for GUI agent via stochastic exploration and intent-aware reasoning. *CoRR*, abs/2509.15738.

Weize Liu, Minghui Liu, Sy-Tuyen Ho, Souradip Chakraborty, Xiyao Wang, and Furong Huang. 2026. Agentic critical training. *CoRR*, abs/2603.08706.

Julien Pourcel, Cédric Colas, and Pierre-Yves Oudeyer. 2025. Self-improving language models for evolutionary program synthesis: A case study on ARC-AGI. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Jiadai Sun, Xinyue Yang, Yu Yang, Shuntian Yao, Wei Xu, Jie Tang, and Yuxiao Dong. 2025. Webrl: Training LLM web agents via self-evolving online curriculum reinforcement learning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*.

Dingfeng Shi, Jingyi Cao, Qianben Chen, Weichen Sun, Weizhen Li, Hongxuan Lu, Fangchen Dong, Tianrui Qin, King Zhu, Minghao Liu, Jian Yang, Ge Zhang, Jiaheng Liu, Changwang Zhang, Jun Wang, Yuchen Eleanor Jiang, and Wangchunshu Zhou. 2025. Taskcraft: Automated generation of agentic tasks. *CoRR*, abs/2506.10055.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. 2021. Alfworld: Aligning text and embodied environments for interactive learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, and 1 others. 2025. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*.

Liangcai Su, Zhen Zhang, Guangyu Li, Zhuo Chen, Chenxi Wang, Maojia Song, Xinyu Wang, Kuan Li, Jialong Wu, Xuanzhong Chen, Zile Qiao, Zhongwang Zhang, Huifeng Yin, Shihao Cai, Runnan Fang, Zhengwei Tao, Wenbiao Yin, Chenxiong Qian, Yong Jiang, and 3 others. 2025. Scaling agents via continual pre-training. *CoRR*, abs/2509.13310.

Kimi Team. 2025a. Kimi K2: open agentic intelligence. *CoRR*, abs/2507.20534.

Qwen Team. 2025b. Qwen3 technical report. *CoRR*, abs/2505.09388.

Chengying Tu, Xuemiao Zhang, Rongxiang Weng, Rumei Li, Chen Zhang, Yang Bai, Hongfei Yan, Jingang Wang, and Xunliang Cai. 2025. A survey on LLM mid-training. *CoRR*, abs/2510.23081.

Ruoyao Wang, Peter A. Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. Scienceworld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 11279–11298. Association for Computational Linguistics.

Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Henry Wu, Zhennan Shen, Zhuokai Li, Ryan Li, Xiaochuan Li, Junda Chen, Boyuan Zheng, Peihang Li, Fangyu Lei, Ruisheng Cao, and 23 others. 2025a. Opencua: Open foundations for computer-use agents. *CoRR*, abs/2508.09123.

Zhaoyang Wang, Yiming Liang, Xuchao Zhang, Qianhui Wu, Siwei Han, Anson Bastos, Rujia Wang, Chetan Bansal, Baolin Peng, Jianfeng Gao, and 1 others. 2025b. Adapting web agents with synthetic supervision. *arXiv preprint arXiv:2511.06101*.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. 2025c. RAGEN: understanding self-evolution in LLM agents via multi-turn reinforcement learning. *CoRR*, abs/2504.20073.

Mengzhou Wu, Yuzhe Guo, Yuan Cao, Haochuan Lu, Songhe Zhu, Pingzhe Qu, Xin Chen, Kang Qin, Zhongpu Wang, Xiaode Zhang, Xinyi Wang, Wei Dai, Gang Cao, Yuetang Deng, Zhi Gong, Dezhi Ran, Linyi Li, Wei Yang, and Tao Xie. 2026. Ui-oceanus: Scaling GUI agents with synthetic environmental dynamics. *CoRR*, abs/2604.02345.

Yu Xia, Yiran Shen, Junda Wu, Tong Yu, Sungchul Kim, Ryan A. Rossi, Lina Yao, and Julian J. McAuley. 2025. SAND: boosting LLM agents with self-taught action deliberation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pages 3062–3077. Association for Computational Linguistics.

Zonghan Yang, Shengjie Wang, Kelin Fu, Wenyang He, Weimin Xiong, Yibo Liu, Yibo Miao, Bofei Gao, Yejie Wang, Yingwei Ma, Yanhao Li, Yue Liu, Zhenxing Hu, Kaitai Zhang, Shuyi Wang, Huarong Chen, Flood Sung, Yang Liu, Yang Gao, and 2 others. 2025. Kimi-dev: Agentless training as skill prior for swe-agents. *CoRR*, abs/2509.23045.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, and 16 others. 2025. DAPO: an open-source LLM reinforcement learning system at scale. *CoRR*, abs/2503.14476.

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

Siyu Yuan, Zehui Chen, Zhiheng Xi, Junjie Ye, Zhengyin Du, and Jiecao Chen. 2025. Agent-r: Training language model agents to reflect via iterative self-training. *CoRR*, abs/2501.11425.

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, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, Findings of ACL, pages 3053–3077. Association for Computational Linguistics.

Ji Zeng, Dayuan Fu, Tiantian Mi, Yumin Zhuang, Yaxing Huang, Xuefeng Li, Lyumanshan Ye, Muhang Xie, Qishuo Hua, Zhen Huang, Mohan Jiang, Hanning Wang, Jifan Lin, Yang Xiao, Jie Sun, Yunze Wu, and Pengfei Liu. 2026. davinci-dev: Agent-native mid-training for software engineering. *CoRR*, abs/2601.18418.

Yunpeng Zhai, Shuchang Tao, Cheng Chen, Anni Zou, Ziqian Chen, Qingxu Fu, Shinji Mai, Li Yu, Jiaji Deng, Zouying Cao, Zhaoyang Liu, Bolin Ding, and Jingren Zhou. 2025. Agentevolver: Towards efficient self-evolving agent system. *CoRR*, abs/2511.10395.

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, and 11 others. 2025a. Agent learning via early experience. *CoRR*, abs/2510.08558.

Zijing Zhang, Ziyang Chen, Mingxiao Li, Zhaopeng Tu, and Xiaolong Li. 2025b. RLVMR: reinforcement learning with verifiable meta-reasoning rewards for robust long-horizon agents. *CoRR*, abs/2507.22844.

A Environments

A.1 ALFWorld

ALFWorld (Shridhar et al., 2021) is a text-based embodied household environment built on ALFRED and TextWorld. In the standard human tasks, each episode starts from a household scene and a task instruction, such as finding, cleaning, heating, cooling, examining, or placing objects. At each step, the agent receives a textual observation describing the consequence of its previous actions. The environment also exposes a set of admissible commands, which are used for environment exploration by the random policy. The agent must output a command inside an `<action>` span, and the command is executed by the simulator.

We use the ALFWorld text environment for standard human task training and evaluation. Episodes terminate according to the environment success signal, and the reward returned from the environment is 1.0 for successful completion and 0.0 otherwise.

The STATE2STATE training reuses the same environment as the standard human tasks but replaces the original natural-language task instruction with a target observation as the state-reaching objective. Each task specifies a source game file for environment initialization and a target observation string. After each action, the current textual observation is compared with the target observation after normalization. The episode succeeds and terminates when the normalized observation exactly matches the normalized target. Thus, STATE2STATE provides a binary, rule-based state-matching reward without using the original ALFWorld task success verifier.

Training and evaluation are run with parallel environment workers. For grouped RL, repeated workers in the same group load the same task instance, enabling multiple rollouts from the same initial configuration.

A.2 ScienceWorld

ScienceWorld (Wang et al., 2022) is a text-based interactive environment for elementary-level science tasks. Each episode presents a scientific goal that may require navigation, object inspection, object manipulation, container operations, device activation, and experiment execution. The agent receives a textual observation at each step and must issue one valid action. The action space is built from action templates, possible objects, and valid action-object combinations exposed by the simu-

lator. In the prompt, we present the generic action templates.

In the standard ScienceWorld environment, each task is specified by a task name and variation. The simulator maintains a task score in the range from 0 to 100. In our reward design, we reward only perfect completion of the task: score 100 from the environment will receive a 1.0 for successful completion and 0.0 otherwise. In our main experiments, ScienceWorld evaluation reports success rates on sampled in-distribution and out-of-distribution task partitions.

The STATE2STATE training uses the same ScienceWorld environments but defines the objective as reaching a target observation string, without access to the original environment task instruction. Each STATE2STATE task JSON provides the original task name, task variation, and `task_target_state`. After each action, the current observation is compared against the target observation using exact string equality under normalization. If the strings match, the episode terminates with reward 1; otherwise the reward remains 0. This setting turns explored ScienceWorld observations into verifiable state-reaching objectives.

B STATE2STATE Datasets

This section describes the construction of the STATE2STATE datasets used for environment-derived mid-training. Both datasets are generated from environment exploration rather than expert demonstrations or human-written task instructions. The resulting examples specify a reproducible environment configuration and a target observation, and success is verified by matching the current observation to the target observation.

B.1 ScienceWorld Dataset

Exploration. The ScienceWorld STATE2STATE dataset is built from random exploration trajectories generated with 2560 exploration episodes. The exploration seed is 42, and each episode allows up to 300 environment steps.

The exploration policy samples from environment actions after several validity and compatibility filters. Invalid actions, non-actionable objects, and semantically incompatible action-object pairs are removed before sampling. Because ScienceWorld has comparatively large maps, exploration uses an explore-then-operate two-phase strategy: a spatial-exploration phase ending at 20% of the episode, and during the rest 80% steps we encourage the scientific and state-changing actions.

Processing and filtering. The dataset construction process loads all exploration trajectories and collects observations from each trajectory's set of unique observations. For each observation, the pipeline records provenance information including source trajectory, source step, task name, task variation, seed, and exploration strategy. It then removes invalid or uninformative observations using a keyword blacklist and a short-text heuristic for error-like messages.

Sampling. ScienceWorld target selection follows a diversity-first procedure. The first phase ensures that each unique valid observation appears at least once. If more tasks are needed, later phases add additional instances subject to a maximum observation frequency of 5. The target number is 8,000 tasks. Each exported task contains a task identifier, the original ScienceWorld task name and variation, a `task_target_state` field, and metadata describing the source trajectory and step.

Statistics. The final ScienceWorld dataset contains 8,000 tasks, split into 7,872 training tasks and 128 validation tasks. All 8,000 target observations are unique, giving a diversity ratio of 1.0. The largest task-name groups are `test-conductivity` (2,900), `inclined-plane-friction-named-surfaces` (1,630), `find-living-thing` (674), `find-non-living-thing` (574), and `measure-melting-point-known-substance` (540). Target observations have median length 72 characters, 90th percentile length 500, 95th percentile length 811, and maximum length 1,276. The source-step distribution has median 116, 90th percentile 185, 99th percentile 199, and maximum 200.

B.2 ALFWorld Dataset

Exploration. The ALFWorld STATE2STATE dataset is built from random exploration over the ALFWorld training task list. The production run uses seed 42, 256 parallel environments and 20 episode batches, yielding 5,120 trajectories. Each trajectory allows up to 500 steps. The random explorer samples from the environment-provided admissible commands. Since the distribution of each action verb can be very uneven, we adopt a verb-weighted sampling mechanism: commands

are grouped by inferred verb, assigned per-verb weights, and normalized by the number of admissible commands for that verb.

Processing and filtering. Candidate target states are collected from all trajectory steps whose index is at least 10; no upper step bound is applied. Observations are filtered with substring bans for welcome banners and error-like feedback, including messages such as unknown actions, invalid actions, and impossible commands. For each candidate, the selection script infers a verb from observation text using a regular expression matching phrases of the form `You <verb>`.

Deduplication and sampling. To encourage the state-changing actions, we use different deduplication scopes for different verb categories in ALFWorld. For state-changing verbs like `clean`, `cool` and `heat`, duplicate observations are removed only within the same trajectory, preserving similar outcomes reached in different initial games. For all other verbs, duplicate observations are removed globally across trajectories. The final dataset is sampled without replacement from the candidate pool using verb-weighted sampling, with preference towards state-changing actions like heat or cool. Each exported task contains a task identifier, a source game file for environment initialization, and a `task_target_state` field.

Statistics. The final ALFWorld dataset contains 4,256 tasks, split into 4,000 training tasks and 256 validation tasks. The final verb distribution is: `arrive` (733), `clean` (435), `cool` (423), `move` (423), `turn` (396), `pick` (381), `open` (361), `sliced` (320), `see` (280), `are` (206), `heat` (204), `other` (51), and `close` (43). Target observations are shorter than those in ScienceWorld: the median length is 44 characters, the 90th percentile is 104, the 95th percentile is 135, and the maximum is 360.

B.3 Representative Cases

The following examples show representative STATE2STATE targets sampled from the case files in `resources/appendix_2`. They are chosen to cover different task families, observation types, and action effects.

ScienceWorld Case 1: `boil`; room inventory

```
This room is called the greenhouse.
In it, you see:
the agent
a substance called air
a bee hive. The bee hive door is open. In
      the bee hive is: nothing.
a sink, which is turned on. In the sink
     is: the ground, a hole (containing a
     substance called water).
soil
You also see:
A door to the hallway (that is open)
A door to the outside (that is open)
```

ScienceWorld Case 2: `power-component`; inventory

```
In your inventory, you see:
a blue light bulb, which is off.
its anode is connected to: a terminal 1
     on agent.
its cathode is connected to: a terminal 2
      on agent.
a green light bulb, which is off.
its anode is connected to: a terminal 1
     on door.
its cathode is connected to: a terminal 1
      on drain.
a substance called wood
```

ScienceWorld Case 3: `find-animal`; action feedback

```
You dunk the flower pot 9 in the flower
     pot 5, moving the liquids (water,
     water, water, water, water) to the
     flower pot 9.
```

ALFWorld Case 1: `arrive`; navigation/listing

```
You arrive at countertop 1.
On the countertop 1,
you see a bowl 2, a bread 3,
a bread 2, a bread 1,
a fork 1, a houseplant 1,
a kettle 2, a knife 2, a knife 1.
```

ALFWorld Case 2: `open`; container state

```
You open the fridge 1.
The fridge 1 is open.
In it, you see a apple 2, a bread 1,
a egg 2, a egg 1, a mug 1,
a plate 2, a potato 1,
a tomato 2, and a tomato 1.
```

Dataset	Tasks	Train	Val	Traj.	Median Len.	Max Len.
ScienceWorld	8,000	7,872	128	2,560	72	1,276
ALFWorld	4,256	4,000	256	5,120	44	360

Table 7: Summary statistics for the STATE2STATE datasets.

ALFWorld Case 3: heat; state-changing action

```
You heat the potato 1
using the microwave 1.
```

C RL Algorithms

This section briefly summarizes the RL algorithms used in our experiments. All of them are group-based policy optimization methods: for each task or target state, the policy samples multiple rollouts under the same initial condition, and learning is driven by relative performance within the group.

C.1 GRPO

Group Relative Policy Optimization (GRPO) (Shao et al., 2024) is the main RL backbone used in our STATE2STATE and downstream human-task training experiments. Compared with PPO, GRPO removes the learned value function and estimates the baseline from a group of sampled responses or trajectories for the same prompt. In our agent setting, for a task x, we sample G rollouts $\{\tau_i\}_{i=1}^{G}$ from the policy and obtain environment returns $\{R_i\}_{i=1}^{G}$. The group-relative advantage is computed by normalizing returns inside the group:

$$\hat{A}_i = \frac{R_i - \mathrm{mean}(\{R_j\}_{j=1}^{G})}{\mathrm{std}(\{R_j\}_{j=1}^{G})}.$$

The policy is then updated with a clipped importance-ratio objective, with the same rollout-level advantage assigned to the generated tokens/actions in that rollout. This critic-free design is well suited to sparse verifiable rewards, because successful and failed attempts on the same task provide a direct contrastive signal.

C.2 DAPO Dynamic Sampling

DAPO (Yu et al., 2025) introduces several modifications to GRPO for large-scale RL, and in this work we only use the dynamic sampling mechanism. The motivation is that a rollout group with identical outcomes gives no useful group-relative gradient: if all G rollouts fail or all G rollouts succeed, the group provides no meaningful relative comparison. Dynamic sampling therefore over-samples rollout groups and keeps only informative groups before an update. For binary verifiable rewards, this corresponds to retaining groups satisfying

$$0 < \sum_{i=1}^{G} \mathbf{1}[R_i > 0] < G.$$

In our implementation, STATE2STATE training repeatedly samples groups until enough non-identical outcome groups are collected, up to a maximum generation-batch budget. This increases the density of effective comparisons for sparse state-reaching rewards.

C.3 GiGPO

Group-in-Group Policy Optimization (GiGPO) (Feng et al., 2025) is designed for multi-turn LLM agent training, where a single terminal reward may be too coarse for assigning credit to intermediate actions. GiGPO preserves the episode-level grouping used by GRPO, but adds a second, step-level grouping mechanism. After collecting multiple trajectories for the same task, it identifies repeated environment states across trajectories as anchor states and groups the actions taken from the same anchor state. This allows the algorithm to compare locally different actions that were chosen under the same state condition, without requiring extra per-state rollouts.

GiGPO combines an episode-level advantage $A^E(\tau_i)$ with a step-level advantage $A^S(a_t^{(i)})$:

$$A(a_t^{(i)}) = A^E(\tau_i) + \omega A^S(a_t^{(i)}),$$

where ω controls the strength of step-level credit assignment. The resulting advantage is used in a clipped policy-gradient objective similar to GRPO. In this paper, GiGPO is used as an alternative downstream RL backbone in the analysis section to test whether STATE2STATE remains beneficial when the optimizer already provides more fine-grained agent credit assignment.

D Baselines

We include two trajectory-based baselines that use expert-style supervision rather than online RL from STATE2STATE objectives.

D.1 Distillation SFT

The Distillation SFT baseline collects demonstration trajectories by running a strong closed-source LLM Qwen-Plus in ALFWorld and ScienceWorld with the same environment interface used during training. At each step, the model receives the full prompt constructed by the environment manager, including the task description, current observation, recent interaction history, and admissible actions, and returns a response containing thoughts and an `<action>` span. We record the prompt and response for each active step, keep successful trajectories, and convert the resulting step-level pairs into supervised fine-tuning data format.

D.2 Agent Early Experience

Agent Early Experience (AEE) (Zhang et al., 2025a) augments expert trajectories with step-level self-reflection supervision. In our experiment, we adopt the self-reflection variant. For each expert step, AEE reconstructs the environment state by replaying the expert prefix, samples K alternative admissible actions excluding the expert action, and executes the expert and alternative actions in parallel environment copies. An LLM then observes the original situation, the expert action and next state, and an alternative action and next state, and writes a self-reflection explaining why the expert action is preferable. The final SFT example trains the model to output the reflection followed by the expert `<action>` tag. In our experiments, AEE serves as a stronger trajectory-based baseline that provides additional reasoning supervision beyond direct action imitation.

E Training Setups and Prompts

This section presents the key parameters and prompts used for training. In all settings, the agent is expected to reason and then emit an executable action inside `<action>` and `</action>` tags. Curly-braced fields are filled by the environment manager at rollout time. Our standard training prompts mainly follow the previous work (Feng et al., 2025) with slight modifications. The STATE2STATE training prompts are modified from the standard training prompts.

E.1 Key RL Training Hyperparameters

All RL training runs, including STATE2STATE mid-training and downstream standard-task training, share most optimization settings across model sizes and phases. We adopt verl (Sheng et al., 2024) open-source RL framework and vllm (Kwon et al., 2023) inference engine for high-throughput RL training. We optimize the actor with learning rate $1e^{-6}$, enable KL loss with coefficient 0.01, and do not add KL penalties directly into the environment reward. We use GRPO-style RL algorithm in both training phases. We set the train batch size to 16 and the GRPO group size is set to 8. We deploy the environment via ray framework and enable the vllm rollout server to interact with the environments in parallel. For validation rollouts, we use sampling with temperature 0.1 for stability, and for training rollouts, we set the temperature to 1.0 for balanced exploration and exploitation.

For ALFWorld training, we use a maximum prompt length of 4096 and response length of 1024. For ScienceWorld training, we use longer contexts, with maximum prompt length between 12000 and response length 1024. STATE2STATE runs use dynamic group filtering to continually collect informative rollout groups. We set the maximum number of generation batches to 4 for ALFWorld and ScienceWorld Qwen3-8B STATE2STATE training, and to 6 for ScienceWorld Qwen3-4B training. For the STATE2STATE training, we train the model for 80 steps maximum. And for standard task training, we train the model for 400 steps in ScienceWorld and 300 steps ALFWorld, determined by the convergence steps of the baseline training (directly RL on the environments without STATE2STATE). As for the max_steps in a single episode, we set max_steps to be 40 for ALFWorld and 30 for ScienceWorld.

E.2 Standard ALFWorld Training Prompt

Standard ALFWorld Training Prompt

```
You are an expert agent operating in the
    ALFRED Embodied Environment. Your
    task is to: {task_description}
Prior to this step, you have already
    taken {step_count} step(s). Below are
     the most recent {history_length}
    observations and the corresponding
    actions you took: {action_history}
You are now at step {current_step} and
    your current observation is: {
    current_observation}
```

```
Your admissible actions of the current
     situation are: [{admissible_actions
     }].

Now it's your turn to take an action.
You should first reason step-by-step
     about the current situation.
Once you've finished your reasoning, you
     should choose an admissible action
     for current step and present it
     within <action> </action> tags.
```

E.3 STATE2STATE Training Prompt

STATE2STATE Training Prompt

```
You are an expert agent operating in the
     ALFRED Embodied Environment.
Your goal is to reach the following
     TARGET STATE (the environment
     observation text must match it):
{target_state}
You need to perform actions to make the
     environment observation the same as
     the given target state. After your
     each action, the environment
     observation will be updated. Only if
     the environment observation is
     exactly the same as the target state,
      the task is considered successful.

Prior to this step, you have already
     taken {step_count} step(s). Below are
      the most recent {history_length}
     observations and the corresponding
     actions you took: {action_history}
You are now at step {current_step} and
     your current observation is: {
     current_observation}
Your admissible actions of the current
     situation are: [{admissible_actions
     }].

The target state is: {target_state}

Now it's your turn to take an action.
You should first reason step-by-step
     about the current situation. This
     reasoning process MUST be enclosed
     within <think> </think> tags.
Once you've finished your reasoning, you
     should choose an admissible action
     for current step and present it
     within <action> </action> tags.
```

E.4 Standard ScienceWorld Training Prompt

Standard ScienceWorld Training Prompt

```
You are an expert agent operating in the
     ScienceWorld environment, which is a
     text-based virtual environment
     centered around accomplishing tasks
     from the elementary science
     curriculum.

Your current task is: {task_description}

Prior to this step, you have already
     taken {step_count} step(s). Below are
      the most recent {history_length}
     observations and the corresponding
     actions you took: {action_history}
You are now at step {current_step} and
     your current observation is: {
     current_observation}

Here are the actions you may take:
[
{"action": "open OBJ", "description": "
     open a container"},
{"action": "close OBJ", "description": "
     close a container"},
{"action": "activate OBJ", "description":
      "activate a device"},
{"action": "deactivate OBJ", "description
     ": "deactivate a device"},
{"action": "connect OBJ to OBJ", "
     description": "connect electrical
     components"},
{"action": "disconnect OBJ to OBJ", "
     description": "disconnect electrical
     components"},
{"action": "use OBJ [on OBJ]", "
     description": "use a device/item"},
{"action": "look around", "description":
     "describe the current room"},
{"action": "look at OBJ", "description":
     "describe an object in detail"},
{"action": "look in OBJ", "description":
     "describe a container's contents"},
{"action": "read OBJ", "description": "
     read a note or book"},
{"action": "move OBJ to OBJ", "
     description": "move an object to a
     container"},
{"action": "pick up OBJ", "description":
     "move an object to the inventory"},
{"action": "put down OBJ", "description":
      "drop an inventory item"},
{"action": "pour OBJ into OBJ", "
     description": "pour a liquid into a
     container"},
{"action": "dunk OBJ into OBJ", "
     description": "dunk a container into
     a liquid"},
{"action": "mix OBJ", "description": "
     chemically mix a container"},
{"action": "go to LOC", "description": "
     move to a new location"},
{"action": "eat OBJ", "description": "eat
      a food"},
{"action": "flush OBJ", "description": "
     flush a toilet"},
{"action": "focus on OBJ", "description":
      "signal intent on a task object"},
{"action": "wait", "description": "take
     no action for 10 iterations"},
{"action": "wait1", "description": "take
     no action for 1 iteration"},
{"action": "task", "description": "
     describe current task"},
```

```
{"action": "inventory", "description": "
    list your inventory"}
]

Current available actions:
{available_actions}

Tips:
1. You can use look around to see the
    surroundings of the current location.
2. All containers in the environment have
     already been opened, you can
    directly get items from the
    containers.
3. You can explore the environment and
    find the items you need to complete
    the experiment.
**4. Your thinking process should be
    concise, do not waste time on
    irrelevant details.**

Your current task is: {task_description}

You are now at step {current_step} and
    your current observation is: {
    current_observation}

Now it's your turn to take an action. You
     should first reason step-by-step
    about the current situation.
Once you've finished your reasoning, you
    should choose an appropriate action
    for the current step and present it
    within <action> </action> tags.
```

E.5 STATE2STATE Training Prompt

STATE2STATE Training Prompt

```
You are an expert agent operating in the
    ScienceWorld environment, which is a
    text-based virtual environment
    centered around accomplishing tasks
    from the elementary science
    curriculum.
Your task is to reach a target state(the
    environment observation). If the
    target state is reached, the task is
    considered successful. If the task is
     not completed within a certain
    number of steps, the task is
    considered failed.

The target state is: {target_state}

You need to perform actions to make the
    environment observation the same as
    the given target state. After your
    each action, the environment
    observation will be updated. Only if
    the environment observation is
    exactly the same as the target state,
     the task is considered successful.

Prior to this step, you have already
    taken {step_count} step(s). Below are
     the most recent {history_length}
    observations and the corresponding
    actions you took: {action_history}
You are now at step {current_step} and
    your current observation is: {
    current_observation}

Here are the actions you may take:
[
{"action": "open OBJ", "description": "
    open a container"},
{"action": "close OBJ", "description": "
    close a container"},
{"action": "activate OBJ", "description":
     "activate a device"},
{"action": "deactivate OBJ", "description
    ": "deactivate a device"},
{"action": "connect OBJ to OBJ", "
    description": "connect electrical
    components"},
{"action": "disconnect OBJ to OBJ", "
    description": "disconnect electrical
    components"},
{"action": "use OBJ [on OBJ]", "
    description": "use a device/item"},
{"action": "look around", "description":
    "describe the current room"},
{"action": "look at OBJ", "description":
    "describe an object in detail"},
{"action": "look in OBJ", "description":
    "describe a container's contents"},
{"action": "read OBJ", "description": "
    read a note or book"},
{"action": "move OBJ to OBJ", "
    description": "move an object to a
    container"},
{"action": "pick up OBJ", "description":
    "move an object to the inventory"},
{"action": "put down OBJ", "description":
     "drop an inventory item"},
{"action": "pour OBJ into OBJ", "
    description": "pour a liquid into a
    container"},
{"action": "dunk OBJ into OBJ", "
    description": "dunk a container into
    a liquid"},
{"action": "mix OBJ", "description": "
    chemically mix a container"},
{"action": "go to LOC", "description": "
    move to a new location"},
{"action": "eat OBJ", "description": "eat
     a food"},
{"action": "flush OBJ", "description": "
    flush a toilet"},
{"action": "focus on OBJ", "description":
     "signal intent on a task object"},
{"action": "wait", "description": "take
    no action for 10 iterations"},
{"action": "wait1", "description": "take
    no action for 1 iteration"},
{"action": "inventory", "description": "
    list your inventory"}
]

Current available actions:
{available_actions}

Tips:
```

```
1. You can use look around to see the
    surroundings of the current location.
2. All containers in the environment have
     already been opened, you can
    directly get items from the
    containers.
3. You can explore the environment and
    find the items you need to complete
    the experiment.
**4. Your thinking process should be
    concise, do not waste time on
    irrelevant details.**

The target state is: {target_state}

Now it's your turn to take an action. You
     should first reason step-by-step
    about the current situation(Your
    thinking process should be concise
    and focus on the key information).
    After you've finished your reasoning,
     you should choose an appropriate
    action for the current step and
    present it within <action> </action>
    tags.
```

F MobileWorld Experiment

This section provides additional details for the MobileWorld experiment reported in Table 6.

F.1 Environment and Evaluation

MobileWorld (Kong et al., 2025) is a mobile GUI benchmark that requires agents to interact with Android applications through visual observations and executable UI actions. Compared with the text-only environments used in our main experiments, MobileWorld introduces richer visual observations, cross-application interaction patterns, and longer-horizon GUI navigation. For our evaluation, we use the GUI-only subset, comprising 117 instances which focus exclusively on GUI operation and omit user interaction and MCP tasks.

We use MAI-UI-8B (Kong et al., 2025) as the base model due to its competent GUI operation capability among open-source models, which provides a good foundation for conducting STATE2STATE training. Given that MobileWorld contains only evaluation tasks and lacks a separate training set, while constructing additional human-annotated training tasks with reliable verifiers would introduce substantial manual costs, we restrict our methodology to STATE2STATE training in this environment rather than performing downstream RL on human-specified tasks. This setup effectively positions MobileWorld as a stress test to evaluate whether STATE2STATE can yield robust environment-learning signals without relying on human-authored training tasks.

F.2 STATE2STATE Data

We adopt a rule-based exploration strategy within MobileWorld. At each step, the strategy randomly selects a clickable XML element, effectively executing a randomized depth-first search (DFS) through the application states. Each collected state comprises a target screenshot and its corresponding UI hierarchy XML, which jointly serve as the reward signal for STATE2STATE training. In total, we collected 365 target states across 14 applications. These were partitioned into a training set of 313 states and a validation set of 52 states, providing a robust dataset for learning from the MobileWorld environment.

F.3 Reward and Training Setup

During the training phase, the model receives a target screenshot within its task instruction. At each subsequent interaction step, it operates purely on visual inputs, observing only the current screen image. The underlying UI hierarchy XML remains strictly hidden from the policy and is reserved solely for reward computation. In our training, both target screenshots and online screen observations are resized to 540×1200 to keep the visual context length manageable during multi-turn rollouts. In evaluation, we use the original image size for standard evaluation settings.

To determine whether the final GUI state matches the target state, the STATE2STATE reward combines structural and visual similarity signals. For the structural component, we normalize the XML by removing position- and instance-specific attributes, converting it into a compact semantic sequence composed of resource identifiers, text, content descriptions, and boolean states (e.g., selected or checked). We then calculate a fuzzy sequence similarity score between the current and target XML sequences. For the visual component, we directly compute SSIM-based similarity score between two screenshots. The final reward is computed as a weighted combination, heavily favoring the structural signal with an XML similarity weight of 0.8.

We optimize the policy with GRPO, sampling 6 rollouts per target state with a base batch size of 4 target states. This process is distributed across 24 parallel rollout executions due to computational resource constraints. To enhance training stability,

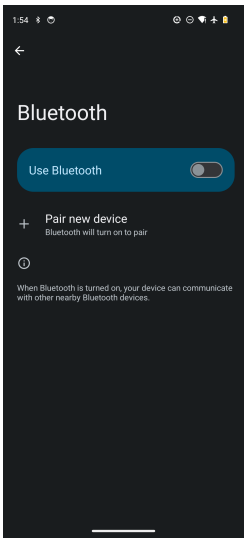
we accumulate gradients over two steps, yielding an effective training batch size of 8 target states. Additionally, we cap the maximum rollout length at 15 interaction steps to prevent excessive sequence lengths and mitigate memory overhead during training. We set the actor learning rate to 10^{-6}, the KL loss coefficient to 0.001, and rollout temperature to 1.0 with top-$p = 0.9$. The reported STATE2STATE model checkpoint is trained for 80 update steps and evaluated on downstream tasks directly, without an additional MobileWorld human-task RL stage.

F.4 Representative Target States

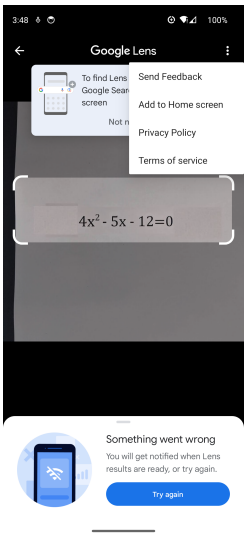
Figure 4 shows four representative MobileWorld STATE2STATE targets. These examples illustrate the type of state-reaching objective used during training: the agent is not given a human-written task instruction, but must interact with the phone until the current screen matches the target state. The targets include system settings, camera-based search, shopping, and file-management states, covering both built-in Android interfaces and application-specific UI modes. Together, they show that the environment-derived targets span diverse visual layouts, interaction contexts, and application domains.

G The Use of Large Language Models (LLMs)

Large Language Models (LLMs) were used only for language polishing, including grammar checking and improving word- and sentence-level readability. All substantive aspects of the paper, including the research ideas, method design, experiments, analysis, and conclusions, were developed and conducted by the authors.

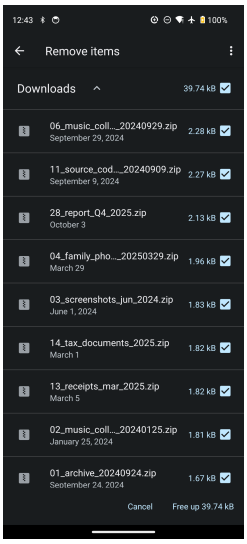


(a) Bluetooth settings.



(b) Google Lens.

(c) Shopping product page.



(d) File cleanup selection.

Figure 4: Representative MobileWorld STATE2STATE target states used for environment-derived training. The model observes a target screenshot and learns to reach a matching GUI state through multi-turn interaction. The examples cover system settings, visual search, shopping, and file-management interfaces.