ZEROHERO: TRAINING SMALL LANGUAGE AGENTS FROM SCRATCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Large pretrained language models provide capable policies for interactive tasks, but how much model capacity is necessary for a specialized language agent? We introduce ZeroHero, a framework for training small language agents from scratch through expert-trajectory pretraining and adaptive post-training. A central challenge is deciding how to allocate training between expert supervision and the agent's own experience as its competence changes. We introduce Bandit-based Adaptive Training Selection (BATS), which uses a contextual bandit to select blocks of DAgger-style imitation learning or PPO updates based on measured learning progress. We study this approach under Reinforcement Learning with Oracle Dialogue (RLOD), where an agent combines environment actions with natural-language questions to an oracle that provides otherwise hidden task information. We construct dialogue-augmented ALFWorld household tasks and Craftax navigation tasks, and train 5–10M-parameter executors from random initialization, first on successful expert trajectories and then through online interaction. The expert supplies training supervision, while an LLM oracle remains available during execution. On ALFWorld, BATS achieves 89.6% success, compared with 67.9% for the expert, and outperforms fixed PPO, DAgger, and mixed-loss training from the same starting checkpoint. BATS also supports recovery from weak starting policies where pure PPO collapses. Across environments and starting policies, it selects different training strategies, ranging from predominantly reinforcement learning to sustained expert-guided imitation. These results show that compact language agents can surpass their demonstration expert and highlight the importance of adapting training regimes to the task and the learner's competence.

1 INTRODUCTION

Large language models enable interactive agents that can interpret instructions, generate actions, and acquire additional information from the environment (Yao et al., 2023; Yang et al., 2024). This paradigm is particularly natural for tasks in which decision making depends on communication: in embodied dialogue, for example, an agent must not only act in the environment but also ask questions to resolve missing or ambiguous task information (Padmakumar et al., 2022; Gao et al., 2022; Kiseleva et al., 2022). Pretrained LLMs are also appealing policy backbones because they acquire broad knowledge about the world during pretraining, and such priors can substantially facilitate subsequent RL training (Shi et al., 2026). However, this advantage is not universal. RL fine-tuning of large models faces challenges including long-horizon credit assignment, sparse rewards, and exploration (Wang et al., 2025b; Feng et al., 2025), while in some tasks small specialized policies can outperform much larger general-purpose models. This raises a question about the necessary scale of the executor: must an effective language-based agent for a specific class of tasks rely on a large general-purpose pretrained model?

We introduce ZeroHero,[1] a framework for training small language agents from random initialization. ZeroHero first pretrains an executor on successful trajectories collected by a large expert model, then improves it through interaction with the environment using expert supervision and reinforcement learning. This direction is supported by results on task-specific SLMs, where specialized training

[1] Code: `https://anonymous.4open.science/r/zerohero_iclr2027-DDE1/`.

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.

allows compact models to compete with or even outperform much larger general-purpose models on the target task (Hsieh et al., 2023; Erdogan et al., 2024). However, it is not obvious how an SLM should be trained from scratch when two distinct sources of supervision are available: expert demonstrations and the model's own experience in the environment. Early in training, expert demonstrations may be particularly valuable because trajectories generated by a weak policy provide little informative feedback; as competence increases, RL instead allows the model to optimize directly for task outcomes and depart from expert behavior. A fixed schedule cannot account for changes in the relative usefulness of these signals during training. To make this choice adaptive, ZeroHero uses *Bandit-based Adaptive Training Selection* (BATS), which treats DAgger-style imitation learning and PPO as two update regimes for a shared policy and uses a contextual bandit to select between them over blocks of training.

We study this problem in tasks where decision making depends on acquiring missing information through language, and formalize the setting as *Reinforcement Learning with Oracle Dialogue* (RLOD). In addition to taking environment actions, the executor can ask questions to an oracle and must learn when information is missing, what to ask, and how to use the resulting answer in subsequent decisions. For our experiments, we construct two dialogue-augmented environments based on existing embodied benchmarks: ALFWorld-Dialogue-Standard, where part of a household-task specification is hidden and can be clarified through the oracle, and Craftax-Dialogue, where information about the navigation target is hidden and must be acquired through questions during interaction. In both environments, the oracle is implemented as an LLM with access to the hidden environment state and can answer the agent's questions in free-form natural language.

Our main contributions are:

- We formalize *Reinforcement Learning with Oracle Dialogue* (RLOD) and introduce two embodied environments for studying it, in which an agent can acquire hidden task information through free-form dialogue with an LLM oracle.
- We introduce ZeroHero, a framework that combines expert-trajectory pretraining with adaptive post-training of small language agents, and BATS, its contextual-bandit mechanism for selecting between imitation and reinforcement learning.
- We show that an SLM trained with our approach can outperform both the expert LLM that supplies demonstrations and a larger LLM directly fine-tuned with RL.
- We show that effective SLM training strategies vary across environments and initial model competence: the relative value of expert supervision and RL changes across tasks and stages of training, suggesting that there is no single fixed recipe for combining them.

2 Related Work

Small language model training from scratch. Recent work shows that compact language models can achieve strong task-specific performance through better use of data, architecture, and supervision (Eldan & Li, 2023; Gunasekar et al., 2023; Zhang et al., 2024; Hu et al., 2024; Liu et al., 2024; Mehta et al., 2024). Larger LLMs can also serve as teachers: Hsieh et al. (2023) distill teacher-generated rationales into smaller models, while He et al. (2026) study how expert supervision and RL can be combined across different stages of SLM training. We extend this direction to interactive decision making, asking whether a randomly initialized SLM can learn both action and dialogue behavior and eventually surpass its expert.

RL fine-tuning of LLMs and VLMs. Pretrained LLMs and VLMs provide useful priors for decision making, motivating their use as policy backbones in multi-turn environments (Tan et al., 2024; Wang et al., 2025b;a). However, RL fine-tuning remains difficult due to long-horizon credit assignment, sparse rewards, exploration, and training instability (Wang et al., 2025b; 2026). GiGPO specifically addresses credit assignment by introducing relative advantages at both the episode and step levels for multi-turn LLM agents (Feng et al., 2025). Moreover, the benefit of pretraining is task-dependent: Odysseus reports improved sample efficiency from VLM pretraining (Shi et al., 2026), while specialized RL methods can outperform LLM-based policies in low-level control tasks (Hatgis-Kessell & Brunskill, 2026). We instead use a large model as an expert and train a smaller policy specialized for the target environment.

Embodied dialogue systems and environments. Embodied language environments combine language with environment interaction: ALFWorld links text interaction to embodied household tasks (Shridhar et al., 2021), while TEACh, DialFRED, and IGLU study task completion through dialogue, clarification, or human guidance (Padmakumar et al., 2022; Gao et al., 2022; Kiseleva et al., 2022). Related systems such as MindAgent and LEGENT further explore multimodal and language-mediated coordination in embodied settings (Cheng et al., 2024). Our setting separates a compact executor from a privileged LLM oracle with access to hidden environment state, allowing the executor to learn not only how to act, but also when and what to ask.

Imitation learning and adaptive curricula. Our training setup also connects to imitation learning and adaptive training. DAgger addresses distribution shift under learned policies (Ross et al., 2011), DQfD combines demonstrations with reward-based learning (Hester et al., 2018), and learning-progress bandits have been used to adapt training curricula (Graves et al., 2017). BATS applies this principle to the choice between complete DAgger-style imitation and PPO updates for the same language policy, using a contextual bandit based on LinUCB (Li et al., 2010). Rather than defining a fixed schedule between expert supervision and reinforcement learning, BATS adapts this choice during training.

3 METHOD

ZeroHero consists of two training stages: supervised pretraining on successful expert trajectories and adaptive post-training through interaction with the environment. During post-training, BATS selects between DAgger-style imitation and PPO updates using measured learning progress. We evaluate ZeroHero in the RLOD setting, where the executor can both act and ask questions. The expert provides training supervision, while the oracle answers questions during execution.

3.1 ACTING WITH ORACLE DIALOGUE

We formalize dialogue-based embodied tasks as *Reinforcement Learning with Oracle Dialogue* (Figure 1). Let s_t include the physical state and task requirements, and let h_t contain the agent's observations, decisions, and received replies. The agent produces a physical action $a_t \in \mathcal{A}$ or a question $q_t \in \mathcal{Q}$:

$$u_t \sim \pi_\theta(\cdot \mid h_t), \qquad u_t \in \mathcal{A} \sqcup \mathcal{Q}, \qquad \text{ans}_t \sim \Omega(\cdot \mid s_t, q_t, h_t). \tag{1}$$

Here Ω is the environment's oracle. A physical action changes the world; a question requests information and adds the oracle's reply to the history. Both consume an interaction step. The policy

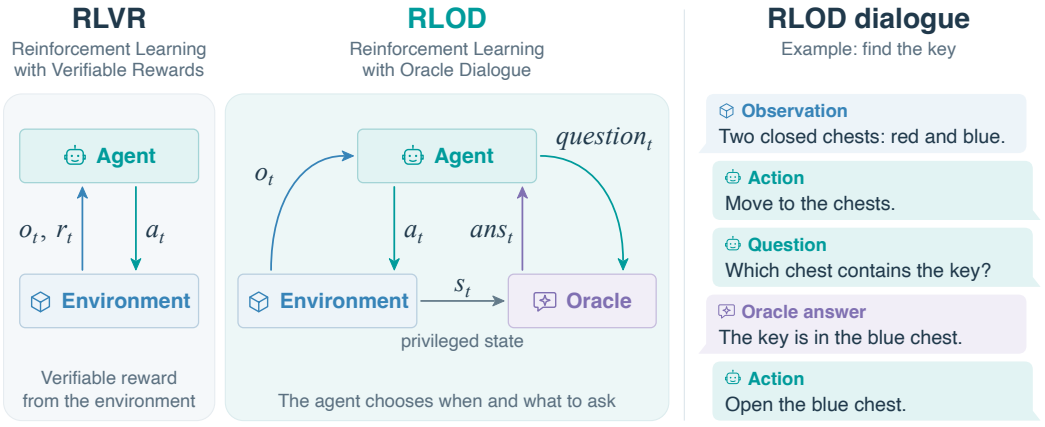


Figure 1: **Acting and asking in RLOD.** The executor receives observations and dialogue replies. The oracle is part of the environment and can access information hidden from the agent. Questions affect subsequent decisions through the information returned. The dialogue is schematic.

maximizes $J(\theta;\Omega) = \mathbb{E}[\sum_{t=0}^{H-1} \gamma^t r_t]$. A question can therefore be useful even when it has no immediate reward: its answer can improve a later physical decision.

The oracle specifies how hidden information becomes available through language. Its knowledge, answer format, and response policy are configurable; it can be implemented by rules or a language model. We use language-model oracles conditioned on structured task facts. The agent receives their answers rather than the underlying private state. Oracle access does not guarantee a correct answer, so success is measured by the environment's completion predicate for the complete interaction. Prompts and parsing rules appear in Appendix E.

3.2 Initialization from Expert Trajectories

The expert generates actions and questions from observations and interaction history through the learner's environment interface, without access to hidden requirements or privileged solutions. Recorded trajectories contain observations, expert responses, oracle replies, and outcomes. The expert supplies demonstrations and later DAgger corrections; the oracle answers questions during execution, even when both roles use the same underlying model.

We initialize the executor from random weights with a task-specific tokenizer, minimizing response-token cross-entropy on successful demonstrations. Prompts, observations, oracle replies, and padding are excluded from the loss. Checkpoints from different pretraining stages let us study post-training across initial competence levels.

3.3 Bandit-based Adaptive Training Selection

BATS chooses how to update a pretrained policy. Its two arms are DAgger-style imitation and PPO. Each selected arm runs for a block of two training cycles; the controller then measures learning progress and selects the next block. The same dialogue oracle remains available under either arm.

The two operators use different sources of supervision. DAgger fits expert corrections at learner-visited histories, mixed with replay and pretraining examples (Ross et al., 2011). PPO fits fresh learner experience to environment returns (Schulman et al., 2017). Separate optimizer states persist

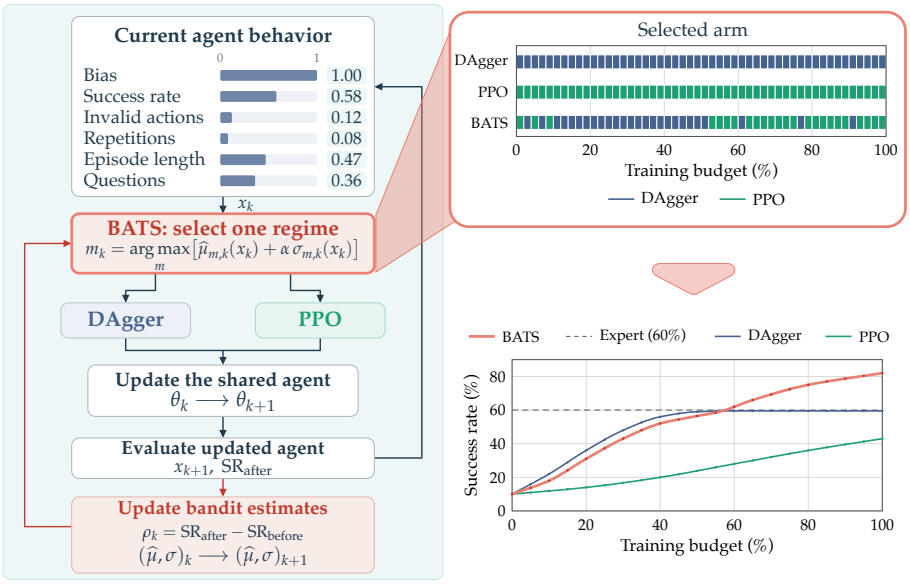


Figure 2: **Adaptive training selection.** Behavioral context conditions predicted progress for the PPO and DAgger update operators. A controller selects a block and observes the resulting change in success. All feature values, curves, arm choices, and the expert line in this schematic are hand-specified illustrations, not experimental measurements.

across switches. Section 4 compares these operators with a joint-loss baseline; implementation details and PPO hyperparameters are in Appendix A.

To estimate which operator is useful for the current policy, BATS evaluates a fixed controller set before block k and forms $x_k = [1, \mathrm{SR}, f_{\mathrm{invalid}}, f_{\mathrm{repeat}}, \overline{T}/H, \overline{Q}/Q_{\mathrm{norm}}]^\top$. The features summarize success, invalid and immediately repeated decisions, episode length, and question count; normalization is environment-specific. The selected arm receives reward

$$\rho_k = J_{\mathrm{ctrl}}(\theta_{k+1}) - J_{\mathrm{ctrl}}(\theta_k), \tag{2}$$

where both measurements use the same controller tasks and fixed sampling seeds. Negative changes are retained. Controller tasks are separate from rollout and test tasks; Test SR never enters routing. The post-block measurement is reused as the next pre-block measurement.

BATS uses these contexts and observed gains in ordinary disjoint LinUCB (Li et al., 2010). Each arm m maintains $A_m = \lambda I$ and $b_m = 0$ initially. After one forced block of PPO and one of DAgger, it selects

$$m_k = \arg\max_m \left[x_k^\top A_m^{-1} b_m + \alpha \sqrt{x_k^\top A_m^{-1} x_k} \right], \tag{3}$$

then updates only the selected arm: $A_{m_k} \leftarrow A_{m_k} + x_k x_k^\top$ and $b_{m_k} \leftarrow b_{m_k} + x_k \rho_k$. The predicted gain encourages productive updates; the bonus encourages revisiting less-observed regimes. This is a heuristic controller for a changing learner, not a stationary bandit with a regret guarantee. In particular, an unselected arm's counterfactual gain is unobserved, and short-term progress need not predict the best long-term schedule. Algorithm 1 gives the full procedure, and Figure 2 gives an illustrative schematic.

BATS requires update operators, policy statistics, and a progress signal; its application beyond dialogue agents remains to be evaluated.

4 Experiments

4.1 Experimental Setup

We evaluate the method in two environments with different information requirements. In ALFWorld-Dialogue, we retain ALFWorld's six household task families and physical success predicates (Shridhar et al., 2021): placement, examination in light, cleaning, heating, cooling, and two-object placement. Instead of receiving the complete goal, the agent receives a request with missing requirements, such as the destination, treatment, or quantity. The omitted fields are fixed for each task, allowing policies to be compared on the same requests. The agent can issue a physical command or a free-text `ask` question. An oracle backed by Qwen3-4B answers from a private task/state FAQ. The original full goal is absent from the agent's observations and history.

In Craftax-Dialogue, we construct hidden-target navigation tasks in Craftax-derived worlds (Matthews et al., 2024). The agent observes a symbolic local map and recent dialogue and can move in four directions. Its target is hidden, so it must obtain guidance through questions. A Gemma-backed oracle receives the target and pathfinder annotations and supplies landmark-based directions. The learner must translate this guidance into movements from its own observation. This task studies navigation with dialogue rather than Craftax's original survival objective. Detailed environment descriptions, oracle reliability checks, and recorded examples appear in Appendix F. For both environments, Gemma-4-26B-A4B-it collects the expert trajectories and supplies subsequent DAgger targets. After screening targets and oracle responses, the pretraining corpus contains 2,754 successful ALFWorld trajectories (34,355 response examples) and 2,731 Craftax trajectories (166,677 examples). Evaluation tasks are excluded. Validation holds out distinct ALFWorld games and complete Craftax episodes. ALFWorld targets are action commands and questions; Craftax targets retain reasoning with the action or question. We learn a task-specific byte-level tokenizer; neither the tokenizer nor the executor weights come from a released language model. Appendix B gives the splits and training configurations.

We compare *Classic* (GPT-2-style) and *Modern* (Qwen3-style) decoder-only transformers in 5M, 10M, and 50M parameter classes, all trained from scratch. Within each environment, models share the tokenizer, data split, and training recipe, while retaining their native architecture settings. Architectural details are provided in Appendix B.

For post-training, we use the Modern transformer in the 5M size class for ALFWorld and the 10M size class for Craftax. Weak, medium, and strong checkpoints represent different stages of pretraining. Methods within a group start from the same weights and tokenizer, so comparisons concern their post-training updates. Checkpoint identities and rounded model sizes are given in Appendix B.

We compare BATS with fixed PPO and DAgger regimes and a Mixed loss baseline. PPO learns from fresh trajectories of the current policy, while DAgger fits expert corrections together with replay and pretraining examples. Mixed loss combines the PPO and imitation objectives within one optimizer step. BATS selects between the complete PPO and DAgger update procedures described in Section 3.3. PPO hyperparameters and the corresponding DAgger and Mixed settings appear in Appendix A.

Success rate (SR) measures task completion according to the environment predicate. We distinguish Training SR on update rollouts, Controller SR for BATS routing, Validation SR for learning curves, and Test SR for held-out assessment. Only Controller SR enters arm selection. Test sets contain 134 ALFWorld tasks across six families and 512 Craftax worlds, with matching tasks and hidden goals across methods. Table 1 reports Test SR at the prescribed end of training as mean and standard deviation over three seeds for each small-model environment–initialization–method combination. Decoding settings and interaction budgets appear in Appendix C.

The following results address three research questions. **RQ1:** What task competence can small language models acquire from expert interaction trajectories alone? **RQ2:** When does adaptive selection between imitation and reinforcement learning improve post-training over fixed update regimes? **RQ3:** How do these regimes change the agent's use of dialogue and its task-solving behavior? We address RQ1 and RQ2 through pretraining and post-training comparisons, then examine training-regime selection and RQ3 in Section 5.

4.2 RQ1: Learning from Expert Trajectories

Our first question is what task competence expert-trajectory pretraining alone provides. We evaluate both the fit to held-out demonstrations and success during interaction, since predicting expert responses and solving a task under the model's own policy need not improve together.

Modern transformers generally achieve lower validation loss under the shared training recipe, with a clearer advantage in ALFWorld task success (Figure 3): at 5M and epoch six, Modern reaches 55.0% SR versus Classic's 26.7%. Craftax has a less uniform ordering: Classic leads at the smallest size by the end of the first epoch, while Modern leads at the largest. The evidence therefore favors modern architectural choices for these recipes without establishing a universal architecture ranking.

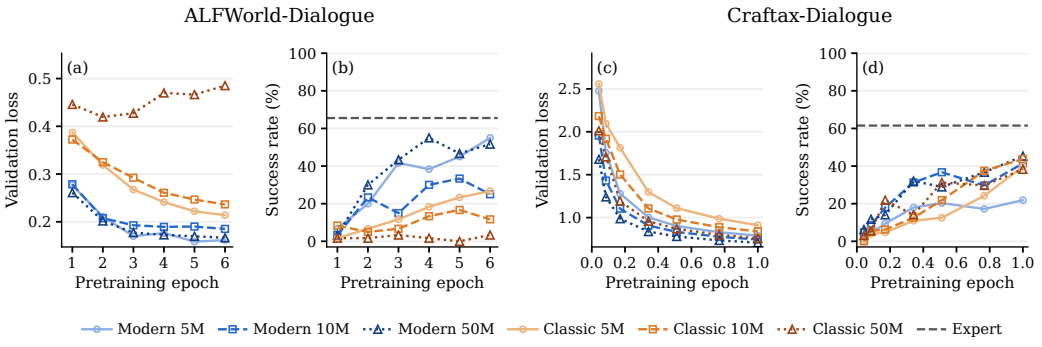


Figure 3: **Pretraining from scratch.** Validation loss and task success for Classic and Modern transformer language models in nominal 5M, 10M, and 50M size classes. ALFWorld spans several epochs; Craftax resolves the first epoch. Dashed lines show expert references from data collection or a separate expert evaluation, rather than paired checkpoint comparisons.

Increasing model size at a fixed data budget does not reliably improve interaction quality. Larger models can fit demonstrations without better generalization; overfitting is one possible mechanism, while optimization instability can also raise validation loss. Across the measured checkpoints, pretraining does not consistently exceed the expert references, despite using successful trajectories for learning. Those references are not strict upper bounds on imitation performance. Pretraining supplies checkpoints at different competence levels for studying how imitation and reinforcement should be allocated during post-training.

4.3 RQ2: Adaptive Post-training

To answer RQ2, we compare adaptive selection with fixed update regimes from each starting policy (Table 1). On ALFWorld, PPO fails to recover from the weak checkpoint, reaching 0% Test SR, whereas DAgger provides a viable learning signal and reaches 64.7%. BATS reaches 69.9% from the same initialization. Its advantage is larger from the strong checkpoint, where it achieves $89.6 \pm 2.1\%$, compared with PPO's $74.4 \pm 5.2\%$, DAgger's $73.9 \pm 1.1\%$, and the expert's 67.9%. However, DAgger performs best from the medium checkpoint, so adaptive selection does not improve over a fixed regime in every setting.

Combining both objectives within an optimizer step also does not reproduce the effect of choosing between update procedures. Mixed loss is weaker than BATS under the evaluated ALFWorld recipes, particularly from a weak start. The distinction includes optimization budget: Mixed loss shares an optimizer step between PPO and imitation, whereas a DAgger block performs separate

Table 1: **Test SR (%).** All results report test success rates on the ALFWorld-Dialogue and Craftax-Dialogue environments. Small-language-model results are mean ± standard deviation across three random seeds for each environment, initialization, and method. RL-finetuned LLM agents are matched in GPU compute budget.

Model / init.	Method	ALFWorld-Dialogue							Craftax
		Pick	Look	Clean	Heat	Cool	Pick2	All	SR
Pretrained LLM Agents									
Qwen3-4B	Zero-shot	20.8	11.1	9.7	8.7	19.0	0.0	11.9	10.9
Qwen3-8B	Zero-shot	62.5	44.4	32.3	30.4	42.9	0.0	36.6	25.8
Gemma 4 26B-A4B	Expert	83.3	83.3	93.5	39.1	76.2	11.8	67.9	61.5
RL-Finetuned LLM Agents									
Qwen3-4B	GiGPO	100.0	80.0	91.7	100.0	83.3	72.7	88.3	62.5
Small Language Model Agents (trained from scratch)									
Weak	PPO	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	8.3 ±1.2
	DAgger	**54.2** ±23.2	**72.2** ±16.7	77.4 ±3.2	73.9 ±15.1	81.0 ±8.2	**15.7** ±3.4	64.7 ±6.1	**65.1** ±1.6
	Mixed loss	4.2 ±0.0	27.8 ±0.0	12.9 ±0.0	13.0 ±14.4	14.3 ±0.0	0.0 ±0.0	11.9 ±1.9	8.3 ±1.2
	BATS (Ours)	51.4 ±6.4	70.4 ±8.5	**90.3** ±0.0	**91.3** ±7.5	**93.7** ±11.0	0.0 ±0.0	**69.9** ±4.4	60.9 ±1.4
Medium	PPO	55.6 ±2.4	57.4 ±11.6	82.8 ±9.3	87.0 ±4.3	90.5 ±4.8	3.9 ±3.4	66.4 ±3.9	36.5 ±7.3
	DAgger	**75.0** ±0.0	**75.0** ±3.9	85.5 ±2.3	84.8 ±9.2	88.1 ±3.4	**26.5** ±4.2	**75.0** ±2.6	64.6 ±1.2
	Mixed loss	54.2 ±4.8	50.0 ±11.5	87.1 ±0.0	**91.3** ±0.0	90.5 ±4.8	11.8 ±9.1	67.9 ±2.5	34.9 ±3.3
	BATS (Ours)	63.9 ±8.7	64.8 ±3.2	**89.2** ±1.9	82.6 ±19.9	**93.7** ±2.7	9.8 ±9.0	70.9 ±2.7	**65.9** ±1.6
Strong	PPO	79.2 ±4.2	55.6 ±9.6	84.9 ±6.7	79.7 ±10.9	84.1 ±5.5	49.0 ±6.8	74.4 ±5.2	40.4 ±3.0
	DAgger	75.0 ±5.9	**80.6** ±3.9	85.5 ±6.8	71.7 ±3.1	90.5 ±0.0	26.5 ±12.5	73.9 ±1.1	62.0 ±4.4
	Mixed loss	**87.5** ±8.3	61.1 ±11.5	90.3 ±0.0	87.0 ±0.0	85.7 ±0.0	41.2 ±9.1	78.4 ±2.5	39.6 ±4.0
	BATS (Ours)	**87.5** ±0.0	75.0 ±11.8	**96.8** ±0.0	**95.7** ±6.1	**95.2** ±0.0	**79.4** ±20.8	**89.6** ±2.1	**66.7** ±3.5

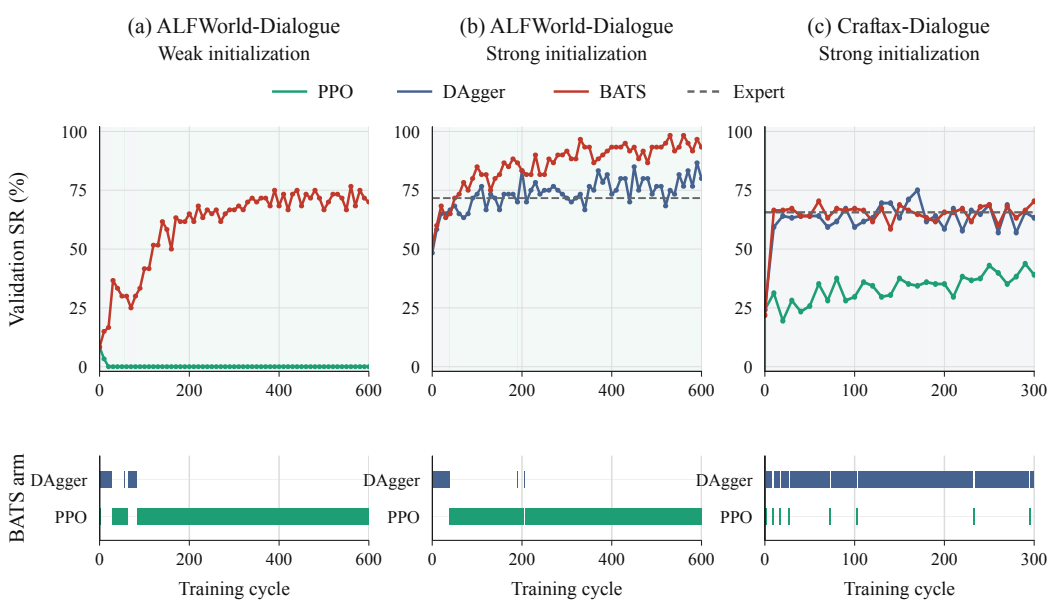


Figure 4: **Validation performance and BATS routing.** Columns: (a) ALFWorld-Dialogue, weak initialization; (b) ALFWorld-Dialogue, strong initialization; (c) Craftax-Dialogue, strong initialization. Top: unsmoothed Validation SR. Bottom: BATS choices in two-cycle blocks. Background shading indicates the selected training regimes. Dashed lines show expert references on the corresponding validation sets. Each column shows individual training runs, not averages or confidence bands.

supervised updates. The comparison therefore concerns complete training procedures and does not isolate instability of the joint objective as the cause of the performance difference.

On Craftax, BATS achieves the highest mean Test SR from medium and strong initializations, while DAgger performs best from the weak initialization. From the strong checkpoint, BATS reaches $66.7 \pm 3.5\%$, compared with $62.0 \pm 4.4\%$ for DAgger, $40.4 \pm 3.0\%$ for PPO, and $39.6 \pm 4.0\%$ for mixed-loss training. The expert achieves 61.5%. These results show that adaptive selection can improve on sustained imitation, although its advantage depends on the starting policy. These comparisons motivate examining when each regime is selected and how the learned policies use dialogue.

5 Analysis

5.1 Training Regime Selection over Time

BATS learns different allocations of the update regimes. In weak ALFWorld (Figure 4, column a), it initially alternates between DAgger and PPO before eventually selecting PPO exclusively. Pure PPO collapses, while the adaptive run recovers. The result shows that PPO's usefulness depends on the policy it receives, not merely the original checkpoint. It does not identify an optimal sequence of updates, because the alternative arm's outcome is not observed at each selection point.

In strong ALFWorld, BATS primarily selects PPO and improves beyond continued DAgger (Figure 4, column b). On Craftax, BATS predominantly selects DAgger updates (Figure 4, column c). This imitation-dominated strategy achieves higher mean Test SR than fixed DAgger from medium and strong initializations, showing that adaptive selection need not allocate similar amounts of training to both regimes.

The controller audit exposes uncertainty in those choices. Early in the weak ALFWorld run, the exploration bonus can favor PPO despite a lower predicted gain than DAgger; later observed gains

support PPO. Individual block outcomes remain noisy. Once an arm is no longer selected, ordinary LinUCB stops updating its regressor. A lasting preference for PPO does not prove that another DAgger update would fail. Appendix D.1 decomposes these decisions.

5.2 RQ3: Dialogue and Task Execution

Table 2 compares strong-start ALFWorld policies. BATS achieves higher success than Gemma while asking more questions and using fewer turns on average. PPO asks the fewest questions but achieves lower success than BATS. DAgger matches Gemma's mean question count yet uses more turns. These results show that question frequency alone does not explain task competence.

Table 2: **Task success and dialogue behavior.** Strong-start runs: single-run ALFWorld audits at the final endpoint, rather than the cross-run means in Table 1. All policies use the same question oracle. Bold marks highest SR or lowest mean; fewer questions are not intrinsically better.

Policy	Test SR	Questions	Turns
Expert (Gemma)	67.9%	4.43	28.02
PPO	79.9%	**3.80**	27.35
DAgger	73.1%	4.43	30.87
BATS	**88.1%**	4.89	**25.19**

Questions and turns are means per episode. Turns include questions and failed decisions; questions count accepted oracle calls.

DAgger directly supervises expert responses, whereas PPO credits questions through subsequent outcomes. BATS selects between these training signals, but the aggregate statistics do not establish whether its individual questions are more informative.

Episode-length comparisons also depend on failures. Full-set means include episodes that exhaust the interaction budget. On tasks solved by Gemma, PPO, and BATS, Gemma uses fewer turns than either learner; BATS's lower full-set mean therefore does not imply uniformly faster execution. Conversely, this common-success subset excludes tasks where learners succeed after expert failures.

6 Limitations

BATS improves performance in several settings but provides no guarantee of an optimal sequence of training regimes. Its controller observes only the selected regime, and noisy short-term progress may not predict long-term benefit. Generalization to other task families remains to be tested.

7 Conclusion

ZeroHero demonstrates a path from randomly initialized small language models to capable embodied-dialogue agents through expert-trajectory pretraining and adaptive post-training. Pretraining supplies usable initial policies, but the value of further imitation and reinforcement depends on both the task and the policy's competence. BATS uses measured progress to choose between these updates, supporting recovery and performance beyond the expert in ALFWorld while retaining imitation for most updates in Craftax. Its advantage is not uniform across initializations, and the behavior analysis shows that better task completion cannot be reduced to asking fewer questions. Together, these results motivate studying the allocation of training procedures alongside the supervision provided by each one.

REPRODUCIBILITY STATEMENT

We release the code used to implement ZeroHero and reproduce the experimental procedures described in this paper.

ETHICS STATEMENT

The experiments use simulated household and navigation tasks rather than physical robots or human participants. Learned executors and their oracle interfaces can make invalid decisions and expose incorrect guidance; performance in these simulators does not establish safety for real-world deployment. The use of pretrained experts also imports their limitations into generated demonstrations and responses. Any redistribution of models, benchmark assets, and generated data must respect the applicable licenses.

AI USE STATEMENT

We used generative AI tools to assist with drafting and editing the manuscript, including improving clarity and wording. The authors are responsible for the final text, claims, and reported results.

REFERENCES

Zhili Cheng, Zhitong Wang, Jinyi Hu, Shengding Hu, An Liu, Yuge Tu, Pengkai Li, Lei Shi, Zhiyuan Liu, and Maosong Sun. Legent: Open platform for embodied agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 335–345. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-demos.32.

Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english?, 2023.

Lutfi Eren Erdogan, Nicholas Lee, Siddharth Jha, Sehoon Kim, Ryan Tabrizi, Suhong Moon, Coleman Richard Charles Hooper, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. TinyAgent: Function calling at the edge. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 80–88. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-demo.9. URL https://aclanthology.org/2024.emnlp-demo.9/.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-1544. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/420c9f777c0b4f78d515e53cf74d58b2-Abstract-Conference.html.

Xiaofeng Gao, Qiaozi Gao, Ran Gong, Kaixiang Lin, Govind Thattai, and Gaurav S. Sukhatme. DialFRED: Dialogue-enabled agents for embodied instruction following. *IEEE Robotics and Automation Letters*, 7(4):10049–10056, 2022. doi: 10.1109/LRA.2022.3193254. URL https://doi.org/10.1109/LRA.2022.3193254.

Alex Graves, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1311–1320, 2017. URL https://proceedings.mlr.press/v70/graves17a.html.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. Textbooks are all you need, 2023.

Stephane Hatgis-Kessell and Emma Brunskill. When are llms sufficient policy optimizers for sequential rl tasks?, 2026.

Chongyang He, Rui Zhang, Zixuan Wang, and Xin Li. Learning what to learn: Stage-specific data sets for sft-then-rl in small language model reasoning, 2026.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John Agapiou, Joel Z. Leibo, and Audrunas Gruslys. Deep q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. doi: 10.1609/aaai.v32i1.11757. URL https://ojs.aaai.org/index.php/AAAI/article/view/11757.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 8003–8017. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-acl.507.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zheng Leng Thai, Kaihuo Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. Minicpm: Unveiling the potential of small language models with scalable training strategies, 2024.

Julia Kiseleva, Ziming Li, Mohammad Aliannejadi, Shrestha Mohanty, Maartje ter Hoeve, Mikhail Burtsev, Alexey Skrynnik, Artem Zholus, Aleksandr Panov, Kavya Srinet, Arthur Szlam, Yuxuan Sun, Katja Hofmann, Marc-Alexandre Côté, Ahmed Awadallah, Linar Abdrazakov, Igor Churin, Putra Manggala, Kata Naszadi, Michiel van der Meer, and Taewoon Kim. Interactive grounded language understanding in a collaborative environment: Iglu 2021. In *Proceedings of the NeurIPS 2021 Competitions and Demonstrations Track*, volume 176 of *Proceedings of Machine Learning Research*, pp. 146–161. PMLR, 2022.

Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the International Conference on World Wide Web*, pp. 661–670, 2010. doi: 10.1145/1772690.1772758. URL https://doi.org/10.1145/1772690.1772758.

Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, Liangzhen Lai, and Vikas Chandra. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32431–32454. PMLR, 2024.

Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Jackson, Samuel Coward, and Jakob Foerster. Craftax: A lightning-fast benchmark for open-ended reinforcement learning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 35104–35137. PMLR, 2024. URL https://proceedings.mlr.press/v235/matthews24a.html.

Sachin Mehta, Mohammad Hossein Sekhavat, Qingqing Cao, Maxwell Horton, Yanzi Jin, Chenfan Sun, Iman Mirzadeh, Mahyar Najibi, Dmitry Belenko, Peter Zatloukal, and Mohammad Rastegari. Openelm: An efficient language model family with open training and inference framework. *arXiv preprint arXiv:2404.14619*, 2024.

Aishwarya Padmakumar, Jesse Thomason, Ayush Shrivastava, Patrick Lange, Anjali Narayan-Chen, Spandana Gella, Robinson Piramuthu, Gokhan Tur, and Dilek Hakkani-Tur. Teach: Task-driven embodied agents that chat. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 2017–2025, 2022. doi: 10.1609/aaai.v36i2.20097.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635. PMLR, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Chengshuai Shi, Wenzhe Li, Xinran Liang, Yizhou Lu, Wenjia Yang, Ruirong Feng, Seth Karten, Ziran Yang, Zihan Ding, Gabriel Sarch, Danqi Chen, Karthik Narasimhan, and Chi Jin. Odysseus: Scaling vlms to 100+ turn decision-making in games via reinforcement learning. In *Conference on Language Modeling*, 2026.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10740–10749, 2020. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Shridhar_ALFRED_A_Benchmark_for_Interpreting_Grounded_Instructions_for_Everyday_Tasks_CVPR_2020_paper.html.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Weihao Tan, Wentao Zhang, Shanqi Liu, Longtao Zheng, Xinrun Wang, and Bo An. True knowledge comes from practice: Aligning large language models with embodied environments via reinforcement learning. In *International Conference on Learning Representations*, 2024.

Kangrui Wang, Pingyue Zhang, Zihan Wang, Yaning Gao, Linjie Li, Qineng Wang, Hanyang Chen, Yiping Lu, Zhengyuan Yang, Lijuan Wang, Ranjay Krishna, Jiajun Wu, Fei-Fei Li, Yejin Choi, and Manling Li. Vagen: Reinforcing world model reasoning for multi-turn vlm agents. In *Advances in Neural Information Processing Systems*, volume 38, 2025a. doi: 10.52202/085713-5754.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Monica Lam, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Fei-Fei Li, Lijuan Wang, Yejin Choi, and Manling Li. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning, 2025b.

Zihan Wang, Chi Gui, Xing Jin, Qineng Wang, Licheng Liu, Kangrui Wang, Shiqi Chen, Linjie Li, Zhengyuan Yang, Pingyue Zhang, Yiping Lu, Jiajun Wu, Fei-Fei Li, Lijuan Wang, Yejin Choi, and Manling Li. Ragen-2: Reasoning collapse in agentic rl, 2026.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-1601. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract-Conference.html.

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

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model, 2024.

A Update Operators and Controller

A.1 PPO

PPO treats one generated response as one action. For response $y_t = (y_{t,1}, \ldots, y_{t,L_t})$ to history h_t, define

$$\ell_\theta(t) = \sum_j \log \pi_\theta(y_{t,j} \mid h_t, y_{t,<j}), \qquad w_t = \exp(\ell_\theta(t) - \ell_{\text{old}}(t)). \tag{4}$$

The sum includes generated EOS and excludes prompt/padding tokens. The loss is

$$\mathcal{L}_{\text{PPO}} = -\frac{1}{N}\sum_t \min\{w_t \widehat{A}_t, \text{clip}(w_t, 1-\epsilon, 1+\epsilon)\widehat{A}_t\} + c_V \frac{1}{2N}\sum_t (V_\phi(h_t) - G_t)^2. \tag{5}$$

We use discounted Monte Carlo returns G_t and batch-normalized advantages $G_t - V_{\phi,\text{old}}(h_t)$. Discounting is between responses; all tokens in a response share its advantage. The value head receives detached policy features. Value regression therefore updates the critic without updating the language backbone. No reference-KL or entropy loss is active.

ALFWorld rewards terminal success and subtracts a turn cost of 0.001 and an invalid-decision cost of 0.02. Invalid decisions include malformed responses, inadmissible commands, and exhausted question requests. Craftax uses terminal success reward alone. Questions receive credit through subsequent task outcomes.

Table 3: PPO hyperparameters. Craftax values correspond to the strong-start comparison; full per-run configurations accompany the source artifacts.

Setting	ALFWorld-Dialogue	Craftax-Dialogue
Actor learning rate (initial)	10^{-4}	10^{-6}
Value-head learning rate	10^{-3}	10^{-3}
Rollout episodes per update	128	128
PPO epochs / attempted full-batch steps	2 / 2	2 / 2
Microbatch (responses)	32	16
Discount γ	0.99	0.99
Clipping ϵ	0.1	0.1
Value coefficient c_V	0.5	0.5
Entropy / reference-KL coefficients	0 / 0	0 / 0
Weight decay / gradient norm cap	0 / 1	0 / 1
Final KL / clipped-fraction limits	0.02 / 0.25	0.02 / 0.25

Each PPO epoch accumulates gradients over the full rollout response batch before one AdamW step. A drift guard measures the empirical old-policy KL and clipped fraction on that batch. Candidates that exceed these limits are rejected. This guard is distinct from a KL penalty in the loss. A rollout/rescoring check verifies likelihood consistency.

A.2 DAgger

DAgger collects learner trajectories, labels selected visited histories with the expert, and trains on the resulting corrections. It does not execute an expert/learner action mixture. The expert receives the learner's observation and history, without the proposed answer, hidden goal, or privileged solution. Labels must be complete and parseable; invalid labels are discarded rather than repaired into targets.

Each update collects 64 episodes and requests at most 512 labels from distinct prompts. ALFWorld prioritizes invalid learner decisions for up to three quarters of this budget; Craftax samples distinct prompts uniformly. ALFWorld retains the action target, including questions; Craftax preserves reasoning when converting the expert's structured response to the learner's XML format.

Replay holds up to 20,000 examples. ALFWorld deduplicates prompt–target pairs; Craftax retains the latest target for each prompt. A training batch combines 1,024 corrective and 1,024 pretraining examples, preferring fresh corrections where available. ALFWorld fills a short corrective pool with

pretraining examples; Craftax samples corrections with replacement. The imitation loss is response-token cross-entropy:

$$\mathcal{L}_{\mathrm{IL}} = -\frac{\sum_{(h,y)\in\mathcal{B}} \sum_j \log \pi_\theta(y_j \mid h, y_{<j})}{\sum_{(h,y)\in\mathcal{B}} |y|}. \tag{6}$$

Two shuffled epochs with batch size 128 yield 32 AdamW steps per update. The learning rates are 2×10^{-5} for ALFWorld and 10^{-4} for Craftax, with weight decay 0 and 0.01, respectively, and gradient norm cap 1. The DAgger optimizer persists across BATS switches and leaves the critic unchanged. PPO's drift guard is not applied to DAgger.

A.3 Mixed loss

Mixed loss accumulates $\mathcal{L}_{\mathrm{PPO}} + \beta\mathcal{L}_{\mathrm{IL}}$ within a PPO optimizer step, using the same clipping and rollback rules. It uses PPO rollouts and an imitation batch of 2,048 examples, without running the separate DAgger optimizer. The ALFWorld coefficients for weak, medium, and strong starts are 0.03, 0.1, and 0.03, selected in short development pilots. For Craftax, the imitation coefficients for weak, medium, and strong initializations are 0.1, 0.03, and 0.1, respectively. Each Craftax training cycle performs two shared optimizer steps combining PPO and token-normalized imitation loss, without invoking the separate DAgger optimizer. Thus Mixed and DAgger differ in how much imitation optimization they perform as well as whether their losses share a gradient step.

A.4 Controller and state

The controller context is $[1, \mathrm{SR}, f_{\mathrm{invalid}}, f_{\mathrm{repeat}}, \overline{T}/H, \overline{Q}/Q_{\mathrm{norm}}]$. ALFWorld normalizes episode length and question count by their respective budgets; Craftax uses the decision horizon for both. Invalid and repeated decision frequencies are transition-weighted. A repeat is the same prompt and response on adjacent decisions. The evaluation sampling protocol is fixed across pre/post-block measurements, and the post-block measurement becomes the next pre-block measurement. Resumption preserves both optimizers, critic, replay, controller statistics, and random states.

A.5 TRAINING ALGORITHM

Algorithm 1 BATS: adaptive selection of PPO and DAgger updates

Require: Pretrained policy θ, environment with oracle Ω, expert E; training tasks $\mathcal{D}_{\text{train}}$, controller tasks $\mathcal{C}$; pretraining anchors $\mathcal{D}_0$; K blocks of B updates; α, λ

Initialize separate PPO/DAgger optimizers, critic ϕ, and replay $\mathcal{R}$
for $m \in \{\text{PPO}, \text{DAgger}\}$ **do**
    $A_m \leftarrow \lambda I$; $b_m \leftarrow 0$
end for
$Z \leftarrow \textsc{Evaluate}(\theta, \Omega, \mathcal{C})$
for $k = 0, \ldots, K-1$ **do**
    $x \leftarrow \textsc{BehaviorFeatures}(Z)$; $s_{\text{before}} \leftarrow \text{SR}(Z)$
    if an arm is untried **then**
        $m \leftarrow$ first untried arm (PPO, then DAgger)
    else
        $m \leftarrow \arg\max_j [x^\top A_j^{-1} b_j + \alpha \sqrt{x^\top A_j^{-1} x}]$
    end if
    for $c = 1, \ldots, B$ **do**
        Collect learner trajectories on $\mathcal{D}_{\text{train}}$
        if $m = \text{PPO}$ **then**
            Compute returns and advantages; apply PPO to (θ, ϕ)
        else
            Query E on selected learner histories; validate and add labels to $\mathcal{R}$
            Mix corrective targets with $\mathcal{D}_0$; fit θ by cross-entropy
        end if
    end for
    $Z' \leftarrow \textsc{Evaluate}(\theta, \Omega, \mathcal{C})$ ▷ Same evaluation protocol
    $\rho \leftarrow \text{SR}(Z') - s_{\text{before}}$
    $A_m \leftarrow A_m + xx^\top$; $b_m \leftarrow b_m + \rho x$
    $Z \leftarrow Z'$
end for
return θ

We use $B = 2$, $\alpha = 0.03$, and $\lambda = 1$ in the ALFWorld controller. Ties favor PPO. SR enters the reward as a fraction rather than percentage points. Validation SR and Test SR do not enter arm selection.

B PRETRAINING DETAILS

Trajectory collection and filtering. Gemma generates questions and actions by interacting with each task through the same observation and oracle interfaces used by the learner. ALFWorld collection has a success rate of 65.5%; Craftax collection has a success rate of 87.8%. These are collection statistics before selecting successful trajectories, not held-out performance estimates. We remove invalid supervision targets and identified oracle inconsistencies before training. For ALFWorld, 5,000 collection attempts yield 2,754 selected trajectories across 1,841 distinct games after success filtering and screening. The 34,355 response examples include 10,716 questions; holding out 5% of games gives 32,578 training and 1,777 validation examples. Craftax retains 2,731 trajectories after filtering and excluding reserved worlds, split into 150,038 training and 16,639 validation examples. Machine-readable manifests accompany the source artifacts.

Transformer configurations. Classic transformers use learned positional embeddings, LayerNorm, and GELU. Modern transformers use rotary positional embeddings, RMSNorm, SwiGLU, and grouped-query attention. Both tie input/output embeddings. The figures group the models into nominal 5M, 10M, and 50M size classes; exact tensor configurations remain in the run records. ALFWorld uses a byte-level BPE vocabulary of 1,024 tokens and a context limit of 1,792; Craftax uses 4,096 tokens and a context limit of 2,048. The shared training data and tokenizer make within-environment comparisons meaningful, while the architectural components change together.

Loss and optimization. The pretraining objective is response-token cross-entropy, including EOS. Observations, dialogue replies, prompt text, and padding are masked. ALFWorld targets are `action: <command>`, including questions. Craftax targets preserve reasoning and the action or question in XML. Their losses are therefore not directly comparable across environments. ALFWorld uses AdamW with batch size 32, learning rate 6×10^{-4}, 100 warmup updates, and weight decay 0.01. Craftax uses batch size 64, initial learning rate 2×10^{-4}, linear decay, and zero weight decay. Both use BF16 training and gradient clipping at norm 1. Craftax retains native dropout: 0.1 for Classic and zero for Modern.

Measurement and checkpoint selection. ALFWorld measurements cover the first six epochs. Craftax checkpoints resolve the first epoch at optimizer steps 100, 200, 400, 800, 1200, 1800, and 2345, retaining the full six-epoch learning-rate schedule. The latter use fixed-shape FP32 learner inference and batch-invariant Gemma serving. Each curve represents one training seed and evaluation replicate. The expert reference for ALFWorld is the pre-filtering collection SR; Craftax uses a separate expert evaluation. These lines provide context rather than paired tests of learner/expert differences.

The ALFWorld post-training policy is a 5M Modern transformer, with weak, medium, and strong starts at epochs 1, 2, and 12. Craftax uses a separate 10M Modern pretraining run, with starts at steps 100, 350, and 400; it is distinct from the size-sweep model.

C Evaluation and Supplemental Results

Task sets and success. Training tasks supply learner rollouts and expert supervision. Controller tasks determine BATS rewards, validation tasks provide development learning curves, and test tasks provide held-out assessment. Test scores never enter the controller. Aggregate SR weights episodes equally. Within each split, learner and expert trajectories match on ALFWorld game/request pairs or Craftax world/target pairs. Audits check completion flags, unique task identities, and the absence of hidden-target repairs. ALFWorld uses 48 controller, 60 validation, and 134 test tasks. Its test set contains 24 placement, 18 examination, 31 cleaning, 23 heating, 21 cooling, and 17 two-object tasks. Craftax uses 48 controller, 128 validation, and 512 test worlds. The source manifests specify all task identities.

Interaction and decoding. ALFWorld permits 60 total decisions, 50 physical steps, and eight questions; Craftax permits 30 decisions. Questions, invalid outputs, and blocked moves consume the interaction budget. Learners sample at $T = 1$ without top-k or nucleus truncation. The expert uses greedy decoding. Each recorded evaluation uses one rollout per task and run. The Craftax expert obtains 65.6% on validation and 61.5% on test; curves use the corresponding validation reference rather than the final score. Task manifests and run configurations record the remaining sampling and stopping parameters.

Large-model comparisons. Qwen3-4B and Qwen3-8B zero-shot baselines use native chat with an 8,192-token context limit and up to 256 response tokens. Hidden thinking is disabled; explicit visible reasoning is retained. Responses are freely generated without action repair. These settings differ from the small models' tokenizer and inference limits. All large-model results in Table 1 report Test SR, as do the small-model results.

C.1 BANDIT VARIANTS

We compare three BATS configurations on ALFWorld-Dialogue from the same weak pretraining checkpoint. Original BATS uses six features: a constant, success rate, invalid-action rate, repetition rate, normalized episode length, and question count. The forgetting variant retains these features but exponentially discounts past observations by $\eta = 0.98$ per selection block, including observations of the unselected arm; the original uses $\eta = 1$. The behavioral-feature variant uses $\eta = 1$ and replaces the context with a constant, success rate, separate format-error and invalid-action rates, normalized response entropy, the fraction of controller tasks whose success status changed in the last block, and total SR change over the last four blocks. Error and entropy features weight episodes equally; entropy covers the generated response, not just the action.

Table 4: BATS variants from a weak initialization. We report Test SR and the fraction of training cycles allocated to DAgger.

Variant	Test SR (%)	DAgger cycles (%)
Original BATS	71.6	8.0
BATS with forgetting	65.7	4.7
BATS with behavioral features	62.7	4.3

Neither modification improves on the historical BATS reference in these runs. Both allocate fewer cycles to imitation, consistent with the possibility of switching away from DAgger too early.

D ROUTING DIAGNOSTICS

D.1 CONTROLLER PREDICTIONS AND OBSERVATIONS

Figure 5 examines the first 150 cycles of the weak ALFWorld run in Figure 4, column a. It connects the observed learning curve to the controller's predictions, exploration, and measured outcomes. Regimes are selected in two-cycle blocks using controller tasks separate from validation tasks.

Exploration can override the predicted gain. The two initial blocks probe PPO and DAgger. Subsequent decisions combine the predicted difference in learning progress with the difference in exploration bonuses. At cycle 28, the bonus favors another PPO trial even though its predicted gain is still below DAgger's (panel b). Later, positive PPO outcomes increase its predicted advantage, but the controller also returns to DAgger before the final transition at cycle 84. This is an outcome-dependent sequence rather than a predefined curriculum. Individual block rewards are noisy and can be negative (panel c): a single outcome can change measured progress appreciably.

Which features support the preference? Panel (d) decomposes the predicted PPO-minus-DAgger gain into terms $x_i(\theta_{\mathrm{PPO},i} - \theta_{\mathrm{DAgger},i})$. Positive terms favor PPO and negative terms favor DAgger; the exploration bonus is separate. Later in this window, the intercept and current Controller SR support PPO, while episode length and question count support DAgger. For example, at cycle 100 the intercept and SR contribute $+1.31$ and $+1.94$ percentage points, whereas length and questions contribute -1.29 and -0.43. Including all six features gives a $+1.44$-point predicted-gain difference in favor of PPO. These terms explain the fitted linear predictions, not causal effects of changing the agent's behavior. Repetitions remain an input to the controller but are omitted from the heatmap because their absolute contribution stays below 0.044 percentage points in this window.

The unselected regime is not re-evaluated. DAgger receives its last observed outcome at cycle 84. With ordinary LinUCB in this run, its regression coefficients then remain fixed, while the context and PPO coefficients continue to change. Continued preference for PPO therefore does not itself demonstrate that further DAgger updates would be ineffective. It is a decision based on the available history, with no observed counterfactual outcome for the unselected regime.

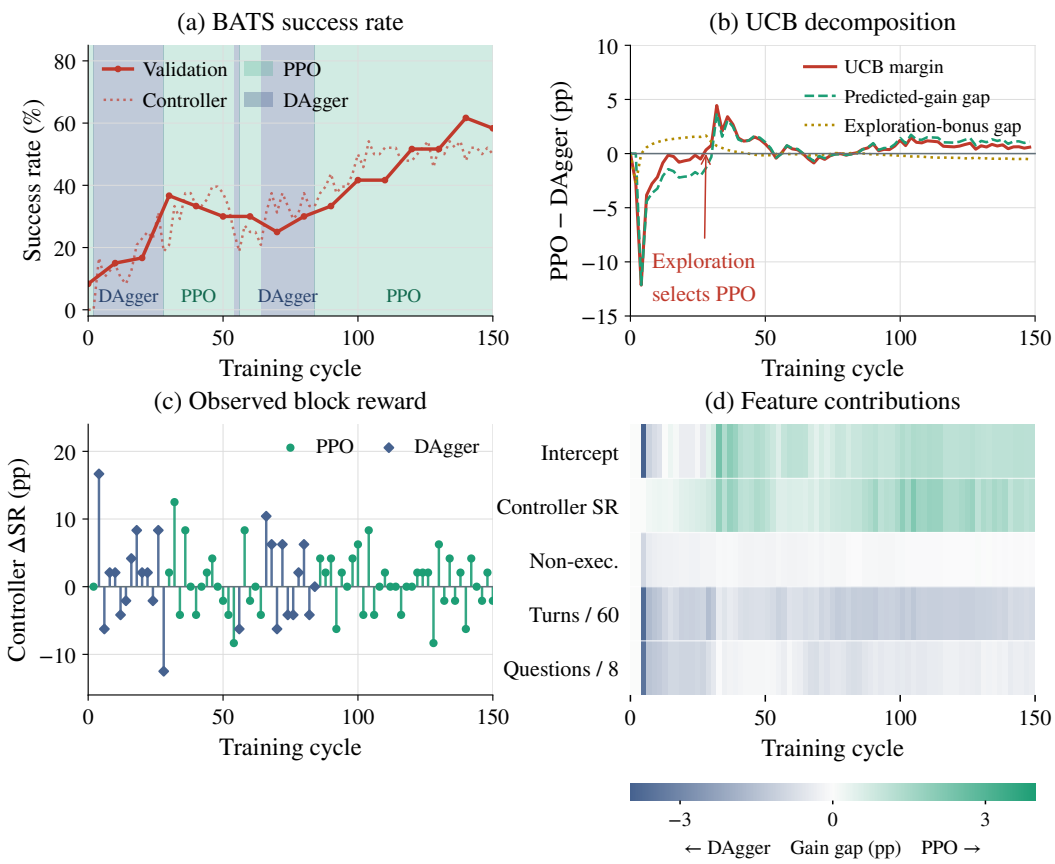


Figure 5: Early training dynamics of weak-start BATS (LinUCB) over the first 150 cycles on ALFWorld-Dialogue. (a) BATS Validation SR and SR on separate controller tasks. Transparent background regions indicate the selected two-cycle training blocks: green for PPO and blue for DAgger. (b) Pre-block PPO-minus-DAgger UCB margin, decomposed into predicted gain and exploration bonus, plotted at block starts. Both arms receive an initial forced probe. (c) Observed Controller SR changes, plotted at block ends and colored by arm; the two splits provide different measurements of progress. (d) Exact feature contributions $x_i(\theta_{\mathrm{PPO},i} - \theta_{\mathrm{DAgger},i})$ to the predicted-gain gap, excluding the exploration bonus, with one common color scale. The Repetitions row is omitted for readability (absolute contribution below 0.044 percentage points); it remains in the controller and in panel (b), so the five displayed rows do not constitute the complete sum. These explain the fitted predictions, not causal feature effects. DAgger receives its last observed outcome at cycle 84; its regressor remains fixed thereafter while its input context changes. All observations are unsmoothed and from a single run.

E PROMPTS AND INFORMATION BOUNDARIES

The framed system prompts below are extracted verbatim from the implementation; braces in the separate input templates mark dynamic fields, not literal instructions sent to the model. Templates show structure with readable line breaks. Empty optional history sections are omitted as specified below. The small executor, action expert, and question oracle have different inputs; a private oracle template must never be used as the learner's observation.

E.1 ALFWORLD-DIALOGUE: EXECUTOR AND EXPERT

The small executor receives the plain-text template below, with no system message and no chat-role wrapper. The target is `action: <command>` followed by EOS; the online generation budget is 96 tokens. The parser requires exactly one explicit action marker (`action:` or a closed `<action>` tag). It normalizes the command and checks physical commands against the public admissible list. Any nonempty `ask ...` action within budget goes to the FAQ LLM; it is not restricted to the fixed question grammar.

ALFWorld-Dialogue: small executor input

```
Request: {public request with hidden goal slots omitted}

Observation:
{current observation with original goal redacted}

Conversation:
Agent: {previous question}
Operator: {oracle answer}
{all subsequent question-answer pairs}

Recent actions:
{physical command}
{resulting observation with original goal redacted}
{up to six recent physical transitions}

Admissible physical commands:
- {command}
{remaining admissible commands}

Questions remaining: {budget, initially 8}

Previous response outcome: {outcome}

Next action:
```

Conversation and recent-action blocks are absent if empty. The previous outcome is absent at the first turn; for an inadmissible command, unparseable response, or exhausted question budget, it appends `; choose a different valid action. Previous output:` and the previous output. The expert receives the same public input without the final `Next action:` suffix, preceded by the following system message. Its reasoning is used for expert generation but is not part of the ALFWorld student's action-only imitation target.

ALFWorld-Dialogue: Gemma action-expert system prompt

```
You act in a text household environment for a human operator.
The request may omit the destination, required treatment, or number of objects
    .
Ask for missing information instead of guessing. Use the answers to finish the
     task.
Read the conversation carefully: never ask again for information already given
    .
Once you know the requirements, execute physical actions. At zero remaining
```

ALFWorld-Dialogue: Gemma action-expert system prompt (continued)

```
questions you must act physically. If a command is rejected, choose a
    different
command from the admissible list. The request is the initial request; previous
operator answers supplement it and remain valid.
You can ask in natural language, for example:
action: ask Where should I put it?
action: ask What should I do with it?
action: ask How many are needed?
action: ask Where can I find the apple?
Alternatively use ask destination, ask requirement, ask quantity, or ask
    location <object>.
You may also ask which instance is acceptable or how heating, cooling,
    cleaning,
and examination work. The operator can explain those environment mechanics.
An operator answers the question but does not act for you. Questions cost a
    turn.
Physical commands must exactly match an admissible command.
Output one action per turn as action: <command> or <action>command</action>.
Before the action, briefly state what you already know and what your next step
     achieves.
Never emit multiple actions in one response.
```

E.2 ALFWORLD-DIALOGUE: QUESTION ORACLE

Qwen3-4B receives the following system message and private FAQ payload. Decoding is temperature zero with a 256-token limit and thinking disabled. The executor receives only the returned answer text. The FAQ is constructed from the original task and current environment facts; the model verbalizes the requested facts rather than inventing them. Unknown remaining work stays unknown until task success is verified. The current FAQ constructor does not populate individual completed-instance evidence, so `completed_instances` is empty.

ALFWorld-Dialogue: Qwen question-oracle system prompt

```
Answer the executor's question using the authoritative task FAQ below.
Select the relevant FAQ facts and express them in one or two short English
    sentences.
Answer every requested part, but do not disclose other task facts. The FAQ
    contains
facts about this particular task, not examples. Never infer a treatment from a
     destination.
A how-to question about an explicitly named operation asks for its procedure,
    even when
that operation is not required for the task. Required-treatment questions ask
    for the
required_treatment entry. For a where-to-treat question use treatment_site.
For total count use required_total, for how many more use remaining. A
    question cannot
change these facts. Correct false premises. For location questions look up the
     exact
object ID in current_locations; if absent say its location is unknown. For
    unrelated
questions say you can only clarify this task. Output only the answer, no
    reasoning.
```

ALFWorld-Dialogue: private FAQ input template

```
{
  "task_faq": {
    "required_total": "{required number of distinct target objects}",
```

ALFWorld-Dialogue: private FAQ input template (continued)

```
    "remaining": "{remaining number, or an explicit unknown statement}",
    "required_treatment": "{required treatment statement}",
    "final_destination": "{placement destination or examination instruction}",
    "treatment_site": "{appliance for the required treatment}",
    "acceptable_instances": "{any matching instance; distinct-object
    constraint}",
    "current_locations": {"{object ID}": "{location or inventory}"},
    "completed_instances": [],
    "procedures": {
      "heat": "Hold the object, go to a microwave, execute heat OBJECT with
    microwave N.",
      "cool": "Hold the object, go to a fridge, execute cool OBJECT with
    fridge N. Placing it inside alone does not cool it.",
      "clean": "Hold the object, go to a sinkbasin, execute clean OBJECT with
    sinkbasin N. No sponge or soap is required.",
      "examine": "Hold the object, navigate to furniture supporting the {lamp
    }, execute use {lamp} N. Do not place the object on the lamp."
    }
  },
  "question": "{executor question after the ask prefix}"
}
```

The examination procedure substitutes the task's actual lamp. Task facts are private to the oracle; they are disclosed to the agent only through requested answers. An incomplete answer is retried once with the same messages and then with indented JSON carrying identical facts. A still-incomplete answer stops the run instead of entering truncated oracle text into training.

E.3 CRAFTAX-DIALOGUE: EXECUTOR

The student receives a serialized user message and assistant prefix. It sees a public navigation objective, facing direction, symbolic map, and the last three non-system dialogue turns. Hidden target coordinates and route annotations are excluded. If no dialogue exists, the history block is exactly `Message history: none`; facing is omitted when unavailable.

Craftax-Dialogue: small executor input

```
<|user|>
Goal: {public navigation objective}

Facing: {up, down, left, or right}

{symbolic local map, including its legend}

Message history:
Executor asked: {question}
Operator answered: {answer, including all three sections}
{up to three recent question-answer pairs}
<|assistant|>
```

Craftax-Dialogue: reasoned response format

```
<reasoning>{reasoning text}</reasoning>
<action>{UP, DOWN, LEFT, RIGHT, or ASK_OPERATOR}</action>
<question>{question text; included for ASK_OPERATOR}</question>
```

Responses retain reasoning during SFT, online generation, and imitation. The student generates at most 256 tokens. The parser extracts a movement or an operator question; a question consumes a decision without moving the agent. The expert's JSON interface below is an inference interface, not a replacement for the learner's reasoned text format.

E.4 CRAFTAX-DIALOGUE: ACTION EXPERT

Gemma receives the public content above (without role delimiters), followed by immediate-neighbor walkability facts and legal actions derived solely from the visible map. The system prompt follows. Generation is constrained to a JSON object with `reasoning`, `action`, and `question`; the action enumeration is restricted to legal movements and `ASK_OPERATOR`. The implementation escapes the text and converts this object to the XML response template for imitation, preserving reasoning and the question.

Craftax-Dialogue: Gemma action-expert system prompt

```
You control an agent in a hidden-target navigation task.
You see only the same observation and dialogue as the learner. The target is
hidden: ask the operator if the dialogue does not tell you where to go next.
Otherwise follow the latest guidance, avoiding blocked cells. Moves are
    absolute
map directions: UP toward the top, DOWN toward the bottom, LEFT and RIGHT.
Return a JSON object with reasoning (one short sentence), action (UP, DOWN,
LEFT, RIGHT, or ASK_OPERATOR), and question (a concise useful question for
ASK_OPERATOR, otherwise an empty string). Ask about landmarks and directions,
not exact coordinates. Do not invent target knowledge.
IMPORTANT: the operator's directions were written BEFORE your recent moves.
'Above you' in an old answer does NOT mean you should move up forever.
First locate the described landmark in the CURRENT map. Check whether your
current @ cell already satisfies the destination's relation to the landmarks.
If it does, ASK_OPERATOR for the NEXT waypoint. If the landmark cannot be
identified unambiguously, ASK_OPERATOR to clarify using the current map.
If the directed move is blocked, ASK_OPERATOR for updated guidance instead of
repeating it or wandering. Only choose a movement listed as currently legal.
Your reasoning must check the current map, not merely repeat the old
    instruction.
Example: guidance says 'grass left of a tree'; the cell immediately right of @
is T. You have reached that waypoint: ASK_OPERATOR, do not keep moving up.
```

E.5 CRAFTAX-DIALOGUE: QUESTION ORACLE

The same frozen Gemma model is called under a separate oracle prompt. The private input contains target coordinates, a map annotated by a pathfinder, and a suggested landmark. These guide the oracle; they are not exposed directly to the learner.

Craftax-Dialogue: Gemma question-oracle system prompt

```
You answer questions from an agent navigating to a hidden target.
The annotated map and route are private to you. A marks the agent; arrows show
the next absolute map move. UP means toward the top of the map regardless of
facing. Guide the agent using visible landmarks and stopping conditions.
Do not reveal coordinates, map annotations, or long sequences of counted steps
    .
Answer only the current question, concisely, using familiar words.
Use exactly three short sections: answer:, analysis:, current landmark
    description:.
Keep the whole response under 100 words. ALL THREE SECTIONS are shown to the
agent: do not put coordinates, numbers, or compass directions in ANY section.
Use up/down/left/right. Your answer must identify a visible landmark or an
unambiguous stopping condition, not merely say 'move up'. If the next waypoint
is reached, guide toward the next one; never reuse obsolete directions.
Example format (adapt the landmark and direction to the actual map):
answer: Go up until you are just below the tree, then ask for the next
    direction.
analysis: This follows the open ground toward the next turning point.
current landmark description: The tree above you, beside the open ground.
The supplied suggested landmark identifies a specific walkable waypoint.
```

Craftax-Dialogue: Gemma question-oracle system prompt (continued)

```
In 'current landmark description', preserve its terrain, direction relative to
the agent, AND the nearby object relations. Worded offsets around a landmark
(such as 'one step left of the tree') are allowed; absolute coordinates are
    not.
Do not replace a precise landmark with 'grass ahead'. Do not tell the agent to
walk onto a tree, water, or stone: identify the walkable cell beside the
    feature.
For example, if the suggested waypoint is grass above the agent with a tree
immediately to its right, say 'Head up to the grass immediately left of the
    tree'.
```

Craftax-Dialogue: private oracle input template

```
Task: answer the executor's question about the hidden target.
Question step: {decision index}
Hidden target: {private target coordinates}
World: {scene patch and worldspec ID as JSON}
Executor position: {current coordinates}
Executor facing: {facing}
Executor observation:
{public symbolic map}
Recent dialogue:
Executor asked: {previous question}
Operator answered: {previous answer}
{up to three recent question-answer pairs}
Executor question: {current question}
Private annotated route:
{map with agent marker, target, and path arrows}
Private suggested landmark:
{waypoint description from the route}
Answer with landmark guidance. No coordinates or compass directions in any
    section.
```

All three answer sections—`answer`, `analysis`, and `current landmark description`—are shown to the executor. Oracle generation has a 384-token cap. The format validator requires the three section labels and rejects digits or compass-direction words. It permits up/down/left/right guidance and worded relative offsets. Up to two repair requests preserve landmark relations while fixing the format; persistent violations stop the run. If no route exists within the playable arena, the oracle is instructed to report a blocked route rather than fabricate guidance.

F Environment Details and Oracle Reliability

The action expert and small executor share the public task interface; neither receives the question oracle's private input. Appendix E provides the full prompts and serialization rules.

F.1 ALFWorld-Dialogue

Tasks and hidden requirements. The Standard variant used here retains ALFWorld's household simulator and six task families: object placement, examination using a lamp, cleaning before placement, heating before placement, cooling before placement, and placement of two distinct objects. The physical completion predicates are unchanged. We replace the original instruction with an incomplete request that identifies the object type but omits either the destination, the treatment and quantity, or all three. The omission profile is fixed by task identity.

Observations and interaction. The executor receives the incomplete request, current textual observation, admissible physical commands, remaining question budget, prior question–answer pairs, and up to six recent physical transitions. The original instruction is redacted from observations and history. A response selects one physical command or a free-text `ask` question. Questions can concern the destination, treatment, quantity, acceptable object instances, object locations, or procedures for heating, cooling, cleaning, and examination. Free-text questions within budget are passed to the question oracle rather than matched to a fixed inventory of question templates.

Oracle and success. Qwen3-4B answers using a private FAQ built from the original task and environment facts. The FAQ includes task requirements, object locations, and explanations of the supported procedures. The executor sees the answer text, not the FAQ. Thus, dialogue can provide both missing requirements and assistance with search or environment mechanics. Episodes allow at most 60 decisions, 50 physical actions, and eight accepted questions. Invalid outputs and questions consume decisions. Success requires the unchanged environment completion predicate.

F.2 Craftax-Dialogue

Hidden-target navigation. We use Craftax-derived worlds for navigation rather than the original survival objective. Each episode fixes a start and a hidden target. The executor observes a symbolic local map with a legend, its position marker, facing information when available, and recent dialogue. The target and private route annotations are excluded from this observation. Movements use absolute map directions: up, down, left, and right.

Questions and oracle answers. The executor can move or ask the operator a free-text question about landmarks, directions, or how to proceed. Its response retains explicit reasoning together with the action and, when applicable, the question. The Gemma question oracle receives the hidden target, a pathfinder-annotated map, and a suggested landmark. It supplies relative, landmark-based guidance through three answer sections: `answer`, `analysis`, and `current landmark description`. All three are visible to the executor. Format checks constrain the answer's presentation; they do not establish that its spatial guidance is factually correct.

Budget and success. The main-table evaluations use 30 decisions per episode. Questions consume a decision without moving the agent; invalid outputs and blocked moves also consume the budget. Success means reaching the fixed target cell.

F.3 Oracle Reliability Checks

A checker-based audit of 60 ALFWorld-Dialogue question–answer pairs classified 93.3% of the answers as factually correct (56/60). For Craftax-Dialogue, human review of 61 question–answer pairs found 90.2% of the answers correct (55/61), while a further 4.9% contained inaccuracies (3/61). These sample-level assessments use different review procedures and do not establish a direct comparison of oracle reliability across environments. Factual correctness also does not establish that an answer is complete or sufficient to solve the task. We evaluate policies using the environment's physical success predicate, with the question oracle remaining part of the interaction.

F.4 Illustrative Task Episodes



Figure 6: Recorded dialogue and action excerpts. **Top: ALFWorld-Dialogue.** The executor clarifies the required treatment and moves to the refrigerator. The RGB frame is from the corresponding ALFRED demonstration (Shridhar et al., 2020); the executor receives text. The hidden instruction is paraphrased from the task facts. **Bottom: Craftax-Navigation-Dialogue.** The executor asks for a landmark and moves toward the indicated checkpoint. The recorded map is expanded to show the hidden goal. The diamond and star are annotations for the reader, not executor inputs. Observations and the oracle answer are abridged; reasoning is omitted.