Reward Prediction with Factorized World States

Anonymous ACL submission

Abstract

Agents must infer action outcomes and select actions that maximize a reward signal indicating how close the goal is to being reached. Supervised learning of reward models could introduce biases inherent to training data, limiting generalization to novel goals and environments. In this paper, we investigate whether well-defined world state representations alone can enable accurate reward prediction across domains. To address this, we introduce STATEFACTORY, a factorized representation method that transforms unstructured observations into a hierarchical object-attribute structure using language models. This structured representation allows rewards to be estimated naturally as the semantic similarity between the current state and the goal state under hierarchical constraint. Overall, the compact representation structure induced by STATEFACTORY enables strong reward generalization capabilities. We evaluate on REWARDPREDICTION, a new benchmark dataset spanning five diverse domains and comprising 2,454 unique action-observation trajectories with step-wise ground-truth rewards. Our method shows promising zero-shot results against both VLWM-critic and LLM-as-a-Judge reward models, achieving 60% and 8% lower EPIC distance, respectively. Furthermore, this superior reward quality successfully translates into improved agent planning performance, yielding success rate gains of +21.64% on `AlfWorld` and +12.40% on `ScienceWorld` over reactive system-1 policies and enhancing system-2 agent planning.

1 Introduction

We are interested in building agents with planning capabilities that generalize across different goals and environments in zero-shot. Many real-world scenarios do not allow agents to perform massive trial-and-error to obtain explicit reward signals, making model-free approaches, *e.g.,* reinforcement learning from verifiable rewards (RLVR) (Guo et al., 2025a) not applicable. This limitation necessitates *world models* (Ha and Schmidhuber, 2018; Richens et al., 2025) that predict the consequences of actions as future world states to guide policy learning (Hafner et al., 2023) or plan search (LeCun, 2022; Hansen et al., 2023).

To leverage the state predictions of world models, agents need to evaluate how a predicted state aligns with the goal, typically via a scalar reward, and then maximize it (M. Moerland et al., 2023). Our work focuses on accurately predicting such rewards with effective *generalization*. One direct approach is to train models (Chen et al., 2025) or reward prediction heads on top of world models (Hafner et al., 2023) using task-specific supervision. However, explicit supervision may introduce biases and overfitting (Di Langosco et al., 2022; Gao et al., 2023; Casper et al., 2023; Sharma et al., 2023), limiting generalization to novel environments. A promising alternative is to estimate task progress by measuring the semantic distance between the current state and a desired goal state (Liu et al., 2022; Schaul et al., 2015), avoiding explicit reward modeling and enabling zero-shot planning (Ma et al., 2023a; Wang et al., 2023a).

There are two bottlenecks in using world-state distances as reward signals. First, constructing an abstract state representation space (M. Moerland et al., 2023) whose geometry accurately reflects task progress (Trott et al., 2019; Wu et al., 2018) is nontrivial. Prior successes leveraged representations from visual foundation models, such as DINO-WM (Zhou et al., 2024), JEPA-WMs (Terver et al., 2025), and RoboCLIP (Sontakke et al., 2023). Yet, how to extend from low-level continuous planning to high-level language-based agent planning (Wang et al., 2024a; Luketina et al., 2019; Cao et al., 2024), especially for procedural tasks (Chen et al., 2025) that require stronger semantic and temporal abstraction (Sutton, 1998), remains

an open challenge.

Second, for agents that operate in text space, rigorous evaluation of reward quality (especially in terms of fine-grained, step-wise proximity to the goal state) is difficult due to the lack of suitable benchmarks. Existing datasets mainly focus on sparse, outcome-oriented rewards, which can make it difficult to systematically assess quality of rewards and how they guide the planning process.

We address both challenges by introducing a new **state representation method** and a **dedicated benchmark** for rigorous reward evaluation. Specifically, we propose STATEFACTORY, a semantic factorization framework that factorizes world states into hierarchical object–attribute structures. Within this framework, unstructured observations are decomposed into atomic semantic factors using large language models. Reward signals are then naturally estimated by measuring semantic similarity between the current and goal states through hierarchical routing. This process effectively distills unstructured observations into compact abstractions, thereby improving zero-shot reward generalization.

To enable fine-grained evaluation of reward prediction, we introduce REWARDPREDICTION, a benchmark spanning five interactive environments: `AlfWorld` (Shridhar et al., 2020) for robotics planning, `ScienceWorld` (Wang et al., 2022) for scientific reasoning, `TextWorld` (Côté et al., 2018) for text-based games, `WebShop` (Yao et al., 2022a) for website navigation, and `BlocksWorld` (Valmeekam et al., 2023a) for classical planning. In total, there are 2,454 unique trajectories, each with step-wise action–observation pairs and scalar rewards, allowing any predicted reward to be evaluated via the EPIC distance (Gleave et al., 2020) to the ground-truth rewards.

Overall, the representation structure in STATEFACTORY demonstrates strong reward generalization capabilities. While supervised reward models exhibit a **138% increase in reward prediction error** in novel domains on REWARDPREDICTION, STATEFACTORY achieves superior zero-shot performance, reducing EPIC distance by 60% and 8% compared to VLWM-critic and LLM-as-a-Judge, respectively. These improvements in reward quality turn directly into enhanced agent planning performance, yielding success rate gains of +21.64% on AlfWorld and +12.40% on ScienceWorld over reactive system-1 policies, while consistently strengthening system-2 agent planning.

The contributions are summarized as follows:

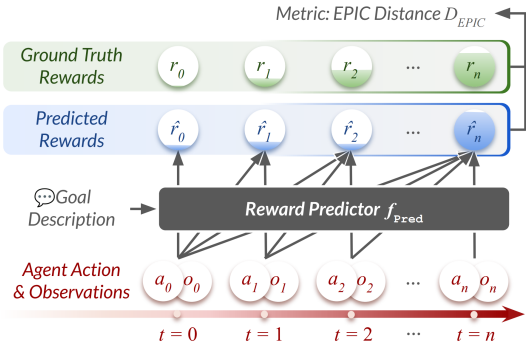

Figure 1: **REWARDPREDICTION Benchmark Protocol.** Given a textual goal description, the model computes step-wise progress estimates $[\hat{r}_t]_{t=0}^{n}$ from sequences of action-observation pairs. Reward predictions are compared against ground-truth $[r_t]_{t=0}^{n}$ using EPIC distance (Gleave et al., 2020) to quantify alignment.

- We introduce REWARDPREDICTION benchmark, covering 2,454 unique trajectories with step-wise ground truth reward across five text-based environments.
- We propose STATEFACTORY, a new representation method to transform flat text descriptions to structured hierarchies for generalizable reward prediction.
- We demonstrate that STATEFACTORY representation is able to enhance agent planning performance when combined with a world model and action proposal.

2 Related Work

Goal Conditioned Reinforcement Learning. Accurate progress measurement is critical for policy optimization (Yu et al., 2025). While early manual engineering (Lillicrap et al., 2015) and recent supervision-heavy methods (e.g., human feedback (Ouyang et al., 2022) or LLM-synthesized rewards (Xue et al., 2025)) suffer from reward hacking or poor generalization, recent unsupervised approaches extract signals from pre-trained representations via explicit reward modeling (e.g., vision-language alignment (Sontakke et al., 2023)) or implicit latent world model costs (Terver et al., 2025), enabling dense progress indications.

State Representation in World Models. Faithful world representation is pivotal for agent learning (Wang et al., 2023b). Unstructured representations (Chen et al., 2025) often retain excessive noise, while object-centric frameworks (Feng et al., 2025) may fail to capture fine-grained attribute dynamics for precise progress tracking (Wang et al.,

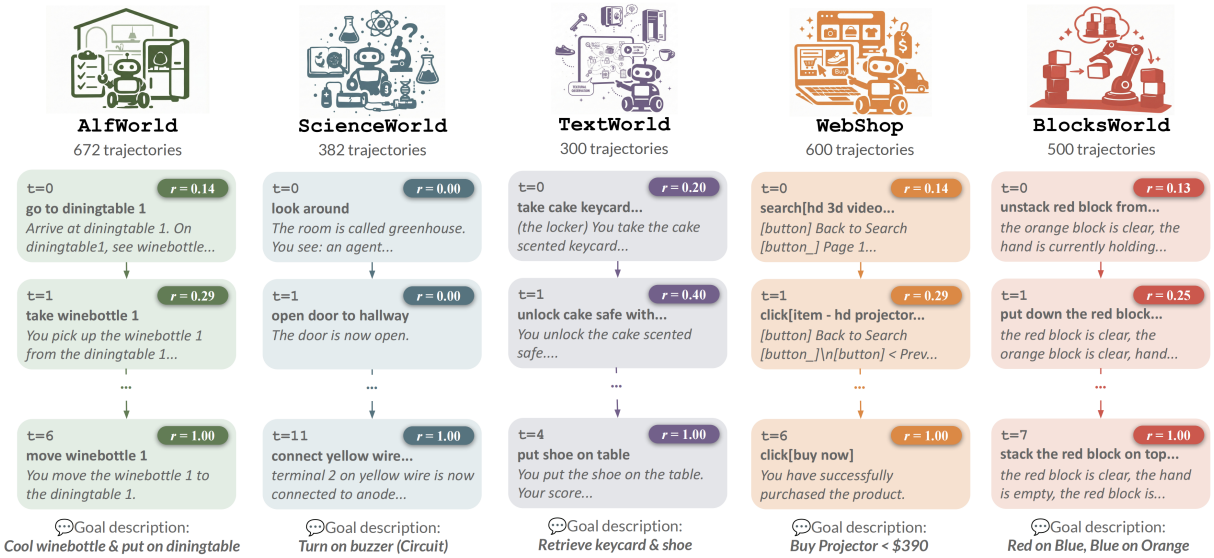

Figure 2: **REWARDPREDICTION Benchmark Examples.** Representative trajectories across five diverse domains. Each column displays a task with actions, observations, and the ground-truth task progress score ($R \in [0, 1]$).

2024b). As a remedy, recent work explores object-attribute factorization (Rozanov and Rei, 2025) for better state tracking.

Language-based Agent Learning. Sequential decision-making extends to digital domains like web navigation (Deng et al., 2023) and OS control (Zhang et al., 2025a). While model-free policies (Park et al., 2023) struggle with long-horizon consistency (Wang and Li, 2023), model-based RL uses LLMs as implicit world models to simulate transitions (Li et al., 2025) for search-based planning (e.g., MCTS). Furthermore, CoSPlan (Grover et al., 2025) and offline RL (Kumar et al., 2020) show that explicit state tracking and implicit value estimation can refine policies from sub-optimal data (Zhang et al., 2025b). Extended discussion of related works is in Appendix A.

3 The REWARDPREDICTION Benchmark

We introduce the REWARDPREDICTION benchmark to evaluate zero-shot reward prediction across different domains. In the following sections, we first present the formal task formulations (§3.1), and then describe the implementation details of benchmark construction (§3.2). Figure 1 provides an overview of the benchmark evaluation protocol, and Figure 2 presents the representative examples of different domains.

3.1 Formulation

Since we are interested in building objective-driven goal-reaching agents, we consider a Goal-Augmented Markov Decision Process (GA-MDP) (Liu et al., 2022) with Partial Observation (Kaelbling et al., 1998) constraints: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{R}, \mathcal{G}, \phi, \Omega \rangle$. Here $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{O}$ respectively denote state, action, and observation space. $\mathcal{T}$ is the underlying transition function of the environment. The function $\Omega : \mathcal{S} \rightarrow \mathcal{O}$ maps underlying states into agent observations. Each observation o_t usually partially contains the information of the ground truth states s_t. $\mathcal{R}$ denotes reward function and $\mathcal{G}$ is the goal space–we consider language goal in this work. Given a textual goal description $g \in \mathcal{G}$ and an agent that reaches s_t, the corresponding reward $\mathcal{R}(s_t, g)$ should measure how close is the agent from achieving the goal. With the mapping $\phi : \mathcal{G} \rightarrow \mathcal{S}$ that interpret goals and translate it into state space, it should have $\mathcal{R}(s_t, g) = 1$ if the goal is reached, *i.e.,* when $||s_t - \phi(g)|| \leq \epsilon$ with a small ϵ (Andrychowicz et al., 2017). To allow representing richer task progress information, we adopt the *distance-to-goal* reward (Trott et al., 2019; Liu et al., 2022):

$$\mathcal{R}(s_t, g) = 1 - ||s_t - \phi(g)||. \quad (1)$$

Prediction Task. At each time t, an agent π perceives an observation o_t, takes an action a_t, and obtains the reward r_t. We can collect offline trajectories of agent interacting with environments to achieve goals, each expressed as $\langle g, [o_t, a_t, r_t]_{t=0}^{n} \rangle$. Based on collected goals and trajectories, we define the REWARDPREDICTION task (Fig. 1), which requires a reward predictor f_{pred} to estimate ground

truth rewards based on the goal to achieve, past observations, and action history:

$$f_{\texttt{pred}}(g, o_{[0:t]}, a_{[0:t]}) \rightarrow \hat{r}_t. \quad (2)$$

Iterating over every step, we obtain a sequence of predicted rewards $[\hat{r}_t]_{t=0}^{n}$. Prediction of each step could be independent (*stateless*), but alternatively, $f_{\texttt{pred}}$ is also allowed to keep and access intermediate results from previous steps in the trajectory (*stateful*), *e.g.,* allowing them to maintain a memory of belief states for more efficient prediction.

Evaluation Metric. We adopt the Equivalent Policy-Invariant Comparison (EPIC) distance (Gleave et al., 2020; Frick et al., 2024) between ground truth rewards and predicted rewards as the metric. EPIC distance is an approach to quantify the difference between reward functions, providing a policy-invariant measure that preserves the fine-grained magnitude information essential for robust planning. Drawing on Rocamonde et al. (2023), which simplifies EPIC to a closed-form function of Pearson correlation for state-only rewards, we formulate the metric as:

$$D_{\text{EPIC}} = \frac{1}{\sqrt{2}}\sqrt{1 - \rho(\hat{R}, R)}, \quad (3)$$

where ρ represents Pearson correlation, $\hat{R} = [\hat{r}_t]_{t=0}^{n}$ are the predicted rewards and $R = [r_t]_{t=0}^{n}$ are ground truths. We report average EPIC distance across all trajectories as the final performance.

3.2 Implementation

To ensure robustness across diverse physical and semantic landscapes, we construct this benchmark by collecting *task-balanced* interaction trajectories directly from a suite of five real text-based environments. To mitigate heuristic reward hacking, we employ a paired data construction strategy. Specifically, each instance consists of a positive-negative pair tailored to the same task. The positive anchor is derived from an expert trajectory with deterministic progress gradients(Yang et al., 2024) ($r_t = t/T$) (Ziakas and Russo, 2025). Crucially, we augment these sequences with random interaction steps at the boundaries, inherit native environment rewards, to decouple trajectory length from reward signals. Complementing this, the negative baseline is generated via a random policy to represent failure trajectories ($r_t = 0$). This structural coupling forces the model to explicitly distinguish meaningful semantic progress from stochastic noise within the same task context. The final

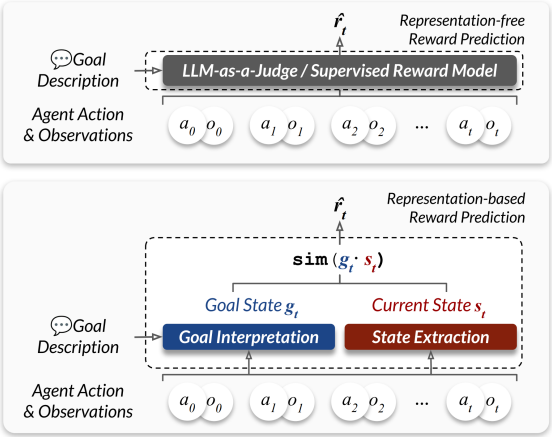

Figure 3: **Two paradigms of reward prediction.** Unlike representation-free methods that regress rewards directly from raw inputs (top), representation-based frameworks derive progress signals by measuring alignment between current state s_t and goal state g_t (bottom).

evaluation set comprises **2,454 trajectories**, as illustrated in Figure 2. For the supervised training stage, we scale up data generation to produce approximately 350,000 trajectories. Full construction details are provided in Appendix B.

`AlfWorld` (Shridhar et al., 2020). An embodied household instruction-following benchmark covering six standard task types, including *Pick & Place*, *Examine*, *Cool*, *Heat*, *Clean*, and *Movable*. We collect 672 expert trajectories.

`ScienceWorld` (Wang et al., 2022). A science-oriented text environment requiring multi-step causal reasoning and tool use across topics such as thermodynamics and life sciences. We curate 382 trajectories from the test split.

`TextWorld` (Côté et al., 2018). A procedural text-game environment involving object manipulation and lock-and-key puzzles. We use its native engine to generate 300 state-change trajectories.

`WebShop` (Yao et al., 2022a). An e-commerce navigation benchmark where agents search for products matching specified attributes. We subsample 600 successful human demonstration trajectories as positive anchors.

`BlocksWorld` (Valmeekam et al., 2023a). A classical planning benchmark for spatial reasoning. We translate symbolic PDDL states into natural language and obtain 500 trajectories.

4 Reward Prediction Methods

In this section, we explore different approaches to solving the REWARDPREDICTION benchmark. We

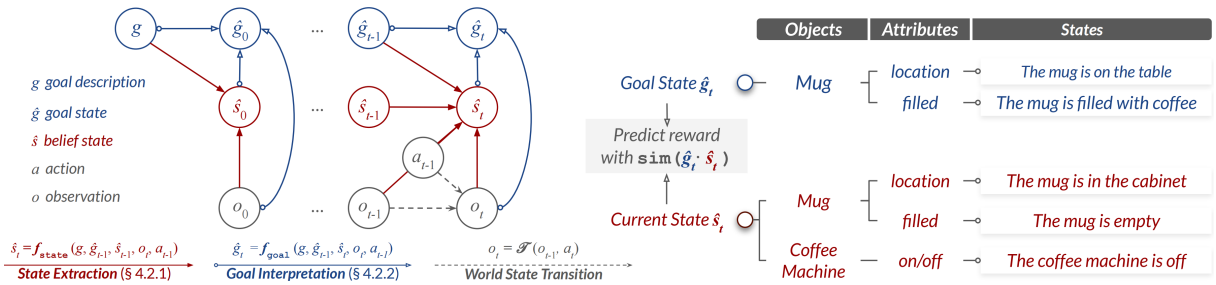

Figure 4: **StateFactory Framework.** Left: recurrent state extraction and goal interpretation. Right: Our approach factorizes states into objects and attributes, deriving rewards via semantic similarity between $\hat{s}_t$ and $\hat{g}_t$.

introduce two families of methods: *representation-free* methods and *representation-based* methods, as illustrated in Fig. 3. The key distinction lies in whether the reward predictor constructs explicit world state representations (belief states) and explicit goal interpretations (approximating ϕ). Despite architectural differences, methods can also be categorized by whether they leverage the RewardPrediction training set, resulting in two regimes: *supervised* predictors and *zero-shot* predictors.

4.1 Representation-free Methods

4.1.1 Finetuned Language Models

Inspired by reward models used in Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022) and the self-supervised critic used in the Vision Language World Model (VLWM) (Chen et al., 2025), we train language models with a linear head as $f_{\texttt{pred}}$ to predict the reward. Specifically, we generate paired samples from the training set of RewardPrediction, where the positive sample has a reward that is higher than the negative sample. We then train the language model with a contrastive ranking loss (Chen et al., 2025) with a reward centering term (Naik et al., 2024). The reward model here is fully stateless. Given input goal description and (partial) trajectory at each time step, it predicts the reward independently.

4.1.2 LLM-as-a-Judge Prompting

This approach leverages prompted LLM-as-a-Judge (Zheng et al., 2023) as generative reward predictor (Guo et al., 2025b; Gunjal et al., 2025; Whitehouse et al., 2025). We provide goal and trajectory to an LLM, with additional guidance describing the reward prediction task, to instruct it to respond with a scalar reward on a scale of 0 to 1. See Appendix Table 8 for the full prompt. In addition to this simple stateless baseline, the flexibility of prompting allows us to implement a stateful variant: the LLM is instructed to construct and manage belief state over time–a summary as memory that tracks the world state.

4.2 Representation-based Method (StateFactory)

StateFactory predicts task progress by decomposing the GA-MDP $\mathcal{M}$ into three integrated layers: *world state transition*, *state extraction* ($\hat{s}_t$), and *goal interpretation* ($\hat{g}_t$). As illustrated in Fig. 4, raw observations $o_t = \Omega(s_t)$ are distilled into structured object-attribute states $\hat{s}_t$, while the textual goal $g \in \mathcal{G}$ is iteratively grounded into a dynamic goal state $\hat{g}_t$. The reward signal is derived via semantic alignment: $\hat{r}_t = \text{sim}(\hat{g}_t, \hat{s}_t)$, where sim denotes the hierarchical matching score defined in Sec. 4.2.3. This factorization ensures zero-shot rewards remain grounded in physical transitions while staying responsive to goal description.

4.2.1 State Extraction

Unlike unstructured representations that retain task-irrelevant noise (Hao et al., 2023; Chen et al., 2025) or simple object-centric approaches that fail to capture fine-grained attribute dynamics (Yoneda et al., 2024; Feng et al., 2025), our framework explicitly separates entity identity from evolving attributes. StateFactory treats state extraction as a recurrent, goal-conditioned update process. We employ a tracking function $f_{\texttt{state}}$ to dynamically refine the semantic state $\hat{s}_t$ by integrating the raw observation o_t with both execution history and task progress:

$$\hat{s}_t = f_{\texttt{state}}(g, \hat{g}_{t-1}, \hat{s}_{t-1}, o_t, a_{t-1}), \qquad (4)$$

where a_{t-1} and $\hat{s}_{t-1}$ provide temporal context for consistency, while the goal context $(g, \hat{g}_{t-1})$ acts as an attentional prior to filter task irrelevant details. The resulting state $\hat{s}_t$ is structured as a set of N object instances $\{e_i\}_{i=1}^{N}$:

$$e_i = \left\langle \mathbf{d}_i, \{(\boldsymbol{\alpha}_{i,l}, \mathbf{v}_{i,l})\}_{l=1}^{L_i} \right\rangle, \qquad (5)$$

Table 1: **Main Results.** We compare STATEFACTORY with baselines on REWARDPREDICTION. STATEFACTORY performs well among evaluated zero-shot methods and is competitive with supervised baselines. **Bold** and underline denote the best and second-best results.

Method	Backbone	Training Data	Reasoning	Zero-shot	Reward Prediction Error ($D_{\text{EPIC}} \downarrow$)					
					AlfWorld	ScienceWorld	WebShop	BlocksWorld	TextWorld	Overall
Monotonic Baseline	–	–	–	✓	0.505	0.509	0.507	0.504	0.504	0.506
Supervised RM	Qwen2.5-1.5B	AlfWorld	N/A	✗	0.212	0.542	0.618	0.596	0.414	0.476
		ScienceWorld			0.506	0.305	0.661	0.706	0.580	0.552
		WebShop			0.707	0.624	**0.242**	0.706	0.707	0.597
		BlocksWorld			0.556	0.471	0.596	**0.472**	0.556	0.530
		TextWorld			0.523	0.658	0.678	0.604	0.203	0.533
		All Domains			**0.178**	**0.283**	0.285	0.489	**0.177**	**0.282**
VLWM-critic	Llama3.2-1B	VLWM Trajectories	N/A	✓	0.823	0.673	0.636	0.663	0.896	0.738
LLM-as-a-Judge	Qwen3-14B	–	✗	✓	0.395	0.542	0.403	0.466	0.316	0.424
			✓		0.370	0.371	0.356	0.436	0.211	0.349
	GPT-OSS-20B		Low		0.368	0.394	0.376	**0.362**	0.119	0.324
			Medium		0.366	0.391	0.374	0.363	**0.115**	0.322
STATEFACTORY	GPT-OSS-20B	–	Medium	✓	**0.285**	**0.288**	**0.286**	0.427	0.201	**0.297**

where $\mathbf{d}_i$ denotes a textual object identifier or reference description (*e.g., "Mug"*), and each pair $(\boldsymbol{\alpha}_{i,l}, \mathbf{v}_{i,l})$ denotes an attribute key and its value (*e.g., "location: on the table"*).

4.2.2 Goal Interpretation

Effective goal interpretation must bridge abstract instructions with evolving physical states. Traditional methods typically fix the goal representation at initialization (Mahmoudieh et al., 2022; Xie et al., 2023), which often creates an "illusion of progress" as they fail to adapt to environmental changes during task execution (Xue et al., 2025).

In contrast to static methods, STATEFACTORY treats goal interpretation as an iterative, state-aware process. As illustrated in the blue pathway of Figure 4, we employ a function $f_{\texttt{goal}}$ to dynamically update the goal state $\hat{g}_t$ based on the goal description $g \in \mathcal{G}$ and the current context:

$$\hat{g}_t = f_{\texttt{goal}}(g, \hat{g}_{t-1}, \hat{s}_t, o_t, a_{t-1}). \quad (6)$$

4.2.3 Hierarchical Routing

We derive reward $\hat{r}_t$ through hierarchical routing between goal state $\hat{g}_t$ and current state $\hat{s}_t$, using $\cos(\cdot,\cdot)$ for text-embedding cosine similarity.

Object Matching. For each goal object instance $e_k \in \hat{g}_t$, we identify its corresponding object instance in the current state $\hat{s}_t$. We search for the candidate e_i that maximizes the joint consistency of identity and attributes:

$$\hat{r}_{k,t} = \max_{e_i \in \hat{s}_t} \left(\cos(\mathbf{d}_k, \mathbf{d}_i) \cdot \psi_{\text{attr}}(e_k, e_i)\right) \quad (7)$$

where $\cos(\mathbf{d}_k, \mathbf{d}_i)$ measures identity similarity, for correct object selection before state evaluation.

Attribute Matching. We compute state satisfaction ψ_{attr} by averaging the similarity between each goal attribute value $\mathbf{v}_k$ and its semantically aligned counterpart $\hat{\mathbf{v}}_i$ in candidate:

$$\psi_{\text{attr}}(e_k, e_i) = \frac{1}{|A_k|} \sum_{(\boldsymbol{\alpha}_k, \mathbf{v}_k) \in A_k} \cos(\mathbf{v}_k, \hat{\mathbf{v}}_i), \quad (8)$$

where A_k is the attribute set of e_k, and $\hat{\mathbf{v}}_i$ is the value in e_i whose attribute key best matches $\boldsymbol{\alpha}_k$.

Reward Prediction. The global reward signal $\hat{r}_t$ is computed by aggregating the local fulfillment scores across all goal object instances to quantify the overall task progress:

$$\hat{r}_t = \frac{1}{|\hat{g}_t|} \sum_{e_k \in \hat{g}_t} \hat{r}_{k,t}. \quad (9)$$

5 Experiments

In this section, we evaluate STATEFACTORY across diverse planning domains to address following questions: 1) whether our zero-shot approach provides more accurate reward signals than baselines; 2) whether it generalizes better to new domains compared to supervised models; 3) how structural granularity contributes to the performance improvement; 4) how robust the framework is to different design choices (e.g., embeddings, reasoning capacity); and 5) whether these signals can effectively guide online planning to improve success rates.

5.1 Baselines and Setup

We standardize experiments on REWARDPREDICTION using gpt-oss-20b for state extraction and all-MiniLM-L6-v2 for alignment. We compare STATE-

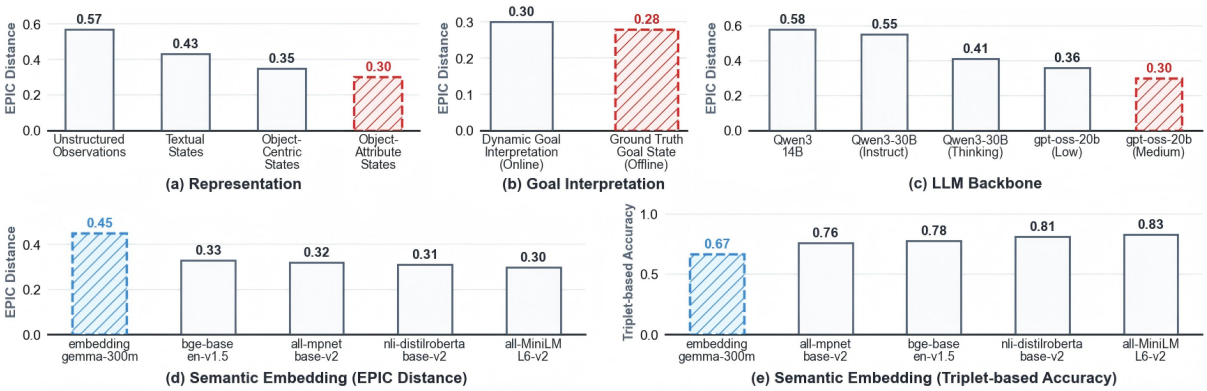

Figure 5: StateFactory ablations. Panels (a) to (d) report D_{EPIC}, and panel (e) reports Triplet-based Accuracy, the fraction of triplets where the anchor is closer to the positive sample than to the negative sample. Lower D_{EPIC} and higher Accuracy are better; red and blue hatched bars mark the best and worst configurations.

Factory against three distinct categories of baselines: (1) a naive `Monotonic Baseline`; (2) supervised approaches, including VLWM-critic (Chen et al., 2025) and other reward models explicitly trained on RewardPrediction; and (3) LLM-as-a-Judge. Full details are provided in Appx. D.

5.2 Main results on RewardPrediction

We evaluate reward prediction with EPIC distance (D_{EPIC}), where lower values indicate better alignment with ground-truth progress. Table 1 presents a comprehensive comparison of StateFactory against both supervised and representation-free baselines on RewardPrediction benchmark.

Supervised Reward Models. Table 1 reveals a severe generalization bottleneck of supervised approaches. While these models achieve exceptional precision within their training domains (e.g., 0.212 in `AlfWorld`), they exhibit a **138%** average error increase on unseen tasks. This confirms that supervised critics tend to overfit domain-specific surface patterns rather than learn a generalized notion of progress, limiting zero-shot generalization.

Representation-free Methods. Training-free baselines avoid overfitting but face challenges. Reasoning traces show that LLMs implicitly maintain belief states and extract goals at each step, improving reward prediction. However, probabilistic generation leaves these reward signals discrete and volatile. While they perform slightly better in structured domains like `TextWorld` (where tasks resemble a "checklist") and `BlocksWorld` (suitable for reverse engineering), this implicit tracking struggles in open-ended environments like `ScienceWorld`.

Representation-based Method. StateFactory addresses the dual challenges of reward generalization and signal continuity. By explicitly decomposing states into structured object-attribute representations, our method transforms reward estimation from discrete generation into continuous semantic-distance computation. The fine-grained hierarchical factorization structure not only effectively mitigates noise in flat text observations, but also enables StateFactory to capture subtle progresses that implicit methods could miss, thus yielding a robust estimator that generalizes across different domains without training. As can be observed, StateFactory achieves an average D_{EPIC} of **0.297**, not only superior to the best representation-free baseline (0.322) but also surpassing single-domain supervised models in generalization. Notably, its zero-shot performance remains close to the all-domain supervised model, with Appx. D.5 further verifying prompt robustness.

5.3 Ablation Studies of StateFactory

Importance of Abstraction. Transitioning from *unstructured observations* to extracted *textual states* yields a significant distance reduction ($0.57 \rightarrow 0.43$) in Fig. 5(a), showing that raw observations contain excessive distractors, whereas state extraction isolates task-relevant information. However, unstructured text remains susceptible to *score dilution*, where critical state changes are often overshadowed by the total sentence length.

Factorization Granularity. As shown in Fig. 5(a), while *object-centric* states reduce dilution by isolating objects (0.35), but still hit a precision ceiling from *attribute entanglement*. For example, merging properties into one embedding (e.g., "blue mug on the table") makes location changes harder to track when color acts as noise. State-

FACTORY overcomes this by refining the granularity to explicit *object-attribute* states. By decomposing observations into **minimal semantic units**, our method eliminates semantic interference and achieves the best performance (0.30).

Goal Interpretation. We compare our online goal interpretation with a post-hoc offline oracle. The oracle first processes the full expert trajectory and uses the final-step state as the ground-truth goal state, while our online setting must update the goal state incrementally during interaction. As shown in Fig. 5(b), the online setting only increases D_{EPIC} by **0.02** over the offline oracle (0.30 vs. 0.28), suggesting that our goal interpretation reliably recovers target requirements.

LLM Backbone. We investigate the impact of LLM backbone capabilities on state factorization, distinguishing inference ability (reasoning-enhanced modes) from model scale. As shown in Fig. 5(c), both dimensions correlate with alignment precision. First, enabling "thinking" modes yields significant gains: *gpt-oss-20b thinking medium* improves from 0.36 to 0.30, and *qwen3-30B-thinking* (0.41) outperforms its standard counterpart (0.55), confirming that extended inference helps decompose complex states. Second, performance generally improves with model size within the same model family, showing that reasoning ability is grounded in model size. Consequently, STATEFACTORY is positioned to scale with future advances in both model size and reasoning strategies.

Semantic Embeddings. One advantage of STATEFACTORY is that any text-embedding model can be applied on top of the hierarchical object-attribute structure to compute final rewards. Here, we investigate the correlation between the discriminative power of the embedding and the performance of STATEFACTORY. This discriminative power is measured by **triplet-based accuracy** (Dumpala et al., 2024), derived from triplets (details in Appx. D.6.3) comprising a state description (anchor), a semantic paraphrase (positive), and a contradiction (negative). The metric is defined as the fraction of triplets where the anchor is closer to the positive than to the negative. As shown in Fig. 5(d-e), higher triplet accuracy correlates with performance: discriminative models (e.g., `all-MiniLM`) yield superior alignment ($D_{\text{EPIC}} \approx 0.30$), whereas weaker embeddings like `gemma-300m` (acc$= 0.67$) fail ($D_{\text{EPIC}} = 0.45$). This indicates that STATEFACTORY relies on embedding-space discriminability.

Table 2: **Planning success rates.** STATEFACTORY improves ReAct across all tested domains.

Domain	Method	Success Rate (%)
`AlfWorld`	ReAct (Yao et al., 2022b)	34.33
	ReAct + STATEFACTORY	**55.97**
`BlocksWorld`	ReAct (Yao et al., 2022b)	85.00
	ReAct + STATEFACTORY	**93.00**
`ScienceWorld`	ReAct (Yao et al., 2022b)	22.63
	ReAct + STATEFACTORY	**35.03**

5.4 Utility for Agent Planning

We posit that the superior reward prediction capability of STATEFACTORY translates into better agent planning. To verify this, we evaluate REACT + STATEFACTORY, which integrates our reward signals into ReAct baseline. Unlike standard ReAct, which relies solely on internal reasoning, our method augments decision-making by using STATEFACTORY to predict post-action reward of each candidate and rank the top-K actions. As summarized in Table 2, this approach yields consistent gains in tested domains. The success rate in `AlfWorld` improves by 20+ percentage points ($34.33\% \rightarrow 55.97\%$), with similar gains in `BlocksWorld` and `ScienceWorld`. We find that explicit reward signals help agent break out of reasoning "deadlocks", reducing long-horizon planning ambiguity where internal reasoning alone fails.

We further integrate STATEFACTORY with LLM action proposals and a learned world model (Li et al., 2025) in MCTS, where it serves as a structured heuristic to guide search-based planning. Details are provided in Appendix D.7.4.

6 Conclusion

In this work, we demonstrate that structured text-based world state representations alone can enable accurate and generalizable reward prediction. We introduce the REWARDPREDICTION benchmark and propose STATEFACTORY, a zero-shot framework that achieves superior prediction accuracy by factorizing observations into hierarchical object-attribute structures. Our results show that STATEFACTORY exhibits clear advantages over supervised models and LLM-as-a-Judge baselines across diverse domains. Furthermore, our state-driven signals yield success rate improvements of up to 21.64% for system-1 policies , while qualitative analysis validates the utility of STATEFACTORY for system-2 agent planning by effectively guiding structured search grounded in physical evidence.

Limitations

Semantic-similarity sensitivity. Since hierarchical routing relies on cosine similarity between text embeddings, linguistic similarity may not always faithfully reflect true task progress at each intermediate step. For example, for the goal *"place the cup on the cupboard"*, an intermediate state *"cup in hand"* may be scored lower than *"cup on the desk"*, because *"desk"* and *"cupboard"* are linguistically closer in the embedding space. This can cause small local reward fluctuations in otherwise reasonable trajectories, e.g., $[0.76, 0.72]$ instead of increasing progress labels such as $[0.68, 0.85]$. **A promising direction** is to replace deterministic routing with more adaptive LLM-based scoring over structured states.

Planning-related limitations. STATEFACTORY can also be affected by vague goals and missing temporal prerequisites. For example, in an `AlfWorld` heating task, the model may miss the microwave if it is not explicitly mentioned. Moreover, the current aggregation emphasizes state fulfillment rather than strict temporal order, so a partially satisfied state may receive non-trivial reward even when key steps such as heating are missing. **A promising direction** is to combine STATEFACTORY with MCTS or ReAct-style planning, where the reward model serves as auxiliary judgment rather than the sole decision criterion, which have been preliminarily validated in Table 2.

Ethics Statement

This work studies zero-shot reward prediction for text-based agent planning. The benchmark is constructed from existing simulated environments and does not involve private user data, human-subject experiments, or personally identifiable information. The main potential risk is that improved reward estimation could be used to guide autonomous agents in unintended settings. We therefore view STATEFACTORY as a research framework for controlled planning benchmarks rather than a deployment-ready safety mechanism.

References

Minttu Alakuijala, Reginald McLean, Isaac Woungang, Nariman Farsad, Samuel Kaski, Pekka Marttinen, and Kai Yuan. 2024. Video-language critic: Transferable reward functions for language-conditioned robotics. *arXiv preprint arXiv:2405.19988*.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight experience replay. *Advances in neural information processing systems*, 30.

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, and 1 others. 2020. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20.

Yuji Cao, Huan Zhao, Yuheng Cheng, Ting Shu, Yue Chen, Guolong Liu, Gaoqi Liang, Junhua Zhao, Jinyue Yan, and Yun Li. 2024. Survey on large language model-enhanced reinforcement learning: Concept, taxonomy, and methods. *IEEE Transactions on Neural Networks and Learning Systems*.

Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, and 1 others. 2023. Open problems and fundamental limitations of reinforcement learning from human feedback. *arXiv preprint arXiv:2307.15217*.

Yevgen Chebotar, Quan Vuong, Karol Hausman, Fei Xia, Yao Lu, Alex Irpan, Aviral Kumar, Tianhe Yu, Alexander Herzog, Karl Pertsch, and 1 others. 2023. Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions. In *Conference on Robot Learning*, pages 3909–3928. PMLR.

Delong Chen, Theo Moutakanni, Willy Chung, Yejin Bang, Ziwei Ji, Allen Bolourchi, and Pascale Fung. 2025. Planning with reasoning using vision language world model. *arXiv preprint arXiv:2509.02722*.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Marc-Alexandre Côté, Akos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, and 1 others. 2018. Textworld: A learning environment for text-based games. In *Workshop on Computer Games*, pages 41–75. Springer.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114.

Lauro Langosco Di Langosco, Jack Koch, Lee D Sharkey, Jacob Pfau, and David Krueger. 2022. Goal

misgeneralization in deep reinforcement learning. In *International Conference on Machine Learning*, pages 12004–12019. PMLR.

Sri Harsha Dumpala, Aman Jaiswal, Chandramouli Shama Sastry, Evangelos Milios, Sageev Oore, and Hassan Sajjad. 2024. Sugarcrepe++ dataset: Vision-language model sensitivity to semantic and lexical alterations. *Advances in Neural Information Processing Systems*, 37:17972–18018.

Fan Feng, Phillip Lippe, and Sara Magliacane. 2025. Learning interactive world model for object-centric reinforcement learning. *arXiv preprint arXiv:2511.02225*.

Evan Frick, Tianle Li, Connor Chen, Wei-Lin Chiang, Anastasios N Angelopoulos, Jiantao Jiao, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. 2024. How to evaluate reward models for rlhf. *arXiv preprint arXiv:2410.14872*.

Leo Gao, John Schulman, and Jacob Hilton. 2023. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866. PMLR.

Adam Gleave, Michael Dennis, Shane Legg, Stuart Russell, and Jan Leike. 2020. Quantifying differences in reward functions. *arXiv preprint arXiv:2006.13900*.

Shresth Grover, Priyank Pathak, Akash Kumar, Vibhav Vineet, and Yogesh S Rawat. 2025. Cosplan: Corrective sequential planning via scene graph incremental updates. *arXiv preprint arXiv:2512.10342*.

Yu Gu, Kai Zhang, Yuting Ning, Boyuan Zheng, Boyu Gou, Tianci Xue, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, and 1 others. 2024. Is your llm secretly a world model of the internet? model-based planning for web agents. *arXiv preprint arXiv:2411.06559*.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. 2025. Rubrics as rewards: Reinforcement learning beyond verifiable domains. *arXiv preprint arXiv:2507.17746*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025a. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Jiaxin Guo, Zewen Chi, Li Dong, Qingxiu Dong, Xun Wu, Shaohan Huang, and Furu Wei. 2025b. Reward reasoning model. *arXiv preprint arXiv:2505.14674*.

David Ha and Jürgen Schmidhuber. 2018. World models. *arXiv preprint arXiv:1803.10122*, 2(3).

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. 2023. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*.

Nicklas Hansen, Hao Su, and Xiaolong Wang. 2023. Td-mpc2: Scalable, robust world models for continuous control. *arXiv preprint arXiv:2310.16828*.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173.

Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. 2020. Interactive fiction games: A colossal adventure. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7903–7910.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. Orpo: Monolithic preference optimization without reference model. *arXiv preprint arXiv:2403.07691*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, and 1 others. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134.

Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hüllermeier. 2024. A survey of reinforcement learning from human feedback.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191.

Yann LeCun. 2022. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*.

Yixia Li, Hongru Wang, Jiahao Qiu, Zhenfei Yin, Dongdong Zhang, Cheng Qian, Zeping Li, Pony Ma, Guanhua Chen, Heng Ji, and 1 others. 2025. From word to world: Can large language models be implicit text-based world models? *arXiv preprint arXiv:2512.18832*.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Minghuan Liu, Menghui Zhu, and Weinan Zhang. 2022. Goal-conditioned reinforcement learning: Problems and solutions. *arXiv preprint arXiv:2201.08299*.

Yuyang Liu, Chuan Wen, Yihang Hu, Dinesh Jayaraman, and Yang Gao. 2025. Timerewarder: Learning dense reward from passive videos via frame-wise temporal distance. *arXiv preprint arXiv:2509.26627*.

Jelena Luketina, Nantas Nardelli, Gregory Farquhar, Jakob Foerster, Jacob Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel. 2019. A survey of reinforcement learning informed by natural language. *arXiv preprint arXiv:1906.03926*.

Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker. 2023. Model-based reinforcement learning: A survey. *Foundations and Trends in Machine Learning*, 16(1):1–118.

Yecheng Jason Ma, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. 2023a. Liv: Language-image representations and rewards for robotic control. In *International Conference on Machine Learning*, pages 23301–23320. PMLR.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023b. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Parsa Mahmoudieh, Deepak Pathak, and Trevor Darrell. 2022. Zero-shot reward specification via grounded natural language. In *International Conference on Machine Learning*, pages 14743–14752. PMLR.

Abhishek Naik, Yi Wan, Manan Tomar, and Richard S Sutton. 2024. Reward centering. *arXiv preprint arXiv:2405.09999*.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, and 1 others. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22.

Jonathan Richens, Tom Everitt, and David Abel. 2025. General agents need world models. In *Forty-second International Conference on Machine Learning*.

Juan Rocamonde, Victoriano Montesinos, Elvis Nava, Ethan Perez, and David Lindner. 2023. Vision-language models are zero-shot reward models for reinforcement learning. *arXiv preprint arXiv:2310.12921*.

Nikolai Rozanov and Marek Rei. 2025. Stateact: Enhancing llm base agents via self-prompting and state-tracking. In *Proceedings of the 1st Workshop for Research on Agent Language Models (REALM 2025)*, pages 367–385.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. 2015. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320. PMLR.

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. 2018. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1134–1141. IEEE.

Pierre Sermanet, Kelvin Xu, and Sergey Levine. 2016. Unsupervised perceptual rewards for imitation learning. *arXiv preprint arXiv:1612.06699*.

Mrinank Sharma, Meg Tong, Tomasz Korbak, David Duvenaud, Amanda Askell, Samuel R Bowman, Newton Cheng, Esin Durmus, Zac Hatfield-Dodds, Scott R Johnston, and 1 others. 2023. Towards understanding sycophancy in language models. *arXiv preprint arXiv:2310.13548*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

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

Avi Singh, Larry Yang, Kristian Hartikainen, Chelsea Finn, and Sergey Levine. 2019. End-to-end robotic reinforcement learning without reward engineering. *arXiv preprint arXiv:1904.07854*.

Charlie Snell, Ilya Kostrikov, Yi Su, Mengjiao Yang, and Sergey Levine. 2022. Offline rl for natural language generation with implicit language q learning. *arXiv preprint arXiv:2206.11871*.

Vlad Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim GJ Rudner, and Yann LeCun. 2025. Learning from reward-free offline data: A case for planning with latent dynamics models. *arXiv preprint arXiv:2502.14819*.

Sumedh Sontakke, Jesse Zhang, Séb Arnold, Karl Pertsch, Erdem Bıyık, Dorsa Sadigh, Chelsea Finn, and Laurent Itti. 2023. Roboclip: One demonstration is enough to learn robot policies. *Advances in Neural Information Processing Systems*, 36:55681–55693.

Richard S Sutton. 1998. Between mdps and semi-mdps: Learning, planning, and representing knowledge at multiple temporal scales.

Basile Terver, Tsung-Yen Yang, Jean Ponce, Adrien Bardes, and Yann LeCun. 2025. What drives success in physical planning with joint-embedding predictive world models? *arXiv preprint arXiv:2512.24497*.

Alexander Trott, Stephan Zheng, Caiming Xiong, and Richard Socher. 2019. Keeping your distance: Solving sparse reward tasks using self-balancing shaped rewards. *Advances in Neural Information Processing Systems*, 32.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2023a. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *Advances in Neural Information Processing Systems*, 36:38975–38987.

Karthik Valmeekam, Sarath Sreedharan, Matthew Marquez, Alberto Olmo, and Subbarao Kambhampati. 2023b. On the planning abilities of large language models (a critical investigation with a proposed benchmark). *arXiv preprint arXiv:2302.06706*.

Danqing Wang and Lei Li. 2023. Learning from mistakes via cooperative study assistant for large language models. *arXiv preprint arXiv:2305.13829*.

Kangrui Wang, Pingyue Zhang, Zihan Wang, Yaning Gao, Linjie Li, Qineng Wang, Hanyang Chen, Chi Wan, Yiping Lu, Zhengyuan Yang, and 1 others. 2025. Vagen: Reinforcing world model reasoning for multi-turn vlm agents. *arXiv preprint arXiv:2510.16907*.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, and 1 others. 2024a. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. Scienceworld: Is your agent smarter than a 5th grader? *arXiv preprint arXiv:2203.07540*.

Ruoyao Wang, Graham Todd, Ziang Xiao, Xingdi Yuan, Marc-Alexandre Côté, Peter Clark, and Peter Jansen. 2024b. Can language models serve as text-based world simulators? *arXiv preprint arXiv:2406.06485*.

Tongzhou Wang, Antonio Torralba, Phillip Isola, and Amy Zhang. 2023a. Optimal goal-reaching reinforcement learning via quasimetric learning. In *International Conference on Machine Learning*, pages 36411–36430. PMLR.

Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, and Yitao Liang. 2023b. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. *arXiv preprint arXiv:2302.01560*.

David Warde-Farley, Tom Van de Wiele, Tejas Kulkarni, Catalin Ionescu, Steven Hansen, and Volodymyr Mnih. 2018. Unsupervised control through non-parametric discriminative rewards. *arXiv preprint arXiv:1811.11359*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Chenxi Whitehouse, Tianlu Wang, Ping Yu, Xian Li, Jason Weston, Ilia Kulikov, and Swarnadeep Saha. 2025. J1: Incentivizing thinking in llm-as-a-judge via reinforcement learning. *arXiv preprint arXiv:2505.10320*.

Lorenz Wolf, Robert Kirk, and Mirco Musolesi. 2025. Reward model overoptimisation in iterated rlhf. *arXiv preprint arXiv:2505.18126*.

Yifan Wu, George Tucker, and Ofir Nachum. 2018. The laplacian in rl: Learning representations with efficient approximations. *arXiv preprint arXiv:1810.04586*.

Teng Xiao, Yige Yuan, Mingxiao Li, Zhengyu Chen, and Vasant G Honavar. 2025. On a connection between imitation learning and rlhf. *arXiv preprint arXiv:2503.05079*.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, and 1 others. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094.

Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. 2023. Text2reward: Reward shaping with language models for reinforcement learning. *arXiv preprint arXiv:2309.11489*.

Tianci Xue, Weijian Qi, Tianneng Shi, Chan Hee Song, Boyu Gou, Dawn Song, Huan Sun, and Yu Su. 2025. An illusion of progress? assessing the current state of web agents. *arXiv preprint arXiv:2504.01382*.

Daniel Yang, Davin Tjia, Jacob Berg, Dima Damen, Pulkit Agrawal, and Abhishek Gupta. 2024. Rank2reward: Learning shaped reward functions

from passive video. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2806–2813. IEEE.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022a. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022b. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Takuma Yoneda, Jiading Fang, Peng Li, Huanyu Zhang, Tianchong Jiang, Shengjie Lin, Ben Picker, David Yunis, Hongyuan Mei, and Matthew R Walter. 2024. Statler: State-maintaining language models for embodied reasoning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15083–15091. IEEE.

Rui Yu, Shenghua Wan, Yucen Wang, Chen-Xiao Gao, Le Gan, Zongzhang Zhang, and De-Chuan Zhan. 2025. Reward models in deep reinforcement learning: A survey. *arXiv preprint arXiv:2506.15421*.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yanda Li, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. 2025a. Appagent: Multimodal agents as smartphone users. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–20.

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, and 1 others. 2025b. Agent learning via early experience. *arXiv preprint arXiv:2510.08558*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. 2024. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, and 1 others. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, and 1 others. 2023. Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory. *arXiv preprint arXiv:2305.17144*.

Mingchen Zhuge, Haozhe Liu, Francesco Faccio, Dylan R Ashley, Róbert Csordás, Anand Gopalakrishnan, Abdullah Hamdi, Hasan Abed Al Kader Hammoud, Vincent Herrmann, Kazuki Irie, and 1 others. 2023. Mindstorms in natural language-based societies of mind. *arXiv preprint arXiv:2305.17066*.

Christos Ziakas and Alessandra Russo. 2025. Test-time adaptation for generalizable task progress estimation. *arXiv preprint arXiv:2506.10085*.

Appendix

A Related Work

Goal Conditioned Reinforcement Learning. Accurately measuring task advancement is essential for policy optimization (Yu et al., 2025). Early approaches relying on manual engineering (Lillicrap et al., 2015; Singh et al., 2019; Andrychowicz et al., 2020) are labor-intensive and notoriously susceptible to reward hacking (Amodei et al., 2016; Kaufmann et al., 2024). To alleviate these issues, recent methods leverage human feedback, including demonstrations and preferences (Christiano et al., 2017; Ouyang et al., 2022; Hong et al., 2024; Xiao et al., 2025), or LLM-synthesized rewards (Ma et al., 2023b; Xie et al., 2023; Wang et al., 2025; Xue et al., 2025). However, these methods struggle to generalize to novel domains due to their dependence on task-specific supervision (Wolf et al., 2025). Consequently, the field has shifted towards unsupervised paradigms that extract dense signals directly from pre-trained representations. These approaches can be broadly categorized into two streams based on how they utilize the feature space. The first stream focuses on explicit reward modeling, where scalar signals are derived by quantifying task progress or goal probability. This includes methods such as vision-language alignment (Rocamonde et al., 2023; Sontakke et al., 2023; Alakuijala et al., 2024), discriminative classification likelihood (Sermanet et al., 2016; Warde-Farley et al., 2018) or temporal distances in passive videos (Sermanet et al., 2018; Liu et al., 2025). The second stream employs latent world models for implicit reward prediction (Zhou et al., 2024; Terver et al., 2025; Sobal et al., 2025). Instead of relying on explicit scorers, these methods derive signals by formulating the latent distance to the goal as an intrinsic cost, effectively transforming predictive dynamics into a dense progress indicator for planning.

State Representation in World Models. The performance of language-based agent learning heavily relies on the fidelity of world representation (Valmeekam et al., 2023b; Wang et al., 2023b). Existing approaches address this challenge with varying degrees of structural assumption. Initial works employ *unstructured representations*, typically maintaining the state as raw observations (Yao et al., 2022b; Hao et al., 2023). However, this lack of filtering retains excessive task-irrelevant noise. This information overload obscures critical state changes, making it difficult to track the status of objects consistently. (Valmeekam et al., 2023a). To better organize these observations, recent *object-centric frameworks* decompose environments into discrete entities (Feng et al., 2025; Yoneda et al., 2024). Yet, these approaches typically treat objects as simple units, failing to separate the object's identity from its changing attributes. Consequently, it becomes difficult to track specific changes (e.g., the rising temperature of an ingredient). This lack of detail hinders the measurement of precise progress required for effective learning (Wang et al., 2024b). To capture these fine-grained dynamics, recent studies have explored object-attribute representations, which factorize entities into structured attribute-value pairs to support more accurate state tracking (Zhu et al., 2023; Rozanov and Rei, 2025).

Language-based Agent Learning. Similar to physical control, the scope of sequential decision-making extends to digital environments. Here, interactions such as web navigation (Zhou et al., 2023; Deng et al., 2023), operating system control (Zhang et al., 2025a; Xie et al., 2024) and text-based games (Côté et al., 2018; Hausknecht et al., 2020) are formulated as control problems governed by comparable dynamics. Traditionally, agents operate as *model-free policies*, mapping observations to actions via in-context reasoning (Wei et al., 2022; Yao et al., 2022b; Shinn et al., 2023; Park et al., 2023), yet often struggle with long-horizon consistency and generalization (Wang et al., 2024a; Madaan et al., 2023; Wang and Li, 2023). To address this, recent work adopts the *Model-based RL framework*, validating LLMs as implicit world models that simulate text or code transitions (Zhuge et al., 2023; Li et al., 2025). This predictive capability enables search-based planning (e.g., MCTS) in web agents, outperforming reactive prompting by simulating future states before execution (Gu et al., 2024; Hao et al., 2023; Yao et al., 2023). Furthermore, CoSPlan (Grover et al., 2025) demonstrates that incorporating incremental scene graph updates allows agents to explicitly track state transitions, enabling them to detect and rectify errors in complex visual sequences. Concurrently, to mitigate exploration bottlenecks, the paradigm is shifting to *offline reinforcement learning* (Kumar et al., 2020; Levine et al., 2020). Recent Approaches (Zhang et al., 2025b; Nakano et al., 2021) refine policies directly from sub-optimal offline data without extensive online interaction. This leverages implicit value frameworks (Snell et al., 2022) to distill op-

Table 3: Train and evaluation splits of REWARDPREDICTION. We report positive–negative task pairs. Each pair contains one positive and one negative trajectory for the same task.

Domain	Eval Pairs	Train Pairs	Expert Source	Disjointness Control
AlfWorld	336	2,795	Official task solutions / planner	Held-out task split, balanced over task types
ScienceWorld	191	1,094	Official environment trajectories	Official test/train split
WebShop	300	374	Official successful human demonstrations	Disjoint human demonstration subset
TextWorld	150	1,500	Procedurally generated expert trajectories	Disjoint fixed random seeds
BlocksWorld	250	2,100	Planner/PDDL-derived trajectories	Official/generator split with fixed seeds

timal behaviors from mediocre traces, a principle now extended to self-generated agent experiences (Chebotar et al., 2023).

B Benchmark Details and Data Construction

B.1 Data Generation and Progress Labeling

We construct REWARDPREDICTION to evaluate step-wise goal-progress estimation from action–observation histories. The labels should be interpreted as *trajectory-derived progress labels*, rather than direct environment reward functions. The monotonic interpolation is applied only to the core expert segment, while random boundary actions and negative trajectories are included to reduce shortcuts based only on trajectory length.

Progress Label Derivation via Hybrid Labeling. (i) **Expert Segment Densification:** Following prior work on trajectory-derived progress signals (Yang et al., 2024; Ziakas and Russo, 2025), we assign dense progress labels to the core expert segment via linear interpolation, formulated as $r_t = t/T_{\text{expert}}$. This provides a consistent step-wise signal for evaluating whether predicted rewards follow expert-derived task progress. (ii) **Random Step Grounding:** The appended random steps inherit native environment feedback when available. This makes trajectory length alone insufficient, since random actions before or after the expert segment may not correspond to meaningful progress.

Negative Sample Generation & Filtering. Negative trajectories are generated via a random policy for the same task. We filter out random trajectories that overlap with, or match sub-segments of, expert demonstrations to reduce accidental successes. These trajectories provide a non-progress baseline and help test whether predictors distinguish semantic progress from interaction noise.

Native Observations, Actions, and Feedback. We keep observations and admissible actions from the original environments without manual rewriting. When native reward or score feedback is available, it is used for random boundary steps and filtering. This preserves the structure of the source environments while keeping the benchmark focused on progress prediction.

B.2 Train & Evaluation Splits

Table 3 summarizes the train and evaluation split used in REWARDPREDICTION. We report positive–negative task pairs unless otherwise specified. Each evaluation pair contains one positive trajectory derived from an expert or high-quality demonstration and one negative trajectory generated by random interactions for the same task. Thus, the evaluation split contains 1,227 task pairs, corresponding to 2,454 trajectories in total. Training and evaluation instances are constructed from non-overlapping official splits when available, or from disjoint fixed random seeds for procedurally generated environments.

B.3 Evaluation Task Environments

To evaluate our method across a diverse set of challenges, ranging from household navigation to scientific reasoning, we incorporated five distinct interactive environments:

- **AlfWorld:** AlfWorld is an embodied text-based environment covering tasks like cleaning, heating, and cooling objects, challenging agents to ground high-level language instructions into sequential navigation and manipulation actions within simulated domestic scenes.
- **ScienceWorld:** ScienceWorld simulates scientific experiments across interconnected rooms, demanding a deep understanding of physics, scientific causality, and multi-step logical reasoning to interact with diverse objects.
- **WebShop:** WebShop is a massive simulated e-commerce environment built on real-world Amazon data, testing the agent's ability

Table 4: **Positive `AlfWorld` sample.** Task: *"put a hot potato in diningtable."* The expert segment (t = 0-5) receives interpolated dense rewards, while the final random augmentation ($t = 6$) inherits the native success signal.

Step	Action (a_t)	Observation (o_t)	GT (R_t)
0	go to countertop 3	You arrive at countertop 3. On the countertop 3, you see a bread 1... a potato 1...	0.17
1	take potato 1...	You pick up the potato 1 from the countertop 3.	0.33
2	go to microwave 1	You arrive at microwave 1. The microwave 1 is closed.	0.50
3	heat potato 1...	You heat the potato 1 using the microwave 1.	0.67
4	go to diningtable 1	You arrive at diningtable 1. On the diningtable 1, you see a bowl 3...	0.83
5	move potato 1...	You move the potato 1 to the diningtable 1.	1.00
6	**go to toaster 1**	**You arrive at toaster 1. On the toaster 1, you see nothing.**	**1.00**

Table 5: **Negative `AlfWorld` sample.** Task: *"put a hot potato in diningtable."* A random trajectory fails to advance the task state and receives zero reward.

Step	Action (a_t)	Observation (o_t)	GT (R_t)
0	go to stoveburner 1	You arrive at stoveburner 1. On the stoveburner 1, you see a pot 1.	0.00
1	go to cabinet 7	You arrive at cabinet 7. The cabinet 7 is closed.	0.00
2	go to drawer 6	You arrive at drawer 6. The drawer 6 is closed.	0.00
3	open drawer 6	You open the drawer 6. The drawer 6 is open. In it, you see nothing.	0.00
4	go to cabinet 8	You arrive at cabinet 8. The cabinet 8 is closed.	0.00
5	open cabinet 8	You open the cabinet 8. The cabinet 8 is open. In it, you see nothing.	0.00
6	go to toaster 1	You arrive at toaster 1. On the toaster 1, you see nothing.	0.00

to process noisy web text, match complex multi-attribute user intents, and perform long-horizon navigation to successfully complete purchases.

- **`TextWorld`:** `TextWorld` is a procedural text-adventure framework with partially observable environments, requiring agents to utilize memory and map abstract textual clues to sequential puzzle-solving strategies.
- **`BlocksWorld`:** `BlocksWorld` is a fundamental planning environment where the agent must logically unstack and restack colored blocks on a table to reach a specific target configuration, testing basic spatial and sequential reasoning under strict physical constraints.

B.4 Domain-Specific Sourcing and Preprocessing

We tailored the sampling and filtering strategy for each domain to ensure dataset quality and balance:

- **`AlfWorld` (Uniform Sampling):** We randomly sampled **336** tasks from the training set, representing approximately 10% of the total data. To prevent task bias, we ensured a uniform distribution across the six distinct task types, resulting in exactly 56 positive-negative trajectory pairs per task type.
- **`ScienceWorld` (Validity Filtering):** We selected **191** trajectories from the test set. During selection, we explicitly filtered out specific tasks (e.g., *task-4-grow-fruit* and *task-4-grow-plant*) where the native oracle could not reliably reach a task progress of 1.0 due to environmental limitations. Additionally, we clipped terminal failure scores (e.g., -100) to 0 to maintain reward boundedness.
- **`WebShop` (Human Demonstrations):** We leveraged human demonstrations as the expert policy. From a total pool of 711 successful human trajectories, we randomly subsampled **300** high-quality instances to serve as positive samples, ensuring a diverse coverage of e-commerce navigation behaviors.
- **`TextWorld` (Procedural Generation):** We utilized the native engine to procedurally generate **150** unique text-adventure instances, focusing on cooking and treasure-hunting quests that require multi-step reasoning.
- **`BlocksWorld` (Subsampling & PDDL):** Sourced from the `BlocksWorld` suite, we randomly subsampled **250** instances from the official 500-instance dataset. We utilized the authorized PDDL-to-text translator to convert symbolic states (e.g., `on(a, b)`) into natural language descriptions, ensuring consistency with other environments.

B.5 Data Sample Visualization

To provide a granular view of our hybrid data construction and labeling strategy, we transform raw environment interactions into structured, timestep-wise trajectory tables. **Table 4 and Table 5** visualize a representative positive-negative trajectory pair from the `AlfWorld` domain, illustrating how our pipeline converts sparse environmental feedback into a dense, semantically-rich supervision signal.

Case Analysis: Positive Trajectory. Table 4 visualizes a positive trajectory consisting of agent actions (a_t), environment observations (o_t), and derived ground-truth rewards (R_t^*). This sample exemplifies our hybrid supervision approach:

- **Expert Segments ($t = 0$ to 5):** The core sequence follows an optimal expert policy. As per our methodology, these steps are assigned **monotonically increasing dense rewards** (from 0.17 to 1.00) via linear interpolation, providing a granular gradient that captures task progress.
- **Random Augmentation ($t = 6$):** The final step is an appended random action (`random_post`). Crucially, this step strictly inherits the native environment success signal ($R = 1.0$), ensuring the model learns to maintain the success state even amidst non-expert exploration.

Case Analysis: Negative Trajectory. In contrast, **Table 5** displays a negative trajectory generated via a stochastic random policy. Despite the agent's active interaction with the environment (e.g., opening drawers), the actions fail to advance the task state toward the goal. Following our *Strict Filtering Protocol*, these steps are uniformly assigned a **flat ground truth of** 0, establishing a robust non-progressing baseline that represents a clean failure case.

C Hierarchical Routing

As described in Section 4.2.3, the dense reward $\hat{r}_t$ is derived through hierarchical routing between the goal state $\hat{g}_t$ and the current state $\hat{s}_t$. We use $\cos(\cdot, \cdot)$ to denote cosine similarity between text embeddings. The routing process searches for the candidate object that maximizes a composite object–attribute score.

Semantic Key Alignment (Inner Loop). To address potential linguistic discrepancies between goal descriptions and environment observations, we do not assume fixed key correspondences. For a required attribute pair $(\boldsymbol{\alpha}_k, \mathbf{v}_k) \in A_k$ of goal object e_k, and a candidate object e_i, we dynamically identify the most relevant attribute key in e_i via hard-max alignment:

$$l^* = \arg\max_{l} \cos(\boldsymbol{\alpha}_k, \boldsymbol{\alpha}_{i,l}), \tag{10}$$

and define the aligned value as $\hat{\mathbf{v}}_i = \mathbf{v}_{i,l^*}$. This allows attributes with different names but similar meanings to be compared consistently.

Attribute Satisfaction Scoring. With the keys aligned, we compute the attribute satisfaction $\psi_{\text{attr}}(e_k, e_i)$ by averaging the cosine similarity between goal values and their aligned state values:

$$\psi_{\text{attr}}(e_k, e_i) = \frac{1}{|A_k|} \sum_{(\boldsymbol{\alpha}_k, \mathbf{v}_k) \in A_k} \cos(\mathbf{v}_k, \hat{\mathbf{v}}_i). \tag{11}$$

Here, A_k denotes the attribute set of goal object e_k, and $\hat{\mathbf{v}}_i$ is selected separately for each goal attribute key $\boldsymbol{\alpha}_k$.

Joint Object Selection. To determine the local fulfillment score $\hat{r}_{k,t}$, we evaluate every candidate object e_i in the current state $\hat{s}_t$. We define a composite alignment score $\Phi(e_k, e_i)$ that couples identity matching with attribute satisfaction:

$$\Phi(e_k, e_i) = \underbrace{\cos(\mathbf{d}_k, \mathbf{d}_i)}_{\text{Identity}} \cdot \underbrace{\psi_{\text{attr}}(e_k, e_i)}_{\text{Attribute}}. \tag{12}$$

This multiplicative form downweights candidates with either mismatched identities or unsatisfied attributes. The system selects the candidate that maximizes this joint score:

$$\hat{r}_{k,t} = \max_{e_i \in \hat{s}_t} \Phi(e_k, e_i). \tag{13}$$

Global Aggregation. Finally, the total reward $\hat{r}_t$ is the mean of the maximized local fulfillment scores across all goal objects in $\hat{g}_t$, as defined in Eq. 9.

D Experimental Implementation Details

D.1 Monotonic Baseline (Rule-based Heuristic)

The *Monotonic Baseline* serves as a non-parametric, rule-based reference to evaluate the necessity of semantic reward modeling. In this configuration, we bypass environmental feedback and instead apply a naive temporal heuristic to both positive and negative trajectories. Specifically, for any trajectory of length T, the reward at each time

Table 6: **Task-specific training details.** Dataset scale, compute resources, and training duration across tasks. Global batch size (BS) scales with GPU count.

Task	Data Config		Compute Setup		Cost
	Size	Ep.	GPUs (Mem)	BS	Hours
AlfWorld	35k	3	2×A6000 (80G)	16	∼4h
TextWorld	35k	3	2×A6000 (48G)	4	∼4.5h
ScienceWorld	35k	3	2×A100 (80G)	16	∼6h
BlocksWorld	35k	3	2×A100 (80G)	32	∼7h
WebShop	35k	2	2×A100 (80G)	16	∼10h
Combined	**175k**	**2**	**4×A100 (80G)**	**32**	**∼20h**

step t is assigned via linear interpolation: $r_t = t/T$. By enforcing this strictly increasing progress signal regardless of the actual state-action semantics, this baseline tests whether a simple "time-equals-progress" assumption can suffice for the task. It provides a lower bound for our evaluation, highlighting the limitations of ignoring the distinction between expert goal-oriented actions and aimless exploration.

D.2 Supervised Reward Model

In this section, we detail the dataset curation, model architecture, and training protocols employed for the reward model (RM) baselines.

D.2.1 Dataset Construction and Input Format

We curated preference datasets across five distinct interactive environments: `AlfWorld`, `TextWorld`, `ScienceWorld`, `Blocks World`, and `WebShop`. We collected 35,000 preference pairs for each environment, yielding a cumulative dataset of 175,000 pairs.

Each sample comprises a raw text trajectory containing a high-level goal, a sequence of actions, and corresponding state observations. Deviating from standard instruction-tuning formats, we eschew separate system prompts; instead, the input is fed as a flattened interaction history. The input format is structured as follows:

```
<GOAL> [Task Description] </GOAL>
<ACTION> [Action_1] </ACTION>
<STATE> [Observation_1] </STATE> . . .
```

We trained six distinct reward models: five task-specific models (35k pairs each) and one comprehensive model ("All-Combined") trained on the union of all datasets (175k pairs).

D.2.2 Model Architecture and LoRA Configuration

We employ **Qwen2.5-1.5B** (Hui et al., 2024) as the backbone architecture. To manage computational costs efficiently and mitigate potential catastrophic forgetting, we utilize **Low-Rank Adaptation (LoRA)** (Hu et al., 2022) in lieu of full-parameter fine-tuning.

Table 7: **Common hyperparameters.** Settings fixed across all experiments.

Parameter	Value
Base Model	`Qwen2.5-1.5B`
Method	LoRA (`SEQ_CLS`)
Precision	`BF16`
Max Length	8192
LoRA Configuration	
Rank (r) / Alpha (α)	16 / 32
Target Modules	All Linear Layers
Optimization	
Optimizer	`AdamW`
Learning Rate	1×10^{-5}
LR Scheduler	Cosine
Batch Size	16

The reward model is implemented by appending a scalar classification head to the base model. LoRA adapters are applied to all linear layers with a rank of $r = 16$ and an alpha scaling factor of $\alpha = 32$. The training objective minimizes the negative log-likelihood of the preferred trajectory rankings:

$$\mathcal{L}(\theta) = -\mathbb{E}_{(y_w, y_l)\sim\mathcal{D}}\big[\log \sigma(r_\theta(y_w) - r_\theta(y_l))\big]. \quad (14)$$

where y_w and y_l denote the chosen and rejected trajectories, respectively.

D.2.3 Training Setup and Hyperparameters

All models were implemented using the Hugging Face `trl` library. To accommodate the extensive context requirements of agent interaction histories, we set the maximum sequence length to **8192** tokens and utilized mixed-precision training (BF16) with gradient checkpointing to optimize memory efficiency.

As detailed in **Table 6**, the hardware infrastructure varied due to resource availability, ranging

Table 8: **Detailed Baseline Prompt Templates.** We follow the same **Full-History Protocol** for both baseline variants. The CoT variant uses a standard helper system prompt, while the Zero-Shot variant injects a **[Without Thinking]** constraint to suppress reasoning.

Detailed Prompt Templates for Baseline Implementation

Part I: Chain-of-Thought (CoT) Variant	Part II: Zero-Shot / No-Thinking Variant
System Prompt You are a helpful assistant. Please output the final answer in JSON format.	**System Prompt** You are a helpful assistant. Please output the final answer in JSON format. **[Without Thinking]**
User Prompt You are an expert in task progress evaluation. Please judge the agent's progress (0–1) based on the full interaction history.	**User Prompt** You are an expert in task progress evaluation. Judge the agent's progress (0–1) based on the full interaction history provided below.
Input Fields **[Task Description]** `{task_description}` **[Interaction History]** `{history_log}`	**Input Fields** **[Task Description]** `{task_description}` **[Full Interaction History]** `{history_log}`
Evaluation Requirements 1. Review the full history strictly. 2. Evaluate task progress. 3. Provide a scalar score in $[0, 1]$.	**Evaluation Requirements** • Determine progress from the final state. • **Do NOT analyze, reason, or explain.** • Output the score immediately.
Output Format `{"reward": 0.5}`	**Output Format** `{"reward": 0.5}`

from dual NVIDIA RTX A6000 configurations to clusters of NVIDIA A100 GPUs. Consequently, while the effective batch size and training duration varied by task, the core optimization hyperparameters (**Table 7**) were kept constant to ensure fair comparison. Models were generally trained for 3 epochs, with the exception of `WebShop` and the `Combined` model, which were trained for 2 epochs.

Table 7 outlines the shared configuration, while **Table 6** provides the specific hardware setup, batch sizes, and training duration for each task.

D.3 LLM-as-a-Judge

To ensure the **maximal fidelity and empirical rigor** of the evaluation, we implement a direct trajectory evaluation protocol where the *Reward Evaluator* is provided with the full, uncompressed interaction history. All evaluating models are configured with a **context window of 8,192 tokens** to accommodate long-horizon trajectories without information loss. Furthermore, we set the **sampling temperature to** 0.01 across all variants to enforce near-deterministic outputs, thereby ensuring the high stability and reproducibility of our reward signals. To systematically investigate the impact of reasoning on reward judgment, we employ two distinct control mechanisms:

1. Prompt-Level Reasoning Ablation (e.g., Qwen-2.5-14B).

For general-purpose LLMs, we modulate reasoning capabilities through explicit system-level instructions, allowing us to isolate the effects of internal Chain-of-Thought (CoT) processes. **Table 8** provides the detailed prompt templates used for these variants:

- **Intuitive Mode (No-Thinking):** The system prompt imposes strict negative constraints, forcing the model to provide an immediate judgment based on "first-order intuition" without intermediate analysis (e.g., *"Directly output the score. Do NOT analyze, reason, or explain"*).
- **Analytical Mode (CoT):** All negative constraints are removed, and the model is explicitly instructed to perform a comprehensive review of the trajectory history and justify its scoring logic through step-by-step reasoning before arriving at a final value.

2. Test-Time Compute Scaling (Reasoning Effort Control). For models with native reasoning-specialized architectures (e.g., GPT-OSS-20B), we modulate the depth of internal verification and search processes by adjusting the `reasoning_effort` parameter. This mechanism allows for the scaling of the **test-time compute budget** independently of the input prompt structure. In our experiments, we compare a *Baseline* (Low reasoning effort) against an *Enhanced* (Medium

Table 9: Prompt rephrasing robustness of STATEFACTORY. We report average EPIC distance and the change relative to the original prompt. Lower is better.

Prompt Variant	Avg. D_{EPIC}	Δ vs. Original
Original	0.2938	+0.0000
Reordered	0.3160	+0.0222
Plain	0.2935	-0.0003
Concise	0.2974	+0.0036
Soft-constraints	0.2906	-0.0032

Table 10: **Rule-violation audit for STATEFACTORY.** We report first-run violations, rescue rates for repaired cases, and persistent errors remaining after repair.

Rule	Steps	First Error	Rescue Rate	Persist. Error
Goal vs. Plan	25,042	0.54%	96.77%	0.02%
Immutability	22,588	1.41%	87.12%	0.18%
Source Grounding	25,042	1.38%	90.94%	0.12%
Goal Leakage	25,042	0.33%	93.67%	0.02%

reasoning effort) configuration. Notably, as these models generate reasoning within a native "reasoning content" accessible via metadata, we can evaluate the logical derivation without requiring explicit "analysis" fields in the final text response. This allows us to maintain a clean, standardized output format—consisting solely of the reward score—across all models while still capturing the full depth of latent reasoning. By systematically varying the "reasoning effort" parameter, we can observe how increased latent search steps and self-correction cycles contribute to a more precise grounding of the reward signal, effectively decoupling the model's evaluative accuracy from its prompt-based instructions.

D.4 STATEFACTORY

Hyperparameters. To ensure experimental reproducibility and minimize stochastic variance, we utilize **gpt-oss-20b** as the primary reasoning backbone for all evaluative tasks. The model is deployed via the vLLM inference engine, with the sampling temperature fixed at a near-deterministic $T = 0.01$. We maintain a consistent context window of 8,192 tokens.

Embedding Model. Semantic similarity measures within STATEFACTORY are computed using **all-MiniLM-L6-v2**. This specific encoder was selected based on our systematic ablation study (see Figure 5), where it demonstrated superior alignment fidelity for environment-specific state representations. Despite its compact 384-dimensional vector space, it outperformed larger alternatives in capturing the underlying semantic distance.

D.4.1 STATEFACTORY Inference Pipeline

Our proposed method transforms raw observations into structured states via a multi-stage pipeline. **Table 11** details the **Semantic State Factorization** process (Modules A & B). **Table 12** details the **Dynamic Goal Interpretation** (Module C).

D.5 Prompt Robustness & Violation Audit

Prompt rephrasing robustness. We first evaluate whether STATEFACTORY is sensitive to superficial prompt phrasing. Starting from the original prompt, we construct four semantics-preserving variants: *plain* natural phrasing, *reordered* instructions, a *concise* version with substantially reduced length, and a *soft-constraints* version that weakens strict imperative expressions. We evaluate each variant on 30 tasks per environment. As shown in **Table 9**, the performance differences remain small, suggesting that the method is not overly sensitive to prompt wording.

Rule-violation audit. Prompt rephrasing alone does not verify whether the LLM actually follows the required extraction rules. We therefore conduct a systematic rule-violation audit over the full REWARDPREDICTION benchmark. We focus on the most critical rules in state extraction and goal interpretation, excluding JSON formatting errors because the output schema is enforced by the structured output format.

For state extraction, we check *Source Grounding* (whether extracted states are grounded in previous states or current observations) and *Goal Leakage* (whether current states are copied directly from the goal state). For goal interpretation, we check *Goal vs. Plan* (whether the goal state mistakenly contains intermediate plans rather than final conditions) and *Immutability* (whether goal updates introduce new goals rather than refining the initial blueprint).

We use a separate LLM evaluator for the initial audit. To verify reliability, all flagged violation cases are manually reviewed, and an additional 5% of non-violation cases are randomly sampled for human verification. The human verification agrees with the LLM evaluator on the checked cases. As shown in **Table 10**, first-run violation rates are low, and most violations can be recovered by a second-run repair step, resulting in a low persistent error.

Table 11: **Semantic State Factorization Templates.** This pipeline constitutes the first half of the STATEFACTORY method. Module A grounds raw observations into structured states, while Module B filters and refines these states to maintain a coherent, task-relevant history.

STATEFACTORY Pipeline Part I: Semantic State Factorization

MODULE A: World State Extraction Template **(Perception & Grounding)**

System Instruction: `{system_instruction}`

Role: You are a state extraction assistant. Your goal is to extract the current state of the agent based on the provided information, following the rules below:

Rules:

1. **Output Format:** Output a SINGLE JSON object. It must contain two keys: "thinking" and "current_state".
   - "thinking": "For each object, cite the exact phrase from 'Observation' that justifies its state. Verify that the object actually exists in 'Observation' or 'Previous States'. EXPLICITLY REJECT any information that appears ONLY in 'Previous Goal State'."
   - "current_state": `{output_format_des}`
2. **Relevance:** Only extract objects that are present in the Observation, or Previous States, and that are directly relevant to the Task Description. Include only those objects that actively contribute to task progress or are essential for understanding the current task state. Do not extract from Task Description or Previous Goal State.
3. **Previous Goal State:** This is provided ONLY for context verification. IT IS A FORBIDDEN SOURCE. You must NOT copy, infer, or hallucinate any object or state from 'Previous Goal State' into 'current_state'. If an object is in the goal but has not been observed yet, ignore it.
4. **Action Observation:** The immediate feedback observed by the agent after performing an action. This may be None if no observable outcome occurs.
5. **Previous States:** Represents the accumulated task-relevant states from previous steps. Use it to maintain continuity in the agent's world model:
   - If an object from Previous States appears in the current Observation, update its state with the most recent information.
   - If an object from Previous States does not appear in the current Observation, retain its previous state in Current State, assuming it remains unchanged.
   - Only include objects in Current State if they are either (a) newly observed and task-relevant, or (b) carried over from Previous States.
   - Do not include objects that are neither present in the current Observation nor in Previous States.

Input:

Task Description: `{task_description}`
Previous Goal State: `{prev_goal_state}`
Action: `{last_action}`
Observation: `{observation}`
Previous States: `{prev_states}`

Output Example:

`{output_format}`
`{output_format_des}`

MODULE B: Task-Relevant World State Extraction Template **(Filtering & Refinement)**

System Instruction: `{system_instruction}`

Role: You are a task-relevant state extraction assistant. Given the historical task-related states (Previous States), the current step's new states (Current State), the actions taken (Action History), and the Task Description, extract and update the relevant world states according to the following rules:

Rules:

1. Extract task-relevant states from the Current State. Exclude non-factual statements. Do not extract from Task Description. A state is considered strongly relevant only if it directly or clearly indirectly affects whether the task can be completed. If uncertain about a state's relevance, exclude it.
2. Combine the results from Rule 1 with the previous historical states (Previous States) to form an updated set of task-relevant historical states. Note that Previous States may be None, in which case the output should be based exclusively on the results from Rule 1.
3. Refine the combined state set by preserving all states that are relevant to the current task, even if they are intermediate or partially superseded. Only remove or update a state if it represents an outdated or incorrect version of the same information about a specific object. In such cases, retain the most accurate and up-to-date state while discarding its obsolete counterparts. Avoid eliminating states merely because more specific or complete information has emerged, as long as the original states contribute to the task context or support traceability of reasoning.
4. **Output Format:** Only output the JSON content—no additional text, explanation, or formatting. Follow the specified JSON format below exactly:
   `{output_format}`
   `{output_format_des}`

Input:

Task Description: `{task_description}`
Previous States: `{prev_states}`
Current State: `{current_state}`
Action History: `{action_history}`

Table 12: **Dynamic Goal Interpretation Template.** As the second stage of the STATEFACTORY method, this component maintains a persistent yet evolving definition of success, strictly distinguishing between the agent's "plans" and the task's "goals".

STATEFACTORY Pipeline Part II: Dynamic Goal Interpretation

MODULE C: Dynamic Goal Interpretation Template **(Goal State Management)**

System Instruction: `{system_instruction}`

Role: You are an expert in task goal state extraction. Your mission is to generate an Evolving Goal State: a minimal, accurate JSON specification that defines only the FINAL, STATIC BLUEPRINT FOR TASK SUCCESS, derived exclusively from the Task Description.

Core Principle: Goal vs. Plan (CRITICAL)

- **Goal (Legal):** A concrete, factual state that must be objectively true at the moment of success AND is explicitly or implicitly required by the Task Description. This includes all explicit milestones in a multi-step task (e.g., "First do A, then do B").
- **Plan / Blocker (Illegal):** An intermediate step, prerequisite, enabling action, or blocker that YOU (the LLM) deduce is necessary, but is NOT one of the final goals listed in the Task Description.
- **Constraint:** Your Goal State MUST ONLY contain Goals. It MUST NEVER contain Plans or Blockers.

Rules & Instructions:

1. **Output Format:** You must output only the JSON content. Do not include any additional text, explanations, comments, or markdown formatting. Adhere strictly to this JSON structure:
   - Output a SINGLE JSON object containing two keys:
   - "thinking": A step-by-step analysis string. First, analyze the 'Task Description' to identify the ultimate success conditions. Second, verify that every goal attribute comes from the Task, NOT just because it exists in 'Current State'. REJECT any state that appears solely because it is currently true in 'Current State'.
   - "goal_state": `{output_format_des}`
   - Example of desired output structure: `{output_format}` `{output_format_des}`
2. **Core Logic: Blueprint Creation (Step 0)**
   - **a. Step 0: Complete Task Translation:**
     - At Step 0, you MUST translate ALL final goal verbs and required milestones from the Task Description into their final physical states.
     - If a task has multiple required parts (e.g., "First do A, then do B"), ALL parts must be translated into final states and included in the blueprint from the beginning.
     - This blueprint is the COMPLETE and FINAL definition of success.
   - **b. State-Change Filter:** Use this only at Step 0 to exclude "trivial truths".
3. **Core Logic: Blueprint Evolution (Step $\geq$ 1)**
   - **a. GOAL PERSISTENCE:**
     - The Goal State created at Step 0 is the static blueprint for success.
     - Once a goal is added at Step 0, it MUST REMAIN for the entire episode.
     - DO NOT remove goals just because they are achieved in the Current State.
   - **b. STRICT IMMUTABILITY:**
     - You are STRICTLY FORBIDDEN from adding new goals after Step 0.
     - The blueprint from Step 0 is considered complete. Do not add goals for plans, prerequisites, or blockers that you deduce.
   - **c. LEGAL EVOLUTION:** The ONLY two legal modifications to the Goal State after Step 0 are:
     1. **Task Milestone Addition:**
        - *Condition:* The Task Description contains a multi-step requirement (e.g., "First do A, then do B"), and the Observation confirms that step A is now complete.
        - *Action:* You are now allowed to add the goal for step B to the Goal State. This is legal because the goal for B originates from the task description.
     2. **Refinement / Anchoring:**
        - *Condition:* A goal in the Goal State is generic, and the Observation first identifies its specific instance.
        - *Action:* You MUST update the Goal State to "anchor" the generic goal to the specific, observed object. WARNING: Do not copy irrelevant attributes from Current State unless the Task specifically requires them.
4. **Sources of Truth (Inputs)**
   - **Task Description:** Sole source of ultimate intent.
   - **Observation:** Used only to ground generics or reveal implicit preconditions.
   - **Previous Goal State:** The baseline to refine—never discard or contradict it without task-level justification.
   - **Current State & Action History:** REFERENCE ONLY. Provide contextual world facts and recent action history that may inform phrasing, object references, or state granularity, but they must not introduce new goal requirements.

Input:

Task Description: `{task_description}`
Current State: `{current_state}`
Observation: `{observation}`
Action History: `{action_history}`
Previous Goal State: `{prev_goal_state}`

Command: Now generate the updated GoalState based on the above rules.

Table 13: **Prompt variations for representation structural granularity.** We use the same backbone and input context across all levels, varying only *output schema* and *description constraints* to isolate effect of structural depth.

Ablation Study: Output Representation Schemas

LEVEL 1&2: UNSTRUCTURED OBSERVATIONS (RAW INPUT) & TEXTUAL STATES (FLAT DENOISING) *(Denoising only, no hierarchy)*

Description Constraint: Every element must be a grammatically complete English sentence that states only observable, factual, and specific information—no speculation or abstraction. Each sentence must explicitly name its subject to ensure clarity and self-containment.
Output Schema:
"⟨*world_state_description*⟩"

LEVEL 3: OBJECT-CENTRIC STATES (NESTED LIST) *(Entity grouping, entangled attributes)*

Object Definition: Refers to specific or abstract entities identified in the observations. Objects must be directly grounded in the observed input.
Description Constraint: A list of distinct attributes or facts associated with the identified object. Each item must be a standalone, grammatically complete English sentence that includes its own subject.
Output Schema:

```
[
    { "object": { "⟨obj_desc⟩": [ "⟨world_state_desc⟩", ... ] } },
    { "object": { "⟨obj_desc⟩": [ "⟨world_state_desc⟩" ] } }
]
```

LEVEL 4: OBJECT-ATTRIBUTE STATES (FULL FACTORIZATION) *(Ours: STATEFACTORY)*

Object Definition: Refers to specific or abstract entities identified in the observations. Only entities that are explicitly present or clearly implied should be included.
Description Constraint: Use key-value pairs to describe object attributes. The **"value"** (⟨*world_state_desc*⟩) must NOT assume the key is part of the sentence. It must be a complete sentence with subject and verb. The **"key"** (⟨*state_name*⟩) must be short and summative.
Output Schema:

```
[
    { "object": { "⟨obj_desc⟩": [
          { "⟨state_key⟩": "⟨world_state_desc⟩" },
          { "⟨state_key⟩": "⟨world_state_desc⟩" }
      ] }
    }
]
```

D.6 Ablation

D.6.1 Representation

To rigorously evaluate the impact of state representation, we implemented a **Representation-Controlled Ablation**. We established a baseline using the raw environment output and compared it against three levels of LLM-processed abstractions. Crucially, strictly controlling the representation format allows us to decouple the benefits of *denoising* from *structuring*.

The four levels of granularity are defined as follows. Details are in the **Table 13**:

- **Level 1: Unstructured Observations (Raw Input).** This baseline utilizes the raw, noisy output from the environment directly as the state representation. It serves as the control group for "zero processing."
- **Level 2: Textual States (Flat Denoising).** The model outputs a list of independent sentences. **Critically, this level maintains the same unstructured format as Level 1** (flat natural language) but applies the "relevance filtering" constraint. Comparing Level 1 and Level 2 isolates the performance gain attributed solely to **semantic denoising** without the introduction of hierarchical structure.
- **Level 3: Object-Centric States (Nested List).** The model groups sentences under specific object keys. This introduces **entity awareness** and tests the value of hierarchical grouping, but retains the "entangled attributes" limitation within the descriptions.
- **Level 4: Object-Attribute States (Full Factorization).** The model strictly separates descriptions into atomic Key-Value pairs. This is the standard STATEFACTORY configuration, which tests the value of complete **attribute disentanglement**.

D.6.2 LLM Backbone

To isolate the contribution of the underlying LLM's capabilities, we conducted an ablation study by swapping the core backbone of the STATEFACTORY pipeline while keeping all other distinct modules (e.g., prompt constraints, extraction protocols) constant. We evaluated distinct configurations varying in two dimensions: **parametric scale** (e.g., 14B vs. 30B) and **inference strategy** (Standard Instruct vs. Reasoning-Enhanced/Thinking modes).

As shown in **Table 18**, both dimensions strongly correlate with alignment precision (measured by EPIC Distance ↓).

- **Impact of Reasoning:** Enabling "thinking" modes yields significant gains. For instance, shifting from *GPT-OSS-20B (Low)* to *(Medium)* improves the average EPIC distance from 0.36 to 0.30. Similarly, *Qwen3-30B-Thinking* (0.41) substantially outperforms its standard instruction-tuned counterpart (0.55), confirming that extended inference capability is crucial for decomposing complex environmental states into structured factors.

- **Impact of Scale:** Performance generally improves with parameter count within the same model family. Comparing *Qwen3-14B* (0.58) to *Qwen3-30B* (0.55), we observe a natural improvement grounded in model size.

Consequently, STATEFACTORY is positioned to scale continuously with future advancements in both model size and reasoning strategies.

D.6.3 Semantic Embedding

As discussed in the main text, the reliability of STATEFACTORY hinges on the embedding model's ability to maintain **Semantic Discriminability**—the capacity to cluster semantically equivalent states while segregating distinct or contradictory ones. To quantify this, we utilize the **Triplet-based Accuracy** metric.

Probing Dataset Construction. We derived a synthetic probing dataset, $\mathcal{D}_{probe}$, from STATEFACTORY's intermediate outputs. For each unique state s_{anc} (Anchor), we generated a triplet $(s_{anc}, s_{pos}, s_{neg})$:

- **Anchor (s_{anc}):** A raw factorized state description (e.g., *"The apple is sliced"*).

- **Positive (s_{pos}):** A semantically equivalent but syntactically distinct paraphrase generated via an LLM. We explicitly instruct the model to vary vocabulary and structure while strictly preserving the truth value (e.g., *"Slices of apple are present"*).

- **Negative (s_{neg}):** A state sampled from the same episode's history describing a different object or a contradictory attribute (e.g., *"The apple is whole"* or *"The table is dirty"*). This serves as a distractor to test discriminative power.

Metric Definition: Triplet-based Accuracy. Let $\mathcal{D}$ denote the set of collected triplets, where each triplet $t = (s_{anc}, s_{pos}, s_{neg})$ consists of an anchor, a positive paraphrase, and a negative distractor. Let $d(\cdot)$ be the embedding function and $\text{sim}(\cdot, \cdot)$ denote cosine similarity. The accuracy is defined as:

$$\text{Accuracy} = \frac{1}{|\mathcal{D}|} \sum_{t \in \mathcal{D}} \mathbb{I}\big[\text{sim}(d(s_{anc}), d(s_{pos})) > \text{sim}(d(s_{anc}), d(s_{neg}))\big] \tag{15}$$

where $\mathbb{I}[\cdot]$ is the indicator function.

Qualitative Case Studies: Decision Boundaries.

To provide deeper insight into performance disparities, we analyze three representative cases from our probing dataset (**Table 14**). These examples illustrate the operational boundaries of different embedding architectures, focusing on the binary **Success** criterion ($sim(anc, pos) > sim(anc, neg)$) versus **Failure**.

Geometric vs. Co-occurrence (Case 1). In spatial tasks, the negative sample often shares high lexical overlap with the anchor (e.g., "blue block" and "orange block" appear in both). Discriminative models trained on NLI data robustly identify the contradiction between "on" and "standing over". Models like *embeddinggemma-300m* also pass this test, though they struggle to establish a confident separation compared to discriminative baselines.

Topic Clustering vs. Truth Value (Case 2). Case 2 illustrates a critical divergence. The negative sample describes the same object ("tissuebox") but in a contradictory location ("on sidetable"). Embeddings like *embeddinggemma-300m* fail here, likely because they cluster sentences based on shared topics rather than truth conditions. In contrast, discriminative models successfully reject the distractor, strictly adhering to the semantic state.

Table 14: **Polarity decision boundaries.** We evaluate whether embedding models rank semantic paraphrases above distractors. While most models handle spatial contradictions, topic-distraction failures show that some architectures prioritize lexical co-occurrence over factual truth conditions.

Qualitative Analysis: Success vs. Failure on Challenging Triplets

Category	Triplet (Anchor / Positive / Negative)	Polarity Check ($Sim_{pos} > Sim_{neg}$?)
Case 1: Spatial Relations (`BlocksWorld`)	**Anc:** "The blue block is on the orange block." **Pos:** "The blue block is positioned on top of..." **Neg:** "The agent is clutching the blue block while standing over the orange block."	✓ **Success** *all-MiniLM-L6-v2, all-mpnet-base-v2, nli-distilroberta-base-v2* ✓ **Success** *embeddinggemma-300m*
Case 2: Topic Distraction (`AlfWorld`)	**Anc:** "The tissuebox 1 is examined..." **Pos:** "...holding the tissuebox under the lamp" **Neg:** "The tissuebox 1 is on sidetable 1"	✓ **Success** *all-MiniLM-L6-v2, all-mpnet-base-v2* ✗ **Failure** *embeddinggemma-300m* *(confused by topic overlap)*

D.6.4 Comprehensive Ablation Results

Table 15: **Ablation Study I: Representation.** We compare varying state representation granularity across five domains. The results demonstrate that structured representations significantly outperform unstructured text, with our **Object-Attribute** approach achieving the lowest EPIC Distance (↓).

State Representation	EPIC Distance (↓)					
	`AlfWorld`	`ScienceWorld`	WebShop	`BlocksWorld`	`TextWorld`	Overall
Unstructured Observations (Raw)	0.49	0.45	0.52	0.88	0.50	0.57
Textual States (Flat Denoising)	0.31	0.35	0.42	0.72	0.37	0.43
Object-Centric States (Nested)	0.33	0.33	0.40	0.45	0.26	0.35
Object-Attribute States (Ours)	**0.28**	**0.29**	**0.29**	**0.43**	**0.20**	**0.30**

Table 16: **Ablation Study II: Goal Interpretation.** Comparison between **Offline (Oracle)** settings using ground-truth goal states and **Online (Dynamic)** setting using inferred ones. The minimal gap indicates our Dynamic Goal Interpretation is robust without requiring privileged information.

Goal Source Setting	EPIC Distance (↓)					
	`AlfWorld`	`ScienceWorld`	WebShop	`BlocksWorld`	`TextWorld`	Overall
Offline (Oracle Goal State)	0.26	0.27	0.26	0.40	0.22	0.28
Online (Dynamic Goal State)	0.28	0.29	0.29	0.43	0.20	0.30

Table 17: **Ablation Study III: Semantic Embeddings.** We evaluate StateFactory using five different embedding backbones. We report the **EPIC Distance** (↓) on downstream tasks and the **Triplet Accuracy** (↑) on our synthetic probing dataset. Results show a strong correlation: models with higher Triplet Accuracy consistently yield lower EPIC distances.

Embedding Backbone	Probing Metric	Downstream Performance (EPIC ↓)					
	Triplet Acc. (↑)	`AlfWorld`	`ScienceWorld`	`WebShop`	Blocks	`TextWorld`	Overall
nli-distilroberta-base-v2	0.83	**0.28**	**0.29**	**0.29**	0.47	**0.20**	0.31
all-mpnet-base-v2	0.76	**0.28**	**0.29**	**0.29**	0.54	**0.20**	0.32
all-MiniLM-L6-v2	0.81	**0.28**	**0.29**	**0.29**	**0.43**	**0.20**	**0.30**
bge-base-en-v1.5	0.78	0.32	0.34	0.31	0.41	0.25	0.33
embedding-gemma-300m	0.67	0.45	0.47	0.42	0.52	0.38	0.45

Table 18: **Ablation Study IV: LLM Backbone.** We compare different LLM backbones driving the StateFactory pipeline. **Lower** EPIC distance (↓) indicates better alignment with ground-truth progress. The results highlight that "Thinking" variants and larger parameter scales consistently improve reward precision.

Backbone Model	EPIC Distance (↓)					
	`AlfWorld`	ScienceWorld	`WebShop`	Blocks	`TextWorld`	Overall
Qwen Family (Impact of Scale & Thinking)						
Qwen3-14B	0.53	0.57	0.59	0.68	0.53	0.58
Qwen3-30B-A3B-Instruct-2507	0.53	0.59	0.58	0.53	0.52	0.55
Qwen3-30B-A3B-Thinking-2507	0.36	0.46	0.40	0.52	0.31	0.41
GPT-OSS Family (Impact of Reasoning Effort)						
GPT-OSS-20B (Low)	0.37	0.35	0.34	0.47	0.26	0.36
GPT-OSS-20B (Medium)	**0.28**	**0.29**	**0.29**	**0.43**	**0.20**	**0.30**

D.7 Utility for Agent Planning

D.7.1 System-1 Policy Implementation Details

To rigorously evaluate the efficacy of STATEFACTORY, we implemented a standard ReAct (Reasoning + Acting) agent as the primary baseline and the foundational System-1 component of our framework. The implementation follows the standard interleaving of thought generation and action execution, adapted for the specific constraints of `AlfWorld`, `BlocksWorld`, and `ScienceWorld`.

Action Parsing and Error Recovery. The language model is queried to generate a response in a strict format: `Thought: <reasoning> \n Action: <action>`. We employ a robust regular-expression parser to extract the executable command.

- **Syntactic Error Feedback:** If the output format is malformed, such as missing the "Action:" prefix, the system intercepts the error and returns a synthetic observation: *"Observation: Invalid Format. Please strictly follow the format Thought:... Action:..."*.
- **Semantic Error Feedback:** If the action is syntactically correct but physically impossible, such as trying to pick up a table, the environment's native error message is fed back to the agent, allowing for semantic self-correction.

Action Space Exploration. To facilitate efficient exploration, we explicitly provide the agent with the list of currently valid actions, such as `info['admissible_commands']`, at each step.

D.7.2 System-1 Policy Enhancement via STATEFACTORY

To overcome the ambiguity of sparse environmental signals in the standard ReAct baseline, we augment the System-1 policy with a **dense state-aware reward mechanism**. Unlike the original ReAct loop, which relies solely on textual observations, our framework explicitly quantifies the "progress" of each step via STATEFACTORY. This allows the agent to evaluate the potential utility of candidate actions before finalizing its decision.

Candidate Evaluation & State Abstraction. At each time step t, rather than blindly selecting an action, the agent first identifies a set of candidate actions. For each candidate, we leverage the STATEFACTORY engine to project the potential outcome into a structured semantic state s_{t+1}. By using *Dynamic Goal Interpretation* prompts, we filter out environmental noise and focus on task-relevant state changes.

Differential Reward Calculation. For each candidate transition, we compute an immediate, dense reward r_t that reflects its contribution to the final goal g. The reward is defined by the gain in semantic proximity:

$$r_t = \text{Score}(s_{t+1}, g) - \text{Score}(s_t, g) - \lambda \cdot \mathbb{I}(\text{Repetition}). \quad (16)$$

This differential score captures the *Relative Gain*, positively rewarding actions that satisfy critical sub-goals, such as locating a required tool, while penalizing redundant or cyclical behavior.

Reward-Guided Decision Making. The calculated rewards $\{r_t\}$ are injected back into the ReAct context. The prompt is thus augmented from a standard observation to a `(Candidate Action, Predicted Reward)` assessment. This explicit feedback signal empowers the LLM to perform informed selection, prioritizing actions with high relative gain and pivoting away from unproductive strategies with low or negative reward, without requiring expensive lookahead simulations.

We provide the prompts used to filter and propose the **Top-K candidate actions** in Section D.7.5. These prompts assist the ReAct policy in identifying logical next steps before simulation. To balance exploration and efficiency, we set $K = 3$ for *AlfWorld* and *BlocksWorld*, and $K = 5$ for *ScienceWorld* due to its significantly larger and more complex action space.

D.7.3 System-2 Planning via MCTS and World Models

While the dense reward mechanism enhances the local policy, complex tasks often require counterfactual reasoning—simulating "what if" scenarios without committing to irreversible actions in the real environment. To achieve this, we implement a System-2 planner using Monte Carlo Tree Search (MCTS), grounded by a pre-trained World Model (Li et al., 2025).

Generative Action Proposal (Top-K). Unlike traditional RL environments where the action space is discrete and known, text-based environments feature an unbounded action space. During the expansion phase of MCTS, the World Model cannot list valid moves. Therefore, we utilize the Agent LLM as a *Generator*. It analyzes the current observation and proposes the Top-K, typically $K = 3$, most logical actions. As shown in the prompt templates,

Table 19: Trajectory Trace of Case Study 1: **"Task: Look at CD under the desklamp."** This case demonstrates the internal logic of the StateFactory engine, specifically how it facilitates **dynamic goal interpretation** by anchoring abstract instructions to crystallized physical entities (g_t). By extracting structured states (s_t) from observations, the engine generates **continuous reward gradients** (Relative Gain) that guide the MCTS search. Note the significant reward spikes upon environmental constraint satisfaction (Step 7) and identification of the target entity (Step 8), which effectively steers agent away from redundant exploration towards a successful execution path.

Step	Action	Simplified Obs. & State Extraction ($\hat{s}_t$)	Goal Interpretation ($\hat{g}_t$)	Reward / Gain
1	`go to desk 2`	**Obs:** See **cd 1** on desk 2. **State:** {cd 1: [on desk 2]}	**Entity Discovery:** CD → **cd 1**	+0.29 ↑
...		*Steps 2–5: Initial exploration of the environment*		
6	`go to desk 1`	**Obs:** See **desklamp 1** on desk 1. **State:** {desklamp 1: [on desk 1]}	**Constraint Identification:** Lamp → **lamp 1**	+0.02 ↑
7	`use desklamp 1`	**Obs:** Lamp 1 is turned on. **State:** {lamp 1: [**is turned on**]}	**Condition Met:** cd 1 under desklamp 1	+0.22 ↑
8	`go to desk 2`	**Obs:** Navigate back to retrieve cd 1 with satisfied environmental constraints.	**(Unchanged)**	+0.31 ↑
...		*Steps 9–11: Redundant exploration and state verification*		
12	`take cd 1 from desk 2`	**Obs:** Pick up cd 1 from desk 2. **State:** {cd 1: [in hand]}	**(Unchanged)**	+0.12 ↑
13	`go to desk 1`	**Obs:** Arrive at desk 1 with cd 1. **State:** {cd 1: [under lamp 1]}	**Goal Satisfied.**	1.00 ↑

we enforce a "Mental Scratchpad" step, requiring the agent to explicitly verify inventory status and object visibility before generating commands, thereby reducing hallucinated actions.

Simulation via World Model. Selected actions are executed in a virtual environment simulated by a LLaMA-based World Model (Li et al., 2025). This model predicts the next textual observation o_{t+1} given the history H_t and action a_t.

Truncated Rollout & Evaluation. A major challenge in text simulation is hallucination drift, where errors accumulate over long horizons. To mitigate this, we adopt a conservative *Single-Step Rollout* strategy ($d = 1$). We prioritize accurate, immediate state evaluation over uncertain long-term planning. The simulated outcome is immediately scored by the StateFactory engine, which calculates the reward based on the semantic state change. This score backpropagates to guide the root action selection.

StateFactory as a Semantic Value Function. The efficacy of our truncated planning ($d = 1$) heavily relies on the density of the evaluation signal. In traditional sparse-reward settings, a single-step lookahead is typically insufficient to capture value, as the terminal goal remains distant. However, StateFactory bridges this gap by converting semantic state transitions into immediate scalar feedback, serving as a robust heuristic for MCTS.

D.7.4 Case Study of System-2 Planning

Qualitative Analysis. We further investigate whether accurate reward prediction can facilitate complex planning within a System-2 framework. To this end, we integrate LLM-based action proposals, a learned world model (Li et al., 2025), and StateFactory into Monte Carlo Tree Search (MCTS). In this setup, StateFactory serves as a structured heuristic that guides search with a continuous reward signal rather than relying only on sparse success feedback.

As shown in **Table 19**, the reward signal increases as the agent discovers and moves toward task-relevant entities, such as the target CD. This behavior suggests that StateFactory can provide informative intermediate values even before task completion. The effect is driven by two components: dynamic goal interpretation, which refines abstract instructions into concrete state requirements, and structured state extraction, which identifies task-relevant evidence from the current observation. When a relevant object or constraint is discovered, the reward exhibits an immediate increase, helping MCTS prioritize grounded trajectories and avoid redundant exploration.

D.7.5 Prompt Templates

ReAct Baseline Prompt: ALFWORLD ***(Household Tasks)***

Interact with a household to solve a task. Imagine you are an intelligent agent in a household environment and your target is to perform actions to complete the task goal. At the beginning of your interactions, you will be given the detailed description of the current environment and your goal to accomplish and Task Knowledge to guide you to finish the task.
For each of your turns, you will be given the observation of the last turn and the valid actions you can take right now (Valid Actions). You should first think about the current condition and plan for your future actions, then output your action in this turn. Your output must strictly follow this format: "Thought: your thoughts.\nAction: your next action".
Your response should use the following format:
Thought: <your thoughts>
Action: <your next action>
Here is an example:
You are in the middle of a room. Looking quickly around you, you see a bathtubbasin 1, a cabinet 2, a cabinet 1, a countertop 1, a garbagecan 1, a handtowelholder 1, a sinkbasin 1, a toilet 1, a toiletpaperhanger 1, and a towelholder 1.
Your task is to: put a toiletpaper in toiletpaperhanger.
Thought: I need to place a toilet paper roll on the toilet paper hanger. First, I should check if there is already a roll on the hanger or if it is empty.
Action: go to toiletpaperhanger 1
Observation: On the toiletpaperhanger 1, you see nothing.
Thought: The toilet paper hanger is empty, so I need to find a new roll of toilet paper to put on it.
Action: go to toilet 1
Observation: On the toilet 1, you see a soapbottle 1, and a toiletpaper 1.
Thought: I have found a toilet paper roll on the toilet. I should pick it up so I can place it on the hanger.
Action: take toiletpaper 1 from toilet 1
Observation: You pick up the toiletpaper 1 from the toilet 1.
Thought: Now that I have the toilet paper roll, I need to return to the toilet paper hanger to install it.
Action: go to toiletpaperhanger 1
Observation: On the toiletpaperhanger 1, you see nothing.
Thought: I am back at the toilet paper hanger with the roll in hand. It is time to put the toilet paper on the hanger.
Action: put toiletpaper 1 in/on toiletpaperhanger 1
———————————————————————-
Now, the real task begins.

ReAct Baseline Prompt: BLOCKSWORLD ***(Spatial Logic)***

Interact with a blocks world to solve a stacking task. Imagine you are an intelligent planning agent in a blocks world environment and your target is to perform actions to complete the task goal. At the beginning of your interactions, you will be given the detailed description of the current block configuration and your goal to accomplish.
For each of your turns, you will be given the observation of the last turn and the valid actions you can take right now (Valid Actions). Each valid action is shown as:
<ACTION_ID> -> <human-readable explanation>
You should:
1. Use the explanation to understand what each action does.
2. Output ONLY the <ACTION_ID> in your "Action:" line.
Your response should use the following format:
Thought: <your thoughts>
Action: <ACTION_ID>
Here is an example:
Task: the green block is on top of the red block
Observation: the green block is clear, the red block is on the table, the hand is empty
Valid Actions:
UNSTACK_A_B -> unstack the green block from on top of the red block
PICK-UP_C -> pick up the blue block
Thought: I need to remove the green block from the red block. The action UNSTACK_A_B does exactly that.
Action: UNSTACK_A_B
———————————————————————-
Now, the real task begins.

ReAct Baseline Prompt: SCIENCEWORLD ***(Scientific Discovery)***

You are an intelligent agent conducting a scientific experiment in a simulated environment. Your goal is to complete the given task by performing a sequence of valid actions.
The environment contains several rooms: kitchen, foundry, workshop, bathroom, outside, living room, bedroom, greenhouse, art studio, and hallway.
For each of your turns, you will be given the observation of the last turn and the valid actions you can take right now (Valid Actions). You should first think about the current condition and plan for your future actions, then output your action in this turn.
Your response should use the following format:
Thought: <your thoughts>
Action: <your next action>
Example:
Observation: You are in the kitchen. On the counter, you see a beaker 1 and a note 1.
Valid Actions: ['look around', 'examine beaker 1', 'read note 1', 'teleport to greenhouse', 'pick up beaker 1']
Thought: I should read the note first—it might contain instructions for the experiment.
Action: read note 1
———————————————————————————-

Now, the real task begins.

Action Selection Prompt: ALFWORLD ***(Household Tasks)***

You are an expert agent playing a text-based adventure game.
Your goal is to complete the following Task: `{task}`

Current Observation:
`{obs}`
Interaction History (Actions & Results):
`{history}`
Admissible Actions (Candidates):
`{candidates_with_ids}`

Your task is to identify the top `{k}` actions that are most likely to advance the game state.
Selection Rules:
1. **Analyze History**: Check "Interaction History". Avoid actions that resulted in "Nothing happens" or revealed empty containers. Avoid loops.
2. **Logical Continuity**: Select actions that logically follow the observation, e.g., see target → take it.
3. **Exploration**: If blocked, prioritize navigating to new areas.
4. **Index Only**: Return the ID numbers, e.g., 0, 1, only.

Return JSON Format:
```
{
   "reasoning": "...",
   "selected_ids": [id1, id2, ...]
}
```

Action Selection Prompt: BLOCKSWORLD ***(Spatial Logic)***

You are an expert agent solving a Blocks World planning task.
Your goal is to complete the following Task: `{task}`
Context:
- Observation: `{obs}`
- History: `{history}`
- Candidates: `{candidates_with_ids}`

Format: `[ID]: [ACTION_COMMAND]` → `[description]`
Your Task:
Select top `{k}` IDs based on:
1. **Feasibility**: Respect stacking rules.
2. **Relevance**: Move blocks toward target.
3. **Repetition**: Avoid failed actions.
4. **Strategy**: Free blocked blocks first.

Return JSON Format:
```
{
   "reasoning": "...",
   "selected_ids": [id1, id2, ...]
}
```

Action Selection Prompt: SCIENCEWORLD *(Scientific Discovery)*

You are a strategic planner for a scientific agent.
Context:
- Goal: `{task}`
- Current Observation: `{obs}`
- Recent History: `{history}`

Candidate Actions:
`{candidates_with_ids}`
Selection Strategy:
1. **Analyze History**: Do not repeat failed actions. If not in target room, prioritize navigation.
2. **Strict Focus Protocol**:
- Risk: Focusing on the wrong object triggers a -100 penalty.
- Sequence: Follow "First X, then Y" logic. Do not focus on Y if X is not done.
3. **Filter Noise**: Ignore objects not mentioned in the current goal step.

Output Format:
```
{
   "reasoning": "Explain sequence logic...",
   "selected_ids": [id1, id2, ...]
}
```

System-2 Action Generation: ALFWORLD

You are an expert agent playing AlfWorld, a text-based simulation game.
Your goal is to analyze the current state and generate `{k}` valid, logical next actions to complete the task.

Contextual Information:
Task: `{task}`
Recent History: `{history}`
Current Observation: `{obs}`

Mental Scratchpad (Mandatory Step):
You must analyze the situation step-by-step before generating actions:
1. **Inventory Status**: What are you holding right now?
2. **Current Location**: Where are you?
3. **Visible Objects**: List valid interactable objects/receptacles.
4. **Immediate Sub-goal**: Decide the next logical step (Explore/Interact/Process).

Strict Action Rules:
1. **Object IDs**: Always use specific numbers, e.g., *mug 1*.
2. **Interaction**: Actions only work on visible or inventory objects.
3. **Navigation**: You can *go to* any standard furniture in the room.

Allowed Templates:

- go to [receptacle X]
- take [object X] from [receptacle X]
- put [object X] in/on [receptacle X]
- open [receptacle X] / close [receptacle X]
- toggle [object X] [receptacle X]
- clean [object X] with [receptacle X]
- heat [object X] with [receptacle X]
- cool [object X] with [receptacle X]
- use [object X] / look

Output Format:
Return a JSON object containing your analysis and top `{k}` actions.

```
{
   "analysis": {
      "inventory": "Nothing",
      "current_location": "countertop 1",
      "visible_objects": ["apple 1", "knife 1", "cabinet 1"],
      "sub_goal": "Need to wash the apple, but first I need to take it."
   },
   "generated_actions": ["take apple 1 from countertop 1", "go to sinkbasin 1"]
} Now, generate the response for the current situation.
```

World Model: State-Aware Simulator ***(Internal Representation)***

[System: Ground Truth State]
You act as a deterministic physics engine for AlfWorld. Your goal is to update the environment state based on user actions.
Hidden Environment Details:
- **Receptacles:** On bed 1: book 1, laptop 1; On desk 2: alarmclock 1, mug 1; Drawer 1 (closed): creditcard 1.
- **Toggles:** desklamp 1 is currently OFF.
- **Task Goal:** examine the alarmclock with the desklamp.

[Interaction History]
Initial Observation: You are in a room. You see a bed 1, desk 1, desk 2...
User Action: `go to desk 2`
Observation: On the desk 2, you see a alarmclock 1 and a mug 1.
User Action: `use desklamp 1`
Next Observation (Prediction):

E RewardPrediction Examples

Table 20: **Visualized Positive Sample (`TextWorld`).** Task: *"lift the cuboid keycard... take the knife... place the knife on the board."* The agent gets the key, unlocks the chest, retrieves the knife, and places it on the target.

Step	Action (a_t)	Observation (o_t)	GT (R_t)
0	examine type T box	The type T box looks strong, and impossible to destroy...	0.17
1	take cuboid keycard	You pick up the cuboid keycard from the ground.	0.33
2	unlock cuboid chest...	You unlock the cuboid chest.	0.50
3	open cuboid chest	You open the cuboid chest, revealing a potato and a knife.	0.67
4	take knife...	You take the knife from the cuboid chest.	0.83
5	**put knife on board**	**You put the knife on the board... The End**	**1.00**

Table 21: **Visualized Negative Sample (`TextWorld`).** Task: *"lift the cuboid keycard... take the knife... place the knife on the board."* The agent interacts with irrelevant objects and picks up the wrong keycard.

Step	Action (a_t)	Observation (o_t)	GT (R_t)
0	inventory	You are carrying: a mop.	0.00
1	put mop on board	You put the mop on the board.	0.00
2	close fridge	You close the fridge.	0.00
3	examine mop	The mop would seem to be out of place here	0.00
4	take type T keycard	You pick up the type T keycard from the ground.	0.00
5	examine board	The board is stable. On the board is a mop.	0.00

Table 22: **Visualized Positive Sample (`ScienceWorld`).** Task: *"find a(n) living thing... move it to the yellow box."* The agent successfully navigates outside, identifies the target (baby wolf), and transports it to the goal location.

Step	Action (a_t)	Observation (o_t)	GT (R_t)
0	look around	This room is called the foundry. In it, you see: the agent... a blast furnace...	0.00
1	open door to outside	The door is now open.	0.08
2	go to outside	You move to the outside.	0.17
3	look around	This outside location is called the outside. Here you see: ... a baby wolf...	0.17
4	focus on baby baby wolf	You focus on the baby wolf.	0.67
5	pick up baby baby wolf	You move the wolf to the inventory.	0.75
6	open door to kitchen	The door is now open.	0.75
7	go to kitchen	You move to the kitchen.	0.75
8	open door to hallway	The door is now open.	0.75
9	go to hallway	You move to the hallway.	0.75
10	open door to living room	The door is now open.	0.75
11	go to living room	You move to the living room.	0.83
12	**move baby... to yellow box**	**You move the wolf to the yellow box.**	**1.00**

Table 23: **Visualized Negative Sample (`ScienceWorld`).** Task: *"find a(n) living thing... move it to the yellow box."* The agent performs irrelevant interactions that fail to advance the task objective.

Step	Action (a_t)	Observation (o_t)	GT (R_t)
0	look at door	ERROR: SHOULD USE OVERRIDE FOR PORTAL.	0.00
1	connect orange to outside	terminal 1 on orange is now connected to terminal 1 on outside	0.00
2	connect air to foundry	terminal 1 on air is now connected to terminal 1 on foundry	0.00
3	connect air to steel table	terminal 2 on air is now connected to terminal 1 on table	0.00
4	open door	The door is now open.	0.08
5	disconnect orange	orange has been disconnected	0.08
6	connect drain to foundry	terminal 1 on drain is now connected to terminal 2 on foundry	0.08
7	connect agent to inventory	terminal 1 on agent is now connected to terminal 1 on inventory	0.08
8	connect sink...	terminal 1 on sink is now connected to terminal 1 on blast furnace	0.08
9	look at outside	This outside location is called the outside. Here you see: ...	0.08
10	mix door	That container is empty, so there are no items to mix.	0.08
11	look in steel table	Inside the table is: nothing	0.08
12	mix agent	There is only one thing (inventory)	0.08

Table 24: **Visualized Positive Sample (`AlfWorld`).** Task: *"put some cd on safe."* The agent identifies the target object (CD 2) on the dresser, navigates to the safe, opens it, and deposits the item.

Step	Action (a_t)	Observation (o_t)	GT (R_t)
0	go to dresser 1	You arrive at dresser 1. On the dresser 1, you see a bowl 1... a cd 2...	0.20
1	take cd 2 from dresser 1	You pick up the cd 2 from the dresser 1.	0.40
2	go to safe 1	You arrive at safe 1. The safe 1 is closed.	0.60
3	open safe 1	You open the safe 1. The safe 1 is open. In it, you see nothing.	0.80
4	**move cd 2 to safe 1**	**You move the cd 2 to the safe 1.**	**1.00**

Table 25: **Visualized Negative Sample (`AlfWorld`).** Task: *"put some cd on safe."* The agent wanders to irrelevant locations (laundry hamper, bed, drawers) and fails to interact with the target objects.

Step	Action (a_t)	Observation (o_t)	GT (R_t)
0	go to laundryhamper 1	You arrive at laundryhamper 1. On the laundryhamper 1, you see nothing.	0.00
1	go to bed 1	You arrive at bed 1. On the bed 1, you see a cellphone 2...	0.00
2	go to drawer 11	You arrive at drawer 11. The drawer 11 is closed.	0.00
3	look	You are facing the drawer 11. Next to it, you see nothing.	0.00
4	go to drawer 6	You arrive at drawer 6. The drawer 6 is closed.	0.00

Table 26: **Visualized Positive Sample (`AlfWorld`).** Task: *"clean some egg and put it in fridge."* The agent retrieves the target egg from the fridge, cleans it with the sinkbasin, and places it back into the fridge.

Step	Action (a_t)	Observation (o_t)	GT (R_t)
0	go to fridge 1	You arrive at fridge 1. The fridge 1 is closed.	0.14
1	open fridge 1	You open the fridge 1. In it, you see a apple 1, a bowl 2, a bowl 1, a cup 1...	0.29
2	take egg 4 from fridge 1	You pick up the egg 4 from the fridge 1.	0.43
3	go to drawer 2	You arrive at drawer 2. The drawer 2 is closed.	0.57
4	clean egg 4 with sinkbasin 1	You clean the egg 4 using the sinkbasin 1.	0.71
5	go to fridge 1	You arrive at fridge 1. The fridge 1 is open. In it, you see a apple 1...	0.86
6	**move egg 4 to fridge 1**	**You move the egg 4 to the fridge 1.**	**1.00**

Table 27: **Visualized Negative Sample (`AlfWorld`).** Task: *"clean some egg and put it in fridge."* The agent moves to irrelevant locations, picks up a mug instead of the egg, and never completes the required cleaning or placement.

Step	Action (a_t)	Observation (o_t)	GT (R_t)
0	go to sinkbasin 1	You arrive at sinkbasin 1. On the sinkbasin 1, you see a mug 2.	0.00
1	examine drawer 2	The drawer 2 is closed.	0.00
2	take mug 2 from sinkbasin 1	You pick up the mug 2 from the sinkbasin 1.	0.00
3	go to cabinet 6	You arrive at cabinet 6. The cabinet 6 is closed.	0.00
4	go to drawer 4	You arrive at drawer 4. The drawer 4 is closed.	0.00
5	go to cabinet 6	You arrive at cabinet 6. The cabinet 6 is closed.	0.00
6	go to countertop 1	You arrive at countertop 1. On it, you see a apple 3, a bread 1...	0.00

Table 28: **Visualized Positive Sample (`WebShop`).** Task: *"women ankle strap sandal, black, < 80.00 dollars."* The agent identifies a compliant item, selects the required black attribute, and completes the purchase.

Step	Action (a_t)	Observation (o_t)	GT (R_t)
0	search[sandal ankle...]	...Wedge Sandals... $8.13 ... Gibobby Summer Sandals... $14.71 ...	0.14
1	click[next >]	[button] Back to Search ... Page 2 (Total results: 50) ...	0.29
2	click[next >]	[button] Back to Search ... Page 3 (Total results: 50) ...	0.43
3	click[next >]	[button] Back to Search ... Page 4 ... Gibobby Summer Sandals... $1.87 ...	0.57
4	click[item - gibobby...]	...Gibobby Summer Sandals... Price: **$1.87 to $3.37** ... [button] Buy Now	0.71
5	click[black]	...color [clicked button] black... size...	0.86
6	**click[buy now]**	**You have successfully purchased the product.**	**1.00**

Table 29: **Visualized Negative Sample (`WebShop`).** Task: *"women ankle strap sandal, black, < 80.00 dollars."* The agent performs an irrelevant search and interacts with food items instead of footwear.

Step	Action (a_t)	Observation (o_t)	GT (R_t)
0	search[plant based]	...Plant-Based Bolognese... **$62.99** ... Plant-Based Jerky... **$4.98** ...	0.00
1	click[item - B085...]	...Gardein Ultimate Plant-Based Jerky... Price: $6.29 ...	0.00
2	click[teriyaki]	...flavor name... [clicked button] teriyaki...	0.00
3	click[original]	...flavor name... [clicked button] original...	0.00
4	click[Features]	...Contains one 2.25-ounce package of Gardein...	0.00
5	click[< Prev]	...[clicked button] original... [clicked button] teriyaki...	0.00
6	click[hot & spicy]	...[clicked button] hot & spicy...	0.00

Table 30: **Visualized Positive Sample (`TextWorld`).** Task: *"retrieve key... unlock safe... insert spoon."* The agent identifies the key, unlocks the container, retrieves the target tool, and completes the puzzle.

Step	Action (a_t)	Observation (o_t)	GT (R_t)
0	take key	You pick up the key from the ground.	0.20
1	unlock safe with key	(the safe with the key) You unlock the safe.	0.40
2	open safe	(the safe) You open the safe, revealing a spoon.	0.60
3	take spoon from safe	(the safe) You take the spoon from the safe.	0.80
4	**insert spoon...**	**You put the spoon into the spherical safe... The End**	**1.00**

Table 31: **Visualized Negative Sample (`TextWorld`).** Task: *"retrieve key... unlock safe... insert spoon."* The agent gets distracted by irrelevant objects and consumes food instead of solving the lock puzzle.

Step	Action (a_t)	Observation (o_t)	GT (R_t)
0	examine saucepan	The saucepan is stable.	0.00
1	examine coffee cup	The coffee cup is cheap looking.	0.00
2	examine spherical latchkey	The spherical latchkey is cold to the touch	0.00
3	eat legume	You eat the legume. Not bad.	0.00
4	examine saucepan	The saucepan is stable.	0.00

Table 32: **Visualized Positive Sample (`WebShop`).** Task: *"i want a gluten free packaged meal, and price lower than 50.00 dollars."* The agent verifies the item description and purchases a product meeting the price constraint.

Step	Action (a_t)	Observation (o_t)	GT (R_t)
0	search[gluten free...]	...Banza Chickpea Mac and Cheese... **$35.3** ... Banza Chickpea Pasta Shells... **$24.99** ...	0.20
1	click[item - banza...]	...Banza Chickpea Pasta Shells... Price: **$24.99** ... [button] Description ... [button] Buy Now	0.40
2	click[description]	...high protein, lower carb, gluten free alternative to traditional mac...	0.60
3	click[< prev]	...Banza Chickpea Pasta Shells... Price: **$24.99** ... [button] Buy Now	0.80
4	**click[buy now]**	**You have successfully purchased the product.**	**1.00**

Table 33: **Visualized Negative Sample (`WebShop`).** Task: *"i want a gluten free packaged meal, and price lower than 50.00 dollars."* The agent selects an item priced at $100.0 and fails to complete a valid purchase.

Step	**Action** (a_t)	**Observation** (o_t)	**GT** (R_t)
0	search[i want a gluten...]	...Amy's Vegan... Burrito... **$100.0** ... Annie's Shell Pasta... $39.48 ...	0.00
1	click[item - B01FW...]	...Amy's Vegan, Gluten Free Beans & Rice Burrito... Price: **$100.0** ...	0.00
2	click[Features]	One 5.5 oz. Amy's Bean & Rice... Gluten Free...	0.00
3	click[Back to Search]	[button] Search [button_]	0.00
4	click[Search]	[button] Search [button_]	0.00

Table 34: **Visualized Positive Sample (`BlocksWorld`).** Task: *"the red block is on top of the orange block."* The agent unstacks the obstructing orange block to clear the red block, then stacks red onto orange.

Step	**Action** (a_t)	**Observation** (o_t)	**GT** (R_t)
0	unstack the orange block...	...hand is currently holding orange block, the red block is on the table...	0.25
1	put down the orange block	...hand is empty, the red block is on the table... orange block is on the table...	0.50
2	pick up the red block	...hand is currently holding red block, the blue block is on the table...	0.75
3	**stack the red block...**	**...hand is empty, the red block is on top of the orange block...**	**1.00**

Table 35: **Visualized Negative Sample (`BlocksWorld`).** Task: *"the red block is on top of the orange block."* The agent loops with an irrelevant blue block and fails to change the target block state.

Step	**Action** (a_t)	**Observation** (o_t)	**GT** (R_t)
0	pick up the blue block	...hand is currently holding blue block, the orange block is on top of the red block...	0.00
1	put down the blue block	...hand is empty, the orange block is on top of the red block...	0.00
2	pick up the blue block	...hand is currently holding blue block, the orange block is on top of the red block...	0.00
3	put down the blue block	...hand is empty, the orange block is on top of the red block...	0.00

Table 36: **Visualized Positive Sample (`BlocksWorld`).** Task: *"the red block is on top of the orange block."* The agent first removes the obstructing orange block, clears both target blocks, then moves the red block onto the orange block to satisfy the goal.

Step	**Action** (a_t)	**Observation** (o_t)	**GT** (R_t)
0	unstack orange from red	...hand is holding orange block, red block is on top of blue block...	0.25
1	put down orange	...hand is empty, red block is on top of blue block, orange block is on the table...	0.50
2	unstack red from blue	...hand is holding red block, blue block is on the table, orange block is on the table...	0.75
3	**stack red on orange**	**...hand is empty, red block is on top of orange block...**	**1.00**

Table 37: **Visualized Negative Sample (`BlocksWorld`).** Task: *"the red block is on top of the orange block."* The agent repeatedly unstacks and restacks the orange block on the red block.

Step	**Action** (a_t)	**Observation** (o_t)	**GT** (R_t)
0	unstack orange from red	...hand is holding orange block, red block is on top of blue block...	0.00
1	stack orange on red	...hand is empty, orange block is on top of red block, red block is on top of blue block...	0.00
2	unstack orange from red	...hand is holding orange block, red block is on top of blue block...	0.00
3	stack orange on red	...hand is empty, orange block is on top of red block, red block is on top of blue block...	0.00