# ALIGN AS ACT: STEP-LEVEL CREDIT ASSIGNMENT FOR LLM AGENTS VIA SEQUENTIAL SUBSPACE RESIDUALS

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

## ABSTRACT

Reinforcement learning (RL) enables LLM agents to improve their policies by learning from environmental feedback. However, designing reliable intermediate rewards is challenging, and reliance on sparse terminal rewards can hinder efficient training. To obtain step-level signals without a separately trained evaluator, we first examine the policy's own contextual representations in trajectories on ALFWorld. We find that diverse action–feedback sequences exhibit more broadly distributed hidden-state variation than repetitive sequences. Connecting the geometry of contextual representations with classical linear innovations, we propose Align as Act (AAA), a representation-based method for constructing step-level learning signals. Processing post-action contextual representations in trajectory order, AAA projects each representation onto a subspace of directions retained from earlier interactions within the same trajectory. This separates the history-aligned component from an orthogonal innovation. The innovation energy is then transformed into an intrinsic reward signal, normalized within successful trajectories, and used to construct step advantages. We instantiate this construction as GRPO+AAA and extend it with a GiGPO-style anchor-group correction to obtain GiGPO+AAA. Experiments on ALFWorld and WebShop with Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct show improvements over the corresponding GRPO and GiGPO baselines. With the 1.5B policy, GRPO+AAA improves the ALFWorld success rate by 11.9 percentage points and increases the WebShop score and success rate by 8.8 points and 14.9 percentage points, respectively. Additional experiments and analyses address AAA's performance with an alternative policy backbone, the use of NNLS for failure-side credit assignment, and the role of the history basis in step-reward construction. These results suggest that trajectory-relative representation geometry provides a practical basis for step-level credit assignment in outcome-supervised LLM agent training.

## 1 INTRODUCTION

Large language model (LLM) agents interact with environments to perform household tasks, online shopping, and information seeking (Yao et al., 2023). Reinforcement learning (RL) improves these policies through environmental feedback (Feng et al., 2025), but reliable intermediate rewards are difficult to design. With terminal-outcome supervision, Group Relative Policy Optimization (GRPO) (Shao et al., 2024) assigns the same trajectory-level advantage to every step, providing limited guidance for step-level credit assignment.

Recent methods construct finer-grained credit through state- or history-based grouping (Feng et al., 2025; He et al., 2026), transition-level value differences (Wang et al., 2026b), policy entropy (Wang et al., 2025c), and positive-response representation subspaces (Lin et al., 2026).

In this paper we approach the problem from a different perspective and examine the policy's own contextual representations as a source of step-level signals without a separately trained evaluator. These representations encode the task, recent interactions, and action feedback. In initial-policy ALFWorld (Shridhar et al., 2021) trajectories, diverse action–feedback sequences exhibit more broadly distributed hidden-state variation than repetitive sequences within the same episode. This observation motivates using representation geometry to construct intrinsic rewards.



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.

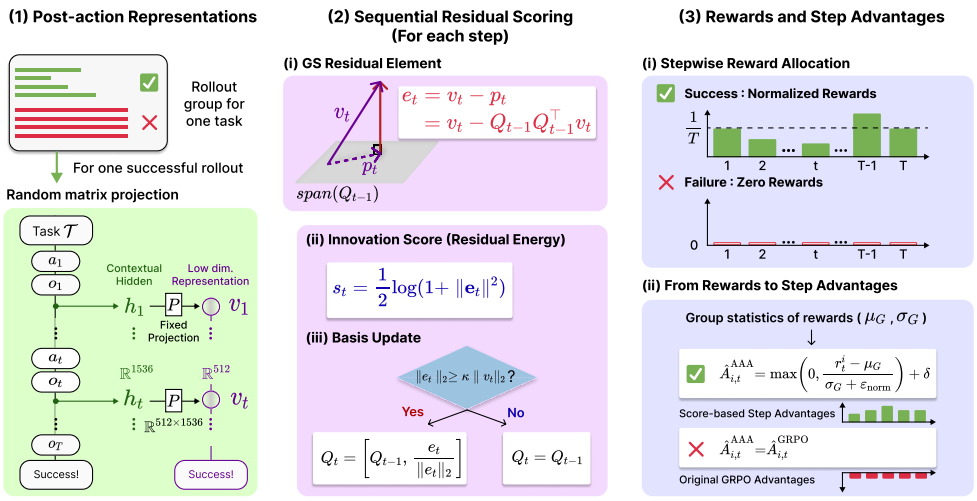



Figure 1: Overview of GRPO+AAA. Sequential residuals produce intrinsic rewards for successful trajectories, while failed rollouts retain their original GRPO advantages.

Realizing this idea, we propose *Align as Act* (AAA), a representation-based step-reward method inspired by classical linear innovations (Kailath et al., 2000) and their connection to entropy of Gaussian models. As outlined in Figure 1, AAA applies a fixed random projection to post-action representations for dimension reduction and processes them in sequential order. At each step, orthogonal projection separates the residual innovation from the component aligned with directions retained from earlier interactions. The basis expands when the relative residual norm meets a threshold. Residual energies are mapped to nonnegative scores and normalized within each successful trajectory to obtain intrinsic rewards.

We incorporate these intrinsic rewards into group-based policy optimization through two variants. GRPO+AAA combines the rewards with group statistics, nonnegativity clipping, and a positive offset to construct step advantages, while failed trajectories retain their original GRPO advantages. GiGPO+AAA adds a GiGPO-style anchor-group correction computed from the same rewards. Neither variant requires step-level annotations or a separately trained evaluator.

Experiments on ALFWorld and WebShop (Yao et al., 2022) with Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct (Yang et al., 2024) show gains over the corresponding GRPO and normalization-matched GiGPO baselines. For the 1.5B and 7B policies, respectively, GRPO+AAA increases ALFWorld success by 11.9 and 9.3 percentage points and WebShop success by 14.9 and 9.4 percentage points, with WebShop score gains of 8.8 and 7.9 points. GiGPO+AAA also improves all aggregate metrics over the matched GiGPO baselines at both model sizes. These gains support the use of trajectory-relative representation geometry for step-level credit assignment. Additional experiments examine an alternative policy backbone and an NNLS-based failure-side extension. Component ablations and score analyses assess the reward design and its association with task progress.

**Our contributions are summarized as follows.** 1) We construct history-relative intrinsic rewards from innovations, i.e., sequential subspace residuals of post-action policy representations. 2) We develop outcome-conditioned step advantages, yielding two algorithms: GRPO+AAA and GiGPO+AAA. 3) We evaluate both variants on two agent benchmarks and analyze the roles of the reward components and retained basis.

## 2 Problem Setup and Background

**Problem Setup** In this paper, we consider a partially-observable sequential decision-making problem of an LLM agent given a task $\mathcal{T}$. At step $t$, the agent generates and executes action $a_t$, after which the environment returns feedback $o_t$. The arrows below indicate the temporal order of interaction:

$$\mathcal{T} \xrightarrow{\text{LLM}} a_1 \xrightarrow{\text{Env.}} o_1 \xrightarrow{\text{LLM}} a_2 \xrightarrow{\text{Env.}} o_2 \cdots \xrightarrow{\text{Env.}} o_T,$$

where $a_t$ and $o_t$ are described in natural language. The policy $\pi_\theta(a_t|c_t)$ conditions on the actual chat-template formatted input context $c_t$, containing the task and available recent interaction history, rather than only the preceding observation. This context may truncate the full history $\mathcal{H}_{t-1} = (\mathcal{T}; a_1, o_1, \ldots, a_{t-1}, o_{t-1})$.

The episode ends at environment termination or the step limit, with length $T$ and a terminal verifier providing a success/failure outcome. We use this terminal supervision and policy representations to construct step-level learning credit. When an actual next prompt $c_{t+1}$ is formed after feedback $o_t$, $h_t \in \mathbb{R}^d$ denotes the contextual hidden read from that prompt. The final feedback $o_T$ may have no subsequent prompt, so $h_T$ is not assumed available.

**Innovations Processes** The method of innovations was introduced for time series by Wiener (Wiener, 1958) and has been used widely in signal processing from Kalman filtering to signal detection (Kalman, 1960; Kailath, 1970; Sung et al., 2006; 2008). It can be explained readily under the linear-estimation framework. Suppose that we have a sequence of observation vectors $y_1, y_2, \cdots, y_n$ in some Hilbert space $\mathcal{H}$. The innovation $e_t$ provided by $y_t$ at time step $t$ is defined as

$$e_t := y_t - \hat{y}_{t|t-1} \quad \text{s.t.} \quad e_t \perp e_k, \quad \forall k < t, \tag{1}$$

where $\hat{y}_{t|t-1}$ is the optimal linear prediction in the $L_2$-error sense of $y_t$ based on all previous observations $y_1, \cdots, y_{t-1}$. Hence, the innovation $e_t$ is the component of $y_t$ that cannot be linearly predicted from the preceding observations. There exists a transform between the sequence $(y_1, \cdots, y_t)$ and the sequence $(e_1, \cdots, e_t)$ that is causal and causally invertible for all $t$ (Kailath et al., 2000). Thus, the collection of innovations contains all the information of the original observation vectors without any loss but the innovations form a sequence of orthogonal vectors: Obtaining the innovations process can be viewed as the familiar *Gram-Schmidt orthogonalization* as shown in Fig. 2.

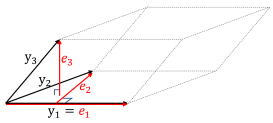



Figure 2: Innovations and parallelotope

Let $y_1, \ldots, y_n$ be zero-mean, square-integrable observation random variables on a common probability space. With correlation $\langle y_i, y_j \rangle = \mathbb{E}[y_i y_j]$ as inner product, they can be viewed as elements (i.e., vectors) of an $L^2$ inner-product space (Grimmett & Stirzaker, 2020). Then, applying Eq. (1) to $y_1, \cdots, y_n$ yields innovations $e_1, \ldots, e_n$ such that $\langle e_t, e_k \rangle = 0$ for all $k < t$ (Kailath et al., 2000).

**Information Decomposition** The maximum total self-information i.e. entropy provided by observation variables $y_1, \ldots, y_n$ with covariance matrix $\Sigma$ is given by (Cover & Thomas, 1991):

$$H(y_1, \cdots, y_n) = \tfrac{1}{2} \log\left((2\pi e)^n \det(\Sigma)\right), \tag{2}$$

which is achieved when $y_1, \ldots, y_n$ are Gaussian. The key property of innovations is as follows: Due to $LDL^T$ decomposition of $\Sigma$ (Golub & Van Loan, 2013), the total entropy (2) can be decomposed as (Kailath et al., 2000; Sung et al., 2006)

$$H(y_1, \cdots, y_n) = \frac{1}{2} \sum_{t=1}^{n} \log(2\pi e R_{e,t}), \tag{3}$$

where $R_{e,t} = \mathbb{E}[e_t^2] = \langle e_t, e_t \rangle$ is the variance of innovation $e_t$. In other words, we have

$$H(y_t|y_1, \cdots, y_{t-1}) = \tfrac{1}{2} \log(2\pi e R_{e,t}). \tag{4}$$

**Geometric Interpretation** Consider the parallelotope generated by edge vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^n$. Let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$. Then, the volume of the parallelotope is given by $|\det(\mathbf{X})|$ (Strang, 2006). The Gram matrix $\mathbf{G}$ of vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n$ is defined as $\mathbf{G}_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$, i.e., $\mathbf{G} = \mathbf{X}^\top \mathbf{X}$. Then, $\det(\mathbf{G}) = (\det(\mathbf{X}))^2$ (Horn & Johnson, 2012); hence $\sqrt{\det(\mathbf{G})}$ equals the parallelotope volume. Now, consider zero-mean random variables $y_1, \cdots, y_n$ and consider the *parallelotope* constructed by random variables $y_1, \cdots, y_n$ as vectors embedded in the higher-dimensional vector space of random variables with correlation as inner product. The Gram matrix of $y_1, \ldots, y_n$ is then the covariance matrix $\Sigma$ with $\Sigma_{ij} = \mathbb{E}[y_i y_j]$. Hence, *the entropy formula (2) can be interpreted as the logarithm of the volume of the parallelotope constructed by observation vectors* $y_1, \cdots, y_n$.

The innovations-based information decomposition eq. (3) is clear under this new geometric interpretation of the entropy formula. The volume of the parallelotope constructed by vectors $y_1, \cdots, y_n$,

i.e., $\sqrt{\det(\Sigma)}$ is nothing but the product of the lengths of orthogonal vectors $e_1, \cdots, e_n$ obtained from Gram-Schmidt orthogonalization of $y_1, \cdots, y_n$, i.e., $\sqrt{\det(\Sigma)} = \prod_{t=1}^{n} ||e_t|| = \prod_{t=1}^{n} \sqrt{R_{e,t}}$, as shown in Fig. 2, yielding the geometric equivalence of (2) and (3).

Thus, the total information obtained by sequential observations $y_1, \cdots, y_n$ can be decomposed as the sum of the logarithm of each innovation norm. This motivates us to adopt a geometric approach to step reward design and use the logarithm of innovation norm as the decomposed step reward capturing the net new information in a sequence of observations.

## 3 Motivation: Intrinsic Signals from Policy Representations

Now consider a rollout $\tau = (\mathcal{T}; a_1, o_1, \ldots, a_T, o_T)$ sampled from LLM agent policy $\pi_{\theta_{\text{old}}}$, performing the given task $\mathcal{T}$ by interacting with the environment. Note that this rollout forms a sequence of observations, described in the previous section. In this LLM agent setting, observations are given in natural language, whereas the geometric framework in the previous section operates on vectors. LLM transformers already convert sequences of words into *vector representations*, which we read at a fixed position in the final layer. These representations lie in a common vector space, allowing us to apply the geometric view of innovations from the previous section to construct step-wise rewards for agentic rollouts.

To examine this idea, we extract the hidden representation vector $h_t \in \mathbb{R}^d$ at a fixed boundary following the complete user-side prompt at each time step in the final layer (the extraction procedure and its sensitivity to the extraction position are provided in Appendices C.2 and B.3, respectively), and test how variation in these representation vectors relates to interaction patterns. For this we consider repetitive and diverse action–feedback windows from the same initial policy, using 357 matched pairs across 125 tasks in ALFWorld. For a window $\mathcal{W}$, let $\bar{h}_{\mathcal{W}}$ be its mean hidden representation, and let $X_{\mathcal{W}} = [h_t - \bar{h}_{\mathcal{W}}]_{t \in \mathcal{W}}$ be the matrix of centered representations. We measure the directional spread using the empirical participation ratio, equivalent to the $R_0$ effective rank of the empirical covariance (Bartlett et al., 2020, Definition 3):

$$D_{\mathcal{W}} = \frac{\operatorname{tr}(X_{\mathcal{W}}^{\top} X_{\mathcal{W}})^2}{\|X_{\mathcal{W}}^{\top} X_{\mathcal{W}}\|_F^2}.$$

Here, a higher $D_{\mathcal{W}}$ indicates that within-window representation variation is distributed across more effective directions, and $h_t,\ t \in \mathcal{W}$ form a larger-volume parallelotope.

Figure 3(a) compares task-level means of $D_{\mathcal{W}}$ for matched repetitive and diverse windows, with an overall task-equal difference of 0.690. Figure 3(b) shows positive mean paired differences $\Delta D = D_{\mathcal{W}^{\text{div}}} - D_{\mathcal{W}^{\text{rep}}}$ across all six task types, with 95% task-bootstrap confidence intervals. These results indicate that hidden-state variation in diverse interaction windows has a higher effective dimensionality on average than in repetitive windows, reflecting interaction diversity in the geometry of contextual policy representations.

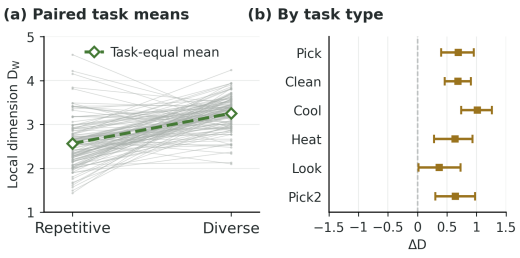



Figure 3: Directional spread of hidden-state variation in repetitive and diverse interaction windows.

Together with the linear-innovation framework in Section 2, this observation motivates using components unexplained by selected earlier representation directions to construct step-level intrinsic signals. Additional analysis details are provided in Appendix B.1.

## 4 Align as Act: Step-Level Credit Construction

In this section, we present Align as Act (AAA), an intrinsic-reward framework for training LLM agents that decomposes sparse, trajectory-level task returns into step-wise innovation signals. At a high level, AAA operates as follows: At each step, the agent embeds the current observation and context in the form of language into a representation vector, computes the innovations, and assigns an intrinsic reward based on the innovations. Hence, this reward encourages actions that reveal new, non-redundant information about the environment.

### 4.1 Innovations as History-Relative Residuals

In Sec. 3, we observe that the geometry of representation vectors $h_t,\ t = 1, \cdots, T$ indeed captures the informativeness of actions in a rollout trajectory. Building on this insight, we formalize the idea to develop a step reward design capturing the new information gained at each step from geometry.

Note that when the embedding dimension $d$ is sufficiently high compared to $T$, the relative geometry among $h_t, t = 1, \cdots, T$ is invariant after a fixed random projection due to Johnson-Lindenstrauss property (Dasgupta & Gupta, 2003). So, we first project $h_t \in \mathbb{R}^d$ to a lower-dimensional space using a fixed random projection $P \in \mathbb{R}^{m\times d}$ to reduce computational complexity:

$$v_t = Ph_t \in \mathbb{R}^m.$$

Within the projected space, let $Q_{t-1} \in \mathbb{R}^{m\times k_{t-1}}$ contain the orthonormal directions retained from earlier steps, with $Q_0 = \varnothing$. Then, the Gram–Schmidt decomposition of current projection $v_t$ with respect to this retained span is given by

$$p_t = Q_{t-1}Q_{t-1}^\top v_t, \qquad e_t = v_t - p_t.$$

Here, $p_t$ is the orthogonal projection of $v_t$ onto $\mathrm{span}(Q_{t-1})$, and $e_t = v_t - p_t$ is the corresponding orthogonal residual, as shown in Fig. 1, which we call the *representation innovation*. Its squared norm, $\|e_t\|^2$, defines the *residual energy*, where $\|\cdot\|$ denotes the Euclidean norm for finite-dimensional vectors.

### 4.2 Sequential Scoring and Basis Update

Consider a group of $G$ rollouts generated for the same task, where $\tau^i$ denotes rollout $i$ and $T_i$ its length. The logarithmic dependence on innovation variance in Eq. (4) motivates assigning each available post-action residual the raw score

$$s_t^i = \tfrac{1}{2}\log\big(1 + \|e_t^i\|^2\big), \qquad t = 1, \ldots, T_i - 1. \tag{5}$$

The constant 1 inside the logarithm yields a nonnegative score that increases with $\|e_t^i\|^2$ and equals zero when $e_t^i = 0$. After computing $s_t^i$ with the pre-update basis $Q_{t-1}^i$, we form $Q_t^i$ by appending the normalized residual $e_t^i/\|e_t^i\|$ to $Q_{t-1}^i$ only if $e_t^i \neq 0$ and $\|e_t^i\| \geq \kappa\|v_t^i\|$, with some $\kappa$; otherwise, $Q_t^i = Q_{t-1}^i$. This criterion makes computation efficient by neglecting insignificant residual directions in the basis extension. Note that the current raw score is evaluated at every step, including those that leave the basis unchanged.

### 4.3 Outcome-Conditioned Intrinsic Rewards

We condition intrinsic reward allocation on whether a rollout succeeds or fails, using the innovation scores $s_t^i$ to distribute rewards across the steps of successful trajectories. For a successful rollout $\tau^i$ with $T_i > 2$, we first assign the mean interior score to the first and last steps before normalizing across the trajectory. Then, the boundary-corrected score $\bar{s}_t^i$ is given by

$$\bar{s}_t^i = \begin{cases} m_i, & t = 1 \text{ or } t = T_i, \\ s_t^i, & 2 \leq t \leq T_i - 1. \end{cases} \quad \text{where} \quad m_i = \frac{1}{T_i - 2}\sum_{t=2}^{T_i-1} s_t^i.$$

Then, we normalize the boundary-corrected scores. Let $S_i := \sum_{k=1}^{T_i} \bar{s}_k^i$. When $S_i > \varepsilon_{\mathrm{norm}}$, we set

$$r_t^i = \frac{\bar{s}_t^i}{S_i}, \qquad t = 1, \ldots, T_i, \tag{6}$$

so that the intrinsic rewards sum to one within each successful rollout with $S_i > \varepsilon_{\mathrm{norm}}$. For numerical stability, we instead set $r_t^i = 0$ for all steps when the sum of the boundary-corrected scores satisfies $S_i \leq \varepsilon_{\mathrm{norm}}$, where $\varepsilon_{\mathrm{norm}} > 0$ is a small positive constant.

Now consider the failed rollout case. In this case, a large innovation in a failed rollout may not reflect a large information step towards solving the task; it may correspond to a big step towards a wrong direction. So, our default reward assignment in this case is

$$r_t^i = 0, \ \ \forall t = 1, 2, \ldots, T_i, \ \ \forall \text{ failed rollout } \tau^i, \tag{7}$$

which leads to the trajectory return-based advantage for a failed trajectory.

We also consider a failure-side extension that fits failed-step residuals using nonnegative combinations of residual directions from a successful rollout in the same task group. The NNLS-based construction is described in Appendix F.3.

### 4.4 Step Advantages and Policy Optimization

Based on the intrinsic rewards $r_t^i$ from all steps in the rollout group, including the zeros assigned to failed and degenerate successful rollouts, we compute their mean and population standard deviation as

$$\mu_G = \frac{1}{\sum_{i=1}^{G} T_i} \sum_{i=1}^{G} \sum_{t=1}^{T_i} r_t^i, \qquad \sigma_G = \left[ \frac{1}{\sum_{i=1}^{G} T_i} \sum_{i=1}^{G} \sum_{t=1}^{T_i} (r_t^i - \mu_G)^2 \right]^{1/2}.$$

Using these statistics, we assign successful steps a clipped group-relative advantage with a positive offset, while retaining the original environment-return-based GRPO advantage $\widehat{A}_{i,t}^{\mathrm{GRPO}}$ for failed steps:

$$\widehat{A}_{i,t}^{\mathrm{AAA}} = \begin{cases} \max\left(0, \dfrac{r_t^i - \mu_G}{\sigma_G + \varepsilon_{\mathrm{norm}}}\right) + \delta, & \text{if } \tau^i \text{ succeeds,} \\ \widehat{A}_{i,t}^{\mathrm{GRPO}}, & \text{if } \tau^i \text{ fails} \end{cases} \tag{8}$$

Here, $\varepsilon_{\mathrm{norm}} > 0$ is again the numerical-stability constant. The nonnegativity operator clips negative group-relative values to zero, and the offset $\delta > 0$ sets the minimum advantage of every successful step to $\delta$. We use $\delta = 0.1$ across environments. Appendix E.4 reports sensitivity to this choice.

For GRPO+AAA, we directly use $\widehat{A}_{i,t}^{\mathrm{AAA}}$ as the per-step advantage in the standard GRPO objective. Appendix C.5 provides details on token-level assignment and the extension to GiGPO.

## 5 Related Work

**Group-based advantage estimation.** GRPO (Shao et al., 2024) estimates relative advantages from rewards within groups of outputs sampled for the same input, avoiding a learned critic. Under outcome supervision, this assigns a shared advantage to all steps of a trajectory. For multi-step agents, GiGPO (Feng et al., 2025) introduces finer comparisons by grouping actions taken from matching anchor states and combining their return-based advantages with trajectory-level advantages. Because matching current states can arise from different interaction histories, HGPO (He et al., 2026) organizes these comparisons into hierarchical groups with increasingly consistent histories and combines advantages across levels.

**Reward construction and gradient weighting.** Other methods derive intermediate learning signals from quantities beyond terminal returns. IGPO (Wang et al., 2026a) uses increases in the policy's log probability of the ground-truth answer to construct turn-level rewards, which are combined with outcome supervision to obtain discounted learning signals. EMPG (Wang et al., 2025c) instead uses current step policy entropy to rescale trajectory advantages and next-step entropy to provide a future-clarity bonus. Using representation geometry, ResRL (Lin et al., 2026) constructs a subspace from successful-response tokens and uses the projection residuals of failed-response tokens to reweight their negative gradients. AAA takes the preceding representations within each trajectory as its comparison reference, using sequential orthogonal residuals to construct intrinsic step rewards. We normalize these rewards within successful rollouts and design outcome-conditioned step advantages for GRPO+AAA and GiGPO+AAA.

Appendix A reviews trajectory refinement, learned process evaluation, rubric-based rewards, intrinsic motivation and exploration, and representation-based learning signals. It also discusses the geometric connection between AAA and determinantal point processes.

Table 1: **Main results on ALFWorld and WebShop.** Within each model-size block, boldface indicates the best aggregate success rate on ALFWorld and the best score and success rate on WebShop. GRPO+AAA is compared against GRPO. GiGPO+AAA uses `mean_std_norm` on ALFWorld and `mean_norm` on WebShop, with $\lambda_d = 1.0$ in both environments. It is compared with $\text{GiGPO}_{\text{w/ std}}$ and $\text{GiGPO}_{\text{w/o std}}$, respectively.

| Type | Method | ALFWorld | | | | | | | WebShop | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Pick | Look | Clean | Heat | Cool | Pick2 | All(Succ.↑) | Score(↑) | Succ.(↑) |
| *Closed-Source Model* | | | | | | | | | | |
| Prompting | GPT-4o | 75.3 | 60.8 | 31.2 | 56.7 | 21.6 | 49.8 | 48.0 | 31.8 | 23.7 |
| Prompting | Gemini-2.5-Pro | 92.8 | 63.3 | 62.1 | 69.0 | 26.6 | 58.7 | 60.3 | 42.5 | 35.9 |
| *Qwen2.5-1.5B-Instruct* | | | | | | | | | | |
| Prompting | Qwen2.5 | 5.9 | 5.5 | 3.3 | 9.7 | 4.2 | 0.0 | 4.1 | 23.1 | 5.2 |
| Prompting | ReAct | 17.4 | 20.5 | 15.7 | 6.2 | 7.7 | 2.0 | 12.8 | 40.1 | 11.3 |
| Prompting | Reflexion | 35.3 | 22.2 | 21.7 | 13.6 | 19.4 | 3.7 | 21.8 | 55.8 | 21.9 |
| RL Training | PPO (with critic) | 64.8±3.5 | 40.5±6.9 | 57.1±4.9 | 60.6±6.6 | 46.4±4.0 | 47.4±1.9 | 54.4±3.1 | 73.8±3.0 | 51.5±2.9 |
| RL Training | RLOO | 88.3±3.0 | 52.8±8.6 | 71.0±5.9 | 62.8±8.7 | 66.4±5.5 | 56.9±4.7 | 69.7±2.5 | 73.9±5.6 | 52.1±6.7 |
| RL Training | GRPO | 85.3±1.5 | 53.7±8.0 | 84.5±6.8 | 78.2±7.9 | 59.7±5.0 | 53.5±5.6 | 72.8±3.6 | 75.8±3.5 | 56.8±3.8 |
| RL Training | $\text{GiGPO}_{\text{w/std}}$ | 94.4±5.9 | 67.5±4.6 | 94.8±3.8 | 94.4±7.8 | 79.8±4.7 | 76.4±5.4 | 86.7±1.7 | 83.1±1.6 | 65.0±3.2 |
| RL Training | $\text{GiGPO}_{\text{w/o std}}$ | 96.0±1.4 | 76.5±3.9 | 91.8±5.5 | 91.3±6.3 | 71.7±8.4 | 79.5±7.7 | 86.1±4.7 | 83.5±1.8 | 67.4±4.5 |
| RL Training | GRPO + AAA | 95.1±5.5 | 76.0±7.1 | 92.2±3.8 | 86.5±5.6 | 86.1±0.8 | 65.7±5.8 | 84.7±0.7 | 84.6±0.2 | **71.7**±0.2 |
| RL Training | GiGPO + AAA | 95.5±3.3 | 74.8±1.3 | 97.2±2.6 | 95.0±4.7 | 86.6±3.1 | 78.2±7.0 | **88.9**±2.0 | **86.6**±1.3 | 71.4±1.8 |
| *Qwen2.5-7B-Instruct* | | | | | | | | | | |
| Prompting | Qwen2.5 | 33.4 | 21.6 | 19.3 | 6.9 | 2.8 | 3.2 | 14.8 | 26.4 | 7.8 |
| Prompting | ReAct | 48.5 | 35.4 | 34.3 | 13.2 | 18.2 | 17.6 | 31.2 | 46.2 | 19.5 |
| Prompting | Reflexion | 62.0 | 41.6 | 44.9 | 30.9 | 36.3 | 23.8 | 42.7 | 58.1 | 28.8 |
| RL Training | PPO (with critic) | 92.3±4.0 | 64.0±8.4 | 92.5±2.4 | 89.5±7.0 | 80.3±2.0 | 68.8±8.3 | 80.4±2.7 | 81.4±3.1 | 68.7±5.1 |
| RL Training | RLOO | 87.6±4.3 | 78.2±8.3 | 87.3±5.8 | 81.3±7.6 | 71.9±5.2 | 48.9±8.4 | 75.5±4.6 | 80.3±3.2 | 65.7±4.0 |
| RL Training | GRPO | 90.8±5.1 | 66.1±6.7 | 89.3±5.4 | 74.7±6.9 | 72.5±5.4 | 64.7±7.3 | 77.6±5.2 | 79.3±2.8 | 66.1±3.7 |
| RL Training | $\text{GiGPO}_{\text{w/std}}$ | 97.7±1.6 | 82.7±7.9 | 98.8±1.6 | 83.7±7.2 | 89.3±8.2 | 79.2±6.6 | 90.8±1.3 | 84.4±2.9 | 72.8±3.2 |
| RL Training | $\text{GiGPO}_{\text{w/o std}}$ | 91.8±5.4 | 88.6±6.3 | 95.9±3.2 | 90.2±2.6 | 86.5±5.5 | 85.2±7.5 | 90.2±2.3 | 86.2±2.6 | 75.2±3.8 |
| RL Training | GRPO + AAA | 98.4±1.4 | 75.6±5.1 | 95.3±5.2 | 88.9±8.1 | 78.3±4.1 | 78.2±7.9 | 86.9±0.5 | 87.2±1.7 | 75.5±2.7 |
| RL Training | GiGPO + AAA | 98.5±1.3 | 84.8±7.4 | 99.4±1.0 | 88.9±1.9 | 89.7±6.5 | 92.6±0.8 | **93.2**±1.4 | **87.4**±1.1 | **77.3**±0.2 |

# 6 EXPERIMENTS

## 6.1 MAIN EXPERIMENT AND RESULTS

**Setup.** We evaluate AAA on ALFWorld (Shridhar et al., 2021) and WebShop (Yao et al., 2022). We compare against the baselines reported by Feng et al. (2025): (1) closed-source prompting baselines, GPT-4o and Gemini-2.5-Pro; (2) open-source prompting baselines, including direct prompting, ReAct, and Reflexion (Shinn et al., 2023); and (3) RL training baselines on the same backbone, including PPO, RLOO (Ahmadian et al., 2024), GRPO, and GiGPO. Our AAA variants are implemented on the official GiGPO codebase[1] and evaluated under the matched benchmark protocol.

**Training Details.** We apply AAA to GRPO and GiGPO, yielding GRPO+AAA and GiGPO+AAA, respectively, as described in Sec. 4.4. Our runs use the same backbone models, environment interfaces, rollout group size, maximum episode length, prompt and response lengths, and final evaluation protocol as the corresponding GiGPO benchmark setting. All main-table AAA results are averaged over three random seeds. Full hyperparameters and implementation details are provided in Appendix D.

**Result.** Table 1 shows that AAA improves aggregate performance over the corresponding GRPO and GiGPO baselines at both model sizes. With Qwen2.5-1.5B-Instruct, GRPO+AAA increases success rates from 72.8% to 84.7% on ALFWorld and from 56.8% to 71.7% on WebShop, while

[1] `https://github.com/langfengq/verl-agent`

improving the WebShop score from 75.8 to 84.6. With Qwen2.5-7B-Instruct, the corresponding gains are 9.3 pp in ALFWorld success, 9.4 pp in WebShop success, and 7.9 points in WebShop score.

AAA also improves the stronger GiGPO baseline, which already includes step-level credit assignment. For the 1.5B model, GiGPO+AAA improves ALFWorld success from 86.7 to 88.9 relative to $\text{GiGPO}_{\text{w/ std}}$. On WebShop, it improves the score from 83.5 to 86.6 and the success rate from 67.4 to 71.4 relative to $\text{GiGPO}_{\text{w/o std}}$. For the 7B model, although the matched GiGPO baseline is already strong, GiGPO+AAA still gives positive aggregate gains on ALFWorld and WebShop. GiGPO+AAA increases ALFWorld success from 90.8 to 93.2, WebShop score from 86.2 to 87.4, and WebShop success from 75.2 to 77.3 relative to the corresponding normalization-matched GiGPO baselines. Overall, these results suggest that AAA not only addresses sparse rewards in practice, but also provides an innovation-based refinement signal for step-wise credit assignment.

**Alternative policy backbone.** To examine whether AAA's gains extend beyond the Qwen family, we evaluate both variants with Llama-3.2-3B-Instruct (AI@Meta, 2024) on ALFWorld, following the training and evaluation protocol used for Table 1. As shown in Table 2, AAA increases the mean success rate from 54.7% to 69.0% for GRPO and from 65.9% to 71.4% for GiGPO, improving both optimizers with this alternative backbone. The improvement is particularly pronounced for GRPO, whose AAA variant also exceeds the GiGPO baseline in mean success rate.

Table 2: ALFWorld success rates with Llama-3.2-3B-Instruct.

| Method | Success (%) |
|---|---|
| Base (ReAct) | $14.3_{\pm 2.4}$ |
| GRPO | $54.7_{\pm 3.6}$ |
| GRPO+AAA | $69.0_{\pm 4.7}$ |
| GiGPO | $65.9_{\pm 2.5}$ |
| GiGPO+AAA | $71.4_{\pm 3.6}$ |

# 7 Analysis

## 7.1 Reward Ablations and Score Analysis

**Component ablation.** Table 3 compares GRPO+AAA with three reward ablations on WebShop using Qwen2.5-1.5B-Instruct, with one training seed and three evaluation seeds. In Eq. (6), each boundary-corrected step score is normalized by the sum of scores within the successful rollout. The *w/o score-sum norm.* variant reduces the mean score by 3.5 points and the success rate by 2.1 percentage points. The *Uniform rewards* variant also lowers mean performance, by 2.4 points in score and 3.4 percentage points in success rate, with substantial variability across evaluation seeds. The largest reductions occur for *w/o success gate*, which lowers the score by 7.3 points and the success rate by 14.9 percentage points. This finding supports the outcome-conditioned design in Section 4.3, as large innovations in failed rollouts need not contribute to task success.

Table 3: Component ablation of GRPO+AAA on WebShop.

| Method | Score | SR (%) |
|---|---|---|
| GRPO+AAA | $84.9_{\pm 1.0}$ | $71.1_{\pm 2.8}$ |
| w/o score-sum norm. | $81.4_{\pm 2.5}$ | $69.0_{\pm 4.3}$ |
| Uniform rewards | $82.5_{\pm 7.2}$ | $67.7_{\pm 12.5}$ |
| w/o success gate | $77.6_{\pm 2.0}$ | $56.2_{\pm 7.4}$ |

**Association with task progress.** We examine the association between AAA's raw innovation scores and task progress on ALFWorld. Steps are labeled *progress*, *redundant*, or *invalid* using environment feedback. Under the text-based rule, *redundant* includes valid steps that do not trigger a tracked goal milestone. Figure 4(a) shows the highest mean AAA scores for progress steps and the lowest for redundant steps in both rollout sets. Figure 4(b) compares AAA with NoQ, which omits the history-basis subtraction and uses $\frac{1}{2}\log(1+\|v_t\|^2)$, and RAW, which uses $\|h_t - h_{t-1}\|$. We measure *progress enrichment* as the progress fraction among the top-$q\%$ scoring steps divided by the overall progress fraction within each seed. A value of one corresponds to random selection. At the top-$5\%$ cutoff, AAA achieves a mean enrichment of $10.62$, whereas NoQ selects no progress-labeled steps in any of the three seeds, yielding zero enrichment.

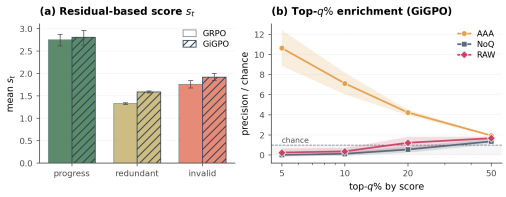



Figure 4: Raw scores and task progress on ALFWorld. (a) Mean AAA score by step category. (b) Progress enrichment for AAA, NoQ, and RAW.

These results support a positive association between our innovations-based scores and annotated task progress (Appendix E.3).

### 7.2 Trajectory Outcomes and Advantage Dispersion

**Paired task outcomes and trajectory length.** We pool matched task-level outcomes across the three evaluation seeds for each base method and its AAA variant. AAA-only successes are at least as frequent as base-only successes in all four comparisons (Figure 5, left) and more frequent in three of them. This indicates that AAA recovers more baseline failures than it loses baseline successes. Among tasks solved by both methods, the AAA variants require fewer environment steps on average (Figure 5, right). The effect is largest for GRPO on ALFWorld, where the AAA-only success share is 15.6% compared with 5.7% for GRPO-only, and the mean trajectory length significantly decreases from 12.4 to 9.6 steps.

**Within-group advantage dispersion.** We track task-specific groups of eight rollouts—all-success, mixed-outcome, and all-fail—during ALFWorld training with Qwen2.5-1.5B-Instruct. For each group, we average step advantages within each rollout and compute the standard deviation across the eight rollout means. Figure 6 shows substantial dispersion in mixed-outcome groups under both methods. In all-success groups, however, dispersion remains near zero for GRPO but stays nonzero for GRPO+AAA. As all-success groups become more common during training, AAA continues to distinguish rollouts with identical terminal outcomes, and this is beneficial for efficient training. Appendix E.6 reports the group composition and experimental details.

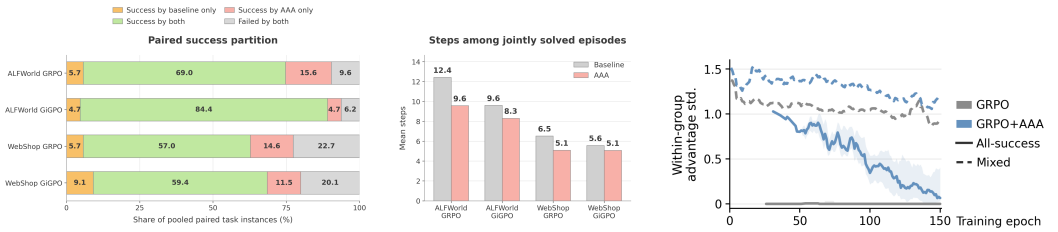



Figure 5: Paired task outcomes (left) and steps among jointly solved episodes (right).




Figure 6: Median dispersion of rollout-mean advantages.

Additional experiments in the appendices cover the effect of the history basis on scores, success offset, and sample efficiency (Appendices E.1, E.4, and E.7), search-augmented question answering (Appendix F.2), and an extension exploiting failed trajectories (Appendix F.3).

## 8 Conclusion

We presented Align as Act (AAA), a step-reward allocation method that uses the policy's own contextual representations. By representing language observations as vectors, AAA brings the geometric perspective of classical linear innovations to step-level reward design. AAA computes Gram–Schmidt residuals relative to a basis of selected earlier representation directions within each trajectory and maps their energies to logarithmic scores motivated by information decomposition. The scores are normalized within successful trajectories to obtain intrinsic rewards, which are combined with group statistics to construct step advantages. Experiments on ALFWorld and WebShop show that GRPO+AAA and GiGPO+AAA improve upon their respective baselines. Our analyses further characterize the role of the history basis and show a positive association between residual-based scores and annotated task progress.

Our evaluation focuses on ALFWorld and WebShop, with supplementary experiments using a Llama backbone and in search-augmented question answering. Evaluating AAA with larger models and a wider range of multi-turn tool-use tasks could help clarify how broadly the method applies. Combining AAA's intrinsic rewards with other group-based advantage estimators and exploring its applicability to mathematical and general language-model reasoning are promising directions for future work. Refining the failure-side extension to make more effective use of unsuccessful trajectories also merits further study.

### AI USE STATEMENT

The core idea of AAA, including its underlying motivation, was developed by the authors through their own analysis and discussions. The authors determined the research methodology, training setup, hyperparameters, experimental design, and analyses. AI-assisted coding tools were used during research execution to assist with the implementation, execution, and debugging of experimental code. Generative AI tools were also used for retrieval and discovery of potentially relevant related work and references, including older foundational and textbook references. All cited sources and bibliographic information were independently verified by the authors. For manuscript polishing, generative AI tools were used to identify grammatical and stylistic issues and to check the consistency of notation, terminology, and mathematical symbols throughout the paper. All AI-assisted code, suggestions, and edits were reviewed and validated by the authors.

### ETHICS STATEMENT

This work evaluates a training-time credit-assignment method for LLM agents entirely in established public benchmark environments. AAA operates on policy representations and rollout outcomes to construct training signals, and does not introduce additional data collection, human interaction, or access to external systems beyond those already defined by the benchmark environments. Our experiments involve neither human subjects nor personal or sensitive information, and do not evaluate deployment in real-world high-stakes settings. Accordingly, we do not identify method-specific ethical risks beyond the broader risks associated with training and deploying LLM agents.

### REPRODUCIBILITY STATEMENT

AAA does not rely on an external learned model or auxiliary module. Rather, its credit computation is based on the classical innovation process and standard linear-algebra operations applied to the policy representations, which allows the method to be implemented within an existing LLM-agent RL pipeline.

For our experiments, we build AAA on top of the publicly released GiGPO codebase (Feng et al., 2025), retrieved on April 12, 2026. We use established public benchmarks, including ALFWorld (Shridhar et al., 2021), WebShop (Yao et al., 2022), and the search benchmarks (NQ, TQ, HotpotQA, 2Wiki, MuSiQue, PopQA) described in Appendix F.2. To facilitate reproduction, we provide the complete AAA algorithms and pseudocode in Appendix C, together with implementation details, training configurations, and hyperparameters in Appendix D.

## REFERENCES

Raja Hafiz Affandi, Emily Fox, and Ben Taskar. Approximate Inference in Continuous Determinantal Point Processes. *Advances in Neural Information Processing Systems*, 26, 2013.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to Basics: Revisiting REINFORCE-Style Optimization for Learning from Human Feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024.

AI@Meta. Llama 3.2 model card, 2024. URL `https://github.com/meta-llama/llama-models/blob/main/models/llama3_2/MODEL_CARD.md`.

Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117(48):30063–30070, 2020.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by Random Network Distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Sanjiban Choudhury. Process reward models for LLM agents: Practical framework and directions. *arXiv preprint arXiv:2502.10325*, 2025.

Thomas M Cover and Joy A Thomas. *Elements of Information Theory*. Wiley-Interscience, 1991.

Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, et al. Agentic reinforced policy optimization. In *The Fourteenth International Conference on Learning Representations*, 2026.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. *Advances in Neural Information Processing Systems*, 2025.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 2013.

Geoffrey Grimmett and David Stirzaker. *Probability and Random Processes*. Oxford University Press, 2020.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. In *The Fourteenth International Conference on Learning Representations*, 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 *The Fourteenth International Conference on Learning Representations*, 2026.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, 2020.

Roger A Horn and Charles R Johnson. *Matrix Analysis*. Cambridge University Press, 2012.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. VIME: Variational information maximizing exploration. *Advances in Neural Information Processing Systems*, 29, 2016.

Zixuan Huang, Xin Xia, Yuxi Ren, Jianbin Zheng, Xuefeng Xiao, Hongyan Xie, Li Huaqiu, Songshi Liang, Zhongxiang Dai, Fuzhen Zhuang, et al. Real-time aligned reward model beyond semantics. *arXiv preprint arXiv:2601.22664*, 2026.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017.

Thomas Kailath. The innovations approach to detection and estimation theory. *Proceedings of the IEEE*, 58(5):680–695, 1970.

Thomas Kailath, Ali H Sayed, and Babak Hassibi. *Linear Estimation*. Prentice Hall, 2000.

Rudolf Emil Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

Wonjoong Kim, Yeonjun In, Sangwu Park, Dongha Lee, and Chanyoung Park. PAIR: Prefix-aware internal reward model for multi-turn agent optimization. *arXiv preprint arXiv:2605.17877*, 2026.

Alex Kulesza and Ben Taskar. Determinantal Point Processes for Machine Learning. *Foundations and Trends in Machine Learning*, 5(2-3):123–286, 2012.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Zihan Lin, Xiaohan Wang, Jie Cao, Jiajun Chai, Li Wang, Xiaodong Lu, Wei Lin, Ran He, and Guojun Yin. ResRL: Boosting LLM reasoning via negative sample projection residual reinforcement learning. *arXiv preprint arXiv:2605.00380*, 2026.

Zeyuan Liu, Jeonghye Kim, Xufang Luo, Dongsheng Li, and Yuqing Yang. Exploratory memory-augmented LLM agent via hybrid on- and off-policy optimization. In *The Fourteenth International Conference on Learning Representations*, 2026.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long papers)*, 2023.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven Exploration by Self-supervised Prediction. In *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 2017.

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. *Advances in Neural Information Processing Systems*, 2023.

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.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and Error: Exploration-Based Trajectory Optimization for LLM Agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024.

Gilbert Strang. *Linear Algebra and Its Applications*. Brooks/Cole, 2006.

Youngchul Sung, Lang Tong, and H Vincent Poor. Neyman-Pearson detection of Gauss-Markov signals in noise: Closed-form error exponent and properties. *IEEE Transactions on Information Theory*, 52(4):1354–1365, 2006.

Youngchul Sung, Xin Zhang, Lang Tong, and H Vincent Poor. Sensor Configuration and Activation for Field Detection in Large Sensor Arrays. *IEEE Transactions on Signal Processing*, 56(2): 447–463, 2008.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 2022.

Vijay Viswanathan, Yanchao Sun, Shuang Ma, Xiang Kong, Meng Cao, Graham Neubig, and Tongshuang Wu. Checklists are better than reward models for aligning language models. *Advances in Neural Information Processing Systems*, 2025.

Guoqing Wang, Sunhao Dai, Guangze Ye, Zeyu Gan, Wei Yao, Yong Deng, Xiaofeng Wu, and Zhenzhe Ying. Information gain-based policy optimization: A simple and effective approach for multi-turn search agents. In *The Fourteenth International Conference on Learning Representations*, 2026a.

Hanlin Wang, Chak Tou Leong, Jiashuo Wang, Jian Wang, and Wenjie Li. SPA-RL: Reinforcing LLM Agents via Stepwise Progress Attribution. *arXiv preprint arXiv:2505.20732*, 2025a.

Hanlin Wang, Jian Wang, Chak Tou Leong, and Wenjie Li. STeCa: Step-level Trajectory Calibration for LLM Agent Learning. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025b.

Jiawei Wang, Jiacai Liu, Yuqian Fu, Yingru Li, Xintao Wang, Yuan Lin, Yu Yue, Lin Zhang, Yang Wang, and Ke Wang. Harnessing uncertainty: Entropy-modulated policy gradients for long-horizon LLM agents. *arXiv preprint arXiv:2509.09265*, 2025c.

Yunan Wang, Minghui Song, Zihan Zhang, Shaohan Huang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, and Qi Zhang. Group-graph policy optimization for long-horizon agentic reinforcement learning. *arXiv preprint arXiv:2606.22995*, 2026b.

Norbert Wiener. *Nonlinear Problems In Random Theory*. The MIT Press, 1958.

Weimin Xiong, Yifan Song, Xiutian Zhao, Wenhao Wu, Xun Wang, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. Watch Every Step! LLM Agent Learning via Iterative Step-level Process Refinement. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024.

An Yang, Baosong Yang, Beichen Zhang, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents. *Advances in Neural Information Processing Systems*, 2022.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

Xuan Zhang, Ruixiao Li, Zhijian Zhou, Long Li, Yulei Qin, Ke Li, Xing Sun, Xiaoyu Tan, Chao Qu, and Yuan Qi. Count counts: Motivating exploration in LLM reasoning with count-based intrinsic rewards. In *The Fourteenth International Conference on Learning Representations*, 2026.

# A Additional Related Work

We discuss complementary approaches to learning from agent interactions, including trajectory refinement, process evaluation, rubric-based rewards, and exploration. We then consider representation-based learning signals and their geometric connections.

**Trajectory and step-level policy refinement.** ETO (Song et al., 2024) iteratively collects unsuccessful agent trajectories and contrasts them with successful trajectories for preference optimization. IPR (Xiong et al., 2024) introduces finer supervision by evaluating alternative actions along expert trajectories with Monte Carlo estimates of subsequent outcomes. The resulting action preferences are used together with trajectory preferences and supervised demonstrations to refine the policy. STeCa (Wang et al., 2025b) uses step-level reward comparisons to detect deviations during exploration and employs LLM-generated reflections to construct calibrated trajectories for further training. These methods improve agent policies by constructing trajectory- or action-level supervision from interaction data, whereas AAA uses post-action representations to construct intrinsic rewards and step advantages for sampled rollouts.

**Learned process evaluation.** AgentPRM (Choudhury, 2025) trains a process evaluator on Monte Carlo continuation returns, yielding action-value estimates that guide iterative policy improvement. SPA-RL (Wang et al., 2025a) learns a progress estimator whose accumulated step contributions predict the trajectory outcome, and combines these contributions with an action-executability signal to provide intermediate rewards. Both approaches learn an evaluator from outcome-based targets. AAA obtains its raw step scores directly from the policy's contextual representations and a sequential orthogonal projection, with terminal outcomes determining how these scores enter reward and advantage construction.

**Rubric-based reward construction.** Rubrics as Rewards (RaR) (Gunjal et al., 2026) uses instance-specific evaluation criteria to construct rewards for on-policy reinforcement learning. Reinforcement Learning from Checklist Feedback (RLCF) (Viswanathan et al., 2025) derives checklists from instructions and combines feedback from AI judges and specialized verifier programs into response-level rewards. These methods structure the criteria used to evaluate generated responses. AAA constructs step-level scores from the policy's contextual representations and their residuals relative to a history basis, with terminal outcomes determining how the scores are used for credit assignment.

**Intrinsic motivation and exploration.** Classical intrinsic-motivation methods supplement task rewards with signals that encourage exploration. VIME (Houthooft et al., 2016) rewards information gain about environment dynamics. Curiosity-driven exploration (Pathak et al., 2017) uses prediction errors in a learned feature space, and random network distillation (Burda et al., 2018) uses errors in predicting the outputs of a fixed random network. For LLM reasoning, MERCI (Zhang et al., 2026) uses a separately trained Coin Flipping Network to estimate state novelty through pseudo-counts. It aggregates the resulting uncertainty estimates into a trajectory-level exploration bonus for policy optimization. For multi-turn LLM agents, ARPO (Dong et al., 2026) uses changes in token-generation entropy after tool feedback to guide adaptive rollout branching. Its advantage attribution accounts for shared prefixes and branched continuations. EMPO$^2$ (Liu et al., 2026) uses self-generated memory to guide subsequent interactions and combines on- and off-policy updates. These methods modify exploration incentives or the collection and reuse of interaction experience. AAA uses its intrinsic signal to allocate credit within collected successful rollouts. The scores are normalized within each successful trajectory, and failed rollouts receive zero intrinsic reward in the base construction.

**Representation-based learning signals.** Some methods train reward predictors that take the policy's hidden representations as input. R2M (Huang et al., 2026) incorporates the policy's hidden states into a reward model and updates lightweight components as the policy evolves. PAIR (Kim et al., 2026) combines a hidden-state probe with an attention-based correction head to predict step correctness in multi-turn agent trajectories, including those with erroneous prefixes. Its probes are trained using correctness labels and remain fixed during policy optimization. AAA computes raw step scores directly from orthogonal residuals relative to the trajectory's history basis. It requires no learned reward-prediction head and uses terminal outcomes to condition reward allocation and advantage construction.

**Relation to Determinantal Point Processes** The Gaussian entropy formula in Eq. (2)–representing the maximum entropy under a second-moment constraint in information theory (Cover & Thomas, 1991)–shares a commonality with Determinantal Point Processes (DPP) (Kulesza & Taskar, 2012) in that both rely on the matrix determinant. In an $L$-ensemble formulation, a DPP defines a distribution over finite point configurations $X = \{x_1, \ldots, x_n\} \subseteq \Omega$. For a continuous ground set $\Omega \subseteq \mathbb{R}^d$, the configuration density satisfies

$$p(X) \propto \det([K(x_i, x_j)]_{1 \le i,j \le n}), \tag{9}$$

where $K$ is a positive-semidefinite likelihood kernel with a finite normalizing constant (Affandi et al., 2013). For a normalized similarity kernel, this determinant measures diversity among the corresponding feature vectors. Typically, the kernel function is designed to measure the similarity between $x_i$ and $x_j$, e.g., Gaussian kernel $\exp(-||x_i - x_j||^2/(2\sigma^2))$, cosine similarity, etc., so that the sampled realization has diversity. In particular, consider the case of cosine similarity as the kernel function:

$$K(x_i, x_j) = \langle \frac{x_i}{||x_i||}, \frac{x_j}{||x_j||} \rangle$$

For this geometric illustration, assume that the nonzero vectors have been explicitly normalized so that $\|x_i\| = 1$ for all $i$. Then, as we noted in Section 2, the matrix $[K(x_i, x_j)]_{1 \le i,j \le n}$ becomes the Gram matrix of vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ (Golub & Van Loan, 2013). Hence, the sampling density of $(x_1, \cdots, x_n)$ is proportional to the square of the volume of the parallelotope constructed by $x_1, \cdots, x_n$. When $x_1, \cdots, x_n$ are similar, i.e., highly correlated with small angles between them, the parallelotope volume is small and the joint sampling density is small. For a fixed number $n \le d$ of unit-norm vectors, mutual orthogonality maximizes the parallelotope volume and the corresponding configuration density whenever such a configuration is feasible. In this way, DPP yields repulsiveness among similar samples and enables dissimilar sampling with high diversity.

DPPs use squared feature-space volumes to define configuration densities. For jointly Gaussian observations, the differential entropy in Eq. (2) equals $\log \sqrt{\det(\Sigma)}$ up to a dimension-dependent additive constant. AAA uses sequential orthogonal residuals to construct step-level learning signals within each trajectory.

# B Additional Representation Analyses

## B.1 Interaction-Related Variation

**Data and readout.** We analyze 433 episodes across 134 tasks from ALFWorld's `valid_unseen` split, generated by the initial Qwen2.5-1.5B-Instruct policy. We use the original contextual hidden representations before random projection, extracted at the fixed boundary described in Appendix C.2.

**Window construction.** Within each episode, we form windows $\mathcal{W}$ of $|\mathcal{W}| = w = 6$ consecutive post-action representations. We categorize a window as *repetitive* if it contains at most two distinct action–feedback text pairs, and as *diverse* if it contains at least five. Windows with three or four distinct pairs are excluded from this comparison. We match nonoverlapping repetitive and diverse windows within the same episode, obtaining 357 matched pairs from 274 episodes across 125 tasks.

**Measurement and aggregation.** For each window, we subtract the within-window mean from the hidden representations and compute the empirical participation ratio $D_{\mathcal{W}}$ defined in Section 3. Higher values indicate variation distributed across more effective directions. For each matched pair, we define

$$\Delta D := D_{\mathcal{W}^{\mathrm{div}}} - D_{\mathcal{W}^{\mathrm{rep}}}.$$

We first average $\Delta D$ within each task and then give each task equal weight in the overall and task-type summaries. Uncertainty is reported using a task-bootstrap confidence interval.

**Results.** The overall task-equal mean of $\Delta D$ is $0.690$, with a 95% confidence interval of $[0.569, 0.808]$. The task-level mean of $\Delta D$ is positive in 109 of the 125 matched tasks, and its task-type mean is positive for all six task types. These results characterize the association between interaction patterns and local representation variation.

### B.2 Effect of Random Projection

We repeat the analysis in Appendix B.1 after applying the fixed random projection $P \in \mathbb{R}^{512\times1536}$. The trajectories, window labels, and matched pairs remain unchanged. We center the projected representations within each window and compute $D_{\mathcal{W}}$ using the same definition and aggregation procedure as in the original space.

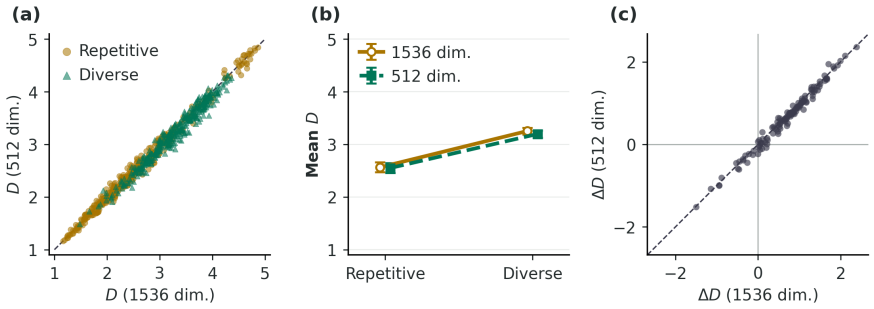



Figure 7: Interaction-related variation before and after random projection from 1536 to 512 dimensions. (a) Window-level $D_{\mathcal{W}}$. (b) Mean $D_{\mathcal{W}}$ by window category. (c) Matched-pair contrasts $\Delta D$. Dashed diagonals in (a) and (c) indicate equality.

Figure 7(a) shows that window-level values of $D_{\mathcal{W}}$ remain close to their original values after projection for both repetitive and diverse windows. Figure 7(b) shows that diverse windows retain a higher mean $D_{\mathcal{W}}$ in the projected space. Figure 7(c) further shows that the matched-pair contrasts remain close to those in the original space, although their mean decreases slightly. Thus, the observed contrast in local directional variation is preserved under this projection, supporting the use of 512-dimensional representations for AAA's subsequent residual computation.

### B.3 Sensitivity to Extraction Position and Layer

**Readout choice.** AAA extracts the contextual representation at the newline immediately after <|im_end|> and before <|im_start|>assistant. This position follows the complete user-side context, including the latest environment feedback, but precedes the assistant header. It therefore provides a consistent boundary at which to read the interaction context across steps. We examine the geometry of this readout relative to nearby positions in the same formatted prompt.

**Setup.** Using Qwen2.5-1.5B-Instruct before RL training, we compare five successive positions: the user-end token <|im_end|>, the following newline used by AAA, the assistant-start token <|im_start|>, the last token of the role text assistant, and the newline after the assistant header. All five positions precede the generated response.

For each position, we measure the projected norm $\|v\| = \|Ph\|$ and the residual-energy fraction $\|e\|^2/\|v\|^2$. The position comparisons use the final-normalized output of layer 27 and the fixed $1536 \to 512$ projection. The layer comparison uses zero-based decoder layers 6, 13, 20, 26, and 27. The separate *27 + RMSNorm* setting applies the final normalization to the same last-layer output and is not an additional decoder layer.

Each episode, position, and layer uses its own sequential history basis. We summarize each episode by the median over finite post-action steps $j \geq 2$, excluding the initial empty-basis step, and give each of the 78 tasks equal weight.

**Results.** Figure 8(a) shows that projected norms depend on the extraction position. The newline after the user end has a relatively large norm, comparable to the newline after the assistant header. In Figure 8(b), the selected readout has a low fraction similar to those at the user-end and assistant-start tokens, whereas the role text and the following newline have larger fractions. Thus, projected norm alone does not determine the fraction of energy unexplained by the history basis. Figure 8(c) shows a pronounced increase near the final decoder layer at the role-text position and the newline after the

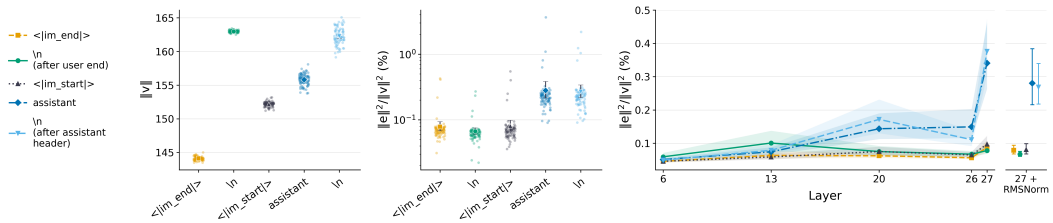



Figure 8: Extraction-position and layer sensitivity before RL training. Left, projected norm $\|v\|$ after final RMSNorm. Middle, residual-energy fraction $\|e\|^2/\|v\|^2$ at the same output. Right, residual-energy fraction across layers. Fractions are expressed as percentages. Error bars and shading show 95% task-bootstrap intervals.

assistant header. The selected readout exhibits a smaller variation in residual-energy fraction across the sampled layers, without the same final-layer increase. These curves compare residual fractions relative to separately constructed history bases, rather than changes in the hidden vectors between layers.

Together with its position before the assistant header, this comparatively stable profile supports the selected newline as a consistent readout for AAA.

### B.4 Success-Conditioned Hidden-State Subspace Overlap: An Effective-Rank Diagnostic Γ

We examine whether successful rollouts for the same task exhibit greater overlap in their observation-conditioned hidden representations than pairs involving failures. We quantify this comparison using an SVD-based effective-rank diagnostic that measures how much additional rank is required when two trajectory representations are combined.

**Diagnostic setup** We conduct this representation-level diagnostic using Qwen3-4B-Instruct-2507 (Yang et al., 2025) with ReAct (Yao et al., 2023) on 133 ALFWorld tasks (Shridhar et al., 2021), using the ETO evaluation codebase (Song et al., 2024). Thus, the exact prompt template and tokenization may differ from those used in GiGPO and our training codebase, but both implementations target the same functional boundary: after the agent has observed the current state and before it generates the next thought and action. For each task, we generate up to five ReAct rollouts and split them into successful and failed trajectories using the environment success label.

**Hidden-state extraction** For each step, we extract the last-layer hidden state at the chat-template transition boundary immediately before the assistant begins producing its next thought and action. The two codebases instantiate this boundary with slightly different prompt templates. In the ETO-style format used for this diagnostic, the current observation appears in the user block, followed by the user-block terminator and the assistant-start boundary:

ETO-style rollout format

```
<|im_start|>user
Observation: On the desk 2, you see a bowl 1 ...
<|im_end|>\n                                    ⟵ extract last-layer hidden state
<|im_start|>assistant\n
Thought: I found a bowl 1 ...
Action: take bowl 1 from desk 2
<|im_end|>
```

This example illustrates that the extracted vector summarizes the observation-conditioned context immediately before the next thought and action are generated. Because the same functional boundary appears at every step of a multi-step ReAct trajectory, it provides a consistent step-wise hidden representation for comparing rollouts.

**Effective-rank diagnostic** For each task, we form all available pairs of rollouts within that task and categorize them as success-success (S-S), success-failure (S-F), or failure-failure (F-F). We do not compare rollouts from different tasks. For a pair $(\tau^i, \tau^j)$, we stack the extracted step hidden states into trajectory matrices

$$\mathbf{H}^i = [\mathbf{h}^i_1, \ldots, \mathbf{h}^i_{T_i}], \qquad \mathbf{H}^j = [\mathbf{h}^j_1, \ldots, \mathbf{h}^j_{T_j}],$$

where $\mathbf{H}^i \in \mathbb{R}^{d \times T_i}$ and $\mathbf{H}^j \in \mathbb{R}^{d \times T_j}$. We then compute the effective rank of $\mathbf{H}^i$, $\mathbf{H}^j$, and the concatenated matrix $[\mathbf{H}^i, \mathbf{H}^j]$ by singular value decomposition. If the singular values of a matrix are $\sigma_1, \ldots, \sigma_T$, its effective rank is the smallest $k$ that explains 90% of the squared singular-value mass:

$$\mathrm{rank}_e := \min \left\{ k : \sum_{\ell=1}^{k} \sigma_\ell^2 \geq 0.9 \sum_{\ell=1}^{T} \sigma_\ell^2 \right\}.$$

We define the effective-rank diagnostic as

$$\Gamma = \frac{\mathrm{rank}_e([\mathbf{H}^i, \mathbf{H}^j])}{(\mathrm{rank}_e(\mathbf{H}^i) + \mathrm{rank}_e(\mathbf{H}^j))/2}.$$

When concatenating $\mathbf{H}^i$ and $\mathbf{H}^j$ requires little additional effective rank relative to the two matrices considered separately, $\Gamma$ is close to 1. Under this diagnostic, lower values are consistent with greater overlap between the dominant hidden-state subspaces, whereas larger values indicate greater effective-rank expansion after concatenation.

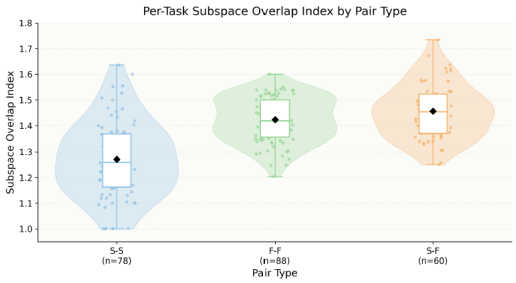



Figure 9: Task-level effective-rank diagnostic $\Gamma$ for S–S, F–F, and S–F rollout pairs. Each point is a task-level aggregate for the indicated pair type rather than a raw rollout pair; violins show the resulting distribution across tasks.

**Task-level aggregation** Figure 9 visualizes a distribution over tasks rather than a distribution over individual rollout pairs. Since the numbers of successful and failed rollouts vary across tasks, we first average the pairwise $\Gamma$ values within each task and pair type, obtaining one task-level $\Gamma$ value whenever that pair type is available. We then plot the resulting task-level values for S–S, F–F, and S–F as violin distributions with box and scatter overlays.

Using the extraction boundary described above and final-layer hidden states, the final figure contains 78 S–S task-level indices, 88 F–F indices, and 60 S–F indices, with mean task-level $\Gamma$ values of 1.271, 1.423, and 1.456, respectively. The lower mean $\Gamma$ for S–S pairs indicates that, under this diagnostic, combining two successful rollouts for the same task introduces less additional effective rank than combining pairs involving at least one failed rollout. This pattern is consistent with greater outcome-associated hidden-subspace overlap among successful rollouts, but it does not identify the shared directions semantically or directly validate AAA's step-wise innovation reward.

# C ALGORITHM DETAILS

Our algorithm AAA is composed of three steps:

1. Semantic vector extraction - Output: $\mathbf{h}_t^i$
2. Innovation computation - Output: $\mathbf{e}_t^i$
3. Advantage computation - Output: $\hat{A}_{i,t}$

We explain each step in detail here. [2] All vectors are column vectors throughout this paper.

## C.1 TIME INDEXING IN THIS PAPER

We use the interaction order defined in the Problem Setup of Section 2; Figure 10 gives the detailed rollout alignment. In standard RL conventions, the observation resulting from action $a_t$ is typically denoted by $o_{t+1}$. In this paper, however, we denote it as $o_t$ to align the time indices of cause (action) and effect (observation) for simplicity. The action $a_t$ is generated from the formatted context $c_t$, which includes the task and available recent history, not just $o_{t-1}$.
Here, $o_t$ denotes the feedback returned after $a_t$, whereas $c_{t+1}$ denotes the full formatted input used for the next policy generation, when such a generation occurs. This prompt combines the feedback with the task and available recent history. The final feedback $o_T$ is not followed by another policy generation and is not separately encoded.

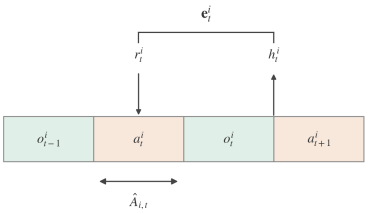



Figure 10: Timing diagram for rollout $\tau^i$.

## C.2 SEMANTIC VECTOR EXTRACTION

Following the interaction order in the Problem Setup of Section 2, we extract $h_t$ from the last user-side token position of the actual next prompt $c_{t+1}$, between feedback $o_t$ and action $a_{t+1}$, at the LLM's final Transformer layer. This readout captures the formatted context containing the feedback induced by action $a_t$. In implementation, the extraction position is the fixed chat-template boundary following the full user-side prompt, specifically the newline after <|im_end|> and before <|im_start|>assistant.

The abbreviated example below illustrates the extraction of $h_{14}$ from the input context $c_{15}$ before the policy generates a response for step 15.

Illustrative Qwen2.5 input: $c_{15}$

```
<|im_start|>user
You are an expert agent ...
Prior to this step, you have already taken 14 step(s).
[Observation 13: ... Action 13: ...]
[Observation 14: ... Action 14: ...]
You are now at step 15 and your current observation is: On the desk 2,
you see a bowl 1.
Your admissible actions are: ...
Now it is your turn to take an action.
<|im_end|>\n                                        ⟵ extract h_14
<|im_start|>assistant\n
```

[2] The step advantages returned by Algorithms 1 and 2 are assigned to the valid response tokens generated at the corresponding steps and used in the standard GRPO objective.

**Algorithm 1** AAA advantage construction for GRPO

**Require:** A task-specific rollout group $\{\tau^i\}_{i=1}^{G}$ with outcomes, old policy $\pi_{\theta_{\text{old}}}$, fixed projection $P \in \mathbb{R}^{m\times d}$, original GRPO advantages $\{\widehat{A}_{i,t}^{\text{GRPO}}\}$, novelty threshold $\kappa$, success offset $\delta$, and numerical constant $\varepsilon_{\text{norm}}$

Initialize the list of intrinsic rewards $\mathcal{C} \leftarrow [\,]$
**for** each rollout $\tau^i$ **do**
  **if** $\tau^i$ succeeds **then**
    Extract post-action representations $\{\mathbf{h}_t^i\}_{t=1}^{T_i-1}$
    $Q_0^i \leftarrow \emptyset$
    **for** $t = 1, \ldots, T_i - 1$ **do**
      $\mathbf{v}_t^i \leftarrow P\mathbf{h}_t^i$
      $\mathbf{e}_t^i \leftarrow \mathbf{v}_t^i - Q_{t-1}^i (Q_{t-1}^i)^\top \mathbf{v}_t^i$
      $s_t^i \leftarrow \frac{1}{2}\log\big(1 + \|\mathbf{e}_t^i\|^2\big)$ Eq. (5)
      $Q_t^i \leftarrow Q_{t-1}^i$
      **if** $\mathbf{e}_t^i \neq \mathbf{0}$ and $\|\mathbf{e}_t^i\| \geq \kappa\|\mathbf{v}_t^i\|$ **then**
        $Q_t^i \leftarrow \left[Q_{t-1}^i,\ \dfrac{\mathbf{e}_t^i}{\|\mathbf{e}_t^i\|}\right]$
      **end if**
    **end for**
    **if** $T_i \leq 2$ **then**
      $r_t^i \leftarrow 0, \quad t = 1, \ldots, T_i$
    **else**
      $m_i \leftarrow \dfrac{1}{T_i - 2}\sum_{t=2}^{T_i-1} s_t^i$
      $\bar{s}_1^i \leftarrow m_i, \quad \bar{s}_{T_i}^i \leftarrow m_i$
      $\bar{s}_t^i \leftarrow s_t^i, \quad t = 2, \ldots, T_i - 1$
      $S_i \leftarrow \sum_{k=1}^{T_i} \bar{s}_k^i$
      **if** $S_i > \varepsilon_{\text{norm}}$ **then**
        $r_t^i \leftarrow \dfrac{\bar{s}_t^i}{S_i}, \quad t = 1, \ldots, T_i$ Eq. (6)
      **else**
        $r_t^i \leftarrow 0, \quad t = 1, \ldots, T_i$
      **end if**
    **end if**
  **else**
    $r_t^i \leftarrow 0, \quad t = 1, \ldots, T_i$ Eq. (7)
  **end if**
  Extend $\mathcal{C}$ with $\{r_t^i\}_{t=1}^{T_i}$
**end for**
$\mu_G \leftarrow \text{mean}(\mathcal{C}), \quad \sigma_G \leftarrow \text{std}_{\text{pop}}(\mathcal{C})$
**for** each rollout $\tau^i$ and step $t = 1, \ldots, T_i$ **do**

$$\widehat{A}_{i,t}^{\text{AAA}} \leftarrow \begin{cases} \max\left(0,\ \dfrac{r_t^i - \mu_G}{\sigma_G + \varepsilon_{\text{norm}}}\right) + \delta, & \tau^i \text{ succeeds}, \\ \widehat{A}_{i,t}^{\text{GRPO}}, & \tau^i \text{ fails} \end{cases} \quad \text{Eq. (8)}$$

**end for**
**return** $\{\widehat{A}_{i,t}^{\text{AAA}}\}_{i,t}$

### C.3 Random Projection

The hidden dimension $d$ is typically large, so AAA stores only a low-dimensional projection

$$v_t = Ph_t \in \mathbb{R}^m, \qquad P = \frac{1}{\sqrt{m}}[p_{ij}] \in \mathbb{R}^{m\times d},$$

where $p_{ij} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1)$. The projection matrix is sampled once and kept fixed during rollout collection and reward computation. We do not backpropagate through $P$.

### C.4 Innovation Computation

Given projected observation vectors $\{v_1, \ldots, v_{T-1}\}$, we initialize $\mathbf{Q}_0 \in \mathbb{R}^{m\times 0}$ as an empty matrix. For each step $t$, let $\mathbf{Q}_{t-1} \in \mathbb{R}^{m\times k_{t-1}}$ contain the $k_{t-1}$ normalized innovation directions retained before step $t$. We compute

$$\mathbf{e}_t = v_t - \mathbf{Q}_{t-1}\mathbf{Q}_{t-1}^\top v_t.$$

The basis is updated as

$$Q_t = \begin{cases} [Q_{t-1},\, e_t/\|e_t\|], & e_t \neq 0 \text{ and } \|e_t\| \geq \kappa\|v_t\|, \\ Q_{t-1}, & \text{otherwise.} \end{cases}$$

where $\kappa = 0.03$ in all reported experiments.

The residual $\mathbf{e}_t$ captures the component of the current observation that is not explained by the retained directions from previous observations. We define the raw innovation score as

$$s_t^i = \frac{1}{2}\log\bigl(1 + \|e_t^i\|^2\bigr)\,.$$

**Boundary handling.** Let $s_t^i$ denote the raw innovation score in Eq. (5). For a successful rollout $\tau^i$ with $T_i > 2$, we assign the two boundary steps the mean interior score

$$m_i = \frac{1}{T_i - 2}\sum_{t=2}^{T_i-1} s_t^i,$$

and define the boundary-corrected scores

$$\bar{s}_t^i = \begin{cases} m_i, & t = 1 \text{ or } t = T_i, \\ s_t^i, & 2 \leq t \leq T_i - 1. \end{cases}$$

The normalized AAA intrinsic rewards are then computed from $\bar{s}_t^i$ as in Eq. (6). For $T_i \leq 2$, the innovation-based rewards are set to zero.

### C.5 Advantage Computation and Policy Optimization

**GRPO+AAA.** We compute the normalized intrinsic rewards using Eqs. (6)–(7) and the boundary handling above. The group statistics include all steps, with zero-valued rewards for failed and degenerate successful rollouts. Equation (8) then gives the step advantages: successful steps receive the clipped group-relative term plus $\delta$, and failed steps retain their original GRPO advantages.

As illustrated in Fig. 10, the post-action context containing $o_t^i$ is encoded as $\mathbf{h}_t^i$ and used to compute the innovation score $s_t^i$ and normalized intrinsic reward $r_t^i$.

**GiGPO+AAA.** Original GiGPO combines an episode-level macro advantage with an anchor-state step-level advantage. In our extension, we instead use the GRPO+AAA advantage $\widehat{A}_{i,t}^{\text{AAA}}$ from Eq. (8) as the base term and add a GiGPO-style anchor-group correction computed from the normalized intrinsic rewards. Thus, GiGPO+AAA adopts the anchor-state grouping mechanism of GiGPO rather than reproducing its original advantage estimator unchanged.

Let $o_{t-1}^{i,\mathrm{raw}}$ denote the history-free current-observation string used as the anchor key for step $t$, with $o_0^{i,\mathrm{raw}}$ denoting the initial observation before the first action. Anchor matching is performed by exact string matching within the $G$ rollouts generated for the same task. Here, $r_t^i$ denotes the normalized rewards from Eq. (6). For each anchor observation $\tilde{o}^{\mathrm{raw}}$, define

$$\mathcal{H}(\tilde{o}^{\mathrm{raw}}) = \left\{(j,k) : o_{k-1}^{j,\mathrm{raw}} = \tilde{o}^{\mathrm{raw}}\right\}, \qquad \bar{r}(\tilde{o}^{\mathrm{raw}}) = \frac{1}{|\mathcal{H}(\tilde{o}^{\mathrm{raw}})|} \sum_{(j,k)\in\mathcal{H}(\tilde{o}^{\mathrm{raw}})} r_k^j.$$

For an anchor group with $|\mathcal{H}(\tilde{o}^{\mathrm{raw}})| > 1$, let its sample standard deviation be

$$\sigma_{\mathcal{H}}(\tilde{o}^{\mathrm{raw}}) = \left[\frac{1}{|\mathcal{H}(\tilde{o}^{\mathrm{raw}})| - 1} \sum_{(j,k)\in\mathcal{H}(\tilde{o}^{\mathrm{raw}})} \left(r_k^j - \bar{r}(\tilde{o}^{\mathrm{raw}})\right)^2\right]^{1/2}.$$

The anchor-group correction is

$$B_{i,t} = \begin{cases} 0, & |\mathcal{H}(o_{t-1}^{i,\mathrm{raw}})| \le 1, \\ r_t^i - \bar{r}(o_{t-1}^{i,\mathrm{raw}}), & \texttt{mean_norm}, \\ \dfrac{r_t^i - \bar{r}(o_{t-1}^{i,\mathrm{raw}})}{\sigma_{\mathcal{H}}(o_{t-1}^{i,\mathrm{raw}}) + \varepsilon_{\mathrm{norm}}}, & \texttt{mean_std_norm}. \end{cases}$$

The final advantage is

$$\widehat{A}_{i,t}^{\mathrm{GiGPO+AAA}} = \widehat{A}_{i,t}^{\mathrm{AAA}} + \lambda_d B_{i,t},$$

where $\lambda_d$ controls the strength of the anchor-group correction.

Thus, GiGPO+AAA keeps the GRPO+AAA base advantage and adds an anchor-group correction; the correction does not replace the base AAA advantage. The correction is computed directly from the AAA rewards, without additional temporal discounting. Failed rollouts have zero AAA rewards as in Eq. (7) but still participate in the anchor-group statistics in $B_{i,t}$. The final advantage is assigned to the valid generated response tokens of step $t$ and optimized using the same GRPO-style objective.

## D Implementation and Training Details

**Codebase and models.** We implement AAA on top of the released GiGPO codebase and use its ALFWorld and WebShop environment interfaces, rollout format, and default optimization recipe unless otherwise specified. We use Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct as the policy backbones. This setup allows us to compare GRPO, GiGPO, and their AAA-augmented variants under the same interaction budget and environment protocol.

**Training and evaluation setup.** Table 4 summarizes the shared training configuration. For both environments, each training batch contains 16 task instances and each instance is expanded into a rollout group of size 8. We train for 150 epochs and use a validation batch size of 128. ALFWorld uses a maximum episode length of 50, while WebShop uses 15 steps. The prompt contains the current observation together with the recent interaction history following the default GiGPO prompt construction. For each reported main result, we train three independent models with different training seeds and evaluate each checkpoint with three independent final-evaluation seeds; we report the resulting mean and standard deviation.

**Computation resource.** All AAA training runs use two H200 GPUs per run. For Table 3 and Table 7 in Appendix E.5, we use two RTX-Pro-6000 GPUs for both training and evaluation. Hardware-dependent runtime settings, such as per-GPU micro-batch sizes, FSDP offloading, and vLLM execution/cache options, are adjusted only to fit the available hardware. We do not change the rollout budget, effective batch size, policy objective, learning rate, KL coefficient, or environment protocol.

Table 4: Shared training settings for ALFWorld and WebShop. Memory-only micro-batch sizes are adjusted by model size and hardware, while the interaction budget and optimization hyperparameters are kept fixed.

| Setting | ALFWorld | WebShop |
|---|---|---|
| Training batch size | 16 | 16 |
| Rollouts per task | 8 | 8 |
| Validation batch size | 128 | 128 |
| Training epochs | 150 | 150 |
| Maximum environment steps | 50 | 15 |
| Maximum prompt length | 2048 | 4096 |
| Maximum response length | 512 | 512 |
| PPO mini-batch size | 256 | 64 |
| Learning rate | $1 \times 10^{-6}$ | $1 \times 10^{-6}$ |
| KL coefficient | 0.01 | 0.01 |
| Invalid-action penalty | 0.1 | 0.1 |

Table 5: AAA-specific hyperparameters and GiGPO integration settings. The GiGPO normalization mode is fixed per environment.

| Hyperparameter | GRPO+AAA | GiGPO+AAA |
|---|---|---|
| Projection dim. $m$ | 512 | 512 |
| Extraction layer | final | final |
| GS threshold $\kappa$ | 0.03 | 0.03 |
| Success offset $\delta$ | 0.1 | 0.1 |
| Normalization constant $\varepsilon_{\text{norm}}$ | $10^{-6}$ | $10^{-6}$ |
| Anchor-group norm. | – | `mean_std_norm` (ALFWorld)<br>`mean_norm` (WebShop) |
| GiGPO weight $\lambda_d$ | – | 1.0 |

**AAA hyperparameters.** Table 5 lists the AAA-specific hyperparameters used in GRPO+AAA and GiGPO+AAA. The projection dimension is fixed to 512. The hidden dimension follows the backbone model: 1536 for Qwen2.5-1.5B-Instruct and 3584 for Qwen2.5-7B-Instruct. We extract hidden states from the final layer at the observation-conditioned assistant boundary described in Appendix C.2. The rollout-local Gram-Schmidt threshold is $\kappa = 0.03$, as defined in Algorithm 1. We use the default success offset $\delta = 0.1$ across ALFWorld and WebShop. For GiGPO+AAA, we use `mean_std_norm` on ALFWorld and `mean_norm` on WebShop. The normalization mode is fixed within each environment across both model sizes and all training seeds, and we set $\lambda_d = 1.0$ in both environments.

# E Additional Ablations and Analyses

## E.1 Effect of the History Basis on Step Scores

We diagnose the effect of history-basis subtraction using alternative next actions from the same environment state and interaction history. We refer to these alternatives as candidates and score the representation obtained after each candidate's feedback is incorporated into the policy context. The pre-action history basis is held fixed across all candidates at that prefix.

**Experimental setup.** We use the initial Qwen2.5-1.5B-Instruct checkpoint on 78 ALFWorld `valid_unseen` tasks, with 13 tasks from each of the six task types. The collection contains 234 episodes. At selected prefixes, we execute alternative actions from the same state and history and extract their post-action representations using the fixed readout in Appendix C.2. We analyze prefixes with at least two eligible nonterminal candidates, obtaining 411 prefixes from 227 episodes and 1,129 candidate interactions.

The model and random projection are fixed throughout the comparison. At each prefix, all candidates use the basis $Q = Q_{t-1}$ accumulated before the decision point. The basis remains unchanged while scoring the alternatives.

**Representation components.** For a prefix with $K$ candidate actions, let $h_j$ denote the post-action representation of candidate action $j$. We compute

$$v_j = Ph_j, \qquad e_j = v_j - QQ^\top v_j, \qquad j = 1, \ldots, K.$$

To examine how the residual calculation affects components shared across candidates and differences among them, we decompose each candidate set into its mean and deviations from that mean, which are described as

$$\bar{v} = \frac{1}{K}\sum_{j=1}^{K} v_j, \qquad \bar{e} = \frac{1}{K}\sum_{j=1}^{K} e_j$$

so that we have $v_j = \bar{v} + (v_j - \bar{v})$ and $e_j = \bar{e} + (e_j - \bar{e})$. We report the percentages of candidate-mean energy and centered-variation energy retained after history-basis subtraction:

$$100\frac{\|\bar{e}\|^2}{\|\bar{v}\|^2}, \qquad 100\frac{\sum_{j=1}^{K}\|e_j - \bar{e}\|^2}{\sum_{j=1}^{K}\|v_j - \bar{v}\|^2},$$

respectively. Note that AAA's residual calculation uses the history-based projection $QQ^\top v_j$.

**Raw-score contrast.** We apply the same logarithmic function to the projected representations and their residuals,

$$s_{v,j} = \tfrac{1}{2}\log\left(1 + \|v_j\|^2\right), \qquad s_{e,j} = \tfrac{1}{2}\log\left(1 + \|e_j\|^2\right).$$

For each prefix, we measure the difference between the highest and lowest candidate scores,

$$C_v = \max_j s_{v,j} - \min_j s_{v,j}, \qquad C_e = \max_j s_{e,j} - \min_j s_{e,j}.$$

**Aggregation and results.** We compute each quantity at the prefix level and average successively within episodes, tasks, and task types. The reported means give equal weight to the six task types. Energy percentages are obtained by averaging the prefix-level ratios. We estimate 95% confidence intervals using 10,000 paired task-cluster bootstrap resamples within task type.

Figure 11(a) shows that the residual calculation leaves $0.0342\%$ of the normalized candidate-mean energy and $60.50\%$ of the normalized centered-variation energy. The history basis therefore removes the mean component much more strongly than the deviations that distinguish the candidates. Figure 11(b) shows that the mean within-prefix score range increases from $0.00320$ for $s_v$ to $0.25860$ for $s_e$. The range increases at 410 of the 411 analyzed prefixes, and the task-level mean change is positive for all 78 tasks. Together, these results show that history-basis subtraction changes the balance between the candidate-mean component and candidate-specific variation, making differences among the resulting raw scores more pronounced.

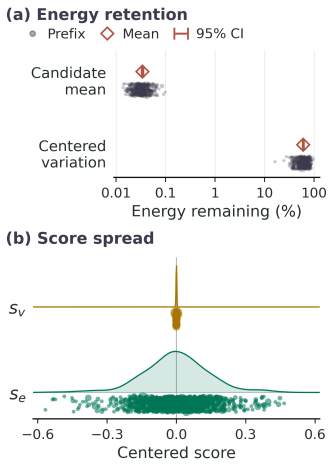



Figure 11: History-basis effects on same-prefix candidates: (a) component-energy retention and (b) centered raw-score distributions.

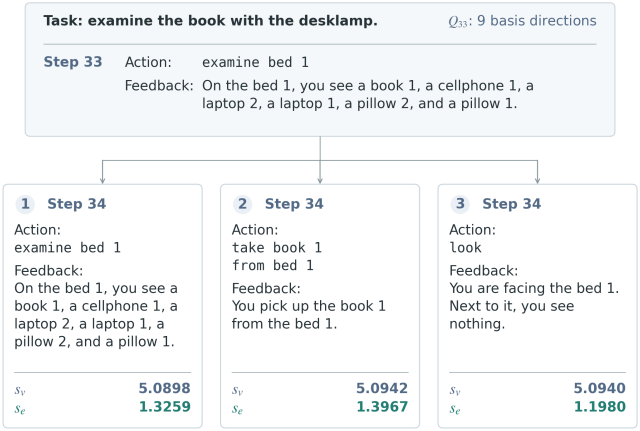



Figure 12: One-step branches from the same ALFWorld state and history. Projected-norm scores $s_v$ and residual scores $s_e$ are compared with $Q_{33}$ fixed across candidates.

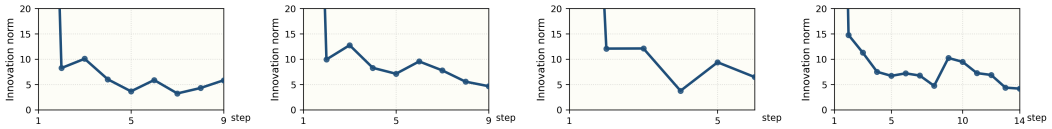
**ALFWorld case study.** In the book-and-desklamp example in Figure 12, we fix the environment state, interaction history, and pre-action prompt after Step 33 and compare three actions at Step 34. All branches use the same model, projection $P$, and history basis $Q_{33}$, which contains nine directions accumulated through Step 33. We compute each branch's post-action representation and residual without updating $Q_{33}$ between candidates. [3]

At the branching point, the agent has just executed `examine bed 1` and observed a book among the objects on the bed. Repeating this action returns the same object listing. The alternative `take book 1 from bed 1` produces feedback confirming acquisition of the book, and `look` reports the agent's position and immediate surroundings. These different interactions yield nearly identical projected-norm scores around 5.09, with $C_v = 0.0043$. Their residual scores are more clearly separated. Book acquisition receives $s_e = 1.3967$, repeated examination receives 1.3259, and `look` receives 1.1980, giving $C_e = 0.1988$.

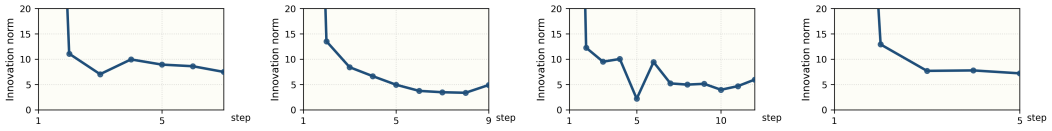
Because the history basis is identical across branches, this separation reflects differences in how the post-action representations align with the same preceding representation subspace. The book-acquisition branch has the largest component outside that subspace, despite its projected norm being close to those of the inspection branches. The example therefore illustrates how history-relative scoring distinguishes alternative next interactions whose differences are weakly expressed by their overall representation norms.

### E.2 Innovations versus Time Step

Fig. 13 visualizes how the innovation norm $\|\mathbf{e}_t\|$ varies over time in selected ALFWorld and WebShop rollouts. Since AAA assigns credit using the innovation extracted from the observation reached after each action, $\|\mathbf{e}_t\|$ can be viewed as a proxy for how much new semantic information the transition reveals relative to the previous trajectory representation. Successive post-action representations yield residuals of different magnitudes relative to the history basis. These differences provide the step-level signal used in AAA's reward construction.




(a) ALFWorld




(b) WebShop

Figure 13: Innovation norm $\|\mathbf{e}_t\|$ over time steps for selected rollouts in (a) ALFWorld and (b) WebShop. Each small panel shows a single trajectory and first-step values are capped at 20 for display.

**ALFWorld trajectory-level interpretation** The following ALFWorld example expands one rollout from Fig. 13 in prompt format. We display each $\|\mathbf{e}_t\|$ value between an action and the observation reached by that action. The bent arrow points back to the preceding action: AAA constructs the innovation after receiving the next-step observation and assigns the resulting score to the action that produced it. Thus, the value reflects the residual magnitude of the newly observed state information, rather than the novelty of the action string itself.

[3] The `look` branch was obtained in a subsequent controlled-action collection under the same pre-action conditions and selected independently of its score.

**ALFWorld trajectory example**

**Task.** Cool some pan and put it in diningtable.

**Step 1** `Your current observation is: -= Welcome to TextWorld, ALFRED! =- You are in the middle of a room. Looking quickly around you, you see a cabinet 11, a cabinet 10, a cabinet 9, a cabinet 8, a cabinet 7, a cabinet 6, a cabinet 5, a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 2, a countertop 1, a diningtable 1, a drawer 3, a drawer 2, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.`
`action: go to diningtable 1`
↥ $\|\mathbf{e}_t\| = 8.76$
**Step 2** `your current observation is: You arrive at diningtable 1. On the diningtable 1, you see nothing.`
`action: go to countertop 1`
↥ $\|\mathbf{e}_t\| = 12.08$
**Step 3** `your current observation is: You arrive at countertop 1. On the countertop 1, you see a bread 2, a butterknife 2, a cup 1, a dishsponge 3, a dishsponge 1, a fork 2, a lettuce 1, a mug 2, a pan 1, a plate 1, a spatula 3, and a spatula 1.`
`action: take pan 1 from countertop 1`
↥ $\|\mathbf{e}_t\| = 12.11$
**Step 4** `your current observation is: You pick up the pan 1 from the countertop 1.`
`action: go to fridge 1`
↥ $\|\mathbf{e}_t\| = 3.75$
**Step 5** `your current observation is: You arrive at fridge 1. The fridge 1 is closed.`
`action: cool pan 1 with fridge 1`
↥ $\|\mathbf{e}_t\| = 9.38$
**Step 6** `your current observation is: You cool the pan 1 using the fridge 1.`
`action: go to diningtable 1`
↥ $\|\mathbf{e}_t\| = 6.46$
**Step 7** `your current observation is: You arrive at diningtable 1. On the diningtable 1, you see nothing.`
`action: move pan 1 to diningtable 1`

This example highlights that the innovation norm does not directly score the importance of an action itself. Instead, it measures how much semantic residual the observation reached after that action contributes beyond the previous trajectory representation. For example, the low value at Step 5 does not imply that "go to fridge 1" is unimportant; rather, the resulting observation, "You arrive at fridge 1. The fridge 1 is closed," is a relatively predictable state transition given the initial observation and the navigation action. In contrast, the pan-acquisition observation at Step 4 and the cooled-pan observation at Step 6 reveal task-relevant state changes more directly, and therefore receive larger innovation norms.

Overall, this qualitative example illustrates that $\|\mathbf{e}_t\|$ is best understood as a signal for newly acquired observation-level semantic information, rather than as a direct score of which action appears important to a human reader.

### E.3 Details of the Task-Progress Analysis

We describe details of the step annotations, scoring rules, and evaluation used in Figure 4.

**Step annotations.** When the goal-condition success-rate signal is unavailable, we annotate ALFWorld interactions using a text-based fallback. The task type, target object, and receptacle are parsed from the gamefile path to define an ordered sequence of task-specific milestones. For example, a heating task tracks pickup, heating, and placement of the target object.

A step is labeled *invalid* if its recorded action-validity flag is zero or its feedback begins with `Nothing happens`. Otherwise, it is labeled *progress* if its feedback matches the next untriggered milestone predicate. Predicates are checked in order and can each be triggered at most once per trajectory. A valid final step of a successful trajectory is also labeled *progress*. All remaining steps are labeled *redundant*. This category can include necessary navigation, inspection, and prerequisite actions that do not trigger a tracked milestone. We utilize these annotations only for analysis.

**Scoring rules.** We compare three raw-score rules on the same hidden representations and step labels: AAA, NoQ, and RAW. NoQ and RAW are alternative scoring rules applied to the recorded representations.

With $v_t = Ph_t$ and $e_t = v_t - Q_{t-1}Q_{t-1}^\top v_t$, the scores are

$$s_t^{\mathrm{AAA}} = \tfrac{1}{2}\log\bigl(1 + \|e_t\|^2\bigr),$$
$$s_t^{\mathrm{NoQ}} = \tfrac{1}{2}\log\bigl(1 + \|v_t\|^2\bigr),$$
$$s_t^{\mathrm{RAW}} = \|h_t - h_{t-1}\|.$$

AAA uses the residual relative to the history basis. NoQ retains the fixed random projection but omits the history-basis subtraction. RAW measures the difference between adjacent hidden representations without either projection.

**Progress enrichment.** For seed $z$, let $\mathcal{S}_z$ denote the analyzed steps and let $y_j = 1$ indicate that step $j$ is labeled *progress*. For scoring rule $m$, let $\mathcal{T}_{z,q}^m \subseteq \mathcal{S}_z$ contain the top-$q\%$ scoring steps. We define the progress base rate and enrichment as

$$b_z = \frac{\sum_{j\in\mathcal{S}_z} y_j}{|\mathcal{S}_z|}, \qquad \mathrm{Enrich}_z^m(q) = \frac{\sum_{j\in\mathcal{T}_{z,q}^m} y_j}{|\mathcal{T}_{z,q}^m|\, b_z},$$

where $q \in \{5, 10, 20, 50\}$ and $\mathrm{Enrich}_z^m(q) = 1$ is the random-selection baseline. Values above one indicate that progress-labeled steps are more frequent in the selected set than in the full analyzed set. Steps are ranked within each seed, pooling its analyzed trajectories. We report the mean and standard deviation of the seed-level enrichment values.

**Results.** Figure 4(a) shows that progress steps have the highest mean raw AAA scores and redundant steps the lowest in both rollout sets. For the enrichment analysis in Figure 4(b), the GiGPO rollouts for 3 random seeds contain 5,005, 5,002, and 4,943 analyzed steps, including 28, 27, and 28 progress steps, respectively. The progress base rates range from $0.540\%$ to $0.566\%$.

At the top-$5\%$ cutoff, AAA achieves an enrichment of $10.62_{\pm 1.78}$, while NoQ selects no progress-labeled steps in any of the three seeds. These results indicate that high residual-based scores are associated with the goal milestones recognized by the annotation procedure.

### E.4 Role of the Success Offset $\delta$

AAA adds the success offset $\delta$ after group-relative normalization and nonnegativity clipping in Eq. (8), providing a uniform floor for successful-step advantages. We vary the offset over $\delta \in \{0, 0.05, 0.1, 0.2, 0.5, 1.0\}$ on WebShop with Qwen2.5-1.5B-Instruct and GRPO+AAA. Each setting uses one training seed (seed 0), and evaluation results are averaged over three evaluation seeds. We report both evaluation performance and the composition of the successful-step advantages logged in the same runs.

Table 6: Effect of the success offset on WebShop performance. Means with standard deviations across three evaluation seeds for each model trained with seed 0.

| $\delta$ | Score | Success rate (%) |
|---|---|---|
| 0 | $83.2_{\pm 0.9}$ | $74.0_{\pm 3.9}$ |
| 0.05 | $82.5_{\pm 4.8}$ | $70.8_{\pm 5.7}$ |
| 0.1 | $84.5_{\pm 1.6}$ | $71.6_{\pm 2.0}$ |
| 0.2 | $86.3_{\pm 2.3}$ | $65.9_{\pm 4.3}$ |
| 0.5 | $87.8_{\pm 1.7}$ | $69.5_{\pm 3.6}$ |
| 1.0 | $72.0_{\pm 2.8}$ | $56.3_{\pm 2.1}$ |

**Evaluation performance.** Table 6 shows that the two evaluation metrics favor different offset values. The highest mean success rate, $74.0\%$, is obtained with $\delta = 0$, and the highest mean score, $87.8$, is obtained with $\delta = 0.5$. The default $\delta = 0.1$ yields a score of $84.5$ and a success rate of $71.6\%$. Increasing the offset to $1.0$ reduces these values to $72.0$ and $56.3\%$. Performance therefore depends on the offset, with no monotonic gain from increasing it. We use $\delta = 0.1$ as a common default across environments.

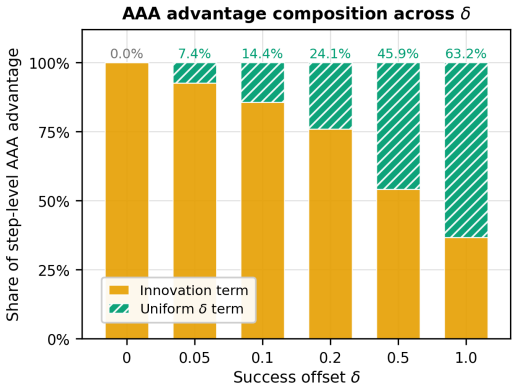



Figure 14: **AAA base step-advantage composition across $\delta$.** Bars decompose summed base step advantage on successful WebShop steps into innovation-dependent and uniform-$\delta$ components.

**Advantage composition.** For these runs, we decompose the summed AAA base advantages on logged successful steps into the clipped innovation-dependent term and the uniform $\delta$ term. Figure 14 reports the proportion of the summed advantage contributed by each component. At the fixed default $\delta = 0.1$, the uniform term accounts for only 14.4% of the summed base advantage, while the remaining 85.6% is innovation-dependent. The uniform term's share approaches parity at $\delta = 0.5$ and exceeds one half at $\delta = 1.0$. These quantities describe the composition of logged base advantages and should not be interpreted as attributions of gradient norm, policy updates, or evaluation performance.

We use $\delta = 0.1$ as a fixed default across environments, with the innovation-dependent term remaining the larger component of the summed advantage in this experiment.

### E.5 Effect of Feature-Mean Centering

We examine whether removing the coordinate mean from each representation affects AAA's performance. For a vector $y \in \mathbb{R}^d$ supplied to the reward computation, the centered variant uses

$$y_{\mathrm{zm}} = y - \bar{y}\mathbf{1}, \qquad \bar{y} = \frac{1}{d}\sum_{j=1}^{d} y_j.$$

This operation sets the mean of the vector's feature coordinates to zero. We apply it to GRPO+AAA and GiGPO+AAA on ALFWorld, keeping the remaining components fixed. The comparison uses a separate set of runs executed on different GPUs from the main experiments.

Table 7: ALFWorld success rates (%) with feature-mean centering. The original and centered settings in each row use the same experimental conditions.

| Setting | Original | w/ $y_{zm}$ |
|---|---|---|
| GRPO+AAA | $74.5_{\pm 3.6}$ | $77.1_{\pm 2.5}$ |
| GiGPO+AAA | $86.5_{\pm 3.3}$ | $82.6_{\pm 5.9}$ |

Table 7 shows a 2.6 pp. increase in mean success for GRPO+AAA and a 3.9 pp. decrease for GiGPO+AAA. The direction of the change therefore depends on the base optimizer. These results provide no consistent performance benefit from feature-mean centering, and we retain the original representations in the main experiments.

### E.6 Within-Group Advantage Dispersion in Homogeneous Outcome Groups

This section provides the experimental details and outcome-group composition for the advantage-dispersion analysis in Figure 6. We use training logs from GRPO and GRPO+AAA with Qwen2.5-1.5B-Instruct on ALFWorld, using training seed 0. Each training epoch contains 16 task-specific

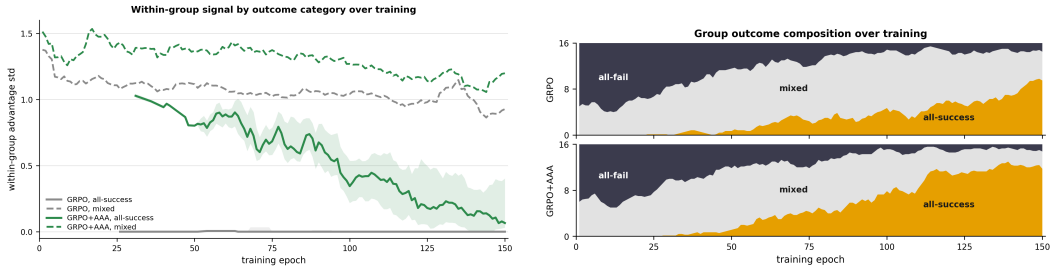



Figure 15: Advantage dispersion and outcome composition during ALFWorld training with Qwen2.5-1.5B-Instruct (training seed 0). Left: median within-group dispersion of rollout-mean advantages, as shown in Figure 6. Right: smoothed counts of the 16 task-specific groups by outcome category at each training epoch.

groups of $G = 8$ rollouts. We classify each group as all-success, mixed-outcome, or all-fail according to its terminal outcomes.

Mixed-outcome groups show substantial dispersion under both GRPO and GRPO+AAA because success and failure already induce different trajectory-level returns. In all-success groups, however, the plain-GRPO signal remains near zero throughout training, since the shared binary outcome provides almost no contrast under group-relative normalization. All-fail groups show the same median behavior. Occasional nonzero values in homogeneous GRPO groups can arise from auxiliary terms such as differing invalid-action penalties rather than from task-completion contrast.

GRPO+AAA retains nonzero dispersion in all-success groups, although the dispersion narrows later in training. The constant success offset $\delta$ cannot produce this variation because it shifts every successful step equally. The remaining dispersion instead reflects trajectory-specific normalized innovation shares, including differences in their allocation across steps and trajectory length. The right panel of Figure 15 shows that all-success groups become more common during training. The nonzero dispersion in Figure 6 therefore occurs in an increasingly common group type whose terminal outcomes alone do not distinguish its rollouts. In this regime, AAA retains a source of within-group differentiation precisely when the sparse outcome signal no longer distinguishes group members. This provides a training-time counterpart to the paired evaluation results in Figure 5, where AAA recovers more baseline failures than it loses baseline successes and yields shorter successful trajectories among jointly solved tasks.

### E.7 Sample Efficiency

We additionally compare the learning dynamics of AAA-augmented training against the corresponding base algorithms: whether AAA also improves the rate at which the policy learns from sparse terminal rewards.

We compare training rollout performance for GRPO and GiGPO with and without AAA. As shown in Figure 16, across both environments and all three metrics, GRPO+AAA, GiGPO, and GiGPO+AAA surpass GRPO from early in training. On WebShop, GRPO+AAA tracks the GiGPO methods in score and success rate and exceeds GiGPO during parts of the score curve. GRPO+AAA thus reaches comparable training performance using its representation-based advantages without anchor-state grouping. On ALFWorld, GiGPO and GiGPO+AAA reach similar training success rates. The final evaluation in Table 1 shows a 2.2 pp. success gain for GiGPO+AAA. These representative runs suggest faster improvement with AAA, particularly when it is added to GRPO.

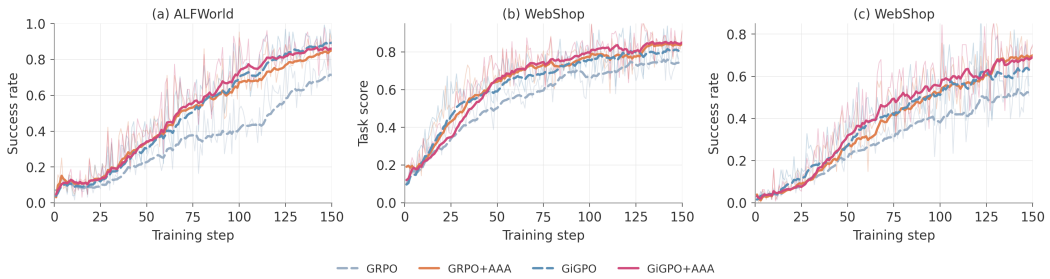



Figure 16: **Sample efficiency of AAA.** Training rollout performance on ALFWorld and WebShop. Each curve represents one Qwen2.5-1.5B-Instruct training run and uses EMA smoothing with $\alpha = 0.1$. Dashed lines denote the base methods, and solid lines denote their AAA variants.

# F Additional Evaluations and Extensions

## F.1 Comparison with HGPO

We compare AAA with HGPO (He et al., 2026) using Qwen2.5-1.5B-Instruct on ALFWorld and WebShop. HGPO is implemented as an extension of the `verl-agent` codebase used in our experiments. Its released training scripts specify 160 epochs. For fair comparison, we rerun HGPO with a 150-epoch training budget to match our AAA experiments. Table 8 reports the results at epoch 150, with means and standard deviations across three training seeds.

Table 8: HGPO and AAA results with Qwen2.5-1.5B-Instruct. Means with standard deviations across three training seeds. AAA entries are from Table 1.

| Method | ALFWorld | WebShop | |
|---|---|---|---|
| | Total success (%) | Score | Success (%) |
| GRPO+AAA | $84.7_{\pm 0.7}$ | $84.6_{\pm 0.2}$ | $71.7_{\pm 0.2}$ |
| GiGPO+AAA | $88.9_{\pm 2.0}$ | $86.6_{\pm 1.3}$ | $71.4_{\pm 1.8}$ |
| HGPO | $89.8_{\pm 1.0}$ | $87.7_{\pm 0.8}$ | $72.7_{\pm 2.2}$ |

HGPO achieves strong performance with hierarchical advantage estimation, yielding higher means than both AAA variants across the three aggregate metrics. AAA's representation-based intrinsic rewards can also be incorporated into HGPO's hierarchical advantage estimator to form HGPO+AAA. We expect this combination to further improve HGPO's performance.

## F.2 Search-Augmented Question Answering

Table 9: Search-augmented QA scores (%) with Qwen2.5-3B-Instruct. Baseline dataset scores are from Feng et al. (2025). GRPO+AAA uses the legacy batch-level aggregation described below. Avg.-6 gives equal weight to the six datasets.

| | Single-Hop QA | | | Multi-Hop QA | | | |
|---|---|---|---|---|---|---|---|
| Method | NQ | TriviaQA | PopQA | HotpotQA | 2Wiki | MuSiQue | Avg.-6 |
| R1-Instruct | 27.0 | 53.7 | 19.9 | 23.7 | 29.2 | 7.2 | 26.8 |
| Search-R1 | 34.1 | 54.5 | 37.8 | 32.4 | 31.9 | 10.3 | 33.5 |
| ZeroSearch | 41.4 | 57.4 | 44.8 | 27.4 | 30.0 | 9.8 | 35.1 |
| StepSearch | – | – | – | 34.5 | 32.0 | 17.4 | – |
| GiGPO | 42.0 | 59.5 | 42.4 | 36.9 | 37.0 | 12.6 | 38.4 |
| GRPO+AAA | 44.0 | 60.8 | 47.1 | 38.9 | 40.3 | 13.9 | 40.8 |

We examine AAA in search-augmented question answering using Qwen2.5-3B-Instruct, which extends the evaluation to a retrieval-based interaction setting and an additional model size. We train GRPO+AAA for 200 iterations in a single run and evaluate the resulting checkpoint with greedy decoding.

**Benchmarks and comparison.** We evaluate on Natural Questions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), PopQA (Mallen et al., 2023), HotpotQA (Yang et al., 2018), 2WikiMultihopQA (Ho et al., 2020), and MuSiQue (Trivedi et al., 2022). The first three datasets form the single-hop group, and the remaining three form the multi-hop group in Table 9. Baseline dataset scores are taken from Table 2 of Feng et al. (2025), and their Avg.-6 values are computed from those reported scores. Avg.-6 is the unweighted mean of the six dataset scores and is reported only for methods with complete coverage.

**Evaluation aggregation.** For GRPO+AAA, we retain the legacy validation aggregation in the released GiGPO codebase, computing each dataset score as the unweighted mean of its per-batch success rates. This can differ from question-level exact match when batches contain unequal numbers of questions from that dataset. We exclude Bamboogle from the comparison because it is identified as the most extreme case in the reported validation aggregation issue. [4]

To assess sensitivity to aggregation, we also compute question-level exact match from the same stored GRPO+AAA predictions, giving each question equal weight within its dataset. For both aggregation methods, we compute Avg.-6 from the six unrounded dataset scores. Batch-level aggregation yields an Avg.-6 of $40.84$, whereas question-level aggregation yields $40.78$; both round to $40.8$ at the precision used in Table 9. The difference computed from the unrounded averages is approximately $0.0593$ percentage points. Among the six retained datasets, the maximum absolute difference between the two aggregation methods is $1.27$ percentage points on PopQA.

**Results.** GRPO+AAA obtains an Avg.-6 of $40.8$, compared with $38.4$ computed from GiGPO's reported scores on the same six datasets. Its batch-level score exceeds the reported GiGPO value on each of the six datasets. StepSearch attains the highest MuSiQue score in the table, $17.4$, followed by GRPO+AAA at $13.9$. The results show competitive performance across both QA groups in this single-run evaluation.

### F.3 NNLS-Based Failure-Side Extension

In the base construction, failed rollouts receive zero intrinsic reward and retain their GRPO advantages (Eqs. (7) and (8)). We extend this construction by using selected residual directions from a successful rollout in the same task group as a reference for failed steps. Nonnegative least squares (NNLS) fits failed-step residuals as nonnegative combinations of these directions, and the fitted energies determine the relative strength of the negative rewards within each failed rollout.

**Post-action scoring representations.** For this extension, $T_i$ denotes the number of executed actions in rollout $\tau^i$. We associate each action $a_t^i$ with a post-action scoring representation $\widetilde{h}_t^i$ extracted from the prompt containing its returned feedback $o_t^i$ and the available interaction context. For nonterminal steps, this representation is read from the next policy prompt. After the final action, we construct a scoring prompt from the terminal feedback and obtain $\widetilde{h}_{T_i}^i$ through an additional prompt-only forward pass. This pass is used only for representation extraction and generates no further action. The rollout therefore has $T_i$ scoring representations, residuals, and advantage-allocation rows.

We compute

$$v_t^i = P\widetilde{h}_t^i, \qquad e_t^i = v_t^i - Q_{t-1}^i (Q_{t-1}^i)^\top v_t^i, \qquad t = 1, \ldots, T_i,$$

using the sequential basis-update rule of AAA.

**Success-derived reference.** Within each task-specific group, we select the shortest successful rollout, breaking ties by rollout index. From its interior steps, we select up to $K = 8$ residuals with

[4] https://github.com/langfengQ/verl-agent/issues/267. The issue concerns the released implementation and requests confirmation of the aggregation used for the paper results.

the largest raw scores, breaking score ties by step index. The first and terminal rows are excluded. All interior residuals are eligible, including those not appended to the history basis $Q$. We normalize the selected residuals to unit length and collect them as columns of $B_S$. Residuals with norm at most $10^{-12}$ contribute zero columns.

The selected directions define the convex cone

$$\mathcal{C}_S = \{B_S\alpha : \alpha \geq 0\},$$

which contains all their nonnegative linear combinations. The coefficients may rescale and combine the reference directions and need not sum to one. Here, success-derived refers to the rollout from which these directions are selected. Each failed rollout computes its residuals using its own history basis, while all residual vectors remain in the same projected space $\mathbb{R}^m$ and are fitted to this common reference.

**Failure-side credit.** For each failed rollout and step $t = 2, \ldots, T_i$, including the terminal step, we solve

$$\alpha^\star_{i,t} = \arg\min_{\alpha\geq 0} \|B_S\alpha - e^i_t\|^2, \qquad y^i_t = B_S\alpha^\star_{i,t},$$

using Lawson–Hanson NNLS. The fitted vector $y^i_t$ is the closest point in $\mathcal{C}_S$ to the failed-step residual $e^i_t$. We use its reconstruction energy to define

$$q_{i,t} = \tfrac{1}{2}\log(1 + \|y^i_t\|^2).$$

The first failure row receives $\rho_{i,1} = -\lambda_F$. For the remaining fitting rows, we standardize $q_{i,t}$ within the failed rollout and clip the standardized values to $[-1, 1]$, obtaining $z_{i,t}$. We set $z_{i,t} = 0$ when the within-rollout standard deviation is at most $\varepsilon$. The resulting rewards are

$$\rho_{i,t} = \lambda_F z_{i,t} - \lambda_F \in [-2\lambda_F, 0], \qquad t = 2, \ldots, T_i.$$

Among the fitted steps of a failed rollout, higher reconstruction energies lead to less negative rewards after within-rollout standardization, subject to the clipping limits.

Successful rollouts use the base boundary correction and score-sum normalization. We compute the shared group mean and population standard deviation from all successful and failed rewards, and set

$$g_{i,t} = \frac{\rho_{i,t} - \mu_G}{\sigma_G + \varepsilon}, \qquad \widehat{A}_{i,t} = \begin{cases} \max(0, g_{i,t}) + \delta, & \tau^i \text{ succeeds}, \\ \min(g_{i,t}, 0) - \delta, & \tau^i \text{ fails}. \end{cases}$$

For groups using the NNLS construction, successful-step advantages are at least $\delta$, and failed-step advantages are at most $-\delta$. The shared statistics also allow failure-side rewards to affect successful-step advantages. Algorithm 2 gives the full construction. The reference configuration uses $\kappa = 0.03$, $\delta = 0.1$, $\varepsilon = 10^{-6}$, and $\lambda_F = 1.5 \times 10^{-3}$.

**Evaluation and results.** We evaluate the extension on ALFWorld and WebShop with the 1.5B and 7B Qwen2.5-Instruct models. Table 10 reports means and standard deviations across three training seeds, using an evaluation average for each trained checkpoint. The GRPO+AAA reference values are taken from Table 1.

Table 10: NNLS failure-side extension. Means with standard deviations across three training seeds.

| Model | Method | ALFWorld | WebShop | |
|---|---|---|---|---|
| | | Succ. (%) | Score | Succ. (%) |
| 1.5B | GRPO+AAA | $84.7_{\pm 0.7}$ | $84.6_{\pm 0.2}$ | $71.7_{\pm 0.2}$ |
| | NNLS extension | $87.8_{\pm 1.4}$ | $85.9_{\pm 3.0}$ | $71.4_{\pm 3.4}$ |
| 7B | GRPO+AAA | $86.9_{\pm 0.5}$ | $87.2_{\pm 1.7}$ | $75.5_{\pm 2.7}$ |
| | NNLS extension | $87.4_{\pm 4.6}$ | $86.9_{\pm 0.2}$ | $75.5_{\pm 0.5}$ |

On ALFWorld, mean success increases from $84.7\%$ to $87.8\%$ for the 1.5B model and from $86.9\%$ to $87.4\%$ for the 7B model. On WebShop, the 1.5B score increases from $84.6$ to $85.9$, and success slightly decreases from $71.7\%$ to $71.4\%$. The 7B score changes from $87.2$ to $86.9$, with mean success remaining at $75.5\%$. The extension therefore yields setting-dependent changes, with the clearest mean gain on 1.5B ALFWorld. Further evaluation of the failure scale and success-reference selection could clarify how to adapt this construction across environments.

**Algorithm 2** NNLS-based failure-side advantage construction

**Require:** A task-specific rollout group $\{\tau^i\}_{i=1}^{G}$ with outcomes, policy encoder, fixed projection $P \in \mathbb{R}^{m\times d}$, fallback step advantages $\{\widehat{A}^{\text{base}}_{i,t}\}$, novelty threshold $\kappa$, advantage offset $\delta$, failure scale $\lambda_F$, reference budget $K = 8$, and numerical constant $\varepsilon$

For each rollout $\tau^i$, extract post-action representations $\{\widetilde{\mathbf{h}}^i_t\}_{t=1}^{T_i}$, including the terminal prompt-only forward pass
Compute $\mathbf{v}^i_t = P\widetilde{\mathbf{h}}^i_t$ and residuals $\mathbf{e}^i_t$ for $t = 1, \ldots, T_i$ using the thresholded Gram–Schmidt procedure in Appendix C.4, with $\kappa$ and a separate $Q^i_0 = \emptyset$ for each rollout
$s^i_t \leftarrow \frac{1}{2}\log(1 + \|\mathbf{e}^i_t\|^2), \quad i = 1, \ldots, G, \quad t = 1, \ldots, T_i$
$S \leftarrow \{i : \tau^i \text{ succeeds}\}$
**if** $S = \emptyset$ **then**
    **return** $\{\widehat{A}^{\text{base}}_{i,t}\}_{i,t}$
**end if**
$i^\star \leftarrow \arg\min_{i\in S}(T_i, i), \quad I \leftarrow \{2, \ldots, T_{i^\star} - 1\}$
**if** $I = \emptyset$ **then**
    **return** $\{\widehat{A}^{\text{base}}_{i,t}\}_{i,t}$
**end if**
Select the top $\min(K, |I|)$ scores in $I$; break ties by step index
Form $B_S$ from the selected $\mathbf{e}^{i^\star}_t / \|\mathbf{e}^{i^\star}_t\|$, using a zero column when $\|\mathbf{e}^{i^\star}_t\| \le 10^{-12}$
**for** each rollout $\tau^i$ **do**
    $\rho_{i,t} \leftarrow 0, \quad t = 1, \ldots, T_i$
    **if** $\tau^i$ succeeds and $T_i > 2$ **then**
        $m_i \leftarrow \dfrac{1}{T_i - 2}\sum_{t=2}^{T_i-1} s^i_t$
        $\bar{s}^i_1 \leftarrow m_i, \quad \bar{s}^i_{T_i} \leftarrow m_i$
        $\bar{s}^i_t \leftarrow s^i_t, \quad t = 2, \ldots, T_i - 1$
        $S_i \leftarrow \sum_{k=1}^{T_i} \bar{s}^i_k$
        **if** $S_i > \varepsilon$ **then**
            $\rho_{i,t} \leftarrow \bar{s}^i_t / S_i, \quad t = 1, \ldots, T_i$
        **end if**
    **else if** $\tau^i$ fails **then**
        $\rho_{i,1} \leftarrow -\lambda_F$
        **if** $T_i \ge 2$ **then**
            **for** $t = 2, \ldots, T_i$ **do**
                $\boldsymbol{\alpha}^\star_{i,t} \leftarrow \arg\min_{\boldsymbol{\alpha}\ge 0}\|B_S\boldsymbol{\alpha} - \mathbf{e}^i_t\|^2$
                $q_{i,t} \leftarrow \frac{1}{2}\log(1 + \|B_S\boldsymbol{\alpha}^\star_{i,t}\|^2)$
            **end for**
            $\mu_i \leftarrow \text{mean}([q_{i,t}]_{t=2}^{T_i}), \quad \sigma_i \leftarrow \text{std}_{\text{pop}}([q_{i,t}]_{t=2}^{T_i})$
            **for** $t = 2, \ldots, T_i$ **do**
                $z_{i,t} \leftarrow 0$
                **if** $\sigma_i > \varepsilon$ **then**
                    $z_{i,t} \leftarrow \text{clip}((q_{i,t} - \mu_i)/(\sigma_i + \varepsilon), -1, 1)$
                **end if**
                $\rho_{i,t} \leftarrow \lambda_F z_{i,t} - \lambda_F$
            **end for**
        **end if**
    **end if**
**end for**
$\mathcal{C} \leftarrow [\rho_{i,t} : i = 1, \ldots, G,\ t = 1, \ldots, T_i]$
$\mu_G \leftarrow \text{mean}(\mathcal{C}), \quad \sigma_G \leftarrow \text{std}_{\text{pop}}(\mathcal{C})$
**if** $\sigma_G \le \varepsilon$ **then**
    **return** $\{\widehat{A}^{\text{base}}_{i,t}\}_{i,t}$
**end if**
$g_{i,t} \leftarrow (\rho_{i,t} - \mu_G)/(\sigma_G + \varepsilon)$
$\widehat{A}_{i,t} \leftarrow \begin{cases} \max(0, g_{i,t}) + \delta, & \tau^i \text{ succeeds}, \\ \min(g_{i,t}, 0) - \delta, & \tau^i \text{ fails} \end{cases}$
**return** $\{\widehat{A}_{i,t}\}_{i,t}$