REMEMBER RIGHT, LOOK AHEAD: HISTORY NOTES WITH FUTURE PREDICTION FOR LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Language-model agents trained with multi-turn reinforcement learning (RL) often see only their last few steps, so they lose what they observed earlier. An agent can keep this information in a short text note that it writes at every step and reads at the next, but RL rewards the agent for completing the task and never checks whether the note is correct. We introduce *History Notes with Future Prediction* (ForeNote), which supervises the note directly. In the text environments we study, everything an agent learns during an episode comes from text feedback, such as a confirmed pickup or an opened drawer, so a small rule-based tracker can read the same feedback and write the correct note, a reference note. In both supervised fine-tuning and RL, ForeNote uses this reference as two training signals: it trains the agent to write its current note correctly (remember right) and to predict the next note after each action (look ahead). At test time, the agent writes and reads its own note without the tracker. With Qwen2.5-3B-Instruct, ForeNote reaches 94.6% and 95.2% success on seen and unseen ALFWorld scenes and solves 59.1% of ScienceWorld tasks. Our analysis on ALFWorld shows where these gains come from. Without a note, an agent is twice as likely to revisit a searched place just after it leaves its five-step window, whereas the note of ForeNote still lists 99.6% of such places, and the agent relies on it: erasing the note at test time costs 8.5 points of unseen success. Without the two signals during RL, unseen success falls by 20.8 points, and giving the agent a correct note at test time does not bring it back. Unseen success also falls by 4.8 points without next-note prediction in RL, and by 3.9 and 15.7 points when the agent instead predicts the next observation or another game's next note.

1 INTRODUCTION

Language-model agents that act over many steps, in household tasks, web shopping, or science experiments (Shridhar et al., 2021; Yao et al., 2022; Wang et al., 2022), must remember what they have already seen. To keep prompts short, multi-turn RL systems often show the policy only its last few steps (Feng et al., 2025; He et al., 2026), so everything older is lost. An agent asked to pick up a keychain must still know which places it has already searched after they have left its window (Figure 1, left). In ALFWorld, an agent trained without a note is twice as likely to go back to a place it has already searched just after that place leaves its five-step window (Section 5.1).

A natural remedy is to let the agent write a short text note at every step and read it at the next (Zhou et al., 2026; Lidayan et al., 2025). The difficulty is teaching it what the note should contain. Supervised fine-tuning (SFT) can teach the agent to write notes, but during RL the task reward, which says whether the episode succeeded, arrives at its end and never checks what the note contains. On ALFWorld, when RL uses only the task reward, the notes omit about 30% of the searched places, and the policy also learns a harmful habit: it picks up and puts back objects the task never mentions (Section 5.2).

Our key observation is that in the text environments we study, everything an agent learns during an episode comes from text feedback, such as a confirmed pickup, an opened container, or a selected product option, so the content of a note can be checked. We introduce *History Notes with Future*

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.

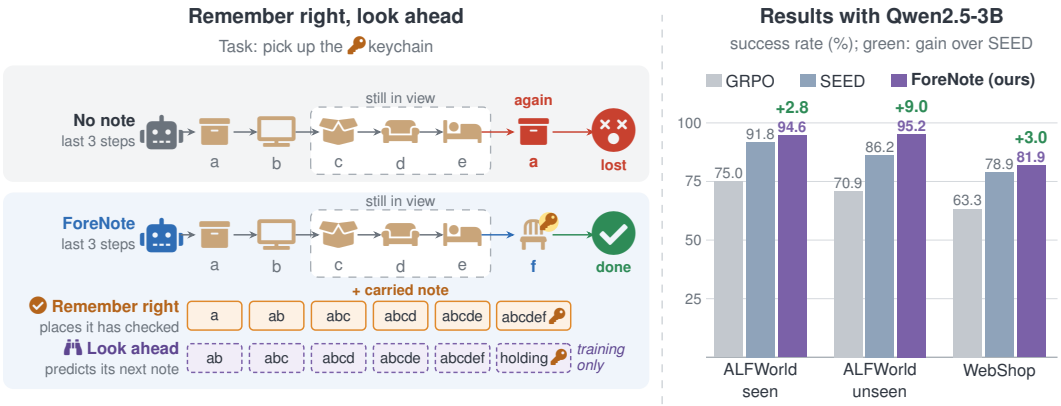


Figure 1: **ForeNote trains the note an agent carries.** Left (illustrative): both agents see only their last few steps. Without a note, the agent returns to a place it has already searched and loops. ForeNote also writes a note at every step; training scores it against what the agent has seen (remember right) and teaches the agent to predict its next note (look ahead, training only). Right: success rate with Qwen2.5-3B-Instruct; GRPO and SEED as reported by Wu et al. (2026); our WebShop result is on the clean test set (Table 1).

Prediction (ForeNote). A small rule-based tracker reads the same feedback and writes the note the agent should have, a reference note, which provides two note signals in both SFT and RL. *History-note supervision* trains the agent to write its current note to match the reference (remember right). *Future prediction* trains the agent to predict the next reference note after its action (look ahead). A note lists only the facts that track task progress, without the wording and incidental detail of observations, so the next note is a compact target in a format the agent already reads and writes. In RL, history-note supervision is a reward and future prediction a loss. The reward compares the notes sampled for the same step with one another, so it cannot correct an error they all share; the loss trains on the reference itself (Figure 4). The tracker and the prediction target are used only in training; at test time, the agent writes and reads its own note.

With Qwen2.5-3B-Instruct, ForeNote reaches 94.6% success on seen and 95.2% on unseen ALFWorld scenes, compared with 91.8% and 86.2% reported for SEED (Wu et al., 2026) (Figure 1, right). It also solves 59.1% of ScienceWorld tasks, and with Qwen3-1.7B it reaches 92.3% on seen scenes. Controlled comparisons on ALFWorld then show where the gains come from. Our contributions are:

- **Direct supervision of the agent's note.** A rule-based tracker turns environment feedback into reference notes, which supervise the agent's current note and give it a compact target to predict, in both SFT and RL.
- **Evidence that the note must be trained during RL.** On ALFWorld, the trained note still lists places that have left the window, and the agent acts on it. From the same SFT checkpoint, dropping the note signals during RL lowers unseen success by 20.8 points, and a correct note given at test time does not bring it back: RL has learned a worse policy.
- **Evidence that the note needs a loss on its reference, not only a reward.** On ALFWorld, removing the next-note loss from RL lowers unseen success by 4.8 points, and the notes stop recording objects already put in place; predicting the next observation or another game's next note instead lowers unseen success by 3.9 and 15.7 points, whereas a loss on the current reference note works about as well.

2 History Notes with Future Prediction

ForeNote trains a language-model agent that, at every step, writes a short note of what it has learned so far and then acts. During training, a rule-based tracker supplies the correct current note and the correct next note (Figure 2); we call the two objectives built on them the *note signals*. We first define the note, then the tracker, then the two signals.

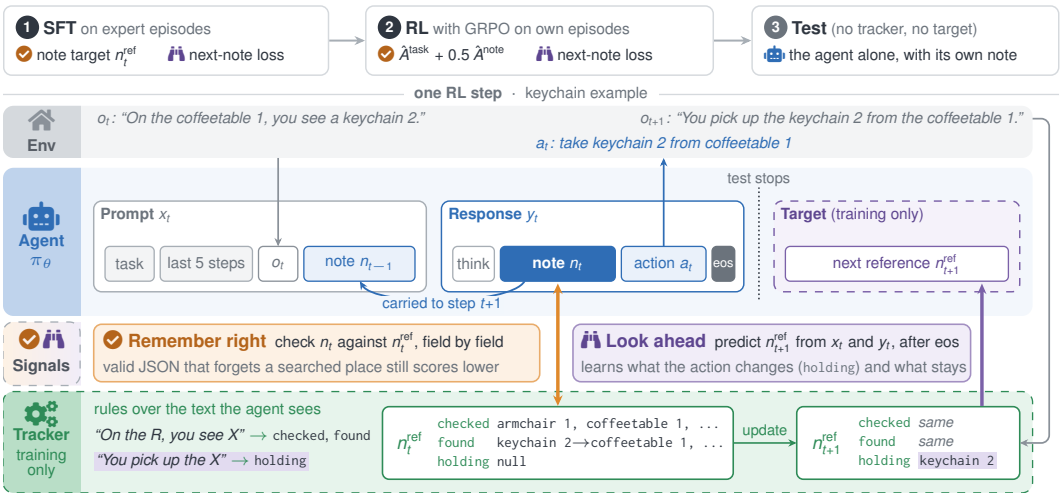


Figure 2: **Training ForeNote.** Top: the two training stages and test. Bottom: one RL step. The agent reads a prompt x_t with the task, recent steps, the observation o_t, and its previous note, and writes a response with a new note n_t and an action a_t. A rule-based tracker, used only in training, turns the same text into reference notes. Remember right: n_t is scored against n_t^{ref}, and the score enters the RL advantage. Look ahead: in training only, the next reference note n_{t+1}^{ref} is appended after the response as a cross-entropy target; at test time, generation stops at the end of the response.

2.1 Acting with a history note

At step t, the agent receives a task g, the current observation o_t, the last h interactions, the admissible actions, and its previous note n_{t-1}. Let x_t denote this prompt. The policy π_θ generates a response y_t containing a reasoning block, an updated note n_t, and an action a_t. The environment executes a_t and returns o_{t+1}; the note is carried into the next prompt. Episodes last at most T steps.

The note records the situation *before* the action. It is a JSON object of at most 256 tokens with five fields chosen for each environment (Appendix A), which record three kinds of information that a short window loses: what the agent has already tried, where its targets are, and which steps of the task are done. In ALFWorld, these are the searched receptacles and opened containers, the locations of target objects, the held object, and completed task events. ForeNote carries a valid note even if the action fails, and a malformed note leaves the previous valid note in place.

2.2 Checking the note with a tracker

Text environments report the outcome of each action, such as a confirmed pickup, an opened container, or a selected product option. A deterministic tracker applies simple rules to these messages to maintain the correct note, the *reference note*:

$$n_t^{\text{ref}} = U\big(n_{t-1}^{\text{ref}}, a_{t-1}, o_t, g\big). \tag{1}$$

The tracker reads only the text and admissible actions the agent has seen, so the reference lists the task-relevant facts the agent has seen or done, including those that have left the short window, and nothing else. The update rules of each tracker take fewer than 100 lines of Python (Appendix B).

We compare the agent's note with this reference field by field. Each field is converted to a set of entries, and its score is the F1 overlap between predicted and reference entries. The note score $s_t \in [0, 1]$ is the mean over fields that are non-empty in either note. Malformed or over-length notes score zero. This score checks content as well as format: forgetting a searched table is an error even when the JSON is valid.

2.3 History-note supervision: checking the current note

History-note supervision (HN) trains the agent to write its current note correctly. During SFT, expert responses contain the reference note as the note to write. During RL, the note score becomes a

note reward: we add a note advantage to the task advantage of group relative policy optimization (GRPO; Shao et al., 2024; Feng et al., 2025). For each task, GRPO samples G trajectories, and the task advantage $\hat{A}_i^{\text{task}}$ compares trajectory i's return with the group mean (Appendix C gives each environment's normalization). The note advantage compares note scores at the same step index:

$$\hat{A}_{i,t}^{\text{note}} = \frac{s_{i,t} - \mu_t}{\sigma_t + \epsilon}, \qquad A_{i,t} = \hat{A}_i^{\text{task}} + \lambda \hat{A}_{i,t}^{\text{note}}, \tag{2}$$

where μ_t and σ_t are computed over the group's trajectories still running at step t. Single-response and zero-variance groups receive zero note advantage. The combined advantage trains the response with a clipped policy-gradient objective (Schulman et al., 2017); we call this HN-GRPO. We use $\lambda = 0.5$.

Because the note advantage compares responses with one another, it cannot reward an entry that they all leave out. Future prediction trains directly on the reference note and has no such blind spot.

2.4 Future prediction: learning the next note

Future prediction (FP) trains the agent to predict how its note will change after its action. After the environment executes a_t, the tracker reads o_{t+1} and writes the next reference note n_{t+1}^{ref}. The policy is trained to predict it from the prompt and its own response:

$$\mathcal{L}_{\text{FP}}(t) = -\frac{1}{|n_{t+1}^{\text{ref}}|} \log \pi_\theta\big(n_{t+1}^{\text{ref}} \mid x_t, y_t\big)\,. \tag{3}$$

The next observation determines the target but is not part of the input. The target follows the response after a delimiter, and the loss is averaged over its tokens.

One transition, two targets. Consider the keychain task in Figure 2. Before `take keychain 2 from coffeetable 1`, the current reference records the keychain's location and an empty hand. History-note supervision checks that the agent's note records these facts. Once the pickup is confirmed, the next reference sets `holding` to `keychain 2`. Future prediction trains the model to anticipate that update from the current response while keeping the rest of the note unchanged. Appendix A gives the complete note.

Why predict a note? A raw observation mixes the outcome of an action with descriptions of objects, pages, and other details. The next note keeps only the task-relevant facts, in the few fields the agent already writes, so predicting it trains the model on what the action changes and what stays the same. Section 5.3 compares this target with the next observation, the current note, and another game's note.

2.5 Training and inference

The two training stages use

$$\mathcal{L}_{\text{SFT}} = \mathcal{L}_{\text{expert}} + \beta \mathcal{L}_{\text{FP}}, \tag{4}$$

$$\mathcal{L}_{\text{RL}} = \mathcal{L}_{\text{PG}}(A) + \alpha \mathcal{L}_{\text{KL}} + \beta \mathcal{L}_{\text{FP}}, \tag{5}$$

where $\mathcal{L}_{\text{expert}}$ is the cross-entropy on expert responses, which contain reference notes, A is the combined advantage in Equation 2, $\mathcal{L}_{\text{KL}}$ regularizes toward the SFT policy with weight $\alpha = 0.01$, and $\beta = 0.1$. During RL, prediction targets come from the same on-policy transitions used to train the policy; their tokens are excluded from the policy-gradient loss.

SFT prompts show the full history of expert trajectories; RL and evaluation prompts show only the last few steps and the agent's own note. At test time, the agent writes its note and action exactly as during RL; the tracker is not used.

3 Experimental Setup

We evaluate ForeNote in three environments and use ALFWorld for the controlled comparisons of Section 5.

Table 1: **Comparison with published results** (%). ALFWorld: seen-scene success by task family (Pick2: two objects) and average. Published rows and method names are from Wu et al. (2026); ForeNote averages 10 inference repeats. WebShop: published protocol for literature rows, frozen clean test set (660 episodes) for ours. Bold/underline: highest/second-highest displayed value per backbone.

	Qwen2.5-3B-Instruct									*Qwen3-1.7B*								
	ALFWorld							WebShop		ALFWorld							WebShop	
Method	Pick	Look	Clean	Heat	Cool	Pick2	Avg	Score	Succ.	Pick	Look	Clean	Heat	Cool	Pick2	Avg	Score	Succ.
Vanilla	44.4	11.1	6.2	15.4	28.6	12.5	21.9	6.7	0.8	25.0	22.2	3.1	0.0	21.4	4.2	12.5	46.5	4.7
Skill-Prompt	51.7	66.7	48.4	0.0	4.3	10.0	28.9	0.2	0.8	10.3	50.0	16.1	0.0	0.0	5.0	9.4	23.0	2.3
OPSD	48.8	41.7	16.7	0.0	15.8	16.7	28.1	11.3	3.1	26.3	33.3	9.1	0.0	4.5	5.3	14.1	47.4	9.3
GRPO	91.2	62.5	96.2	61.9	65.0	47.4	75.0	79.8	63.3	71.1	41.7	36.4	40.0	31.8	31.6	46.1	67.3	38.3
Skill-GRPO	88.9	71.4	58.8	70.6	40.7	29.2	60.2	77.3	60.9	27.6	54.5	22.7	27.3	0.0	19.2	21.1	73.4	46.1
GRPO+OPSD	**100.0**	82.4	85.7	75.0	70.0	60.0	81.2	77.8	66.4	38.2	50.0	30.8	28.6	30.0	21.1	32.0	70.7	38.3
Skill-SD	88.2	50.0	96.2	52.4	65.0	57.9	73.4	75.9	64.0	52.9	37.5	69.2	42.9	60.0	36.8	52.3	81.8	53.9
RLSD	87.9	75.0	90.9	75.0	73.1	68.4	79.7	84.4	66.4	50.0	37.5	61.5	19.0	50.0	21.1	42.2	74.0	50.8
SDAR	97.1	62.5	**100.0**	61.9	75.0	84.2	84.4	85.0	68.0	73.5	25.0	76.9	33.3	40.0	36.8	53.9	76.8	58.6
SEED	**100.0**	**100.0**	**100.0**	**100.0**	70.6	80.0	91.8	88.5	78.9	97.6	**100.0**	**100.0**	80.0	84.2	**90.0**	92.0	87.1	77.3
ForeNote (ours)	**100.0**	96.2	98.1	93.8	**86.8**	**92.5**	**94.6**	**90.0**	**81.9**	**98.3**	81.5	**100.0**	**91.9**	**96.0**	86.3	**92.3**	**90.5**	**81.7**

Table 2: **ScienceWorld.** Published Qwen2.5-3B results use their own protocols and report success rate (SR) or score. Our evaluations report means over three inference repeats; sources, protocols, and standard deviations are in Appendix D (Table 5). –: not reported.

Metric	ForeNote (Ours)	GiGPO	GLIDER	Proact Agent	ExpRAG	AgentGym-RL	GPT-5.6 Sol	Gemini 3.1-P	Gemini 3.6-Flash
Solved / SR ↑	59.1	58.1	–	53.5	8.1	22.5	61.4	65.1	63.6
Score ↑	64.5	–	54.7	–	–	–	74.3	77.8	78.5

Environments and metrics. **ALFWorld** (Shridhar et al., 2021) comprises six families of household tasks. We evaluate all 140 seen and 134 unseen games, with a five-step history and a 30-step limit, and report success macro-averaged over task families. Unseen games take place in rooms absent from training. **ScienceWorld** (Wang et al., 2022) comprises 30 types of science tasks. We use a fixed 140-episode grid with the benchmark's `easy` simplifications, a five-step history, and a 100-step limit. We report solved rate (final score 100) and task score (0–100), macro-averaged over task types; failed episodes count as zero, as in the benchmark's official example code. **WebShop** (Yao et al., 2022) requires buying a product that satisfies an instruction. We use a two-step history and a 15-step limit. The common index-based split is not clean: each worker reshuffles the goal list, so 620 of our 1,280 scheduled test episodes (48%) share their instruction with a task used in SFT or drawn during RL. We therefore report score and success on the other 660 episodes, a frozen clean test set covering 610 instructions (Appendix J).

Training. Our main model is Qwen2.5-3B-Instruct (Yang et al., 2024); we also evaluate Qwen3-1.7B (Yang et al., 2025) in non-thinking mode. SFT uses successful expert trajectories from the environments' own solution paths (Appendix C): 2,000 transitions for 750 optimizer steps on ALFWorld and WebShop, and 11,628 transitions for three epochs on ScienceWorld. RL runs for 150 steps, each with 16 tasks and $G = 8$ trajectories per task, at learning rate 10^{-6}. For Qwen2.5-3B on WebShop, SFT notes are near-perfect, so the note reward, which compares sampled notes, rarely varies (Section 2.3). We therefore use a note-loss variant there, which supervises the note with a cross-entropy loss on the current reference note (weight 0.1) and lets the task advantage train only the tokens outside the note (Appendix J). Appendix C gives the full settings.

Comparisons. Table 1 compares ForeNote with the ALFWorld and WebShop results that Wu et al. (2026) report for the same two backbones: GRPO (Shao et al., 2024), skill-based methods, self-distillation methods such as OPSD (Zhao et al., 2026), and SEED, which, like ForeNote, starts RL from an SFT checkpoint. Table 2 adds published ScienceWorld results and three API models evaluated on our grid. Table 3 separates the contributions of SFT and RL.

Table 3: **Training-stage and objective ablations** (Qwen2.5-3B; %; mean ± s.d. over inference repeats). HN: history-note supervision (current note); FP: future prediction (next note). HN+FP SFT + GRPO uses only the task reward in RL. ALFWorld Mean: mean of seen and unseen; WebShop: clean test set. The WebShop ForeNote row uses the note-loss variant (Section 3). Bold: best per column.

	ALFWorld (success)			WebShop		ScienceWorld	
Method	Mean	Seen	Unseen	Score	Success	Solved	Score
Qwen2.5-3B-Instruct	13.5±2.6	14.4±2.5	12.5±2.7	4.2±2.1	1.1±1.3	8.5±1.7	15.0±2.5
+ SFT	57.3±1.5	60.4±2.3	54.3±4.0	85.5±2.6	74.5±5.4	28.3±1.2	49.8±0.4
+ SFT (HN)	56.7±1.2	58.7±2.2	54.6±2.0	90.0±1.8	78.5±5.7	31.5±0.8	56.3±0.4
+ SFT (HN+FP)	57.8±2.4	61.4±2.7	54.1±3.6	**90.5**±2.0	79.6±6.5	29.0±0.7	53.2±0.4
HN SFT + HN-GRPO	93.0±0.8	**95.0**±1.0	91.1±1.8	88.8±2.1	80.7±5.6	51.5±0.4	62.3±0.3
HN+FP SFT + GRPO	78.2±1.2	82.1±2.0	74.4±1.7	89.9±2.3	75.6±6.2	57.9±0.7	**65.8**±0.5
ForeNote (HN+FP)	**94.9**±0.7	94.6±1.2	**95.2**±1.2	90.0±2.7	**81.9**±3.7	**59.1**±0.8	64.5±0.9

Evaluation. Each trained configuration is evaluated with 10 inference repeats on ALFWorld and WebShop and three on ScienceWorld, at temperature 0.4; matched methods share test episodes and sampling seeds. We report mean ± s.d. over repeats and compare configurations with exact sign-flip tests over repeats on ALFWorld and WebShop, game-level bootstrap intervals for the main ALFWorld contrasts, and episode bootstrap intervals on ScienceWorld (Appendix E). ALFWorld uses the final RL checkpoint and ScienceWorld the checkpoint with the highest validation solved rate. On WebShop, the checkpoint and RL rollout temperature of the 3B ForeNote are chosen on validation goals, and the task-reward-only ablation, which collapses late in RL, is evaluated at step 100 (Appendix J).

4 Main Results

ALFWorld. With Qwen2.5-3B-Instruct, ForeNote reaches 94.6% success on seen scenes and 95.2% on unseen scenes (Table 3); a second training run with other seeds reaches 95.7% and 95.3% (Table 12). On seen scenes it exceeds the 91.8% reported for SEED, with the largest margins on Cool and Pick2, SEED's two weakest task families. On unseen scenes, where SEED reports 86.2%, ForeNote keeps its seen-scene level. With Qwen3-1.7B, ForeNote reaches 92.3% on seen scenes, level with SEED's 92.0%, and 85.6% on unseen scenes (Appendix C).

ScienceWorld and WebShop. On ScienceWorld, ForeNote solves 59.1% of tasks (score 64.5); three API models on the same grid solve 61.4–65.1% (score 74.3–78.5; Table 2). On the clean WebShop test set, it scores 90.0 with 81.9% success (Qwen2.5-3B) and 90.5 with 81.7% (Qwen3-1.7B).

On ALFWorld, the gains are made during RL. Table 3 separates the two training stages. On ALFWorld, the three SFT variants reach similar success (56.7–57.8% averaged over seen and unseen scenes): writing notes, or learning to predict them, adds little before RL. After RL, ForeNote reaches 94.9%, whereas RL on the task reward alone from the same SFT checkpoint reaches 78.2%. These gains are made during RL, and they depend on keeping the note signals there (Section 5.2).

5 Why Training the Note Helps

We now ask what the trained note does for the agent (Section 5.1), why it must be supervised during RL (Section 5.2), what the prediction loss adds (Section 5.3), and where these effects matter (Section 5.4). ALFWorld provides the controlled comparisons.

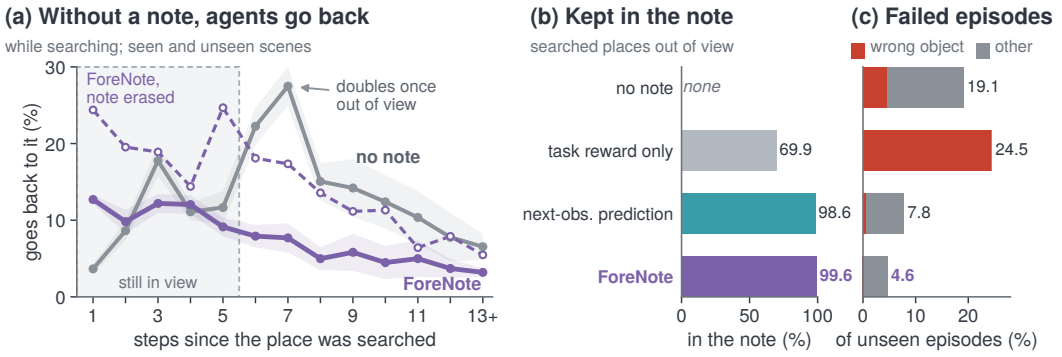


Figure 3: **The trained note remembers places that have left the window** (ALFWorld). (a) While searching for a target object, how often the agent goes back to a place it has already searched, by how many steps earlier it searched it; shaded: the five-step window in the prompt; dashed: ForeNote with its note erased at test time. (b) Searched places outside the window that the note still lists. (c) Share of unseen episodes that fail; red: the agent handled a wrong object. (a) Mean $\pm$ s.d. over 10 inference repeats, (b,c) means; (a,b) pool seen and unseen scenes. Definitions in Appendix E.

5.1 The note records what has left the window

A five-step window tells the agent where it has searched recently, but not earlier. At each step of the search for a target object, we measure how often the agent goes back to a place it has already searched, grouped by how many steps earlier it searched it (Figure 3a). For an agent trained without a note (five-step SFT and GRPO; Table 11), this rate roughly doubles as the place leaves the window: from 11.4% for places searched four or five steps earlier to 24.4% for six or seven (higher in all 10 repeats, $p = 0.002$). ForeNote shows no such rise (10.8% and 7.8%). Its note still lists 99.6% of the searched places that have left the window (Figure 3b), and on unseen scenes it matches the tracker's reference note exactly at 97.7% of steps (Appendix F).

The agent acts on this note. Erasing the carried note before every step at test time roughly doubles how often ForeNote goes back to searched places, even to places still in view (Figure 3a, dashed; Table 6) and lowers success by 4.2 ± 0.8 points on seen and 8.5 ± 2.3 points on unseen scenes (lower in all 10 repeats, $p = 0.002$). Replacing its note with the tracker's reference changes success by only $+0.1$ and $+0.8$ points: its own note is almost as useful to it as the tracker's.

5.2 The note must be supervised during RL

RL on the task reward alone yields a worse agent. From the same SFT checkpoint with the same RL settings, success falls from 94.6% to 82.1% on seen scenes and from 95.2% to 74.4% on unseen scenes (lower in all 10 repeats on both splits, $p = 0.002$). The notes stay well formed but list only about 70% of the searched places at every distance (Figure 3b shows those out of view), and in a third of unseen episodes they log, as completed task events, moves of objects the task does not name. The drop replicates with a second pair of runs on other training seeds (Table 10).

The policy also learns a harmful habit. In 42.5% of unseen episodes, the agent picks up objects the task does not name, often putting them straight back, and every failed unseen episode involves this (Figure 3c). A correct note does not repair this behaviour: carrying the tracker's reference note at every test step leaves success unchanged (81.7% and 74.1%), and agents that also carry it throughout RL reach only 84.1% on unseen scenes with the task reward alone and 75.5% with the note signals, which then train a note the agent never reads (Appendix F). The signals help only when they train the note the policy reads.

Both signals are needed. From the same SFT checkpoint, RL with the note reward alone reaches 90.4% on unseen scenes and RL with the prediction loss alone 83.3%, against 74.4% with neither and 95.2% with both (Table 10).

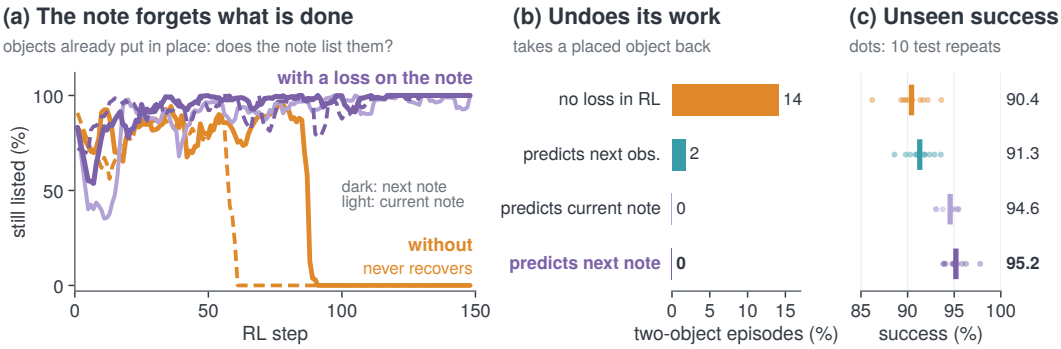


Figure 4: **A loss on the reference note restores what the note reward cannot** (ALFWorld). (a) During RL, the share of objects already put in place that the agent's note lists; dashed: the second run of ForeNote (purple) and the HN run (orange). Once every sampled note leaves them out, the note reward cannot bring them back. (b) Unseen two-object episodes in which the agent takes a placed object back. (c) Unseen success; dots: 10 test repeats.

5.3 What the prediction loss adds

Table 12 isolates the prediction loss. Removing it from RL only, from the same SFT checkpoint, lowers unseen success from 95.2% to 90.4% (4.8 ± 2.2 points, lower in all 10 repeats, $p = 0.002$) and leaves seen success similar (94.2%); without the loss in either stage, history-note supervision reaches 91.1% on unseen scenes. Without the loss, the agent goes back to searched places more often, and its note stops recording objects already put in place, although it still lists 99.1% of the searched places outside the window and the note reward still applies (Figure 4a; Appendix I).

The target of the loss matters. Predicting the next raw observation instead of the next note lowers unseen success to 91.3% (3.9 ± 2.0 points, lower in all 10 repeats, $p = 0.002$) with similar seen-scene success (93.7% and 94.6%). Predicting another game's next note lowers it to 79.4%, below training without the loss. With the current reference note as the RL target, the agent reaches 94.6% on unseen and 92.6% on seen scenes, against 95.2% and 94.6% with the next note.

Why a loss on the note matters. The note reward compares the notes sampled for the same step with one another, so it cannot restore an entry that all of them leave out (Section 2.3). Without the loss, this happens to objects the agent has already put in place: the note stops listing them and never lists them again (Figure 4a). No sampled note lists a placed object after step 88 in the run from the SFT checkpoint of ForeNote (0 of 15,454), nor after step 58 in the HN run (0 of 25,689). A loss on the reference note keeps these entries: 98.7% and 96.6% of them after step 88 in the two runs of ForeNote, and 96.7% with the current note as the target. Trained without the loss from the same SFT checkpoint as ForeNote, the agent undoes its own progress: on unseen two-object tasks, it takes a placed object back in 14.1% of episodes, always after the placement has left its window, whereas ForeNote never does (Figure 4b). A token-level loss on the agent's own reference note, current or next, supplies what a reward that compares responses cannot.

5.4 Where the note signals matter

On ALFWorld, SFT notes are imperfect (note score 0.73 on 48 training games held out for validation), and the note signals matter most on unseen scenes: dropping them costs 20.8 points of unseen versus 12.5 of seen success, and removing the prediction loss from RL 4.8 versus 0.3; erasing the note of ForeNote at test time costs 8.5 versus 4.2.

WebShop needs little memory: agents usually buy after one search and one product page, and SFT notes already match the reference almost perfectly (note score 0.997). After RL, the success and mean score of ForeNote stay close to their SFT values (81.9% and 90.0, from 79.6% and 90.5), and its note loss keeps the note accurate (note score 0.999), whereas without supervision the note score drifts to 0.735 during RL (Table 14).

6 Related Work

Learning language-model agents. ReAct and Reflexion use language to organize interaction and feedback (Yao et al., 2023; Shinn et al., 2023); subsequent work trains multi-turn agents with RL (Zhou et al., 2024; Feng et al., 2025; Xi et al., 2026; Yu et al., 2026b; He et al., 2026; Liu et al., 2026b). Self-distillation transfers reference answers, feedback, or natural-language skills into the policy (Zhao et al., 2026; Yang et al., 2026; Hübotter et al., 2026; Wang et al., 2026b; Lu et al., 2026b); SEED distills hindsight skills the current policy extracts from its own trajectories (Wu et al., 2026). ForeNote instead supervises the agent with a note that a tracker rebuilds from its interactions. Its note reward scores an intermediate output, as in process supervision (Lightman et al., 2024), and is normalized separately from the task advantage (Zhang et al., 2026b; Liu et al., 2026a).

Explicit memory and belief supervision. Agents can replace a growing transcript with a bounded memory or evolving report (Zhou et al., 2026; Yu et al., 2026a; Chen et al., 2026), maintain a textual belief across interactions (Wang et al., 2026a), or detect when a belief deviates during RL (Zou et al., 2026). StateAct prompts an agent to track its state without training (Rozanov & Rei, 2025). Several methods supervise state content directly. Self-notes can be trained on ground-truth notes (Lanchantin et al., 2023); ABBEL checks beliefs against rule-derived statistics (Lidayan et al., 2025); ReBel uses discrepancies between beliefs and observed feedback for dense rewards (Tang et al., 2026); Agent-BRACE trains a belief model with a judge that compares against environment state (Singh et al., 2026); and VAGEN rewards both current-state estimates and future predictions for visual agents (Wang et al., 2025). From History to State uses a hand-written tracker in training and at inference (Xie et al., 2026). ForeNote uses such a tracker only in training; the trained agent writes its own note, which is almost as useful to it as the tracker's (Section 5.1).

Auxiliary prediction and world models. Predictive auxiliary objectives and self-predictive representations are established tools in deep RL (Jaderberg et al., 2017; Schwarzer et al., 2021; Ni et al., 2024). For language agents, world models can predict abstract transitions for inference-time planning (Chae et al., 2025); future states can supervise policies before RL (Zhang et al., 2026a); and next-observation losses or rewards can train the policy during RL (Lu et al., 2026a; Wang et al., 2026c; Yu et al., 2026c). Concurrent work predicts decision-oriented dynamics instead of observations (Cai et al., 2026a), and earlier text-game work learned belief graphs and knowledge-graph world models (Adhikari et al., 2020; Ammanabrolu & Riedl, 2021). A concurrent analysis finds that prediction rewards can favor predictable but unproductive behavior, and that auxiliary losses can help even with non-semantic targets (Wang, 2026). ForeNote instead predicts the agent's own note with a loss rather than a reward, and predicting another game's note hurts (Section 5.3).

7 Discussion and Conclusion

ForeNote treats the note an agent carries as something RL must train, not a scratchpad that the task reward will shape on its own. In the text environments we study, a small tracker can check the note and supply its next value, telling the agent what to remember and how its actions change what it knows. On ALFWorld, the trained agent remembers places that have left its window and acts on them; without these signals, RL leaves a less complete note and a weaker policy, which a correct note given at test time does not repair.

Lessons beyond our agents. First, the state an agent writes should be trained where it is used: supervising a note that the policy does not read, or handing the policy a correct note only at test time, does not reproduce the gains. Second, a reward that compares sampled responses cannot fix an error they all share: under GRPO and other group-relative methods, any scored intermediate output, be it a plan, a belief, or a note, has this blind spot, and a token-level loss on a reference can close it.

Scope and outlook. ForeNote needs a note schema and a reference for each environment. Ours are short rules over text feedback; environments whose feedback is harder to parse may call for learned or model-based references, which need to be available only during training. For agents that must remember, checking the note is a practical complement to the task reward.

AI USE STATEMENT

LLM-based coding assistants were used to help implement and audit experiment, analysis, and figure code, and refine manuscript writing. The authors are responsible for the scientific claims, reported results, citations, and final manuscript.

ETHICS STATEMENT

This work uses publicly available research environments (ALFWorld, ScienceWorld, and WebShop) and open-weight language models. It involves no human subjects or private user data. The agents act only inside simulated environments.

REPRODUCIBILITY STATEMENT

Section 2 defines the agent interface and training objectives. Appendices A and B provide note schemas, prompts, and tracker rules. Appendix C specifies training hyperparameters and compute; Appendix E gives evaluation and statistical procedures. Appendices D and J document benchmark-specific settings and checkpoint selection. Additional component results and training diagnostics appear in Appendices H and I. We will release the code, including the reference trackers, upon acceptance.

REFERENCES

Ashutosh Adhikari, Xingdi Yuan, Marc-Alexandre Côté, Mikuláš Zelinka, Marc-Antoine Rondeau, Romain Laroche, Pascal Poupart, Jian Tang, Adam Trischler, and William L. Hamilton. Learning dynamic belief graphs to generalize on text-based games. In *Advances in Neural Information Processing Systems*, volume 33, pp. 3045–3057, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1fc30b9d4319760b04fab735fbfed9a9-Paper.pdf.

Prithviraj Ammanabrolu and Mark O. Riedl. Learning knowledge graph-based world models of textual environments. In *Advances in Neural Information Processing Systems*, volume 34, pp. 3720–3731, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/1e747ddbea997a1b933aaf58a7953c3c-Paper.pdf.

Guangfeng Cai, Kaibing Yang, Shuo He, Yu Li, Shengtian Yang, Jiaqi Lv, and Lei Feng. Beyond next-observation prediction: Agent-authored world modeling for sequential decision making. *arXiv preprint arXiv:2606.25421*, 2026a.

Yuxuan Cai, Wei Li, Jie Zhou, Qin Chen, Xin Li, Bo Zhang, and Liang He. Ask only when needed: Proactive retrieval from memory and skills for experience-driven lifelong agents. *arXiv preprint arXiv:2604.20572*, 2026b.

Hyungjoo Chae, Namyoung Kim, Kai Tzu-iunn Ong, Minju Gwak, Gwanwoo Song, Jihoon Kim, Sunghwan Kim, Dongha Lee, and Jinyoung Yeo. Web agents with world models: Learning and leveraging environment dynamics in web navigation. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/a00548031e4647b13042c97c922fadf1-Paper-Conference.pdf.

Guoxin Chen, Zile Qiao, Xuanzhong Chen, Donglei Yu, Haotian Xu, Wayne Xin Zhao, Ruihua Song, Wenbiao Yin, Huifeng Yin, Liwen Zhang, Kuan Li, Minpeng Liao, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. IterResearch: Rethinking long-horizon agents with interaction scaling. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/3369a3b268a86804237ef4702ac99ed3-Paper-Conference.pdf.

Juncheng Diao, Zhicong Lu, Peiguang Li, Yongwei Zhou, Changyuan Tian, Qingbin Li, Rongxiang Weng, Jingang Wang, and Xunliang Cai. HIPIF: Hierarchical planning and information folding for long-horizon LLM agent learning. *arXiv preprint arXiv:2606.10507*, 2026.

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, pp. 46375–46408, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/420c9f777c0b4f78d515e53cf74d58b2-Paper-Conference.pdf.

Thomas Palmeira Ferraz, Romain Deffayet, Vassilina Nikoulina, Hervé Déjean, and Stéphane Clinchant. Retrieval-augmented LLM agents: Learning to learn from experience. *arXiv preprint arXiv:2603.18272*, 2026.

Shuo He, Lang Feng, Qi Wei, Xin Cheng, Lei Feng, and Bo An. Hierarchy-of-groups policy optimization for long-horizon agentic tasks. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/2e7eb932f93d5db3b61c7a001e72fd0f-Paper-Conference.pdf.

Zican Hu, Wei Liu, Xiaoye Qu, Xiangyu Yue, Chunlin Chen, Zhi Wang, and Yu Cheng. Divide and conquer: Grounding LLMs as efficient decision-making agents via offline hierarchical reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 24570–24590. PMLR, 2025. URL https://proceedings.mlr.press/v267/hu25q.html.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation. In *International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/64121.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJ6yPD5xg.

Jack Lanchantin, Shubham Toshniwal, Jason Weston, Arthur Szlam, and Sainbayar Sukhbaatar. Learning to reason and memorize with self-notes. In *Advances in Neural Information Processing Systems*, 2023.

Aly Lidayan, Jakob Bjorner, Satvik Golechha, Kartik Goyal, and Alane Suhr. ABBEL: Learning natural-language belief states for memory-efficient interaction. *arXiv preprint arXiv:2512.20111*, 2025.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/aca97732e30bcf1303bc22ac3924fd16-Paper-Conference.pdf.

Xiaoqian Liu, Ke Wang, Yuchuan Wu, Fei Huang, Yongbin Li, Jianbin Jiao, and Junge Zhang. Agentic reinforcement learning with implicit step rewards. In *International Conference on Learning Representations*, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/d1e6739f319be4ba8e748cc05f23bba1-Paper-Conference.pdf.

Zeyuan Liu, Jeonghye Kim, Xufang Luo, Dongsheng Li, and Yuqing Yang. Exploratory memory-augmented LLM agent via hybrid on- and off-policy optimization. In *International Conference on Learning Representations*, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/b589d92785e39486e978fa273d0dc343-Paper-Conference.pdf.

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

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

Tianwei Ni, Benjamin Eysenbach, Erfan Seyedsalehi, Michel Ma, Clement Gehring, Aditya Mahajan, and Pierre-Luc Bacon. Bridging state and history representations: Understanding self-predictive RL. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/666c1861d709bd84e20b6e0e02a2c223-Paper-Conference.pdf.

Sangeun Park and Minhae Kwon. Multi^2: Hierarchical multi-agent decision-making with LLM-based agents in interactive environments. In *International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/65074.

Nikolai Rozanov and Marek Rei. StateAct: Enhancing LLM base agents via self-prompting and state-tracking. In *Proceedings of the 1st Workshop for Research on Agent Language Models (REALM 2025)*, pp. 367–385, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.realm-1.27. URL https://aclanthology.org/2025.realm-1.27/.

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

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2021. URL https://iclr.cc/virtual/2021/poster/2797.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf.

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://iclr.cc/virtual/2021/poster/2973.

Joykirat Singh, Zaid Khan, Archiki Prasad, Justin Chih-Yao Chen, Akshay Nambi, Hyunji Lee, Elias Stengel-Eskin, and Mohit Bansal. Agent-BRACE: Decoupling beliefs from actions in long-horizon tasks via verbalized state uncertainty. *arXiv preprint arXiv:2605.11436*, 2026.

Wenjie Tang, Minne Li, Sijie Huang, Liquan Xiao, and Yuan Zhou. Rewarding beliefs, not actions: Consistency-guided credit assignment for long-horizon agents. *arXiv preprint arXiv:2605.20061*, 2026.

Hanlin Wang, Chak Tou Leong, Jian Wang, and Wenjie Li. Seeing isn't believing: Mitigating belief inertia via active intervention in embodied agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 37796–37815, San Diego, California, United States, July 2026a. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.1884. URL https://aclanthology.org/2026.findings-acl.1884/.

Hao Wang, Guozhi Wang, Han Xiao, Yufeng Zhou, Yue Pan, Jichao Wang, Ke Xu, Yafei Wen, Xiaohu Ruan, Xiaoxin Chen, and Honggang Qi. Skill-SD: Skill-conditioned self-distillation for multi-turn LLM agents. *arXiv preprint arXiv:2604.10674*, 2026b.

Kangrui Wang, Pingyue Zhang, Zihan Wang, Yaning Gao, Linjie Li, Qineng Wang, Hanyang Chen, Chi Wan, 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, pp. 172871–172933, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/fc6688d75adde86b9df910769c1d02e3-Paper-Conference.pdf.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.775. URL https://aclanthology.org/2022.emnlp-main.775/.

Yu Wang. The dark room in the reward channel: Dense prediction rewards collapse GRPO-trained LLM agents – and the channel, not the content, decides what works. *arXiv preprint arXiv:2607.21273*, 2026.

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

Jinyang Wu, Shuo Yang, Zhengxi Lu, Fan Zhang, Yuhao Shen, Lang Feng, Haoran Luo, Zheng Lian, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. SEED: Self-evolving on-policy distillation for agentic reinforcement learning. *arXiv preprint arXiv:2607.14777*, 2026.

Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Jiaqi Liu, Honglin Guo, Yajie Yang, Rui Zheng, Junjie Ye, Jiazheng Zhang, Wenxiang Chen, Wei He, Yiwen Ding, Guanyu Li, Zehui Chen, Zhengyin Du, Xuesong Yao, Yufei Xu, Jiecao Chen, Tao Gui, Zuxuan Wu, Qi Zhang, Xuanjing Huang, and Yu-Gang Jiang. AgentGym-RL: An open-source framework to train LLM agents for long-horizon decision making via multi-turn RL. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/1571ce1ff735be4551db50887043726e-Paper-Conference.pdf.

Haoyang Xie, Xinyuan Wang, Yancheng Wang, Puda Zhao, and Feng Ju. From history to state: Constant-context skill learning for LLM agents. *arXiv preprint arXiv:2605.05413*, 2026.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

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

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

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, pp. 20744–20757, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/82ad13ec01f9fe44c01cb91814fd7b8c-Paper-Conference.pdf.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://iclr.cc/virtual/2023/poster/11003.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and Hao Zhou. MemAgent: Reshaping long-context LLM with multi-conv RL-based memory agent. In *International Conference on Learning Representations*, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/4264ee4376776907c0b87ed70b959585-Paper-Conference.pdf.

Xiao Yu, Baolin Peng, Michel Galley, Hao Cheng, Qianhui Wu, Janardhan Kulkarni, Suman Nath, Zhou Yu, and Jianfeng Gao. Dyna-Mind: Learning to simulate from

experience for better AI agents. In *International Conference on Learning Representations*, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/372bd0e47f2d5bceca7e300e1446849c-Paper-Conference.pdf.

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

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, Jian Xie, Yuxuan Sun, Boyu Gou, Qi Qi, Zihang Meng, Jianwei Yang, Ning Zhang, Xian Li, Ashish Shah, Dat Huynh, Hengduo Li, Zi Yang, Sara Cao, Lawrence Jang, Shuyan Zhou, Jiacheng Zhu, Huan Sun, Jason Weston, Yu Su, and Yifan Wu. Agent learning via early experience. In *International Conference on Machine Learning*, 2026a. URL https://icml.cc/virtual/2026/poster/64488.

Zijing Zhang, Ziyang Chen, Mingxiao Li, Zhaopeng Tu, and Xiaolong Li. RLVMR: Reinforcement learning with verifiable meta-reasoning rewards for robust long-horizon agents. In *International Conference on Learning Representations*, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/baa41b7368408670c6a14e06a04420d9-Paper-Conference.pdf.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. In *International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/64784.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. ArCHer: Training language model agents via hierarchical multi-turn RL. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62178–62209. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhou24t.html.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Bryan Kian Hsiang Low, and Paul Pu Liang. MEM1: Learning to synergize memory and reasoning for efficient long-horizon agents. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/5fc8b3bdfbb9167b5144df5d3fae4616-Paper-Conference.pdf.

Deyu Zou, Yongqiang Chen, Jianxiang Wang, Haochen Yang, Mufei Li, Qing Da, James Cheng, Pan Li, and Yu Gong. Reducing belief deviation in reinforcement learning for active reasoning of LLM agents. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/2468f84a13ff8bb6767a67518fb596eb-Paper-Conference.pdf.

A History Note Schemas and Prompts

Schemas. In every environment the fields fall into three groups, each recording something a short window can lose: what the agent has already *tried*, where its *targets* are, and which task *milestones* are confirmed. Notes are canonical JSON objects with a fixed key order, compact separators, sorted and de-duplicated list fields, and at most 256 tokens under the Qwen2.5 tokenizer. All fields describe the state before the next action.

- **ALFWorld** (key order `checked`, `open`, `found`, `holding`, `done`). Tried: `checked`, receptacles whose contents were explicitly shown, and `open`, containers observed open and not closed since. Targets: `found`, target-type objects mapped to their last explicitly seen location. Milestones: `holding`, the object confirmed as picked up, or `null`, and `done`, confirmed task events (`clean`, `heat`, `cool`, `light`, `place`).
- **ScienceWorld** (key order `room`, `inventory`, `visited`, `focus`, `facts`). Tried: `visited`, rooms entered, with `room`, the room last confirmed. Targets: `focus`, the object of the last confirmed `focus on` action. Milestones: `inventory`, objects confirmed in the inventory, and `facts`, a key–value map of confirmed facts (device power, container states, last container of a moved object, the last instrument reading per target, and connections), where a later confirmation overwrites an earlier one so that long episodes stay within the token limit.
- **WebShop** (key order `query`, `seen`, `candidate`, `options`, `budget_ok`). Tried: `query`, the last search that produced a result page, and `seen`, product ids whose item page was opened. Targets: `candidate`, the product currently displayed. Milestones: `options`, the option selected in each option group of that product (which the page itself does not display), and `budget_ok`, whether its lowest displayed price satisfies the task's price limit.

Prompt. The note block is inserted into the standard agent prompt of each environment, before the instruction to act. For ALFWorld it reads:

```
The carried state below was written before the most recent action and current observation; at
     step 1 it is the empty initial state. It may be stale.
<note>{"checked":[...],"open":[...],"found":{...},"holding":...,"done":[...]}</note>
Using only the task, visible interaction history, current observation, and admissible actions
     above, update all five fields for the current pre-action state. Do not predict the result
      of the action you are about to choose. The updated note must be compact JSON with keys
     in this exact order: checked, open, found, holding, done, and must stay within 256 tokens
     .
Return exactly one <think>...</think> block, then exactly one <note>{...}</note> block, then
     exactly one <action>...</action> block. The action must be selected from the admissible
     actions.
```

Example. In a task *put two keychains in armchair*, after the agent has placed one keychain and arrived at the coffee table, the reference note is

```
{"checked":["armchair 1","coffeetable 1","sidetable 1"],"open":[],"found":{"keychain 2":"
     coffeetable 1","keychain 3":"armchair 1"},"holding":null,"done":["place:keychain 3>
     armchair 1"]}
```

and the expert action is `take keychain 2 from coffeetable 1`. The future-prediction target for this step is the reference note after the pick-up is confirmed, in which `holding` becomes `keychain 2`.

B Reference Tracker and Note Score

Tracker. The tracker U receives only text the agent has seen: the previous reference note, the previous action, the new observation, the admissible actions, and the task text. For ALFWorld it applies the following rules, each triggered by explicit feedback text:

- *Arriving at, opening, or examining* a receptacle whose contents are listed adds it to `checked`; "is open" adds a container to `open`, and "is closed" or "you close" removes it.
- When contents are shown, every listed object whose type matches a target type of the task is recorded in `found` with the current location.

- A pick-up message that matches the executed `take` action sets `holding`; a move message that matches the executed `move` action clears `holding`, updates `found`, and adds a `place` event to `done` if the destination has the task's destination type.
- Clean, heat, cool, and light-switch messages that match both the executed action and the task add the corresponding event to `done`.

Failed actions ("Nothing happens.") change nothing. The ScienceWorld tracker follows the same design and does not use the task text:

- A move message ("You move to the kitchen.") or a room description ("This room is called the kitchen.") sets `room` and adds the room to `visited`.
- "You move the *X* to the inventory." adds *X* to `inventory`; "You move the *X* to the *Y*." removes *X* from `inventory` and records *Y* as its location in `facts`; "You focus on the *X*." sets `focus`.
- Each confirmed state change writes one key of `facts`: device power, container state, pouring, mixing products, wire connections, and instrument readings. A thermometer reading is stored under the target of the executed `use` action, because the message does not name it. Keys are unique, so a later confirmation overwrites an earlier one.
- Messages that match none of these patterns, including failures such as "No known action matches that input.", change nothing. Names are cut to 60 characters and lists and `facts` to 40 entries, and a note over 256 tokens drops entries in a fixed order, object locations first; no reference note on the expert trajectories reaches these bounds.

The WebShop tracker also reads the clickable elements of the page, which the agent sees as its admissible actions:

- The page type is read from the displayed page: search page, result list, item page, an item's description, features, or reviews page, or purchase confirmation.
- A result list reached by `search[`q`]` sets `query` to q. A result list, search page, or purchase page clears `candidate`, `options`, and `budget_ok`; description, features, and reviews pages change nothing.
- An item page reached by clicking a product id sets `candidate`, empties `options`, and adds the id to `seen`. A click on an option value of that page records the value under its option group, which the tracker recovers from the page layout and the clickable elements. `budget_ok` compares the lowest displayed price with the task's price limit and is `null` if either is missing.
- WebShop shows no failure message: an invalid action leaves the page, and hence the note, unchanged, except that a product id clicked on an item page becomes `candidate` even if the click has no effect.

The update functions have 78 (ALFWorld), 81 (ScienceWorld), and 45 (WebShop) lines of Python, not counting helpers that parse the task text and, for WebShop, the page.

Note score. A response is *canonical* if it has exactly one `think`, `note`, and `action` block, the note is canonical JSON within 256 tokens, and the action is non-empty. For a canonical note we convert each field to a set of atoms (list entries, key–value pairs of `found`, or the held object) and compute, per field, the F1 (Dice) coefficient $2\,|P \cap R|/(|P| + |R|)$ between predicted atoms P and reference atoms R. A field is active if $P \cup R \neq \emptyset$. The note score is the mean Dice over active fields, 1 if both notes are entirely empty, and 0 for non-canonical responses.

C Training Details

Expert data. Expert trajectories are the environments' own solution paths: ALFWorld's PDDL walkthroughs, a shortest action path to a highest-scoring product on WebShop, and ScienceWorld's gold action sequences. The tracker adds the reference notes.

No-note SFT+GRPO baseline. This baseline has no notes: it starts GRPO on the task reward from full-history SFT on the expert actions without notes (the SFT row of Table 3), with the same expert data and RL budget as our method, and it sees the same short window as our agents during

Table 4: Training hyperparameters (Qwen2.5-3B-Instruct; backbone-specific details below). Mini-batch sizes count transitions.

	ALFWorld	WebShop / ScienceWorld
Supervised fine-tuning		
Transitions (validation)	2,000 (256)	2,000 (256) / 11,628 (4,070)
Steps; batch size; learning rate	750; 8; 5×10^{-6}	750 / 4,359[a]; 8; 5×10^{-6}
Maximum sequence length	4,096	8,192 / 16,384
Reinforcement learning		
RL steps	150	150 / 150[b]
Tasks per step $\times$ rollouts	16×8	16×8
Learning rate; KL α; entropy coefficient	10^{-6}; 0.01; 0.001	10^{-6}; 0.01; 0.001
Epochs per RL step; mini-batch size	1; 256	1; 64
Sampling temperature (note variants)	0.4	0.4 / 1.0 (App. D)
Sampling temperature (no-note SFT+GRPO)	1.0	1.0
Window h; step limit	5; 30	2; 15 / 5; 100
Prompt tokens; response tokens	2,048; 512	4,096 / 5,120; 512
Invalid-action penalty	0.1	0.1
Note reward weight λ; future-prediction weight β	0.5; 0.1	0.5[c]; 0.1
Outcome reward	10$\times$success	10$\times$success / 10$\times$solved

[a]Three epochs; the checkpoint with the lowest validation loss is used. [b]The checkpoint with the highest validation solved rate is tested (Appendix D). [c]Qwen2.5-3B ForeNote on WebShop: $\lambda = 0$ and a note cross-entropy with weight 0.1 on eight transitions per optimizer update (Appendix J).

RL and evaluation; the no-note run of Table 11 instead uses five-step SFT. On ALFWorld it reaches 90.7% and 84.2% on seen and unseen scenes; HN improves on it by 4.2 and 6.9 points and ForeNote by 3.8 and 11.0 points ($p < 10^{-4}$ for all four; Appendix E). On the clean WebShop test set it reaches a score of 92.6 and 84.3% success, and both note variants stay below it (ForeNote: -2.5 ± 3.0 score points, $p = 0.02$; HN: -3.8 ± 3.4, $p = 0.006$; Appendix J). On ScienceWorld it solves 48.0% of the tasks with a score of 59.1, against 59.1% and 64.5 for ForeNote (Appendix D).

Advantages. Let R_i be the outcome reward of trajectory i (Table 4). Step t of trajectory i scores $R_i - 0.1\, v_{i,t}$, where $v_{i,t} = 1$ if its action is invalid, and its task advantage is this score minus the mean score over all steps of the task's G trajectories; on ALFWorld it is additionally divided by their standard deviation plus $\epsilon = 10^{-6}$, the same ϵ as in Equation 2. This is the episode-level advantage of GiGPO (Feng et al., 2025) with the step-level term switched off, so no discount enters it. The policy-gradient loss $\mathcal{L}_{\mathrm{PG}}$ of Equation 5 also contains an entropy bonus with coefficient 0.001. Within each environment, the no-note baseline and note variants use the same task-advantage definition, except for the Qwen2.5-3B WebShop variant described below. Note scores are grouped by task and step index, standardized on ALFWorld and WebShop, and centred on ScienceWorld. Except for the Qwen2.5-3B WebShop variant, both advantages apply to all response tokens. In that variant, note tokens are excluded from the policy-gradient and KL losses, which are averaged over the tokens outside the note; a step with an invalid action cannot receive a positive task advantage. WebShop's task reward optimizes full-score purchases, while evaluation also measures partial score.

Future prediction. The target follows the policy response and end-of-sequence token, introduced by a fixed delimiter and terminated by another end-of-sequence token. Cross-entropy is averaged over target tokens, which are excluded from the policy-gradient mask. During RL, the loss uses 16 on-policy transitions per optimizer update in each environment, with weight $\beta = 0.1$. The next-observation variant changes only the prediction target, in both SFT and RL. Neither variant generates a prediction during rollout or evaluation.

Sampling temperature. The no-note SFT+GRPO baselines use rollout temperature 1.0 in all three environments. Note variants use 0.4 on ALFWorld and WebShop and 1.0 on ScienceWorld; the runs of Table 11, including the one without a note, use 0.4 and one GPU. For Qwen2.5-3B ForeNote on WebShop, temperature and checkpoint are selected on validation from the candidates described

in Appendix J. Trained policies are evaluated at temperature 0.4; API-model settings are given in Appendix D.

Qwen3-1.7B. These runs use the same expert data, training budgets (SFT step 750 and RL step 150), and evaluation grids as Qwen2.5-3B. On ALFWorld, the Qwen3-1.7B agent reaches $92.3 \pm 1.5\%$ on seen and $85.6 \pm 1.9\%$ on unseen scenes. Qwen3's non-thinking template pre-fills an empty reasoning block; the SFT targets contain the note and action in the corresponding format. Both ALFWorld and WebShop use the note-reward objective ($\lambda = 0.5$) and FP loss ($\beta = 0.1$). The Qwen2.5-3B WebShop current-note cross-entropy variant is not used for Qwen3. No clean-test instruction is used for training or validation; the Qwen3 checkpoints are fixed by training step.

Compute. ALFWorld and WebShop note runs used one H100 GPU per RL run, as did the WebShop SFT+GRPO baseline; the ALFWorld SFT+GRPO baseline and the original WebShop one (Table 13) used eight. ScienceWorld note runs used four, and the ScienceWorld baseline used two with parameter offloading. Measured as GPUs times wall-clock time (for the eight-GPU baselines, from their training logs), SFT of Qwen2.5-3B took 1–2 H100-hours on ALFWorld and WebShop and 14–23 on ScienceWorld, where it used two GPUs. An RL run took 11–26 H100-hours on ALFWorld and WebShop (ForeNote: 19.0 and 13.8), 98–123 for the note runs on ScienceWorld (ForeNote: 123), and about 56 for SFT+GRPO there, plus 4–16 H100-hours of checkpoint-selection validation on ScienceWorld and 3.8 for ForeNote on WebShop. The Qwen3-1.7B runs of ForeNote took 2.2 and 1.9 H100-hours of SFT and 17.6 and 11.9 of RL on ALFWorld and WebShop. Evaluating one checkpoint over all repeats took 0.5–2.2 H100-hours on ALFWorld, 1.0–1.3 on WebShop, and 1.0–1.5 on ScienceWorld. These figures exclude profiling runs and failed attempts.

D SCIENCEWORLD

Published comparisons and API baselines. Table 2 retains each literature result's protocol: GiGPO is the 211-episode, 40-turn re-evaluation of Diao et al. (2026); GLIDER (Hu et al., 2025) is the single-attempt score, on in-distribution goal templates, of the reimplementation by Park & Kwon (2026); ProactAgent adapts policy and memory on an online test stream (Cai et al., 2026b); ExpRAG uses dynamic retrieval with $K = 4$ and an easy index on 149 episodes (Ferraz et al., 2026); AgentGym-RL reports the share of tasks solved on eight task subsets with 20 turns (Xi et al., 2026). These rows are not measurements on our grid.

Our API baselines use the ReAct prompt, public observations and admissible actions, the same 140 episodes, `easy` simplifications, a five-step history, and 100 environment steps. Each has three inference repeats. Gemini-3.1-P and Gemini-3.6-Flash use temperature 0.4; GPT-5.6 Sol requires 1.0. Requests set `max_tokens`=4,096 and retain provider reasoning defaults, whereas trained policies use a 512-token generation cap. Gemini reasoning usage is reported separately and can exceed the request cap. Up to three format attempts are allowed. Unrecovered API errors are not scored as environment failures; incomplete episodes are rerun under the same seed rule. Task sets and environment horizons are matched, while generation budgets and GPT temperature differ.

Table 5: **ScienceWorld evaluation variability.** Mean $\pm$ s.d. over three inference repeats on the same 140-episode grid ($n = 420$ episodes per method).

Metric	ForeNote (Ours)	GPT-5.6 Sol	Gemini-3.1-P	Gemini-3.6-Flash
Solved ↑	59.1±0.8	61.4±2.5	65.1±3.3	63.6±3.7
Score ↑	64.5±0.9	74.3±1.8	77.8±2.7	78.5±2.5

Metrics. ScienceWorld scores progress through required and optional subgoals (Wang et al., 2022). We report scores on a 0–100 scale and count an episode as solved when its final score reaches 100. An incorrect focus or answer can end the episode with simulator score -100; we map this to zero before averaging, following the official example code.[1] Within each repeat, solved rates and scores are macro-averaged over the 29 task types of our grid, which omits the benchmark's use-thermometer task.

[1] `https://github.com/allenai/ScienceWorld`, `examples/random_agent.py`.

Results. The HN and ForeNote rows in Table 3 use rollout temperature 1.0. HN resets a malformed carried note to the empty note; ForeNote and its task-reward-only ablation retain the previous valid note. Relative to ForeNote, the same-SFT task-reward-only ablation changes solved rate by -1.1 points (95% CI $[-6.2, 4.4]$) and score by $+1.3$ ($[-3.7, 6.7]$), using a within-task episode bootstrap. Relative to the no-note SFT+GRPO baseline, the complete recipe improves solved rate by 11.0 points ($[4.4, 17.2]$) and changes score by $+5.4$ ($[-0.7, 11.1]$).

For the selected SFT checkpoints, HN solves 31.5% with score 56.3 and HN+FP solves 29.0% with score 53.2. These use the carry rules of their respective RL policies. With the reset rule applied to both, adding FP at SFT changes solved rate by -2.3 points ($[-6.4, 2.1]$) and score by -3.2 ($[-6.7, 0.2]$). As on ALFWorld, stronger final performance is not explained by an SFT-stage task gain.

Checkpoint selection. Validation uses 87 simulator variations (three per task type) with two repeats, every 10 RL steps for note agents and every 50 steps for the no-note baseline. We select the checkpoint with highest validation solved rate. For ForeNote, configuration selection considered temperature 0.4 with either carry rule and temperature 1.0 with the keep rule; the last setting was selected by validation. Its solved rate was 42–53% from step 50 onward, versus at most 44% for either 0.4 configuration. The main HN row uses temperature 1.0. The selected and final checkpoints are:

Method	Selected step	Selected solved / score	Final solved / score
SFT + GRPO	100	48.0 / 59.1	49.9 / 61.0
HN	150	51.5 / 62.3	51.5 / 62.3
HN+FP SFT + GRPO	140	57.9 / 65.8	57.5 / 65.1
ForeNote	130	59.1 / 64.5	57.7 / 63.9

All cells average three inference repeats; final checkpoints are at step 150.

E Evaluation Protocol and Statistics

Repeats and pairing. Repeat i uses sampling seed $2025 + i$. ALFWorld repeats cover all 140 seen or 134 unseen games (three repeats for the base model, ten for trained policies). WebShop repeats use the frozen clean subset of a shared 10-by-128 schedule (59–78 clean episodes per repeat). We compute paired differences per repeat before rounding, so a printed difference can differ by 0.1 from the difference of two printed means, and test them with exact two-sided sign-flip tests; with ten repeats the minimum attainable p is 0.002. Resampling games within task family instead (10,000 draws, pairing preserved) gives 95% intervals. Tables 7, 10, and 12 list them for their contrasts; for the unseen-scene advantage of ForeNote over RL on the task reward alone with the same seeds and over its own erased note they are $[14.7, 27.1]$ and $[5.6, 11.6]$ points, and $[7.0, 18.3]$ and $[1.9, 6.8]$ on seen scenes. ScienceWorld has three repeats and uses episode bootstrap intervals, resampling episodes within task type and preserving method pairing.

Evaluation uses temperature 0.4, top-p 0.8, top-k 20, and repetition penalty 1.05. The latter three values come from Qwen2.5's generation configuration and are explicitly matched for Qwen3. The ALFWorld evaluation of the no-note SFT+GRPO baseline (Appendix C) uses a different game–seed grid, so its contrasts use a difference of means and an exact two-sided permutation test over two sets of ten repeats.

Comparison with published results. The published rows of Table 1 are transcribed from Wu et al. (2026). Their RL budget is 150 steps with 16 tasks and 8 rollouts per task. We match the history windows, environment step limits, and evaluation temperature, but use expert-trajectory SFT before RL and our own implementation. Our GRPO-from-base reproduction reaches 77.4% seen success on ALFWorld and 86.3 score / 78.9% success on clean WebShop, compared with published values of 75.0% and 79.8 / 63.3. Figure 1 also shows the unseen-scene success that Wu et al. (2026) report for GRPO and SEED (70.9% and 86.2%). In Table 1, OPSD, RLSD, Skill-SD, and SDAR are the methods of Zhao et al. (2026), Yang et al. (2026), Wang et al. (2026b), and Lu et al. (2026b); Skill-Prompt, Skill-GRPO, and GRPO+OPSD are baselines defined by Wu et al. (2026).

For every copied row except SEED's, the printed ALFWorld average differs from the mean of the printed family values (e.g., 75.0 vs. 70.7 for 3B GRPO). We retain the source values. Our own rows use the unweighted mean over six families; episode-weighted seen success is 94.9% for Qwen2.5 and 93.9% for Qwen3. Skill-Prompt has access to a natural-language skill at evaluation.

Carrying notes at evaluation. The main ALFWorld post-RL comparisons use one evaluator, carrying a canonical note even after a failed or invalid action and retaining the previous note after a malformed response. All post-RL ALFWorld evaluations of our agents in Tables 3, 6, 7, 10, and 12, and the Qwen3 agent, share the same game–seed grid. The history-supervision runs of Table 11 and note-SFT evaluations use an earlier evaluator with different invalid-response handling; for an HN checkpoint evaluated with both, the two evaluators differ by less than one point on each split. The no-note baseline evaluator retries malformed responses up to three times. ScienceWorld's method-specific carry rules are stated in Appendix D.

Prompt length on ScienceWorld. With the five-step window, the longest prompt of a test episode has a median of 1,209 tokens (GRPO from the base model, 420 episodes); rendering the full history would give a median of about 4,655 tokens at the last step and at most 16,153. On average, about 58 steps of an episode lie outside the five-step window.

Behaviour metrics. Figure 3 is computed from the evaluation episodes with the reference tracker, which reads only text visible to the agent; shares of episodes pool the episodes of all task families, whereas success rates in the tables average over families. A place counts as *searched* once an observation has shown its contents. A *search step* is one at which the reference note holds nothing and lists no target object that remains to be placed. At each search step, every searched place is one opportunity, labelled by how many steps earlier its contents were last shown, and the step is a return to that place if its action is `go to` that place; panel (a) divides returns by opportunities, excluding the light source that a look task still needs. The prompt shows the last five observations, so places shown up to five steps earlier are still in view. In panel (b), a searched place is kept if the note the agent writes at that step lists it; a malformed note counts as empty. In panel (c), a failed episode involves a *wrong object* if the agent took an object whose type the task does not name, or put an object back where it had just taken it from. The agent without a note is the no-note run of Table 11; its evaluation uses the same games and seeds and produced no malformed responses. For the note interventions (Appendix F), the evaluator carries either the empty note (note erased) or the tracker's reference note into every step; the agent still writes a note, which is scored but not carried, and the weights, evaluator, games, seeds, and sampling settings are those of the corresponding cell.

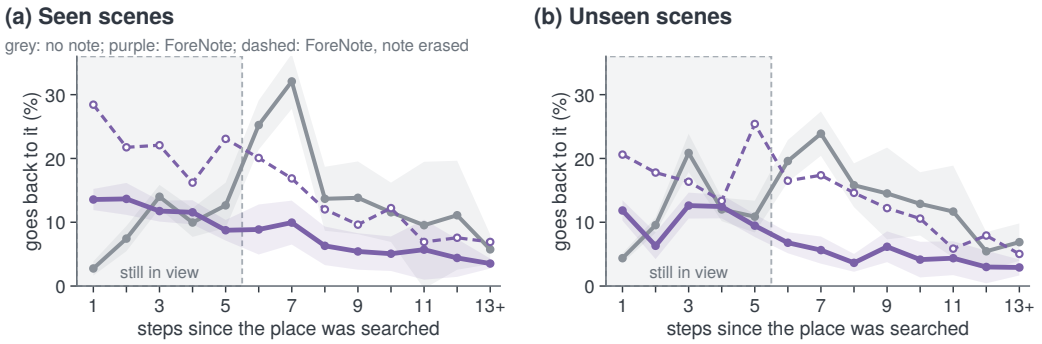


Figure 5: **Figure 3a by scene type.** Same definitions and agents; bands: s.d. over 10 inference repeats.

Figure 5 shows Figure 3a separately for seen and unseen scenes. Without a note, the rate of going back rises from 11.2% to 28.1% on seen scenes and from 11.5% to 21.4% on unseen scenes between places searched four or five and six or seven steps earlier (both higher in all 10 repeats, $p = 0.002$); neither ForeNote line rises.

Placements and re-takes. Figure 4a uses the training rollouts of five RL runs: three with a loss on the reference note (ForeNote, its second run with other training seeds, and the current-note run) and two without it (one from the SFT checkpoint of ForeNote, sharing its first batch of rollouts, and the HN run). At each RL step, the completed placements are the `place:` entries of the reference note's `done` field; we count the share that the sampled response's note lists, over well-formed notes, pooled over 5-step windows. A note-reward group holds the responses for one task at one step index; after the last recorded placement (steps 88 and 58), every response in every group that contains placements omits them (2,412 and 3,876 groups). In Figure 4b, an episode re-takes a placed object if the agent takes an object of a target type from the receptacle where it had put that object; each agent plays 17 unseen two-object games 10 times. On seen scenes, the agent trained without the loss from the SFT checkpoint of ForeNote re-takes a placed object in 11.7% of two-object episodes, all more than five steps after placing it, and ForeNote in 2.5%. Figure 4c shows, for each of the 10 test repeats, the unseen success reported in Section 5.3.

F Test-Time Note Interventions

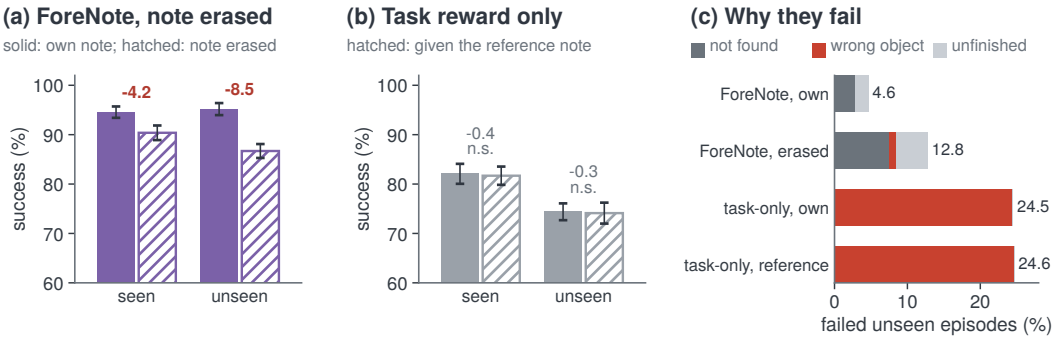


Figure 6: **Changing the carried note at test time** (ALFWorld). (a) ForeNote with its own note (solid) and with the carried note erased before every step (hatched). (b) The task-reward-only agent with its own note (solid) and with the tracker's reference note carried instead (hatched). Numbers above the bars are paired differences over 10 inference repeats; n.s.: $p > 0.05$. (c) Failed unseen episodes by cause (Appendix E).

Each intervention keeps the weights, evaluator, games, seeds, and sampling settings of the corresponding cell and changes only the note carried into each step. Erasing the note of ForeNote lowers success by 4.2 ± 0.8 points on seen and 8.5 ± 2.3 points on unseen scenes (lower in all 10 repeats, $p = 0.002$); most of the added unseen failures are searches that never find the target object (Figure 6c). With the carried note erased, the note ForeNote writes lists only 34% of the searched places still in view, although the window shows them. Carrying the tracker's reference note in place of the note ForeNote writes changes its success by $+0.1 \pm 1.3$ points on seen ($p = 0.77$) and $+0.8 \pm 1.1$ points on unseen scenes (higher in 7 of 10 repeats, $p = 0.02$). Carrying the reference note leaves the task-reward-only agent unchanged: success changes by -0.4 ± 2.2 and -0.3 ± 1.7 points ($p = 0.59$ and 0.61), and 24.6% of its unseen episodes still fail after it handles a wrong object (24.5% with its own note). A correct state at test time does not undo what task-reward-only RL has done to the policy.

Erasing the note of other agents. Table 6 repeats the erasure for the HN agent of Table 3 and the next-observation agent of Table 12. Without its note, each agent loses seen success (4.2, 11.9, and 4.2 points for HN, the next-observation agent, and ForeNote; lower in all 10 repeats, $p = 0.002$). On unseen scenes, only the two agents trained with a prediction loss lose much: 13.0 ± 3.9 and 8.5 ± 2.3 points (lower in all 10 repeats, $p = 0.002$), against 0.6 ± 2.1 for HN ($p = 0.34$). Even with its own note, HN goes back to searched places out of view more often than ForeNote (13.0% vs. 5.3%). Without its note, the next-observation agent goes back to places still in view more than twice as often, and 14.6% of its unseen episodes never find the target object (3.9% with its note).

Reliance on the note does not by itself explain success: the next-observation agent relies on its note the most, yet with its own note it succeeds less often than ForeNote.

Table 6: **Erasing the carried note at test time** (ALFWorld; %, mean over 10 repeats). Success with the agent's own note and with the carried note erased before every step (paired difference in parentheses); going back to a searched place still in view or out of view, seen and unseen scenes pooled (Appendix E); unseen episodes that never find the target object.

		Success		Goes back		Not found
Agent	Note	Seen	Unseen	In view	Out of view	(unseen)
HN	own	95.0	91.1	11.5	13.0	3.7
	erased	90.8 (−4.2)	90.5 (−0.6)	17.4	17.2	4.6
Next observation	own	93.7	91.3	16.1	5.3	3.9
	erased	81.9 (−11.9)	78.3 (−13.0)	36.4	7.0	14.6
ForeNote	own	94.6	95.2	11.4	5.3	2.8
	erased	90.4 (−4.2)	86.7 (−8.5)	21.0	11.9	7.5

Carrying the tracker's note during training. Two control runs repeat the RL of ForeNote from the same SFT checkpoint, seeds, and schedule, but carry the tracker's reference note of each step's pre-action state into the next prompt during training rollouts, validation, and testing; the agent still writes a note, which is never carried. One run keeps the objectives of ForeNote, the other trains on the task reward alone (Table 7, final checkpoints). With the objectives of ForeNote, the agent reaches 91.3% on seen and 75.5% on unseen scenes, level with task-reward-only RL with the same seeds on unseen scenes, although its own notes match the reference on 99.4% of unseen steps; 23.9% of its unseen episodes fail after it handles objects the task does not name (task-reward-only RL: 24.5%; ForeNote: 0%). With the task reward alone, the agent reaches 90.8% and 84.1%. On unseen scenes, this is 9.8 points above the first task-reward-only run, level with the second (83.7%; Appendix H), and 11.0 points below ForeNote; 14.7% of its unseen episodes again fail after it handles objects the task does not name. A correct note in the input, even throughout RL, does not reproduce the effect of supervising the note the agent writes and carries.

Table 7: **The tracker's note in place of the agent's own** (ALFWorld; success %, mean ± s.d. over 10 inference repeats). RL objectives and the note carried into each prompt (test: at test time only; RL+test: also during RL); differences to ForeNote are paired over repeats (repeats higher, exact sign-flip p) with game-level 95% intervals (Appendix E).

				Δ seen		Δ unseen	
Objectives	Carried note	Seen	Unseen		95% int.		95% int.
ForeNote	own	94.6±1.2	95.2±1.2	–	–	–	–
ForeNote	tracker, test	94.7±0.8	95.9±1.3	+0.1 (6/10, .77)	[−0.4, 0.7]	+0.8 (7/10, .02)	[0.0, 1.6]
ForeNote	tracker, RL+test	91.3±2.5	75.5±1.9	−3.3 (1/10, .008)	[−7.4, 0.5]	−19.7 (0/10, .002)	[−25.9, −13.8]
Task only	tracker, RL+test	90.8±1.1	84.1±1.5	−3.8 (0/10, .002)	[−8.2, 0.2]	−11.0 (0/10, .002)	[−15.1, −6.9]
Task only	own	82.1±2.0	74.4±1.7	−12.5 (0/10, .002)	[−18.3, −7.0]	−20.8 (0/10, .002)	[−27.1, −14.7]
Task only	tracker, test	81.7±1.9	74.1±2.1	−12.9 (0/10, .002)	[−18.6, −7.3]	−21.1 (0/10, .002)	[−27.3, −15.0]

Note accuracy at test time. Table 8 compares the note each evaluated agent writes with the tracker's reference note at every test step, where the agent carries only its own note. On ALFWorld the note is about as accurate on unseen scenes as on seen ones, with both backbones, and on WebShop it matches the reference on the clean test instructions. On ScienceWorld it matches the reference exactly on 65% of steps (note score 0.90).

G Case Studies

We show single ALFWorld episodes of four agents: SEED, the agent without a note (Table 11), the agent trained from the SFT checkpoint of ForeNote with the task reward alone (Section 5.2), and ForeNote. The SEED episodes come from our reproduction of SEED with the authors' code on Qwen2.5-3B-Instruct, with GPT-5.5 writing the stage-1 skill annotations; it reaches 86.7% on

Table 8: **The note the evaluated agent writes at test time**, compared with the tracker's reference note at every test step. Malformed: not canonical JSON within 256 tokens. Exact: equal to the reference.

Model	Test set	Steps	Malformed	Exact (%)	Note score
Qwen2.5-3B	ALFWorld seen	13,626	4	98.8	0.999
Qwen2.5-3B	ALFWorld unseen	14,224	28	97.7	0.997
Qwen3-1.7B	ALFWorld seen	13,874	7	98.5	0.998
Qwen3-1.7B	ALFWorld unseen	16,950	12	96.3	0.995
Qwen2.5-3B	WebShop, clean test	3,282	2	99.2	0.998
Qwen3-1.7B	WebShop, clean test	3,328	0	97.6	0.997
Qwen2.5-3B	ScienceWorld test	12,798	599	65.1	0.90

seen and 85.4% on unseen scenes (average over task families, as in Table 1; Wu et al. (2026) report 91.8% and 86.2%). Its evaluation uses our five-step window, 30-step limit, and temperature, but other sampling seeds, so we compare SEED with ForeNote on the same game rather than the same seed. Each episode is chosen by a fixed rule, given below. Table 9 gives the rates behind the cases for all four agents, with the definitions of Figure 3 (Appendix E); for SEED, the reference tracker is run on its recorded episodes.

Table 9: **How the agents fail on ALFWorld** (seen and unseen scenes pooled; mean $\pm$ s.d. over 10 repeats, 15 runs for SEED). Solved and failed: % of episodes; wrong object: the agent took an object the task does not name or put one back; not found: no wrong object, and the target never appears. Returns: chance (%) that a search step goes back to a given searched place, last seen within the five-step window (in view) or earlier (out of view), as in Figure 3a. No effect: actions per episode answered by "Nothing happens."

		Failed			Returns		
Agent	Solved	not found	wrong object	other	in view	out of view	No effect
SEED (our reproduction)	87.2±1.6	6.1±0.6	2.6±0.7	4.1±1.2	3.1±0.5	15.3±0.9	1.1±0.1
No note	85.9±1.0	5.8±0.5	3.1±0.4	5.2±1.0	9.7±0.6	15.8±1.4	2.7±0.1
Task reward only	79.4±0.9	0.0±0.0	20.3±0.9	0.4±0.2	5.4±0.3	6.9±0.5	0.0±0.0
ForeNote	95.1±0.5	3.1±0.5	0.2±0.2	1.6±0.6	11.4±0.7	5.3±0.6	0.0±0.0

SEED fails about as often as the agent without a note, and in a similar way. It rarely goes back to a searched place that is still in its window (3.1%), but once the place has left the window it goes back 15.3% of the time, as often as the agent without a note (15.8%) and three times as often as ForeNote (5.3%). The jump sits at the window edge: 4.3% for places last seen four or five steps earlier and 20.4% for places seen six or seven steps earlier (without a note, 11.4% and 24.4%; ForeNote, 10.8% and 7.8%).

SEED loses track of what it has searched. Figures 7 and 8 use seen games from Pick2 and Cool, the two task families on which SEED is weakest in Wu et al. (2026). Among the 7 seen games that SEED solves in at most half of its 15 runs and ForeNote in at least 9 of its 10 repeats (2 games are the other way round), each is the game of its family that SEED solves least often; we show SEED's first failed run and the first successful repeat of ForeNote. SEED writes its reasoning before each action. In both episodes, this reasoning calls a place unsearched after the place has been searched and has left the window.

Without a note. Figure 9 shows the first unseen pick-and-place game, in evaluation order and on the first repeat, in which the agent without a note goes back to a place it searched six or seven steps earlier while ForeNote solves the same game with the same seed. ForeNote also goes back to places it has searched, but its note lists all of them at every step, and it moves on to places it has not searched.

Task reward only. Figure 10 shows the first failed unseen pick-and-place episode of the agent trained with the task reward alone, on the first repeat, next to ForeNote on the same game and seed.

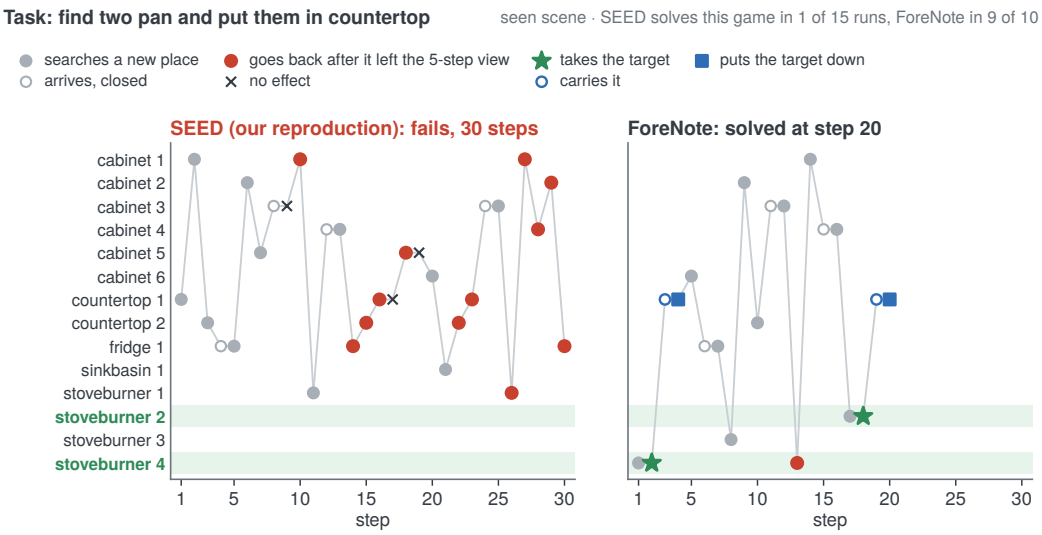


Figure 7: **SEED loses track of what it has searched (Pick2).** Each marker shows where the agent is after a step; green rows mark where the targets are first seen. Quotes are verbatim from the episodes.

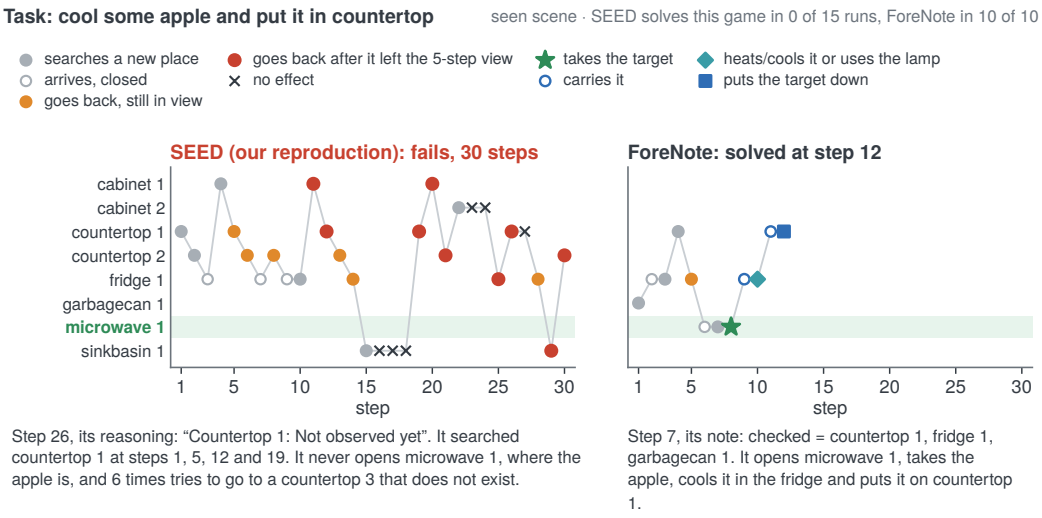


Figure 8: **SEED loses track of what it has searched (Cool).** As Figure 7.

The agent picks up objects that the task does not name and puts them back, and its note records these moves as completed task events (Section 5.2).

Where ForeNote fails. Of the 62 failed unseen episodes of ForeNote, 38 never find the target, 23 hold it without finishing, and one finds it but never picks it up. Figure 11 shows the first episode of each of the two main kinds, on the first repeat in evaluation order. In both, the note matches the tracker's reference at every step, so the failure lies in what the agent does with it: it keeps returning to places the note lists as searched, and it turns on the lamp only while it is not holding the CD.

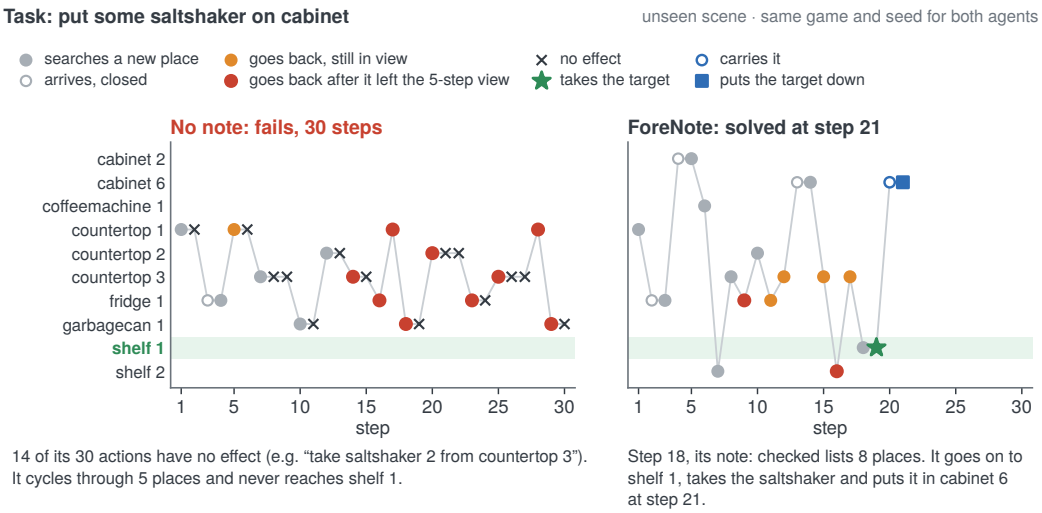


Figure 9: **Without a note.** Same unseen game and seed for both agents.

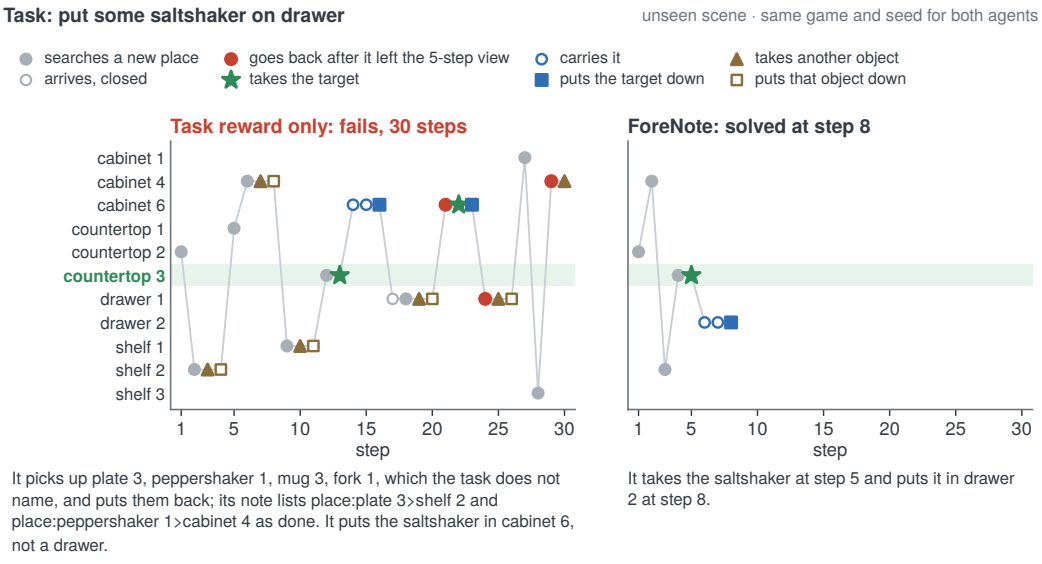
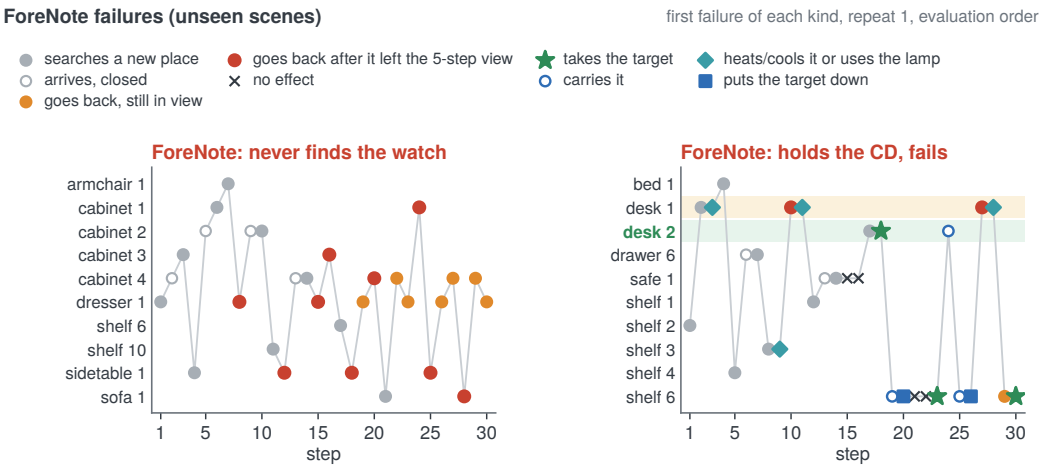


Figure 10: **Task reward only.** Same unseen game and seed for both agents; brown markers: objects the task does not name.



Task: put some watch on safe. No agent ever sees a watch in this game (SEED 15 runs, no note and ForeNote 10 each). Its last note lists 10 searched places, yet it goes to dresser 1 7 times and never opens a drawer.

Task: look at cd under the desklamp. It turns on the lamp at steps 3, 9, 11 and 28, never while holding the CD. After taking the CD (step 18) its note says holding cd 1, but it puts the CD down on shelf 6 instead of using the lamp.

Figure 11: **Where ForeNote fails.** Two unseen games; green row: where the CD is first seen; amber row: the lamp.

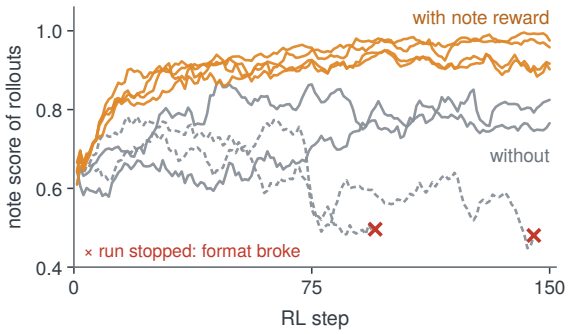


Figure 12: **Note score during RL on ALFWorld**, four runs each with and without the note reward; trailing five-step means. Crosses mark format-regression stops.

H History-Note Supervision: Full Results

Table 11 gives the runs of the five-step series, whose no-note run Section 5.1 uses; unlike Table 3, they use SFT with a five-step window. Figure 12 shows its four note runs, a third five-step pair, and a pair that starts from full-history SFT and was trained with an earlier implementation of history-note supervision. Two of the four runs without the note reward, and none with it, triggered the format-regression stop (at steps 95 and 145), a rule fixed in advance that ends a run once the share of responses in the required output format drops sharply (e.g., from 92% to 69%); the third five-step pair was therefore not evaluated, and the stopped full-history run has no final checkpoint.

Removing both signals from RL. HN+FP SFT + GRPO starts from the SFT checkpoint of ForeNote and trains RL with $\lambda = 0$ and $\beta = 0$, everything else unchanged. On ALFWorld it reaches 82.1% and 74.4% on seen and unseen scenes, against 94.6% and 95.2% for ForeNote on the same games and seeds. Its notes stay well formed (4–5 malformed notes per 100 steps) but miss 123 and 185 items of the reference note per 100 steps on seen and unseen scenes (ForeNote: 1.2 and 2.7), most of them receptacles already searched; in 31% of unseen episodes they also list moves of objects the task does not name as completed events (566 of 729 distinct spurious entries; most others place the target in the wrong receptacle). In 42.5% of its unseen episodes it picks up an object the task does not name. Every failed unseen episode, and 227 of 237 failed seen episodes, involve picking up such objects or putting an object straight back; 16.9% and 24.5% of its episodes fail by reaching the 30-step limit (ForeNote: 5.1% and 4.6%). The task reward does not make the notes complete: on 48 held-out training games, the note score of the SFT checkpoint is 0.73, and during RL it stays between 0.67 and 0.89 for this run, whereas for ForeNote it is at least 0.98 from step 20 on. Its training success fell from 88.0% over steps 91–120 to 78.9% over steps 121–150, while that of ForeNote rose to 94.0%; on the 48 validation games, its best checkpoint is nevertheless the final one. Appendix G shows one of its episodes, and Appendices D and J give its ScienceWorld and WebShop runs. In a second pair of runs, with other training seeds, this agent reaches $90.9 \pm 1.1\%$ and $83.7 \pm 1.7\%$ and ForeNote $95.7 \pm 2.0\%$ and $95.3 \pm 1.3\%$, 4.8 ± 2.3 and 11.5 ± 2.4 points apart (higher for ForeNote in all 10 repeats, $p = 0.002$; game-level 95% intervals $[1.4, 8.2]$ and $[6.6, 16.8]$).

One signal at a time. Table 10 completes the comparison with RL that keeps only one of the two signals, the note reward ($\lambda = 0.5$, $\beta = 0$) or the prediction loss ($\lambda = 0$, $\beta = 0.1$), from the same SFT checkpoint, code, and training seeds as ForeNote.

I Future Prediction: Full Results and Training Dynamics

Table 12 gives the full results. After RL, the notes miss 9.6 and 2.7 reference entries per 100 unseen steps with next-observation and next-note prediction. On WebShop, RL on the task reward alone keeps the note format but lets its content drift (mean note score 0.997 over the first 10 RL steps and

Table 10: **The two note signals during RL** (ALFWorld; success %, mean ± s.d. over 10 inference repeats). All runs start RL from the SFT checkpoint of ForeNote; the note-reward-only run is the SFT-only row of Table 12. Differences to ForeNote with the same training seeds are paired over repeats and computed before rounding (repeats higher, exact sign-flip p), with game-level 95% intervals (Appendix E).

				Δ seen		Δ unseen	
Note reward	Prediction loss	Seen	Unseen		95% int.		95% int.
Training seeds of ForeNote							
–	–	82.1±2.0	74.4±1.7	−12.5 (0/10, .002)	[−18.3, −7.0]	−20.8 (0/10, .002)	[−27.1, −14.7]
✓	–	94.2±1.0	90.4±2.0	−0.3 (3/10, .44)	[−4.7, 3.8]	−4.8 (0/10, .002)	[−8.6, −1.0]
–	✓	90.9±0.9	83.3±1.5	−3.6 (0/10, .002)	[−7.8, 0.6]	−11.9 (0/10, .002)	[−17.7, −6.2]
✓	✓	94.6±1.2	95.2±1.2	–	–	–	–
Second training seeds							
–	–	90.9±1.1	83.7±1.7	−4.8 (0/10, .002)	[−8.2, −1.4]	−11.5 (0/10, .002)	[−16.8, −6.6]
✓	✓	95.7±2.0	95.3±1.3	–	–	–	–

Table 11: **History-note supervision on ALFWorld** (success %, mean ± s.d. over 10 repeats). All runs use SFT, RL, and evaluation with a five-step window; run 2 repeats the note pair with an independent training seed. Bottom: paired differences over repeats (positive repeats out of 10; exact sign-flip p).

Variant	Run	Seen	Unseen
No note	1	90.3±1.7	79.6±1.8
Note interface, task reward only	1	86.0±1.6	74.2±2.2
Note interface + note reward (HN)	1	90.1±1.6	86.3±2.7
Note interface, task reward only	2	91.1±1.4	82.9±1.3
Note interface + note reward (HN)	2	94.5±1.0	85.6±2.0
Note reward on − off, run 1		+4.1±2.6 (9/10, .004)	+12.1±3.6 (10/10, .002)
Note reward on − off, run 2		+3.4±1.9 (9/10, .004)	+2.7±2.9 (8/10, .02)
Interface only − no note, run 1		−4.3±2.5 (0/10, .002)	−5.4±2.8 (0/10, .002)
HN − no note, run 1		−0.2±2.6 (4/10, .79)	+6.7±3.0 (10/10, .002)

0.735 over the last 30; Table 14), whereas supervising the note keeps it higher: 0.91 with the note reward (HN), 0.85 with the note reward and prediction, and 0.999 with the note loss of ForeNote. On ALFWorld, the future-prediction loss falls from a mean of 0.18 at the first RL step to 0.009 at step 30, and after step 40 only 9% of step groups (pooled over steps) contain responses whose note scores differ, so the note reward has little left to teach. Measured on sampled coordinates (the first 4,096 elements of each parameter tensor), the absolute cosine similarity between the prediction gradient and the rest of the policy update (policy-gradient, entropy, and KL terms) is below 0.1 in 72% of the 977 optimizer updates, while the two gradients have comparable norms.

Prediction during RL. The SFT-only row starts RL from the SFT checkpoint of ForeNote and sets $\beta = 0$, keeping the note reward, code, seeds, and training games; its first batch of RL rollouts is identical to that of ForeNote. We fixed its evaluation checkpoint (step 150, as for every ALFWorld cell) and the window-edge analysis below before training; its best-validation checkpoint (step 90) is lower on both splits (90.9% and 82.8%). It goes back to searched places more often than ForeNote, both in view (13.0% vs. 11.4%) and out of view (7.7% vs. 5.3%; higher in all 10 repeats, $p = 0.002$ each). Its note keeps 99.1% of the searched places that have left the window (ForeNote: 99.6%). Its unseen success is 4.8 ± 2.2 points below that of ForeNote (lower in all 10 repeats, $p = 0.002$), mostly on look (-17.8), two-object (-11.8), and pick-and-place tasks (-8.3), while cool and heat tasks gain about 5 points. Between places searched four or five and six or seven steps earlier, its rate of going back rises by 1.5 ± 2.6 points (7 of 10 repeats, $p = 0.09$), against -3.0 for ForeNote and $+13.1$ without a note; the difference from ForeNote (4.5 ± 3.0, $p = 0.004$) meets the criterion we fixed before training: a positive rise, larger than that of ForeNote with $p \leq 0.05$, while the note still lists at least 95% of the searched places out of view. On unseen scenes, 4.7% of its episodes never find the target object (ForeNote: 2.8%) and 1.7% involve a wrong object (ForeNote: none). A post-hoc check of note content shows one systematic omission: the SFT checkpoint records 88%

Table 12: **Future prediction on ALFWorld** (success %, mean ± s.d. over 10 inference repeats). All rows use history-note supervision and full-history SFT; the prediction loss (weight 0.1) applies in SFT and RL to the target named in the row (none for HN), except that three rows start RL from the next-note SFT checkpoint and change only the RL target: another game's next note, the current note, or none (in SFT only; the no-loss run of Figure 4); run 2 uses other training seeds. "Illegal": percentage of steps with an inadmissible action after RL (seen / unseen). Bottom: paired differences after RL, computed before rounding (positive repeats out of 10, exact sign-flip p over repeats), and game-level 95% intervals for unseen scenes.

	After SFT		After RL		
Prediction target	Seen	Unseen	Seen	Unseen	Illegal
None (HN)	58.7±2.2	54.6±2.0	95.0±1.0	91.1±1.8	0.1 / 0.0
Next observation	57.0±3.2	57.3±3.1	93.7±0.8	91.3±1.5	0.0 / 0.1
Another game's next note	61.4±2.7	54.1±3.6	84.5±2.5	79.4±1.4	0.0 / 0.0
Current note	61.4±2.7	54.1±3.6	92.6±1.3	94.6±0.9	0.0 / 0.0
Next note in SFT only	61.4±2.7	54.1±3.6	94.2±1.0	90.4±2.0	0.0 / 0.0
Next note (HN+FP)	61.4±2.7	54.1±3.6	94.6±1.2	95.2±1.2	0.3 / 0.3
Next note (HN+FP), run 2	61.4±2.7	54.1±3.6	95.7±2.0	95.3±1.3	0.0 / 0.1
Difference after RL			Seen	Unseen	95% interval
HN+FP − HN			−0.4±1.8 (4/10, .50)	+4.1±2.9 (10/10, .002)	[−0.3, 8.5]
HN+FP − next observation			+0.8±1.3 (6/10, .09)	+3.9±2.0 (10/10, .002)	[0.5, 7.3]
HN+FP − another game's note			+10.1±3.3 (10/10, .002)	+15.7±1.7 (10/10, .002)	[10.9, 20.5]
HN+FP − current note			+2.0±1.9 (8/10, .01)	+0.6±1.4 (5/10, .19)	[−2.5, 3.5]
HN+FP − SFT only			+0.3±1.2 (7/10, .44)	+4.8±2.2 (10/10, .002)	[1.0, 8.6]

and 83% of completed placements on seen and unseen scenes and ForeNote all of them, but this agent records none, already at step 90, while it records every cleaning, heating, cooling, and light event. The note reward is standardized within groups of responses at the same task and step, so an omission shared by every response in a group receives no advantage, whereas the prediction loss trains on the reference note itself.

Training code. All rows of Table 12 use the implementation of ForeNote and, except the second run, its training seeds. The HN row starts RL from the HN SFT checkpoint and trains without prediction; the row with another game's next note starts from the next-note SFT checkpoint and, at every optimizer update, replaces each transition's prediction target with the next reference note of a transition from a different game; the current-note row does the same with the transition's own current reference note, keeping the weight, the transitions, and the seeds; the second run of ForeNote changes only the training seeds. Every row uses the step-150 checkpoint, and the HN row is also the HN agent of Tables 3 and 6.

Training success. Training success over the last 20 RL steps is similar for ForeNote and HN (94.8% vs. 93.0%), so the unseen-scene difference between them is not visible during training; the SFT-only row reaches 88.9%.

J WebShop Details

Clean test set. WebShop has 6,910 goals corresponding to 6,859 distinct instructions after removing the price clause. Each worker independently shuffles the goal list, so using indices above 500 for training and below 500 for evaluation does not separate instructions: 620 of the 1,280 episodes of our 10-by-128 evaluation schedule (48%) share their instruction with a task used in SFT or drawn during RL. We freeze the 660 episodes (610 instructions) of our 10-by-128 evaluation schedule whose instructions appear in neither our SFT data nor our RL training draws. Every evaluated checkpoint uses exactly this set; all within-paper comparisons are paired. The published WebShop rows in Table 1 retain their source protocol and are not clean-set measurements. Table 13 reports our agents on the clean set.

Table 13: WebShop on the frozen clean test set (score / success %; mean over 10 inference repeats).

Model	Score / success
Qwen2.5-3B-Instruct	4.2 / 1.1
+ SFT	85.5 / 74.5
+ SFT (HN)	90.0 / 78.5
+ SFT (HN+FP)	90.5 / 79.6
GRPO from the base model	86.3 / 78.9
SFT + GRPO	92.6 / 84.3
SFT + GRPO, eight GPUs	93.1 / 86.5
HN SFT + HN-GRPO	88.8 / 80.7
ForeNote	90.0 / 81.9
ForeNote, temperature 1.0	90.4 / 81.5
HN+FP with the note reward	89.5 / 78.3

Table 14: WebShop state supervision (clean-set score / success, mean over 10 repeats). Note score averages the first 10 and last 30 RL steps. All rows use full-history SFT.

RL objective	SFT start	Final checkpoint	Note score
HN	90.0 / 78.5	88.8 / 80.7	$0.997 \to 0.908$
HN+FP, note reward	90.5 / 79.6	89.5 / 78.3	$0.995 \to 0.845$
ForeNote, note loss	90.5 / 79.6	90.0 / 81.9	$0.996 \to 0.999$
Task reward only	90.5 / 79.6	11.5 / 9.5	$0.997 \to 0.735$

Note loss instead of note reward. For Qwen2.5-3B, the WebShop variant supervises the current note by cross-entropy with weight 0.1 on eight on-policy transitions per optimizer update, in addition to FP on 16 transitions. Note tokens are excluded from task policy gradients and KL, and invalid-action steps cannot receive positive task advantages. We select temperature and checkpoint using 512 validation goals whose instructions occur in neither training nor test: candidates are temperatures 0.4 and 1.0 at steps 50, 100, and 150. The selected model is step 150 at 0.4; the 1.0 run is also reported in Table 13.

Compared with the note-reward variant, the selected model improves success by 3.6 ± 3.8 points ($p = 0.016$), with a score difference of 0.6 ± 2.4 ($p = 0.49$). Relative to no-note SFT+GRPO, its score is 2.5 ± 3.0 points lower ($p = 0.02$) and success 2.4 ± 5.0 points lower ($p = 0.16$).

Collapse without the note signals. The HN+FP SFT + GRPO ablation uses the same SFT initialization with only task reward during RL ($\lambda = \beta = 0$, no note loss). Its final checkpoint repeatedly selects product options without purchasing; its note often fails to retain the selected option. At step 150 it achieves score 11.5 and success 9.5%, and 85% of clean episodes end without a purchase. This run has no in-training validation. Table 3 reports its step-100 checkpoint, taken before the collapse and chosen before any test evaluation. That checkpoint achieves score 89.9 and success 75.6%; its final result remains in Table 14.

The non-collapsed trained policies generally purchase after one search and one product page without backtracking. SFT notes have score 0.997 and exactly match the reference on 98.2% of steps; the next note is already predicted exactly on 99% of validation steps.