SkillGen: Learning Domain Skills for In-Context Sequential Decision Making

Ruomeng Ding*
University of North Carolina at Chapel Hill
ruomeng@unc.edu

Wei Cheng
NEC Laboratories America
weicheng@nec-labs.com

Minglai Shao†
Tianjin University
shaoml@tju.edu.cn

Chen Zhao
Baylor University
chen_zhao@baylor.edu

Abstract

Large language models (LLMs) are increasingly applied to sequential decision-making through in-context learning (ICL), yet their effectiveness is highly sensitive to prompt quality. Effective prompts should meet three principles: focus on decision-critical information, provide step-level granularity, and minimize reliance on expert annotations through label efficiency. However, existing ICL methods often fail to satisfy all three criteria simultaneously. Motivated by these challenges, we introduce SkillGen, a skill-based ICL framework for structured sequential reasoning. It constructs an action-centric, domain-level graph from sampled trajectories, identifies high-utility actions via temporal-difference credit assignment, and retrieves step-wise skills to generate fine-grained, context-aware prompts. We further present a theoretical analysis showing that focusing on high-utility segments supports task identifiability and informs more effective ICL prompt design. Experiments on ALFWorld, BabyAI, and ScienceWorld, using both open-source and proprietary LLMs, show that SkillGen achieves consistent gains, improving progress rate by 5.9%–16.5% on average across models. The implementation of SkillGen is available at https://github.com/ruomengd/SkillGen.

1 Introduction

Large language models (LLMs) are increasingly applied to multi-step decision-making tasks across domains such as embodied control [1, 2, 3], text-based games [4, 5, 6], and online shopping [7, 8]. These tasks require agents to operate in dynamic environments, interact with the world through sequences of actions, and pursue long-horizon goals. In contrast to supervised fine-tuning (SFT) methods [9, 10, 11], which depend on large-scale expert demonstrations, in-context learning (ICL) offers a more lightweight and efficient alternative by guiding inference with only a few examples [12, 13]. Consequently, ICL has emerged as a central reasoning paradigm in many LLM-based agent frameworks for decision-making [14, 15, 16, 17]. To make ICL effective for multi-step tasks, prompt design should adhere to three key principles: (1) *Focused*: emphasize decision-critical information while minimizing redundant context; (2) *Granular*: offer fine-grained, step-level guidance that aligns to the evolving task state; (3) *Label-efficient*: replaces costly

Table 1: Comparison of ICL Methods.

ICL Method	Focused	Granular	Label-efficient
Fixed Prompting	✗	✗	✓
Task-level Retrieval	✗	✗	✗
Step-wise Retrieval	✗	✓	✗
Insight Summary	✓	✗	✗
SkillGen (ours)	✓	✓	✓

*This work was done while the first author was affiliated with Georgia Institute of Technology.
†Corresponding author.

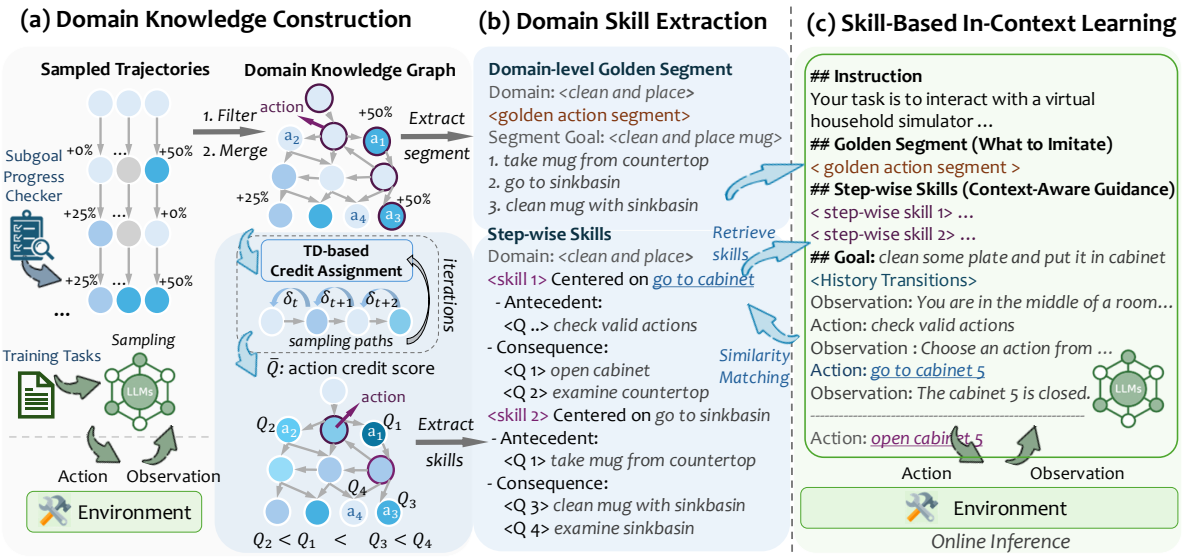


Figure 1: Framework of SkillGen.

expert trajectories with subgoal completion and success signals, which offer a more scalable and structurally aligned form of supervision.

Recent advances in ICL have enhanced performance by shifting from fixed, hand-crafted prompts to more context-sensitive prompt designs [18, 19, 20, 21]. As summarized in Table 1, various methods target different aspects of prompt design. Fixed prompting relies on limited examples that ignore task-specific context. Although label-free, such prompts fail to provide actionable, decision-level guidance. Task-level retrieval methods, such as Synapse [22], retrieve full expert demonstrations based on task metadata and use them as few-shot exemplars. While this improves contextual relevance, the retrieved prompts often include redundant steps and lack step-level resolution, limiting both focus and granularity. Step-level retrieval strategies, such as Trad [23], enhance granularity by retrieving trajectory fragments at each decision step. However, the retrieved actions are often disconnected and lack structural coherence, which undermines decision focus. These approaches often depend on expert-annotated trajectories, thereby reducing their label efficiency. Insight summarization approaches, including Leap [24] and ExpeL [25], generate high-level insights by comparing correct and wrong solutions. Yet these summarized insights are often too abstract to support intermediate decisions, providing limited fine-grained guidance. While each of these approaches addresses different aspects of prompt design, none of them simultaneously meets all three core principles.

To address these limitations, we propose SkillGen, an ICL framework that extracts and applies **domain-oriented** and **action-centric skills**. As illustrated in Figure 1, SkillGen operates in three stages, the first two stages are performed offline, while the third leverages the extracted skills to generate actions step-by-step during inference: (1) *Domain Knowledge Construction* – We construct an action-centric domain knowledge graph from sampled trajectories, effectively capturing the structural dynamics of the task. (2) *Domain Skill Extraction* – Temporal-Difference (TD) based credit assignment is employed to identify actions that consistently contribute to task progress; (3) *Skill-Based In-Context Learning* – During inference, SkillGen combines a golden segment with step-wise skills retrieved based on the current transition history to guide action generation. To support focused reasoning, SkillGen constructs prompts around decision-critical skill segments while filtering out irrelevant context. For fine-grained guidance, it encodes temporal and structural dependencies from the domain graph to retrieve skills that align with the current task history. To promote label efficiency, SkillGen leverages subgoal completion progress as weak supervision and applies TD-based action credits to extract skills, eliminating the need for full expert trajectories.

SKILLGEN improves average progress rate by 5.9%–16.5% across ALFWorld, BabyAI, and ScienceWorld, showing clear gains in sequential decision-making. To conclude, our primary contributions are as follows:

- We address the challenge of designing in-context learning (ICL) prompts that jointly support decision focus, step-level granularity, and label efficiency in sequential decision-making tasks.
- We propose SKILLGEN, a framework that learns domain-oriented skills to support focused, fine-grained ICL. Theoretical analysis shows that high-utility segments enhance task identifiability and ICL prompts.
- Our empirical results show that SKILLGEN consistently improves progress and success rates across various tasks. Ablations show that both the golden segment and step-wise skill retrieval contribute to performance gains.

2 Background

In-Context Learning. Xie et al. [26] model in-context learning (ICL) as an instance of implicit Bayesian inference. In this view, a language model infers a latent task parameter $\phi \in \Phi$ from the observed context $\mathcal{C}$ (e.g., a sequence of demonstrations or interaction history), forming a posterior $p(\phi \mid \mathcal{C})$ [27, 28]. Given a query input $x \in \mathcal{X}$, the model predicts by computing the posterior predictive distribution:

$$p(y \mid x, \mathcal{C}) = \int_{\phi} p(y \mid x, \phi)\, p(\phi \mid \mathcal{C})\, d\phi, \tag{1}$$

where $y \in \mathcal{Y}$ is the predicted output and $p(y \mid x, \phi)$ is the task-specific likelihood. Wies et al. [29] formalize this intuition within a PAC framework by modeling pretraining as sampling from a latent task mixture $D = \sum_{\phi \in \Phi} \pi(\phi) P_\phi$, where $\pi(\phi)$ is the prior of latent task ϕ, and P_ϕ denotes the corresponding data distribution. In this view, ICL serves to recover the underlying task $\phi^\star$ from the prompt, enabling accurate prediction without updating model parameters.

LLMs for Sequential Decision-making. We consider adopting LLMs as autonomous agents for sequential decision-making. In such environments, agents cannot directly observe the underlying state. We model this setting as a Partially Observable Markov Decision Process (POMDP) [30, 31], defined by POMDP $= (\mathcal{S}, \mathcal{A}, \Omega, P, R, O)$, where $\mathcal{S}$ denotes the latent state space, $\mathcal{A}$ the discrete action space, Ω the observation space, P the state transition function, R a sparse, progress-based reward function, and O the observation model. At each time step t, the agent receives a partial observation $o_t \sim O(o_t \mid s_t)$ and selects an action a_t based on the interaction transition: $h_t = \{(o_0, a_0), (o_1, a_1), \ldots, o_t\}$. Given the current history h_t, the LLM generates the next action through a prompting mechanism:

$$\pi(a_t \mid h_t) = \text{LLM}(a_t \mid \text{Prompt}(h_t)). \tag{2}$$

This formulation enables LLMs to serve as decision-making agents in partially observable settings, leveraging contextual information without access to full environment states or explicit policy optimization. The agent aims to maximize cumulative task progress over long-horizon episodes.

3 Method

To enable focused and fine-grained in-context learning, we introduce two forms of decision-critical knowledge derived from structured action knowledge [32, 33, 34]: (1) *Golden segment* – a concise, high-quality action sequence extracted from training trajectories within the task domain, selected for its maximal contribution

toward goal completion; (2) *Step-wise skills* – reusable local patterns centered on a key action, summarizing its typical antecedents and consequences within the domain. For example, for the action *"go to cabinet"*, common antecedents include *"check valid actions"* and *"examine countertop"*, while a likely consequence is *"open cabinet"*.

3.1 Domain Knowledge Construction

To induce domain-level knowledge, we first sample diverse trajectories from LLMs using high-temperature stochastic decoding. Each training instance is denoted by $d = (m, g) \in \mathcal{D}_{\text{train}}$, where m is the task domain information and g is the task goal. For each instance, N trajectories are sampled, forming:

$$\begin{aligned} \mathbb{T}_{\text{train}} &= \{(m, g, \mathcal{T}) \mid (m, g) \in \mathcal{D}_{\text{train}}\}, \\ \mathcal{T} &= \{(o_t, a_t, p_t)\}_{t=0}^{T}, \end{aligned} \tag{3}$$

where each step includes the observation o_t, action a_t, and progress signal p_t. To reduce noise and task-specific variance, trajectories are filtered by discarding invalid actions and those with zero final progress ($p_T = 0$). Actions are abstracted by removing object-specific identifiers (e.g., *"open cabinet 5"* $\rightarrow$ *"open cabinet"*) to reveal transferable patterns across instances, yielding sequences of action–progress pairs. The filtered trajectories for each domain m form:

$$\mathbb{T}_m = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_K\}, \quad \text{with } \mathcal{T}_k = \{(a_t, p_t)\}_{t=0}^{T_k}. \tag{4}$$

From these, a directed domain knowledge graph $\mathbb{G}_m = (\mathbb{V}, \mathbb{E})$ is constructed, where each node $a \in \mathbb{V}$ denotes an action, and each edge $(a_i, a_j) \in \mathbb{E}$ indicates a transition between consecutive actions. All paths share a common start a_{start} and end node a_{end} to ensure structural consistency. Edges are annotated with sets of observed progress deltas:

$$\begin{aligned} \mathcal{P}_\Delta(a_i, a_j) = \{p_{t+1} - p_t \mid (a_t, p_t), (a_{t+1}, p_{t+1}) \in \mathcal{T}, \\ a_t = a_i,\ a_{t+1} = a_j\} \end{aligned} \tag{5}$$

To maintain graph quality, self-loops are removed and low-impact nodes pruned based on $\mathcal{P}_\Delta(a_i, a_j)$ once the graph exceeds a reasonable scale. The resulting graph captures reusable action patterns and domain-level decision dynamics, offering a structured foundation for skill extraction. We extract a *golden segment*—a short action sequence with the highest subgoal progress rate within the domain—to serve as a concise, decision-critical exemplar for focused prompting.

3.2 Domain Skill Extraction

The progress delta $\mathcal{P}_\Delta$, annotated on graph edges, provides sparse, subgoal-level feedback by marking major milestones such as completing a cleaning step or reaching a destination. However, it often fails to capture the contribution of intermediate actions that enable these outcomes. As illustrated in Figure 2, only a few steps in the task *"heat some apple and put it in countertop"* receive positive deltas (e.g., *heating and placing the apple*), while prerequisite actions—like navigating to and opening the microwave—remain uncredited despite being causally necessary. This illustrates the need for credit assignment to assign action value more accurately.

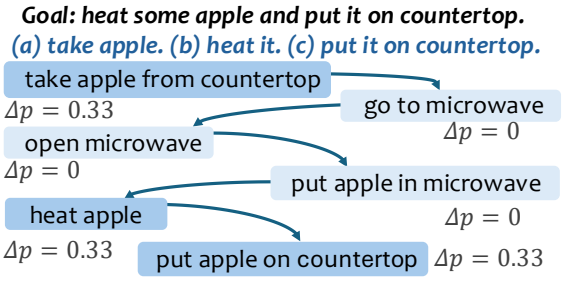


Figure 2: Sparse $\mathcal{P}_\Delta(a_i, a_j)$ reward only subgoal completions, omitting intermediate actions.

To address it, we treat $\mathcal{P}_\Delta$ as delayed rewards and propagate them backward along sampled action paths to estimate fine-grained action utility. While various metrics [35, 36] provide alternative means of estimating action importance, we adopt temporal difference learning with eligibility traces (TD(λ)) [37] to learn an action-value $Q(a)$, capturing the expected long-term credit of each action.

TD-based credit assignment. In the action-centric graph $\mathbb{G}_m = (\mathbb{V}, \mathbb{E})$, each iteration samples a set of action paths from a designated start node a_{start} to an end node a_{end} for downstream credit estimation. Formally, for iteration $i = 1, \ldots, N$, we sample:

$$\tau^{(i)} = (a_1^{(i)}, a_2^{(i)}, \ldots, a_{T_i}^{(i)}) \sim \mathcal{S}_{\text{path}}(a_{\text{start}}, a_{\text{end}}, \mathbb{G}_m), \tag{6}$$

where each $\tau^{(i)}$ is a valid path in $\mathbb{G}_m$ satisfying $(a_t^{(i)}, a_{t+1}^{(i)}) \in \mathbb{E}$, $a_1^{(i)} = a_{\text{start}}$, $a_{T_i}^{(i)} = a_{\text{end}}$. Here, $\mathcal{S}_{\text{path}}$ denotes a uniform distribution over all valid paths from a_{start} to a_{end} in $\mathbb{G}_m$. For each path $\tau = [a_0, a_1, \ldots, a_T]$, action credits are estimated using temporal-difference learning with eligibility traces. At each step t, the reward r_t is defined based on empirical progress deltas. If such records exist, i.e., $\mathcal{P}_\Delta(a_t, a_{t+1}) \neq \emptyset$, a value is uniformly sampled and perturbed with Gaussian noise:

$$r_t \sim \text{Uniform}(\mathcal{P}_\Delta(a_t, a_{t+1})) + \mathcal{N}(0, \sigma^2), \tag{7}$$

where $\text{Uniform}(\cdot)$ denotes a uniform distribution over finite set $\mathcal{P}_\Delta(a_t, a_{t+1})$. Otherwise, the reward defaults to pure noise, $r_t = \epsilon$, treating the transition as a step without observable progress. For each action a_t along path τ, the temporal-difference (TD) error is computed as:

$$\delta_t = r_t + \gamma Q(a_{t+1}) - Q(a_t). \tag{8}$$

To propagate credit backward, the eligibility trace for the current action a_t is incremented, and all actions with nonzero trace values are updated (see detailed algorithm in Appendix B.2):

$$\begin{aligned} E(a_t) &\leftarrow E(a_t) + 1, \quad Q(a) \leftarrow Q(a) + \alpha \delta_t E(a), \\ E(a) &\leftarrow \gamma \lambda E(a), \quad \forall a \in \{a' \mid E(a') > 0\}. \end{aligned} \tag{9}$$

Here, α is the learning rate, γ the discount factor, and λ the trace decay rate. The eligibility trace $E(a)$ accumulates credit for recently visited actions, enabling delayed reward signals to influence earlier decisions across the sampled paths. After learning, $Q(a)$ values are normalized into a dense credit distribution:

$$\bar{Q}(a) = \frac{\max(Q(a), 0)}{\sum_{a'} \max(Q(a'), 0)}, \tag{10}$$

emphasizing both goal-reaching actions and the intermediate steps that enable them. Additional details are provided in Appendix B.1. Structural statistics of domain knowledge graphs are provided in Appendix F.

Skill Extraction. For each action node a in the domain graph, we extract a local skill centered on a using its immediate neighbors and credit scores. We define the *antecedent set* as the set of predecessor actions (incoming edges), and the *consequence set* as the set of successor actions (outgoing edges). Each action is assigned a normalized credit score from TD-based propagation, and both sets are ranked accordingly. As shown in Figure 1 (b), the resulting skill is formulated as:

$$\text{Skill}(a) = (a, \textit{Antecedents}(a), \textit{Consequences}(a)), \tag{11}$$

where a is the central action, and the ranked context captures its typical usage patterns. This structure supports context-aware retrieval of reusable action-centric skills.

3.3 Skill-Based In-Context Learning

Building on structured skills extracted via TD-based credit assignment, we develop a retrieval-augmented prompting strategy that conditions a frozen LLM on both reusable domain knowledge and the agent's interaction history (Figure 1, right). The prompt integrates two levels of contextual information: (i) a domain-level *golden segment*—a concise, high-impact action sequence selected offline to serve as a focused exemplar; and (ii) *step-wise skills*—local, action-centric transitions retrieved based on the most recent action a_{t-1}.

Concretely, at each time step t, given a history transition $h_t = \{(o_0, a_0), \ldots, (o_{t-1}, a_{t-1}), o_t\}$, a pre-trained semantic retriever processes the recent action a_{t-1} and retrieves the most relevant action $\hat{a}$ from the domain graph. The final prompt is formed by concatenating the golden segment from domain m, the retrieved skills $\text{Skill}(\hat{a})$, and the current trajectory history, each serialized into natural language. This composite prompt enables context-aware, credit-guided reasoning without requiring any parameter updates to the LLM. The next action is then sampled autoregressively:

$$\begin{aligned} \Phi &= \text{GoldenSegment}(m) \oplus \text{Skill}(\hat{a}), \\ \pi(a_t \mid h_t, \Phi) &= \text{LLM}(a_t \mid \text{Prompt}(h_t, \Phi)). \end{aligned} \tag{12}$$

3.4 Theoretical Analysis

In sequential reasoning tasks, prompts often include both decision-relevant and irrelevant segments, whereas uninformative tokens may obscure task-specific signals. Following the latent task mixture framework of Wies et al. [29], we study how selecting a focused subset of high-utility content improves task identification. Assume prompts are drawn from a mixture distribution $\mathcal{D} = \sum_{\phi \in \Phi} \pi(\phi) P_\phi$, where ϕ indexes latent tasks, $\pi(\phi)$ is the prior, and P_ϕ the task-specific sequence distribution. Each input $x_t = (g, h_t)$ includes a goal g and history $h_t = \{(o_0, a_0), \ldots, o_t\}$, and $y_t = a_t$ is the action. We decompose the prompt as $p = p_{\text{focused}} \cup p_{\text{irrelevant}}$, defined relative to the ground-truth task $\phi^\star$: $p_{\text{focused}} := p_{\text{focused}} \mid \phi^\star$, capturing informative segments, and $p_{\text{irrelevant}} := p_{\text{irrelevant}} \mid \phi^\star$, capturing segments with little or misleading task evidence. Under Assumptions A.1–A.3 (see Appendix C), we show that the focused portion of the prompt alone suffices for task identifiability. See Appendix C for full derivation.

Theorem 1 (Task Identifiability)**.** *Let $p \sim P_{\phi^\star}^{\otimes k}$ be a prompt sampled from the true task $\phi^\star \in \Phi$, and suppose it admits a decomposition $p = p_{focused} \cup p_{irrelevant}$, where the partition is defined relative to $\phi^\star$. Suppose that $\min_{\phi \neq \phi^\star} \text{KL}(P_{\phi^\star}(p) \| P_\phi(p)) > 8 \log\left(\frac{1}{c_1 \cdot c_2}\right)$. Then, there exists a sample complexity threshold $\tilde{m}_{\mathcal{D}} : (0,1)^2 \to \mathbb{N}$ such that for any $\epsilon, \delta > 0$, if the number of in-context examples $k \geq \tilde{m}_{\mathcal{D}}(\epsilon, \delta)$, the following holds with probability at least $1 - \delta$ over the sampling of p:*

$$\forall \phi \neq \phi^\star, \quad \frac{P_\phi(p_{focused})}{P_{\phi^\star}(p_{focused})} \leq \frac{P_\phi(p)}{P_{\phi^\star}(p)} < \epsilon. \tag{13}$$

This result suggests that removing irrelevant segments sharpens the contrast between tasks, leading to a more task-aligned prompt with reduced ambiguity.

4 Experimental Setup

Datasets. We conduct experiments on three sequential decision-making datasets: ALFWorld [38], BabyAI [39], and ScienceWorld [40]. These benchmarks span household tasks, grid-based navigation, and scientific reasoning, requiring agents to perform multi-turn interactions to achieve final goals. They cover diverse domains

Table 2: Grounding Rate [%] (↑), Progress Rate [%] (↑), Success Rate [%] (↑), and AUPC [0, 1] (↑) on ALFWorld. The best method for each LLM is in **bold**; the second-best method is underlined.

Method	Qwen2.5-7B-Instruct				Qwen-Turbo				GPT-4o-mini			
	GR	PR	SR	AUPC	GR	PR	SR	AUPC	GR	PR	SR	AUPC
0-shot	10.5	6.0	0.8	0.027	56.3	32.2	9.7	0.212	73.7	26.8	1.5	0.184
1-shot	28.1	16.0	2.2	0.095	63.9	55.3	36.5	0.380	77.3	43.3	10.5	0.292
Leap	27.7	21.2	5.2	0.125	66.3	55.6	37.3	0.386	78.6	50.8	11.2	0.348
Synapse (1-shot)	61.5	41.6	17.1	0.278	74.9	54.7	35.8	0.379	76.3	48.8	14.8	0.340
Synapse (3-shot)	71.4	44.8	19.4	0.302	78.4	60.6	47.0	0.421	77.4	52.9	17.8	0.360
Trad	65.4	44.2	22.4	0.296	65.5	54.8	35.8	0.372	79.1	49.1	16.4	0.341
SkillGen (ours)	**84.9**	**68.0**	**55.2**	**0.464**	**85.9**	**67.6**	**53.8**	**0.460**	**83.6**	**55.1**	**29.8**	**0.369**

Table 3: Grounding Rate [%] (↑), Progress Rate [%] (↑), Success Rate [%] (↑), and AUPC [0, 1] (↑) on BabyAI. The best method for each LLM is in **bold**; the second-best method is underlined.

Method	Qwen2.5-7B-Instruct				Qwen-Turbo				GPT-4o-mini			
	GR	PR	SR	AUPC	GR	PR	SR	AUPC	GR	PR	SR	AUPC
0-shot	31.8	21.8	7.1	0.037	50.2	32.7	19.6	0.092	55.3	34.2	22.3	0.129
1-shot	59.2	36.5	18.8	0.112	61.4	37.2	16.3	0.076	76.6	42.6	28.6	0.154
Leap	66.6	46.3	27.7	0.151	68.4	52.9	38.4	0.206	73.1	43.8	29.4	0.170
Synapse (1-shot)	67.2	39.4	21.4	0.153	65.0	55.8	45.5	0.242	86.5	44.9	33.9	0.169
Synapse (3-shot)	**78.6**	44.6	28.6	**0.163**	62.1	50.8	38.4	0.191	**92.8**	49.5	38.4	0.188
Trad	68.2	36.9	19.7	0.115	64.9	46.9	34.8	0.157	87.3	40.9	30.4	0.126
SkillGen (ours)	66.7	**50.0**	**31.2**	0.158	**73.9**	**59.4**	**45.5**	**0.254**	89.5	**57.6**	**41.1**	**0.248**

and increase in complexity—from ALFWorld to ScienceWorld. We employ four-fold cross-validation, partitioning each dataset into four equal splits, using three folds for training and one for testing, and report results averaged over all folds. Based on the subgoal annotations provided by AgentBoard [3], we compute the subgoal achieved rate to quantify the model's step-wise progress. Notably, we use subgoal labels to construct domain graphs but do not rely on full expert trajectories during training or inference.

Evaluation Metrics. All methods are evaluated using four metrics: Grounding Rate (GR), Progress Rate (PR), Success Rate (SR), and Area Under the Progress Curve (AUPC). GR measures whether the agent's action is valid given the current environment state, reflecting its grounding and understanding capabilities. PR quantifies task advancement based on the proportion of subgoals achieved. SR indicates whether the agent successfully completes the entire task. AUPC captures the cumulative task progress over time, rewarding agents that complete tasks more efficiently.

Baselines. We consider the following baselines: *0-shot* asks the agent to perform the task without any in-context examples. *1-shot* provides a single demonstration trajectory as an in-context example. *Leap* [24] enables the agent to self-revise by identifying and learning from mistakes in provided examples. *Synapse* [22] retrieves and prompts entire expert trajectories from memory based on task meta-data. *Trad* [41] guides inference by retrieving observation-action pairs from past interaction history. All baseline methods are built on the *Act* prompting framework [14], chosen for its simplicity and broad compatibility with instruction-following LLMs. To reduce foundation model bias, we evaluate all baselines using three models: Qwen2.5-7B-Instruct [42], Qwen-Turbo [42], and GPT-4o-mini [43]. We use each language model for both sampling and inference. For more experimental details, please refer to Appendix D.

5 Evaluation Results

SkillGen is evaluated on three decision-making tasks with performance, ablation, and sensitivity studies. Appendix E further reports token-cost comparisons, TD-sampling strategies, and an Act/ReAct inference

Table 4: Grounding Rate [%] (↑), Progress Rate [%] (↑), Success Rate [%] (↑), and AUPC [0, 1] (↑) on ScienceWorld. The best method for each LLM is in **bold**; the second-best method is underlined.

Method	Qwen2.5-7B-Instruct				Qwen-Turbo				GPT-4o-mini			
	GR	PR	SR	AUPC	GR	PR	SR	AUPC	GR	PR	SR	AUPC
0-shot	10.8	27.1	9.0	0.136	**28.8**	19.3	4.4	0.113	34.3	44.3	7.7	0.206
1-shot	9.6	19.1	5.5	0.108	11.8	19.1	7.7	0.111	**34.3**	46.8	18.8	0.294
Leap	8.5	25.8	11.1	0.155	4.4	21.8	6.6	0.124	11.4	51.7	21.1	0.330
Synapse (1-shot)	8.6	15.4	4.4	0.090	5.4	16.0	3.3	0.106	14.0	52.8	25.6	0.334
Synapse (3-shot)	6.0	15.3	5.5	0.086	7.2	24.0	10.1	0.155	15.5	60.0	32.4	0.390
Trad	7.3	21.1	4.4	0.135	7.1	29.3	8.8	0.180	16.9	61.4	29.0	0.375
SkillGen (ours)	**16.1**	**46.7**	**23.4**	**0.298**	13.3	**37.7**	**11.1**	**0.242**	25.3	**67.3**	**40.2**	**0.442**

study. Prompt examples and case studies can be found in Appendices G and H. Our key findings are as follows:

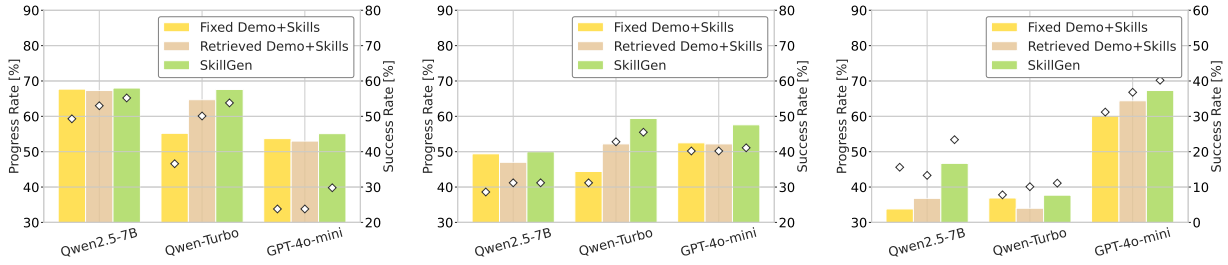
- SkillGen consistently achieves higher PR and SR than baseline methods across all tasks. AUPC measures SkillGen 's efficiency in steady progress beyond final success.
- Step-wise skills and golden segments boost performance, TD-based credit assignment refines skills for better guidance, and subgoal supervision aids decisions via intermediate structure.
- SkillGen demonstrates strong performance across a range of sampling scales and retrieval granularities, with balanced configurations performing consistently well.

5.1 Main Results

ALFWorld. Table 2 presents the comparison of prompting strategies on ALFWorld. While SkillGen achieves the highest GR across all models, the most notable gains appear in PR and SR. On Qwen2.5-7B-Instruct, SkillGen achieves a PR of 68.0 and an SR of 55.2, substantially outperforming the strongest baseline, Synapse (3-shot), which achieves 44.8 (PR) and 19.4 (SR). On Qwen-Turbo, SkillGen reaches 67.6 (PR) and 53.8 (SR), surpassing the second-best Synapse (3-shot) with 60.6 and 47.0, respectively. Similar trends are observed on GPT-4o-mini, where SkillGen boosts SR from 17.8 to 29.8, again outperforming all alternatives. SkillGen outperforms Synapse and Trad by providing more reusable skills, yielding denser step-wise progress, higher subgoal completion, and the best AUPC for efficient task execution.

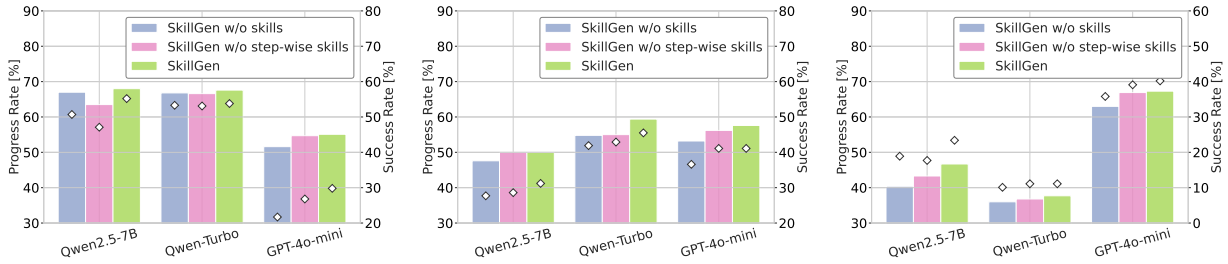
BabyAI. Table 3 shows that SkillGen achieves the highest PR scores across all models—reaching 59.4 on Qwen-Turbo and 57.6 on GPT-4o-mini. For SR, SkillGen shows significant improvements: a +7.0% gain on GPT-4o-mini (41.1 vs. 38.4). On Qwen2.5-7B-Instruct, SkillGen achieves 50.0 (PR) and 31.2 (SR), outperforming Synapse (3-shot) at 44.6 and 28.6, respectively. Compared to *Leap* and *Trad*, SkillGen achieves higher PR and SR—for example, +10.7 SR over *Trad* on GPT-4o-mini—highlighting the advantage of structured skill prompting. These results demonstrate that SkillGen outperforms both insight-summary and trajectory-retrieval baselines in spatially constrained environments.

ScienceWorld. In Table 4, SkillGen achieves a PR of 67.3 and an SR of 40.2 on GPT-4o-mini—a +7.8% improvement in SR over the strongest baseline, Synapse (3-shot), and +11.2% over *Trad*—while also attaining the highest AUPC of 0.442. On Qwen-Turbo, the best baseline is *Trad* with 29.3 (PR) and 8.8 (SR), while SkillGen improves these to 37.7 and 11.1. Notably, naive baselines such as *0-shot* show inflated GR scores by repeatedly issuing generic queries like *"check valid action"*, which accumulate steps but fail to make substantive progress. SkillGen consistently leads in AUPC across models, highlighting its efficiency in making steady progress. These results demonstrate that SkillGen 's structured prompting enables more coherent, goal-directed reasoning and higher task success, even in knowledge-intensive, instruction-heavy scenarios like ScienceWorld.



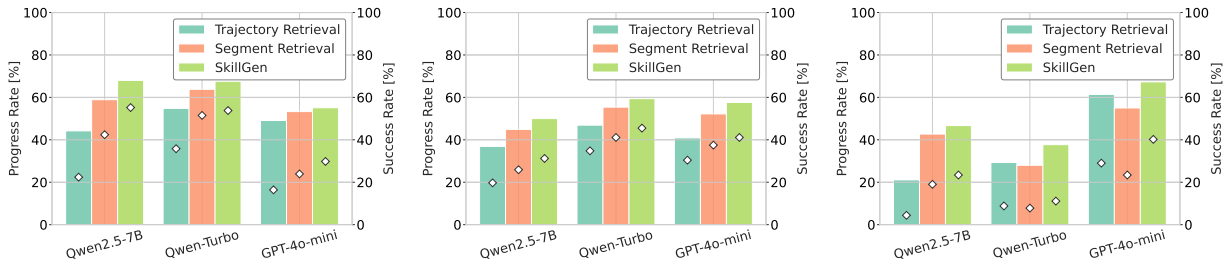
(a) ALFWorld (b) BabyAI (c) ScienceWorld

Figure 3: Ablation study of SkillGen showing the effect of golden segments. Bars represent PR, ◊ markers indicate SR.



(a) ALFWorld (b) BabyAI (c) ScienceWorld

Figure 4: Ablation study of SkillGen showing the effect of step-wise skills. Bars represent PR, ◊ markers indicate SR.



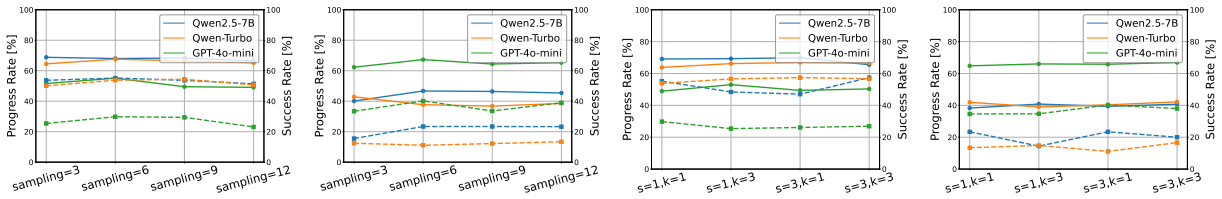
(a) ALFWorld (b) BabyAI (c) ScienceWorld

Figure 5: Ablation study of SkillGen showing the effect of TD-based credit estimation. Bars represent PR, ◊ markers indicate SR.

5.2 Ablation Study

Effect of Golden Segment. To validate the effectiveness of *focused* prompting, we compare three strategies: (i) *Fixed Demo+Skills*, which uses a static, full demonstration augmented with skills; (ii) *Retrieved Demo+Skills*, which retrieves a relevant full trajectory from the training set. In Figure 3, *SkillGen* consistently outperforms the other two strategies across all tasks and model backbones. The largest gain is observed on ALFWorld with Qwen-Turbo, where SkillGen improves SR by +17.2% over Fixed and +3.7% over Retrieved. These results demonstrate that focused, high-impact segments lead to more effective decision-making by eliminating irrelevant context.

Effect of Step-wise Skills. To assess the effect of *granular* prompting, we conduct ablations comparing (i) *SkillGen w/o skills* (golden segment only), (ii) *SkillGen w/o step-wise retrieval* (naive skill injection). In Figure 4, SkillGen consistently achieves the best performance across all datasets and models. It improves



(a) ALFWorld (b) ScienceWorld (c) ALFWorld (d) ScienceWorld

Figure 6: (a) (b): Sensitivity analysis of SkillGen with respect to the number of sampled trajectories per task for training set. (c) (d): Sensitivity analysis of SkillGen with respect to the number of retrieved skills (s) and the number of antecedents (or consequences) per skill (k).

SR over *w/o skills* by +8.1% on ALFWorld and +4.5% on ScienceWorld, highlighting the value of structured skill integration. Step-wise retrieval further boosts SR by +3.0% on ALFWorld and +5.7% on ScienceWorld, showing the benefit of context-aligned guidance. While injecting all domain skills without retrieval yields moderate gains—e.g., +3.6% PR on ScienceWorld. These results underscore that skill utility depends on contextual relevance.

Effect of TD-based Credit Assignment. To assess the effect of TD-based credit assignment, we compare three step-wise retrieval methods: (i) *Trajectory Retrieval* (i.e., *Trad*), which retrieves step-wise observation–action pairs; (ii) *Segment Retrieval*, which retrieves action-only segments without applying credit assignment, using the original sparse progress signal. In Figure 5, SkillGen consistently achieves the highest performance, followed by *Segment Retrieval* with moderate gains. In ALFWorld, SkillGen achieves a SR of 55.2 (Qwen2.5-7B-Instruct), outperforming *Segment Retrieval* by +12.8 SR. On the more compositional ScienceWorld, SkillGen yields substantial gains, achieving a SR of 40.2 on GPT-4o-mini—a 16.8% increase over *Segment Retrieval*. These results demonstrate that TD-based credit assignment is crucial for extracting decision-critical skills.

Effect of Subgoal Annotations. Table 5 presents the results of SkillGen without subgoal annotations, using only fully successful trajectories for skill extraction (see Appendix E Table 12 for the complete results). Subgoal supervision consistently improves performance, particularly on complex tasks such as ScienceWorld, where the SR increases from 32.4% to 40.2% on GPT-4o-mini. Nonetheless, even without subgoal labels, SkillGen outperforms strong baselines. For instance, on ALFWorld with Qwen2.5-7B-Instruct, SkillGen raises the success rate from 19.4% (*Synapse 3-shot*) to 50.1%, while on ScienceWorld, it improves PR from 29.1% to 46.7%. These results demonstrate that SkillGen 's interaction-driven skill extraction enables effective generalization across domains, regardless of the presence of explicit subgoal supervision.

Table 5: Results of SkillGen w/o subgoal annotations.

Dataset	Qwen2.5-7B		Qwen-Turbo		GPT-4o-mini	
	PR	SR	PR	SR	PR	SR
ALFWorld	63.2	50.1	64.6	52.2	32.9	14.2
BabyAI	43.2	32.1	50.5	36.6	56.8	41.1
ScienceWorld	29.1	18.9	27.7	10.0	56.2	32.4

5.3 Parameter Sensitive Analysis

Sampling Scale. To assess SkillGen's robustness to the sampling scale, we perform sensitivity analysis with varying numbers of sampled trajectories per task. In Figure 6 (a) (b), performance remains stable across settings. The minimal configuration (*sampling=3*) already achieves strong SR—e.g., 53.7% on ALFWorld

(Qwen2.5-7B-Instruct). While increasing to *sampling=6* often yields further gains, larger values (*sampling=9,12*) may introduce noise, occasionally leading to slight drops. This suggests that a small, diverse set of trajectories suffices to extract effective skills, supporting robust performance without exhaustive sampling.

Skill Retrieval. As shown in Figure 6 (c) (d), we evaluate SkillGen under varying retrieval settings. The minimal setting (s=1, k=1) already yields strong SR—e.g., 55.2% on ALFWorld (Qwen2.5-7B-Instruct), 34.6% on ScienceWorld (GPT-4o-mini). Increasing either skill count (k) or action scope (s) improves performance in complex tasks, with GPT-4o-mini reaching 40.2% SR under (s=1, k=3). In contrast, diversity-focused settings like (s=3, k=1) help on simpler tasks, but degrade on harder ones (e.g., 14.4% on ScienceWorld with Qwen2.5-7B-Instruct). These results highlight SkillGen's robustness and the need to align retrieval granularity with task complexity.

6 Related Work

Sequential Decision Making with LLMs. Recent advances in LLM-based decision-making have led to interactive agents that operate in multi-turn loops, either selecting actions directly or reasoning before acting [14, 44, 45]. To address long-horizon tasks, many methods incorporate feedback-driven refinement [15, 46, 16, 47] or structured search [4, 48, 49, 50, 17, 51]. Another line of work improves inference by retrieving expert or history information from offline interactions [22, 52, 53, 41]. More recently, self-improving agents construct in-context examples from prior episodes, enabling generalization to unseen tasks without relying on expert demonstrations [54, 55]. These methods highlight a growing emphasis on experience-driven decision-making.

Knowledge-Augmented In-Context Learning. Knowledge augmented methods aim to enrich the prompt with structured information—such as relational or procedural knowledge, to provide stronger inductive bias and support more accurate reasoning. Some approaches guide reasoning with high-level prompts derived from prior interactions [56, 57, 58, 59]. Others inject retrieved graph-based knowledge to support multi-hop inference [60, 61, 62]. LLMs can also self-synthesize reusable strategies from world models for generalization [63, 64, 65, 66], or incorporate procedural knowledge via rule induction [67, 68] and skill reuse [34, 69, 70, 25]. These approaches enrich in-context learning by integrating external or derived task-specific knowledge as decision support.

7 Conclusion

In this study, we explore how to improve in-context learning for sequential decision-making by tackling three key challenges: maintaining decision focus, offering granular guidance, and reducing dependence on expert supervision. To address these issues, we propose SkillGen, a skill-aware prompting framework that leverages structured knowledge and weak supervision to enable fine-grained, context-sensitive reasoning. Our theoretical analysis highlights how decision-critical content supports task identifiability, motivating more principled prompt design. Experiments on ALFWorld, BabyAI, and ScienceWorld show that SkillGen delivers consistent gains without expert demonstrations. We believe our findings offer a foundation for future research on skill-aware prompting to improve generalization, efficiency, and coherence in LLM-based decision-making.

Acknowledgments

Ruomeng Ding, Wei Cheng, and Chen Zhao did not receive any financial support for this work, and Wei Cheng and Chen Zhao contributed only by developing the research ideas, participating in discussions, and providing feedback on the manuscript.

References

[1] Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Erran Li Li, Ruohan Zhang, et al. Embodied agent interface: Benchmarking llms for embodied decision making. *Advances in Neural Information Processing Systems*, 37:100428–100534, 2024.

[2] Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, et al. Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. *arXiv preprint arXiv:2502.09560*, 2025.

[3] Ma Chang, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. Agentboard: An analytical evaluation board of multi-turn llm agents. *Advances in Neural Information Processing Systems*, 37:74325–74362, 2024.

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

[5] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *The Twelfth International Conference on Learning Representations*, 2024.

[6] Martin Klissarov, Devon Hjelm, Alexander Toshev, and Bogdan Mazoure. On the modeling capabilities of large language models for sequential decision making. *arXiv preprint arXiv:2410.05656*, 2024.

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

[8] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*, pages 1–14, 2024.

[9] Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. Fireact: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*, 2023.

[10] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *The Twelfth International Conference on Learning Representations*, 2024.

[11] Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms. In *Findings of the Association for Computational Linguistics*, pages 3053–3077, 2024.

[12] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[13] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[14] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, pages 1–14, 2023.

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

[16] Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. Adaplanner: Adaptive planning from feedback with language models. *Advances in neural information processing systems*, 36:58202–58245, 2023.

[17] Shirley Wu, Shiyu Zhao, Qian Huang, Kexin Huang, Michihiro Yasunaga, Kaidi Cao, Vassilis Ioannidis, Karthik Subbian, Jure Leskovec, and James Y Zou. Avatar: Optimizing llm agents for tool usage via contrastive reasoning. *Advances in Neural Information Processing Systems*, 37:25981–26010, 2025.

[18] Jiachang Liu, Dinghan Shen, Yizhe Zhang, William B Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for gpt-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, 2022.

[19] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*, 2023.

[20] Yiming Zhang, Shi Feng, and Chenhao Tan. Active example selection for in-context learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9134–9148, 2022.

[21] Barys Liskavets, Maxim Ushakov, Shuvendu Roy, Mark Klibanov, Ali Etemad, and Shane K Luke. Prompt compression with context-aware sentence encoding for fast and improved llm inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24595–24604, 2025.

[22] Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *The Twelfth International Conference on Learning Representations*, 2024.

[23] Ruiwen Zhou, Yingxuan Yang, Muning Wen, Ying Wen, Wenhao Wang, Chunling Xi, Guoqiang Xu, Yong Yu, and Weinan Zhang. Trad: Enhancing llm agents with step-wise thought retrieval and aligned decision. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–13, 2024.

[24] Tianjun Zhang, Aman Madaan, Luyu Gao, Steven Zheng, Swaroop Mishra, Yiming Yang, Niket Tandon, and Uri Alon. In-context principle learning from mistakes. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[25] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19632–19642, 2024.

[26] Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*, 2022.

[27] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, 2022.

[28] Fabian Falck, Ziyu Wang, and Chris Holmes. Is in-context learning in large language models bayesian? a martingale perspective. In *Proceedings of the 41st International Conference on Machine Learning*, pages 12784–12805, 2024.

[29] Noam Wies, Yoav Levine, and Amnon Shashua. The learnability of in-context learning. *Advances in Neural Information Processing Systems*, 36:36637–36651, 2023.

[30] Jianliang He, Siyu Chen, Fengzhuo Zhang, and Zhuoran Yang. From words to actions: Unveiling the theoretical underpinnings of llm-driven autonomous systems. In *International Conference on Machine Learning*, pages 17807–17841. PMLR, 2024.

[31] Lingfeng Sun, Devesh K Jha, Chiori Hori, Siddarth Jain, Radu Corcodel, Xinghao Zhu, Masayoshi Tomizuka, and Diego Romeres. Interactive planning using large language models for partially observable robotic tasks. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14054–14061. IEEE, 2024.

[32] Ayush Jain, Norio Kosaka, Kyung-Min Kim, and Joseph J Lim. Know your action set: Learning action relations for reinforcement learning. In *International Conference on Learning Representations*, 2022.

[33] Yan Ding, Xiaohan Zhang, Saeid Amiri, Nieqing Cao, Hao Yang, Andy Kaminski, Chad Esselink, and Shiqi Zhang. Integrating action knowledge and llms for task planning and situation handling in open worlds. *Auton. Robots*, 47(8):981–997, August 2023.

[34] Yuqi Zhu, Shuofei Qiao, Yixin Ou, Shumin Deng, Shiwei Lyu, Yue Shen, Lei Liang, Jinjie Gu, Huajun Chen, and Ningyu Zhang. KnowAgent: Knowledge-augmented planning for LLM-based agents. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 3709–3732. Association for Computational Linguistics, 2025.

[35] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International conference on machine learning*, pages 104–114. PMLR, 2020.

[36] Thomas Mesnard, Theophane Weber, Fabio Viola, Shantanu Thakoor, Alaa Saade, Anna Harutyunyan, Will Dabney, Thomas S Stepleton, Nicolas Heess, Arthur Guez, et al. Counterfactual credit assignment in model-free reinforcement learning. In *International Conference on Machine Learning*, pages 7654–7664. PMLR, 2021.

[37] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44, 1988.

[38] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, pages 1–14, 2021.

[39] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: First steps towards grounded language learning with a human in the loop. In *International Conference on Learning Representations*, volume 105, pages 1–14. New Orleans, LA, 2019.

[40] Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11279–11298. Association for Computational Linguistics, 2022.

[41] Ruiwen Zhou, Yingxuan Yang, Muning Wen, Ying Wen, Wenhao Wang, Chunling Xi, Guoqiang Xu, Yong Yu, and Weinan Zhang. Trad: Enhancing llm agents with step-wise thought retrieval and aligned decision. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–13, 2024.

[42] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[43] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[44] Nikolai Rozanov and Marek Rei. Stateact: Enhancing llm base agents via self-prompting and state-tracking, 2025.

[45] Qiwei Zhao, Dong Li, Yanchi Liu, Wei Cheng, Yiyou Sun, Mika Oishi, Takao Osaki, Katsushi Matsuda, Huaxiu Yao, Chen Zhao, Haifeng Chen, and Xujiang Zhao. Uncertainty propagation on LLM agent. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6064–6073, Vienna, Austria, July 2025. Association for Computational Linguistics.

[46] Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, Yitao Liang, and Team CraftJarvis. Describe, explain, plan and select: interactive planning with large language models enables open-world multi-task agents. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 34153–34189, 2023.

[47] Liyi Chen, Panrong Tong, Zhongming Jin, Ying Sun, Jieping Ye, and Hui Xiong. Plan-on-graph: Self-correcting adaptive planning of large language model on knowledge graphs. *Advances in Neural Information Processing Systems*, 37:37665–37691, 2024.

[48] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, number 16, pages 17682–17690, 2024.

[49] Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 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. Association for Computational Linguistics, 2023.

[50] Yuchen Zhuang, Xiang Chen, Tong Yu, Saayan Mitra, Victor Bursztyn, Ryan A. Rossi, Somdeb Sarkhel, and Chao Zhang. Toolchain*: Efficient action space navigation in large language models with a* search. In *The Twelfth International Conference on Learning Representations*, pages 1–14, 2024.

[51] Dong Li, Xujiang Zhao, Linlin Yu, Yanchi Liu, Wei Cheng, Zhengzhang Chen, Zhong Chen, Feng Chen, Chen Zhao, and Haifeng Chen. Solverllm: Leveraging test-time scaling for optimization problem via llm-guided search. *arXiv preprint arXiv:2510.16916*, 2025.

[52] Yuchen Xiao, Yanchao Sun, Mengda Xu, Udari Madhushani, Jared Vann, Deepeka Garg, and Sumitra Ganesh. O3d: Offline data-driven discovery and distillation for sequential decision-making with large language models. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*, 2023.

[53] Tomoyuki Kagaya, Thong Jing Yuan, Yuxuan Lou, Jayashree Karlekar, Sugiri Pranata, Akira Kinose, Koki Oguri, Felix Wick, and Yang You. Rap: Retrieval-augmented planning with contextual memory for multimodal llm agents. In *NeurIPS 2024 Workshop on Open-World Agents*, 2024.

[54] Vishnu Sarukkai, Zhiqiang Xie, and Kayvon Fatahalian. Self-generated in-context examples improve llm agents for sequential decision-making tasks. *arXiv preprint arXiv:2505.00234*, 2025.

[55] Yitao Liu, Chenglei Si, Karthik Narasimhan, and Shunyu Yao. Contextual experience replay for self-improvement of language agents. *arXiv preprint arXiv:2506.06698*, 2025.

[56] Geunwoo Kim, Pierre Baldi, and Stephen McAleer. Language models can solve computer tasks. *Advances in Neural Information Processing Systems*, 36:39648–39677, 2023.

[57] Tianjun Zhang, Aman Madaan, Luyu Gao, Steven Zhang, Swaroop Mishra, Yiming Yang, Niket Tandon, and Uri Alon. In-context principle learning from mistakes. In *ICML 2024 Workshop on In-Context Learning*, 2024.

[58] Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. Autoguide: Automated generation and selection of state-aware guidelines for large language model agents. *arXiv e-prints*, pages arXiv–2403, 2024.

[59] Mingze Kong, Zhiyong Wang, Yao Shu, and Zhongxiang Dai. Meta-prompt optimization for llm-based sequential decision making. *arXiv preprint arXiv:2502.00728*, 2025.

[60] Costas Mavromatis and George Karypis. Gnn-rag: Graph neural retrieval for large language model reasoning. *arXiv preprint arXiv:2405.20139*, 2024.

[61] Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. Grag: Graph retrieval-augmented generation. *arXiv preprint arXiv:2405.16506*, 2024.

[62] Linhao Luo, Zicheng Zhao, Chen Gong, Gholamreza Haffari, and Shirui Pan. Graph-constrained reasoning: Faithful reasoning on knowledge graphs with large language models. *arXiv preprint arXiv:2410.13080*, 2024.

[63] Ruomeng Ding, Chaoyun Zhang, Lu Wang, Yong Xu, Minghua Ma, Wei Zhang, Si Qin, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. Everything of thoughts: Defying the law of penrose triangle for thought generation. In *ACL (Findings)*, 2024.

[64] Wangchunshu Zhou, Yuchen Eleanor Jiang, Long Li, Jialong Wu, Tiannan Wang, Shuai Wang, Jiamin Chen, Jintian Zhang, Jing Chen, Xiangru Tang, et al. Agents: An open-source framework for autonomous language agents. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024.

[65] Shuofei Qiao, Ningyu Zhang, Runnan Fang, Yujie Luo, Wangchunshu Zhou, Yuchen Jiang, Chengfei Lv, and Huajun Chen. Autoact: Automatic agent learning from scratch for qa via self-planning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 3003–3021, 2024.

[66] Shuofei Qiao, Runnan Fang, Ningyu Zhang, Yuqi Zhu, Xiang Chen, Shumin Deng, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Agent planning with world knowledge model. *Advances in Neural Information Processing Systems*, 37:114843–114871, 2024.

[67] Zhaocheng Zhu, Yuan Xue, Xinyun Chen, Denny Zhou, Jian Tang, Dale Schuurmans, and Hanjun Dai. Large language models can learn rules. *arXiv preprint arXiv:2310.07064*, 2023.

[68] Yudi Zhang, Pei Xiao, Lu Wang, Chaoyun Zhang, Meng Fang, Yali Du, Yevgeniy Puzyrev, Randolph Yao, Si Qin, Qingwei Lin, Mykola Pechenizkiy, Dongmei Zhang, Saravan Rajmohan, and Qi Zhang. RuAG: Learned-rule-augmented generation for large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[69] Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. Automanual: Constructing instruction manuals by llm agents via interactive environmental learning. *Advances in Neural Information Processing Systems*, 37:589–631, 2024.

[70] Jesse Zhang, Jiahui Zhang, Karl Pertsch, Ziyi Liu, Xiang Ren, Minsuk Chang, Shao-Hua Sun, and Joseph J Lim. Bootstrap your own skills: Learning to solve new tasks with large language model guidance. In *Conference on Robot Learning*, pages 302–325. PMLR, 2023.

[71] Yash Chandak, Georgios Theocharous, James Kostas, Scott Jordan, and Philip S. Thomas. Learning action representations for reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 941–950. PMLR, 2019.

[72] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992. Association for Computational Linguistics, November 2019.

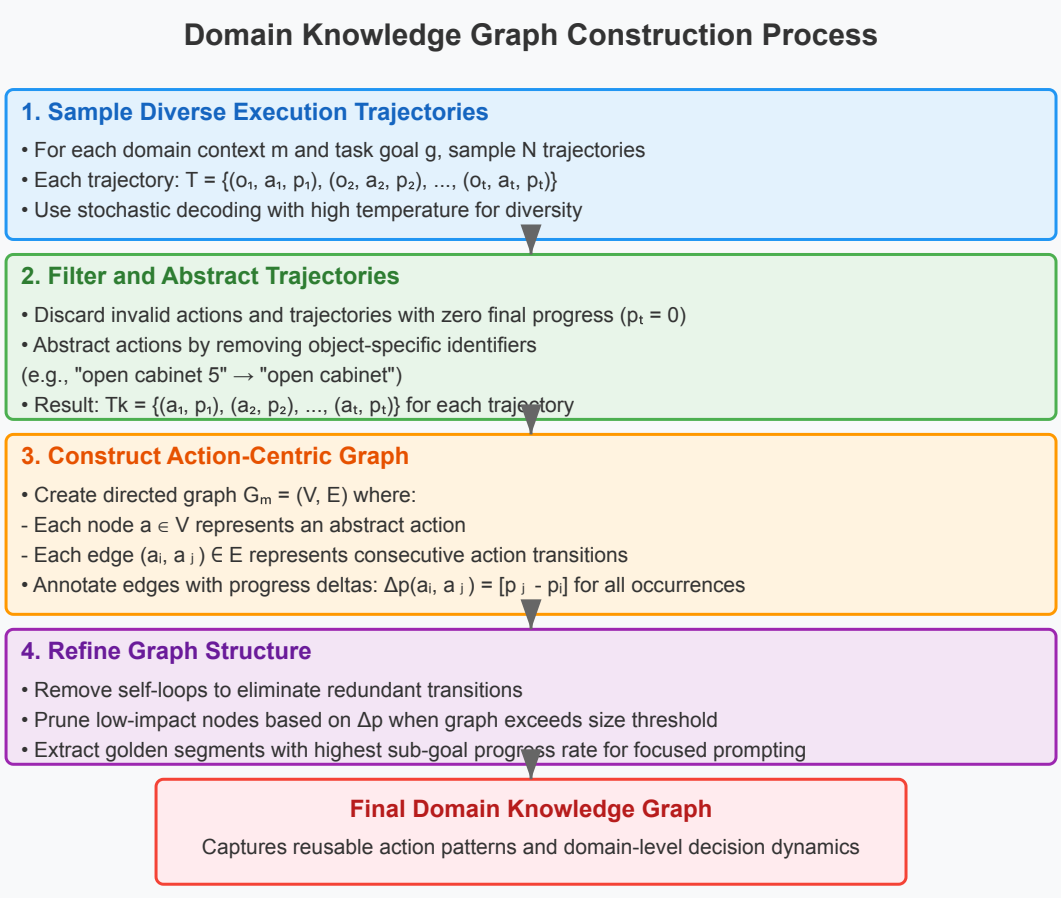


Figure 7: Pipeline of Domain Graph Construction

A Limitations

While SkillGen demonstrates strong performance across diverse tasks and generalizes across both proprietary and open-source LLMs, several limitations remain. SkillGen relies on sampled trajectories with subgoal or progress feedback, which may be unavailable in domains lacking structured intermediate signals. A common workaround is to filter for fully successful trajectories, but this depends on sufficient high-quality samples. Increasing the sampling scale may help, but often yields diminishing returns in quality and incurs high computational cost, while still being constrained by the base model's capabilities. Additionally, constructing domain-level action graphs introduces preprocessing overhead and may not scale well in extensive or dynamic action spaces. Retrieved skills, grounded in local action contexts, may also struggle to generalize to out-of-distribution scenarios or novel task configurations. These limitations do not undermine our core contributions, but point to future directions for enhancing the scalability and robustness of SkillGen.

Algorithm A.1: TD-Based Action Credit Assignment on Refined Domain Knowledge Graph

Require: Refined domain graph $\mathbb{G}_m = (\mathbb{V}, \mathbb{E})$ with progress annotations $\mathcal{P}_\Delta(a_i, a_j)$; TD iterations N; batch size B; learning rate α; discount factor γ; trace decay λ

Ensure: Normalized action credit scores $\bar{Q}(a)$ for all $a \in \mathbb{V}$

$Q(a) \leftarrow 0$, $E(a) \leftarrow 0$ for all $a \in \mathbb{V}$ ▷ Initialize Q-values and eligibility traces for all actions

for $i = 1$ to N **do**

  $\{\tau^{(1)}, \ldots, \tau^{(B)}\} \sim \mathcal{S}^B_{\text{path}}(a_{\text{start}}, a_{\text{end}}, \mathbb{G}_m)$ ▷ Sample a batch of B paths from a_{start} to a_{end}

  for each path $\tau = [a_0, a_1, \ldots, a_T]$ in the batch **do**

    for $t = 0$ to $T - 1$ **do** ▷ Sample noisy reward from empirical progress delta

      if $\mathcal{P}_\Delta(a_t, a_{t+1}) \neq \emptyset$ **then**

        $r_t \sim \text{Uniform}(\mathcal{P}_\Delta(a_t, a_{t+1})) + \mathcal{N}(0, \sigma^2)$

      else

        $r_t \sim \mathcal{N}(0, \sigma^2)$

      end if

      $\delta_t \leftarrow r_t + \gamma Q(a_{t+1}) - Q(a_t)$ ▷ Compute TD error and update eligibility traces

      $E(a_t) \leftarrow E(a_t) + 1$

      for each $a \in \mathbb{V}$ such that $E(a) > 0$ **do**

        $Q(a) \leftarrow Q(a) + \alpha \cdot \delta_t \cdot E(a)$

        $E(a) \leftarrow \gamma \cdot \lambda \cdot E(a)$

      end for

    end for

  end for

end for

$\bar{Q}(a) \leftarrow \dfrac{\max(Q(a), 0)}{\sum_{a' \in \mathbb{V}} \max(Q(a'), 0)}$ for all $a \in \mathbb{V}$ ▷ Normalize Q to obtain action credit scores

return $\{\bar{Q}(a)\}_{a \in \mathbb{V}}$

B Additional Methodological Details

B.1 Domain Knowledge Graph Construction

In Figure 7, we illustrate the pipeline for constructing the domain knowledge graph, including four stages: (1) sampling diverse execution trajectories, (2) filtering and abstracting actions, (3) constructing an action-centric graph, and (4) refining the graph structure. After obtaining the final domain knowledge graph, we use it to perform TD-based action credit assignment, which enables the model to learn utility-aware skill representations for downstream decision-making tasks.

B.2 TD-based Credit Assignment

In Algorithm A.1, we introduce a TD(λ)-based credit assignment procedure over a domain-level action graph. The algorithm repeatedly samples action paths, estimates noisy progress-based rewards, and applies temporal-difference learning with eligibility traces to assign utility-aware scores to abstract actions. Unlike standard TD(λ), which resets traces at the start of each episode, we persist action-level traces across sampled paths to reinforce globally useful action patterns. To prevent unbounded accumulation, we apply exponential decay at each step via $E(a) \leftarrow \gamma \cdot \lambda \cdot E(a)$. This design draws inspiration from prior work on learning shared action values [71], and enables the definition of a state-agnostic action-value function $Q(a)$ that captures the expected long-term utility of each action.

Algorithm A.2: Skill-Based ICL for Sequential Decision Making

Require: Domain skill graph $\mathbb{G}_m$, golden segment Φ_{gold}, frozen LLM, pretrained retriever $\text{Retrieve}(\cdot)$
Ensure: Action sequence $\{a_1, a_2, \ldots, a_T\}$ generated via prompt-based inference

Initialize empty trajectory history $h_0 \leftarrow \emptyset$
for $t = 1$ to T **do**
  Observe current transition history $h_t = \{(o_0, a_0), \ldots, (o_{t-1}, a_{t-1}), o_t\}$
  $\hat{\mathcal{A}} \leftarrow \text{Retrieve}(a_{t-1})$ from $\mathbb{G}_m$ ▷ Top-k relevant actions based on a_{t-1}
  $\Phi_{\text{skill}} \leftarrow \bigoplus_{a \in \hat{\mathcal{A}}} \text{Skill}(a)$ ▷ Concatenate retrieved skills
  $\Phi \leftarrow \Phi_{\text{gold}} \oplus \Phi_{\text{skill}}$ ▷ Assemble multi-level contextual knowledge
  $\text{Prompt}_t \leftarrow \text{Prompt}(h_t, \Phi)$ ▷ Format prompt with history and retrieved context
  $a_t \sim \text{LLM}(a \mid \text{Prompt}_t)$ ▷ Sample next action from LLM
end for
return Action sequence $\{a_1, a_2, \ldots, a_T\}$

B.3 Skill-Based In-Context Learning

In Algorithm A.2, we present a skill-based prompting strategy that guides a frozen LLM using both offline domain knowledge and online interaction history. At each time step, the agent retrieves relevant step-wise skills based on the most recent action, combines them with a domain-level golden segment, and assembles a structured prompt that integrates retrieved knowledge with the current trajectory context. The resulting prompt, serialized in natural language, enables credit-guided, context-aware reasoning without any parameter updates to the model.

C Theoretical Analysis

We adopt the latent task mixture framework of Wies et al. [29], where prompts are sampled from a mixture distribution $\mathcal{D} = \sum_{\phi \in \Phi} \pi(\phi) P_\phi$, with $\phi \in \Phi$ indexing latent tasks, $\pi(\phi)$ the prior, and P_ϕ the task-specific sequence distribution. The following assumptions are used in our analysis.

Assumption A.1 (Approximate Step-wise Independence)**.** *There exists a constant $0 < c_1 \leq 1$ such that for any two strings $s_1, s_2 \in \Sigma^\star$ and any concept ϕ, we have:*

$$c_1 \leq \frac{P_\phi(s_1 \oplus \text{"}\backslash n\text{"}) \cdot P_\phi(s_2)}{P_\phi(s_1 \oplus \text{"}\backslash n\text{"} \oplus s_2)} \leq \frac{1}{c_1}. \tag{14}$$

This assumption implies that the prompt likelihood approximately factorizes over segments: $P_\phi(p) \approx \prod_{t=1}^{T} P_\phi(x_t, y_t)$, up to a bounded multiplicative deviation, due to the string-level composition bounds imposed on concatenated segments.

Assumption A.2 (Global Token Probability Lower Bound)**.** *There exists a constant $c_2 > 0$ such that for all $\phi \in \Phi$, all inputs x_t, and all output tokens y_t, the conditional probability is bounded below:*

$$P_\phi(y_t \mid x_t) > c_2. \tag{15}$$

This assumption ensures that the KL divergence between task distributions is finite and avoids vanishing likelihoods.

Assumption A.3 (Uniform Residual Token Bias)**.** *For all $\phi \neq \phi^\star$ and all $(x_t, y_t) \in p_{\textit{irrelevant}}$, we have:*

$$P_\phi(y_t \mid x_t) \geq P_{\phi^\star}(y_t \mid x_t). \tag{16}$$

This assumption states that irrelevant segments are at least as likely to appear under alternative tasks as under the true task. In other words, they are no more informative about the true task than about any other, and therefore provide no useful signal for identifying the correct task.

Lemma 1 (Task Likelihood Separation [29, Lemma 1])**.** *Let $\mathcal{D} = \sum_{\phi \in \Phi} \pi(\phi) P_\phi$ be a task mixture distribution satisfying Assumptions 1 and 2 . Let $\phi^\star \in \Phi$ be the ground-truth task. Then, there exists a sample complexity threshold $\tilde{m}_\mathcal{D} : (0,1)^2 \to \mathbb{N}$ such that for any $\epsilon, \delta > 0$, if the number of in-context examples $k \geq \tilde{m}_\mathcal{D}(\epsilon, \delta)$, we have:*

$$\mathbb{P}_{p \sim P_{\phi^\star}^{\otimes k}} \left[\forall \phi \neq \phi^\star,\ \frac{P_\phi(p)}{P_{\phi^\star}(p)} < \epsilon \right] \geq 1 - \delta. \tag{17}$$

Lemma 1 guarantees task separability using the full prompt, we now show that task identifiability can be preserved using only the focused portion of the prompt.

Proof of Theorem 1

Proof. By Lemma 1, under Assumptions 1 and 2, there exists a sample complexity threshold $\tilde{m}_\mathcal{D}(\epsilon, \delta)$ such that for any $k \geq \tilde{m}_\mathcal{D}(\epsilon, \delta)$, with probability at least $1 - \delta$ (over the sampling of $p \sim P_{\phi^\star}^{\otimes k}$), we have:

$$\forall \phi \neq \phi^\star, \quad \frac{P_\phi(p)}{P_{\phi^\star}(p)} < \epsilon. \tag{18}$$

Since the prompt decomposes as $p = p_{\text{focused}} \cup p_{\text{irrelevant}}$, we can factor the likelihood ratio as:

$$\frac{P_\phi(p)}{P_{\phi^\star}(p)} = \frac{P_\phi(p_{\text{focused}})}{P_{\phi^\star}(p_{\text{focused}})} \cdot \frac{P_\phi(p_{\text{irrelevant}})}{P_{\phi^\star}(p_{\text{irrelevant}})}. \tag{19}$$

By Assumption 3, we have that for all $\phi \neq \phi^\star$ and for each $(x_t, y_t) \in p_{\text{irrelevant}}$,

$$P_\phi(y_t \mid x_t) \geq P_{\phi^\star}(y_t \mid x_t). \tag{20}$$

Assuming conditional independence across segments within $p_{\text{irrelevant}}$, we can write:

$$P_\phi(p_{\text{irrelevant}}) = \prod_{(x_t, y_t) \in p_{\text{irrelevant}}} P_\phi(y_t \mid x_t), \quad P_{\phi^\star}(p_{\text{irrelevant}}) = \prod_{(x_t, y_t) \in p_{\text{irrelevant}}} P_{\phi^\star}(y_t \mid x_t). \tag{21}$$

Combining these, we obtain:

$$P_\phi(p_{\text{irrelevant}}) \geq P_{\phi^\star}(p_{\text{irrelevant}}), \quad \Rightarrow \quad \frac{P_\phi(p_{\text{irrelevant}})}{P_{\phi^\star}(p_{\text{irrelevant}})} \geq 1. \tag{22}$$

This shows that the irrelevant portion of the prompt weakly favors alternative tasks or, at best, contributes no additional evidence toward the identification of $\phi^\star$. Next, we recall that the full prompt likelihood factorizes over the focused and irrelevant parts:

$$P_\phi(p) = P_\phi(p_{\text{focused}}) \cdot P_\phi(p_{\text{irrelevant}}), \quad P_{\phi^\star}(p) = P_{\phi^\star}(p_{\text{focused}}) \cdot P_{\phi^\star}(p_{\text{irrelevant}}). \tag{23}$$

Taking the ratio, we have:

$$\frac{P_\phi(p)}{P_{\phi^\star}(p)} = \frac{P_\phi(p_{\text{focused}})}{P_{\phi^\star}(p_{\text{focused}})} \cdot \frac{P_\phi(p_{\text{irrelevant}})}{P_{\phi^\star}(p_{\text{irrelevant}})}. \tag{24}$$

Table 6: Task Datasets

Dataset	Task Type	# Environment	# Turns	Action Space	# Subgoal Count
ALFWorld	Household Tasks	134	6	13	3
BabyAI	Grid-based Navigation	112	10	8	4
ScienceWorld	Scientific Reasoning	90	15	21	5

From the inequality above, we know:

$$\frac{P_\phi(p_{\text{irrelevant}})}{P_{\phi^\star}(p_{\text{irrelevant}})} \geq 1, \tag{25}$$

which implies:

$$\frac{P_\phi(p)}{P_{\phi^\star}(p)} \geq \frac{P_\phi(p_{\text{focused}})}{P_{\phi^\star}(p_{\text{focused}})}. \tag{26}$$

Hence, for all $\phi \neq \phi^\star$, we conclude:

$$\frac{P_\phi(p_{\text{focused}})}{P_{\phi^\star}(p_{\text{focused}})} \leq \frac{P_\phi(p)}{P_{\phi^\star}(p)} < \epsilon. \tag{27}$$

□

D Experimental Setup

D.1 Dataset

As shown in Table 6, we evaluate our method on three sequential decision-making benchmarks of increasing complexity: ALFWorld, BabyAI, and ScienceWorld. These datasets span diverse domains—ranging from household tasks to scientific experiments—and vary in interaction length, action space size, and subgoal structure, presenting a progressively harder challenge from ALFWorld to ScienceWorld. To enable robust and fair evaluation across diverse sequential decision-making domains, we adopt a 4-fold cross-validation protocol for all three datasets. For each dataset, the full set of task instances is randomly partitioned into four equal subsets, with shuffling enabled and a fixed random seed (42) to ensure reproducibility. In each fold, one subset is held out as the test set while the remaining three subsets are used for training. This procedure is repeated four times, allowing every task instance to serve as a test sample exactly once. This cross-validation scheme allows us to systematically assess the generalization capability of our method under varying task compositions and minimizes the risk of overfitting to a particular subset of tasks.

ALFWorld. It is an embodied task in simulated household environments, testing agents on exploration, object manipulation, and commonsense reasoning through text-based interaction. It contains 134 tasks across various home settings (e.g., kitchen, bedroom, bathroom), involving objectives like *"put two bars of soap into the trash can"* or *"place the clean egg into the microwave"*. The agent interacts with the environment using natural language actions such as "open the fridge" or "take the egg", and must interpret textual feedback to make decisions. On average, tasks require 6 interaction steps, with an action space of 13 and an average context length of approximately 900 tokens. In addition to the high-level goal, each task is annotated with a sequence of subgoals that define intermediate progress checkpoints. These subgoals correspond to specific, verifiable behaviors—such as locating an object, picking it up, or transforming its state (e.g., cleaning or cooling)—and serve as weak supervision signals during training. For example, the goal *"put a clean soapbar*

in countertop" is decomposed into subgoals: (1) observe a soapbar, (2) pick it up, and (3) clean it. These subgoals are typically specified using regular expressions or fixed language templates matched against textual feedback from the environment.

BabyAI. It evaluates agents in a 20×20 partially observable grid world, focusing on navigation and object interaction. The dataset includes 120 tasks, where agents perceive only their local surroundings and must act based on partial information. Observations are rendered as text (e.g., *"a wall is two steps ahead"*), and actions are also issued as textual commands (e.g., *"go to red ball 1"*, *"open and go through green locked door 1"*). Tasks typically span 10 steps, with an action space of 8 and an average context length of around 1800 tokens—longer than ALFWorld but shorter than ScienceWorld—suggesting a moderate level of language and planning complexity. Each task is defined by a high-level goal (e.g., *"go to the red ball"*), along with a set of subgoals that encode verifiable intermediate signals. These subgoals reflect localized perceptual conditions or agent proximity to targets, such as observing an object or aligning with its position. For instance, a task like *"go to the red ball"* might include subgoals like (1) "There is a red ball " and (2) "There is a red ball right in front of you 1 steps away." These expressions are matched against observation strings during execution, providing weak supervision that allows for fine-grained progress tracking and temporal credit assignment.

ScienceWorld. It provides a rich environment for scientific reasoning, consisting of 90 tasks across 8 rooms (e.g., lab, kitchen, workshop), each equipped with domain-specific tools like thermometers, beakers, and heaters. Tasks involve following experimental procedures and making inferences based on observations, such as *"measure the melting point of orange juice"* or *"observe microbial structures"*. Agents must gather information through interaction and plan a coherent sequence of scientific steps. Tasks average 15 interaction steps, with an action space of 21 and a significantly longer context length of up to 2800 tokens, reflecting the increased complexity of reasoning and language understanding required. Each task is annotated with a sequence of subgoals, which represent semantically intermediate progress. For instance, a task like *"measure the melting point of gallium"* might involve the following subgoals: (1) move the thermometer to the inventory, (2) transfer gallium to a container, (3) apply heat using the stove, (4) wait until a temperature threshold is reached, and (5) focus on the correct output box based on the result. These subgoals serve as weak supervision for learning scientific reasoning progression.

D.2 Evaluation Metrics

Across all experiments in this paper, we evaluate each method on the test set using the metrics below:

Grounding Rate. Grounding Rate measures whether the agent's generated action is valid given the current environment state. It reflects the agent's ability to align textual output with the environment's affordances, indicating its perceptual understanding and language grounding capabilities. It is defined as:

$$\text{Grounding Rate} = \frac{\sum_{t=1}^{T} \mathbb{1}(\text{IsValid}(a_t, \text{Env}_t))}{T}, \tag{28}$$

where T is the total number of interaction steps, a_t is the agent's action at step t, and Env_t is the environment state at that step. The indicator function $\mathbb{1}(\text{IsValid}(a_t, \text{Env}_t))$ returns 1 if the action is valid in the given environment state, and 0 otherwise. Grounding Rate ranges from 0 to 1, with higher values indicating better alignment between the agent's actions and the environment.

Progress Rate. Progress Rate measures the proportion of subgoals successfully completed during task execution, providing a fine-grained indicator of overall progress. Given a task defined by a set of subgoals G, we compute:

$$\text{Progress Rate} = \frac{\sum_{g \in G} \mathbb{1}\left(\exists\, t \leq T \text{ s.t. achieved}(s_t, g)\right)}{|G|}, \tag{29}$$

where achieved(s_t, g) indicates whether subgoal g is satisfied at step t, and $|G|$ is the total number of subgoals. The indicator function returns 1 if the subgoal is achieved at any step. The score ranges from 0 to 1, with higher values indicating greater task completion.

Success Rate. Success Rate is a binary indicator of task completion: it evaluates whether the agent successfully achieves all subgoals by the end of the inference. Unlike Progress Rate, which allows partial credit, Success Rate requires full satisfaction of the task objective:

$$\text{Success Rate} = \mathbb{1}\left(\forall\, g \in G,\ \exists\, t \leq T \text{ s.t. achieved}(s_t, g)\right), \tag{30}$$

where G is the set of subgoals defining the task, and s_t is the agent's state at step t. The indicator returns 1 only if all subgoals are achieved during inference. This score is either 0 or 1 for each task instance.

Area Under the Progress Curve (AUPC). Let a task trajectory be represented as a sequence of progress observations:

$$\mathcal{P} = \{(s_0, p_0), (s_1, p_1), \ldots, (s_n, p_n)\}, \tag{31}$$

where $s_i \in \mathbb{N}$ is the time step and $p_i \in [0, 1]$ is the corresponding progress value. AUPC quantifies how efficiently an agent accumulates progress over time and is computed using the trapezoidal rule:

$$\text{RawAUPC}(\mathcal{P}) = \sum_{i=1}^{n} \frac{p_{i-1} + p_i}{2} \cdot (s_i - s_{i-1}). \tag{32}$$

The raw area is normalized by the total number of steps, ensuring scale-invariant progress comparison:

$$\text{AUPC}(\mathcal{P}) = \begin{cases} \frac{\text{RawAUPC}(\mathcal{P})}{s_n - s_0} & \text{if } s_n > s_0, \\ 0 & \text{otherwise.} \end{cases} \tag{33}$$

A higher AUPC indicates more efficient and consistent task progress, rewarding agents that make early and sustained advancements throughout inference. It yields a scalar score in the range [0,1], reflecting both the speed and smoothness of progress.

D.3 Implementation Details

Sampling Details. We construct the training data by sampling multiple trajectories per task instance using stochastic decoding. The main results are based on 6 sampled trajectories per instance, while 3, 9, and 12 are evaluated in a sensitivity analysis to assess the effect of sampling count. We use temperature-controlled decoding with a default temperature of 1.0 to encourage diversity in action sequences. Lower temperatures (0.3 and 0.7) are included in ablation studies to investigate the impact of more deterministic sampling behavior. All trajectories are truncated to a maximum of 10 steps across datasets (ALFWorld, BabyAI, ScienceWorld) to maintain consistency and reduce variance in long-horizon planning.

Baseline Details. For all baselines, inference is run for up to 20 steps using deterministic decoding (temperature = 0). Each prompt includes a general instruction, contextual knowledge (e.g., demonstrations or insights), and the task's step-wise interaction history. The history window is also set to 20, so no past information is discarded. In the *0-shot* setting, the prompt contains only the instruction and history. The *1-shot* baseline adds a handcrafted demonstration [3], using one shared demo for BabyAI and ScienceWorld, and six domain-specific demos for ALFWorld (one per domain). *Leap* summarizes up to 8 insights from the top-3 diverse success–failure pairs in the training set, which are distilled into general principles and included in the prompt per fold. *Synapse (3-shot)* retrieves the top-3 fully successful trajectories based on task metadata similarity (goal and domain) and includes them in the prompt, while *Synapse (1-shot)* includes only the top-1. *Trad* retrieves the top-3 related nonzero-progress trajectories, decomposes them into (observation, action) pairs, and performs step-wise retrieval. At each step, it selects 3 relevant segments based on the current observation using a sliding window of size 3 (previous, current, and next).

SkillGen Details. During domain graph construction, self-loops are removed before and after pruning. The number of nodes is capped at 30 to avoid extreme cases. If exceeded, intermediate nodes are ranked by the average progress of their incoming edges, and the lowest-ranked are iteratively removed. In temporal-difference (TD) learning, we set $\gamma = 0.95$, $\lambda = 0.9$, $\alpha = 0.05$, and reward noise $\sigma = 0.001$. Q-values are initialized uniformly in $[0.01, 0.05]$. We sample up to 2,000 simple paths (maximum length 20) and run TD updates for 500 iterations, with early stopping if the average Q-value change falls below 10^{-3} for five consecutive steps. An analysis of the domain graphs is provided for reference in Appendix F. For skill extraction stage, the domain-level golden segment is selected as the trajectory with the highest total progress after filtering out invalid steps. For skill retrieval, we set s as the number of skills and k as the number of surrounding steps (antecedents and consequences) per skill. We use $(s{=}1, k{=}1)$ for ALFWorld and BabyAI, and $(s{=}1, k{=}3)$ for ScienceWorld due to its higher complexity.

Computational Resources. All experiments were conducted on a multi-GPU server equipped with 4× NVIDIA RTX A6000 GPUs (49GB each; CUDA 12.7; Driver 565.57.01) and an 80-core Intel Xeon Platinum processor (2 NUMA nodes). Open-source models such as Qwen2.5-7B-Instruct were run locally using HuggingFace Transformers v4.51.3 and PyTorch v2.6.0. Inference with proprietary models—Qwen-Turbo and GPT-4o-mini—was performed via step-wise API calls with batch size 1 to simulate interactive decision-making. Embedding computations were conducted using two models: for one-time task-level metadata embedding, we used OpenAI's `text-embedding-3-small`; for dynamic, step-wise retrieval, we used the `all-MiniLM-L6-v2` model [72]. All experiments were executed under Ubuntu 24.04 with Python 3.9.21, and intermediate results were locally cached to reduce redundant compute. API usage totaled approximately $100 USD, with cost breakdowns based on provider pricing: for GPT-4o-mini, $1.10/*M* input tokens, $0.275/*M* cached input tokens, and $4.40/*M* output tokens; for Qwen-Turbo, minimum rates were $0.05/*M* input tokens and $0.20/*M* output tokens, with a maximum context length of 1,008,192 tokens.

E Additional Experimental Results

E.1 Effect of Step-wise Skills.

Effect of Skills Across Prompting Strategies. Table 7 presents a detailed comparison of skill prompting across different prompting settings—0-shot, 1-shot, and Synapse—on three domains. Across all models and tasks, the addition of skills consistently improves performance, particularly in progress rate (PR), success rate (SR), and AUPC. For example, in ALFWorld, augmenting both 1-shot and Synapse baselines with skills yields substantial gains in SR and AUPC, with the skill-enhanced Synapse achieving the highest scores across all metrics for Qwen2.5-7B-Instruct and Qwen-Turbo. In BabyAI, while Synapse without skills

Table 7: Effect of Skills Across Prompting Strategies. Grounding Rate [%] (↑), Progress Rate [%] (↑), Success Rate [%] (↑), and AUPC [0, 1] (↑). The best method for each LLM is in **bold**.

Method	Qwen2.5-7B-Instruct				Qwen-Turbo				GPT-4o-mini			
	GR	PR	SR	AUPC	GR	PR	SR	AUPC	GR	PR	SR	AUPC
ALFWorld												
0-shot	10.5	6.0	0.8	0.027	56.3	32.2	9.7	0.212	73.7	26.8	1.5	0.184
0-shot + skill	66.0	53.0	28.4	0.350	62.1	47.5	20.9	0.301	62.2	34.3	9.7	0.170
1-shot	28.1	16.0	2.2	0.095	63.9	55.3	36.5	0.380	77.3	43.3	10.5	0.292
1-shot + skill	78.0	**67.7**	49.3	**0.459**	68	55.2	36.6	0.333	77.9	**53.7**	**23.8**	0.343
Synapse	61.5	41.6	17.1	0.278	74.9	54.7	35.8	0.379	76.3	48.8	14.8	0.340
Synapse + skill	**84.0**	67.3	**53.0**	0.457	**83.4**	**64.7**	**50.1**	**0.436**	**81.8**	53.0	23.8	**0.363**
BabyAI												
0-shot	31.8	21.8	7.1	0.037	50.2	32.7	19.6	0.092	55.3	34.2	22.3	0.129
0-shot + skill	41.9	32.2	15.2	0.070	50.6	40.8	26.8	0.134	83.0	51.3	35.7	0.200
1-shot	59.2	36.5	18.8	0.112	61.4	37.2	16.3	0.076	76.6	42.6	28.6	0.154
1-shot + skill	64.7	**49.4**	28.6	0.152	52.4	44.4	31.2	0.132	90.9	**52.5**	40.2	0.198
Synapse	**67.2**	39.4	21.4	0.153	65.0	**55.8**	**45.5**	**0.242**	86.5	44.9	33.9	0.169
Synapse + skill	58.8	47.0	**31.2**	**0.153**	**67.4**	52.2	42.8	0.230	**91.5**	52.2	**40.2**	**0.211**
ScienceWorld												
0-shot	10.8	27.1	9.0	0.136	**28.8**	19.3	4.4	0.113	34.3	44.3	7.7	0.206
0-shot + skill	**13.6**	**38.0**	**16.7**	**0.237**	12.3	**37.1**	10.0	**0.232**	25.9	54.3	18.0	0.313
1-shot	9.6	19.1	5.5	0.108	11.8	19.1	7.7	0.111	**34.3**	46.8	18.8	0.206
1-shot + skill	10.5	33.8	15.6	0.212	9.3	36.9	7.8	0.223	20.1	60.2	31.2	0.386
Synapse	8.6	15.4	4.4	0.090	5.4	16.0	3.3	0.106	14.0	52.8	25.6	0.334
Synapse + skill	10.7	26.8	13.3	0.156	9.1	34.0	**10.1**	0.224	25.1	**64.4**	**36.8**	**0.417**

Table 8: Effect of Step-wise Skills (Supplementary Results for Figure 4). Grounding Rate [%] (↑), Progress Rate [%] (↑), Success Rate [%] (↑), and AUPC [0, 1] (↑). The best method for each LLM is in **bold**.

Method	Qwen2.5-7B-Instruct				Qwen-Turbo				GPT-4o-mini			
	GR	PR	SR	AUPC	GR	PR	SR	AUPC	GR	PR	SR	AUPC
ALFWorld												
SkillGen w/o skills	83.0	67.0	50.7	0.471	86.2	66.8	53.3	0.462	82.0	51.6	21.7	0.354
SkillGen w/o step-wise skills	81.1	63.5	47.1	0.432	83.6	66.6	53.1	0.450	83.9	54.7	26.8	0.380
SkillGen	84.9	68.0	55.2	0.464	85.6	67.6	53.8	0.460	83.6	55.1	29.8	0.369
BabyAI												
SkillGen w/o skills	68.0	47.6	27.7	0.136	74.2	54.8	41.9	0.218	87.9	53.2	36.6	0.207
SkillGen w/o step-wise skills	77.9	50.0	28.6	0.154	75.2	55.0	42.9	0.224	91.1	56.2	41.1	0.234
SkillGen	66.7	50.0	31.2	0.158	73.9	59.4	45.5	0.254	89.5	57.6	41.1	0.248
ScienceWorld												
SkillGen w/o skills	12.8	40.4	18.9	0.250	14.2	36	10.1	0.206	23.3	63.0	35.8	0.412
SkillGen w/o step-wise skills	13.8	43.3	17.7	0.278	10.3	36.8	11.1	0.275	26.4	66.9	39.1	0.424
SkillGen	16.1	46.7	23.4	0.298	13.3	37.7	11.1	0.242	25.3	67.3	40.2	0.442

already performs well, skill augmentation further boosts SR and AUPC, highlighting the complementary

Table 9: Effect of TD-based Credit Assignment (Supplementary Results for Figure 5. Grounding Rate [%] (↑), Progress Rate [%] (↑), Success Rate [%] (↑), and AUPC [0, 1] (↑). The best method for each LLM is in **bold**.

Method	Qwen2.5-7B-Instruct				Qwen-Turbo				GPT-4o-mini			
	GR	PR	SR	AUPC	GR	PR	SR	AUPC	GR	PR	SR	AUPC
ALFWorld												
Trajectory Retrieval	65.4	44.2	22.4	0.296	65.5	54.8	35.8	0.372	79.1	49.1	16.4	0.341
Segment Retrieval	72.0	58.9	42.4	0.408	83.6	63.8	51.5	0.433	81.8	53.3	23.9	0.359
SkillGen	**84.9**	**68.0**	**55.2**	**0.464**	**85.6**	**67.6**	**53.8**	**0.460**	**83.6**	**55.1**	**29.8**	**0.369**
BabyAI												
Trajectory Retrieval	68.2	36.9	19.7	0.115	64.9	46.9	34.8	0.157	87.3	40.9	30.4	0.126
Segment Retrieval	70.6	44.9	25.9	0.125	71.4	55.4	41.1	0.225	89.4	52.2	37.5	0.208
SkillGen	**66.7**	**50.0**	**31.2**	**0.158**	**73.9**	**59.4**	**45.5**	**0.254**	**89.5**	**57.6**	**41.1**	**0.248**
ScienceWorld												
Trajectory Retrieval	7.3	21.1	4.4	0.135	7.1	29.3	8.8	0.180	16.9	61.4	29.0	0.375
Segment Retrieval	13.6	42.7	19	0.266	11	28	7.8	0.167	20.9	55.0	23.4	0.332
SkillGen	**16.1**	**46.7**	**23.4**	**0.298**	**13.3**	**37.7**	**11.1**	**0.242**	**25.3**	**67.3**	**40.2**	**0.442**

Table 10: Effect of Focused Prompting with Golden Segments. Grounding Rate [%] (↑), Progress Rate [%] (↑), Success Rate [%] (↑), and AUPC [0, 1] (↑). The best method for each LLM is in **bold**.

Method	Qwen2.5-7B-Instruct				Qwen-Turbo				GPT-4o-mini			
	GR	PR	SR	AUPC	GR	PR	SR	AUPC	GR	PR	SR	AUPC
ALFWorld												
Static Trajectory	28.1	16.0	2.2	0.095	63.9	55.3	36.5	0.380	77.3	43.3	10.5	0.292
Retrieved Trajectory	61.5	41.6	17.1	0.278	74.9	54.7	35.8	0.379	76.3	48.8	14.8	0.340
Synapse (3-shot)	71.4	44.8	19.4	0.302	78.4	60.6	47.0	0.421	77.4	**52.9**	17.8	**0.360**
Golden Segment	**83.0**	**67.0**	**50.7**	**0.471**	**86.2**	**66.8**	**53.3**	**0.462**	**82.0**	51.6	**21.7**	0.354
BabyAI												
Static Trajectory	59.2	36.5	18.8	0.112	61.4	37.2	16.3	0.076	76.6	42.6	28.6	0.154
Retrieved Trajectory	67.2	39.4	21.4	0.153	65.0	**55.8**	45.5	**0.242**	86.5	44.9	33.9	0.169
Synapse (3-shot)	**78.6**	44.6	**28.6**	**0.163**	62.1	50.8	38.4	0.191	**92.8**	49.5	**38.4**	0.188
Golden Segment	68.0	**47.6**	27.7	0.136	**74.2**	54.8	**b**	0.218	87.9	**53.2**	36.6	**0.207**
ScienceWorld												
Static Trajectory	9.6	19.1	5.5	0.108	11.8	19.1	7.7	0.111	34.3	46.8	18.8	0.206
Retrieved Trajectory	8.6	15.4	4.4	0.090	5.4	16.0	3.3	0.106	14.0	52.8	25.6	0.334
Synapse (3-shot)	6.0	15.3	5.5	0.086	7.2	24.0	10.1	0.155	15.5	60.0	32.4	0.390
Golden Segment	**16.1**	**46.7**	**23.4**	**0.298**	**13.3**	**37.7**	**11.1**	**0.242**	**25.3**	**67.3**	**40.2**	**0.442**

effect of structured prompting. The improvements are especially striking in ScienceWorld, where both 0-shot and 1-shot settings benefit significantly from skill injection, leading to large increases in SR (e.g., from 18.8 to 31.2 on GPT-4o-mini) and smoother task execution as measured by AUPC. While using skills alone (e.g., 0-shot + skill) already provides notable gains over the base 0-shot prompt, the combination with demonstration-based or few-shot prompts yields the most consistent improvements. These results confirm that skill-based prompting enhances grounding and accelerates progress, offering a flexible plug-in benefit

Table 11: Effect of Golden Segment (Supplementary Results for Figure 3.). Grounding Rate [%] (↑), Progress Rate [%] (↑), Success Rate [%] (↑), and AUPC [0, 1] (↑). The best method for each LLM is in **bold**.

Method	Qwen2.5-7B-Instruct				Qwen-Turbo				GPT-4o-mini			
	GR	PR	SR	AUPC	GR	PR	SR	AUPC	GR	PR	SR	AUPC
ALFWorld												
Fixed Demo+Skills	78	67.7	49.3	0.459	68	55.2	36.6	0.333	77.9	53.7	23.8	0.343
Retrieved Demo+Skills	84.0	67.3	53.0	0.457	83.4	64.7	50.1	0.436	81.8	53.0	23.8	0.363
SkillGen	**84.9**	**68.0**	**55.2**	**0.464**	**85.6**	**67.6**	**53.8**	**0.460**	**83.6**	**55.1**	**29.8**	**0.369**
BabyAI												
Fixed Demo+Skills	64.7	49.4	28.6	0.152	52.4	44.4	31.2	0.132	90.9	52.5	40.2	0.198
Retrieved Demo+Skills	58.8	47.0	31.2	0.153	67.4	52.2	42.8	0.230	**91.5**	52.2	40.2	0.211
SkillGen	**66.7**	**50.0**	**31.2**	**0.158**	**73.9**	**59.4**	**45.5**	**0.254**	89.5	**57.6**	**41.1**	**0.248**
ScienceWorld												
Fixed Demo+Skills	10.5	33.8	15.6	0.212	9.3	36.9	7.8	0.223	20.1	60.2	31.2	0.386
Retrieved Demo+Skills	10.7	36.8	13.3	0.156	9.1	34.0	10.1	0.224	25.1	64.4	36.8	0.417
SkillGen	**16.1**	**46.7**	**23.4**	**0.298**	**13.3**	**37.7**	**11.1**	**0.242**	**25.3**	**67.3**	**40.2**	**0.442**

across prompting strategies and model capacities.

Supplementary Results for Figure 7. Table 8 provides supplementary results analyzing the effect of skills on grounding rate (GR), progress rate (PR), success rate (SR), and AUPC across three domains and LLMs. While PR and SR trends are already visualized in the main text, this table complements those findings by reporting GR and AUPC, offering a more comprehensive view of task grounding and execution efficiency. We observe that incorporating both domain-level and step-wise skills generally improves AUPC and GR, reflecting more consistent and timely progress. Notably, the full SkillGen method achieves the highest performance across most metrics and models, especially in ALFWorld and ScienceWorld. In BabyAI, ablations show more mixed effects, highlighting the sensitivity of fine-grained skill application. These results reinforce the contribution of structured, step-aware skill prompting in enhancing task alignment and decision efficiency.

E.2 Effect of TD-based Credit Assignment.

Supplementary Results for Figure 5. Table 9 provides the full numerical results corresponding to Figure 5, showing the effect of TD-based credit assignment across ALFWorld, BabyAI, and ScienceWorld. Consistent with the trends illustrated in the figure, *SkillGen* outperforms both *Trajectory Retrieval* and *Segment Retrieval* across all LLMs and tasks. In ALFWorld, *SkillGen* improves Progress Rate and Success Rate significantly—for example, achieving 68.0 PR and 55.2 SR on Qwen2.5-7B-Instruct, compared to 58.9 PR and 42.4 SR with Segment Retrieval. On the more compositional ScienceWorld, SkillGen reaches 40.2 SR on GPT-4o-mini—a 16.8% absolute gain over Segment Retrieval. These results reinforce the claim that TD-based credit assignment is critical for identifying high-utility actions and enabling effective skill-based prompting.

E.3 Effect of Golden Segment.

Effect of Focused Prompting with Golden Segments. Table 10 provides a deeper analysis of the golden segment's effectiveness by comparing it with various prompt strategies, including static and retrieved trajectories, as well as the 3-shot Synapse baseline. Across all three benchmarks, the *Golden Segment* consistently

Table 12: Effect of Subgoal Annotations. Grounding Rate [%] (↑), Progress Rate [%] (↑), Success Rate [%] (↑), and AUPC [0, 1] (↑). The best method for each LLM is in **bold**.

Method	Qwen2.5-7B-Instruct				Qwen-Turbo				GPT-4o-mini			
	GR	PR	SR	AUPC	GR	PR	SR	AUPC	GR	PR	SR	AUPC
ALFWorld												
Synapse (3-shot)	71.4	44.8	19.4	0.302	78.4	60.6	47.0	0.421	77.4	52.9	17.8	0.360
SkillGen w/o subgoal (sampling=6)	82.0	63.2	50.1	0.440	**87.0**	64.6	52.2	0.439	47.8	32.9	14.2	0.224
SkillGen (sampling=6)	**84.9**	**68.0**	**55.2**	**0.464**	85.6	**67.6**	**53.8**	**0.460**	**83.6**	**55.1**	**29.8**	**0.369**
BabyAI												
Synapse (3-shot)	**78.6**	44.6	28.6	**0.163**	62.1	50.8	38.4	0.191	**92.8**	49.5	38.4	0.188
SkillGen w/o subgoal (sampling=6)	59.6	43.2	**32.1**	0.160	66.1	50.5	36.6	0.221	88.5	58.8	**43.8**	0.232
SkillGen (sampling=6)	66.7	**50.0**	31.2	0.158	**73.9**	**59.4**	**45.5**	**0.254**	89.5	**57.6**	41.1	**0.248**
ScienceWorld												
Synapse (3-shot)	6.0	15.3	5.5	0.086	7.2	24.0	10.1	0.155	15.5	60.0	32.4	0.390
SkillGen w/o subgoal (sampling=6)	11	29.1	18.9	0.175	9	27.7	10	0.147	**27.8**	56.2	32.4	0.357
SkillGen (sampling=6)	**16.1**	**46.7**	**23.4**	**0.298**	**13.3**	**37.7**	**11.1**	**0.242**	25.3	**67.3**	**40.2**	**0.442**

delivers the highest performance in most settings—particularly in Success Rate and AUPC—demonstrating its superiority despite being shorter than full demonstrations. For instance, in ScienceWorld with GPT-4o-mini, the golden segment alone achieves a 40.2% Success Rate and 0.442 AUPC, far surpassing all other methods. These results highlight that focused prompting—not longer or more complex input—is the key to effective decision-making. By isolating high-utility, decision-critical information, our approach substantiates the core principle that focused context matters more than sheer quantity in prompting strategies.

Supplementary Results for Figure 3. Table 11 reports the full results comparing *Fixed Demo+Skills*, *Retrieved Demo+Skills*, and *SkillGen* across three benchmarks. *SkillGen* consistently outperforms the other methods in all settings, validating the benefit of using focused, high-impact segments over full demonstrations. The most pronounced improvements are observed in ALFWorld and ScienceWorld, highlighting that precise, task-relevant context enables more effective reasoning and generalization.

E.4 Effect of Subgoal Annotations

Subgoal annotation significantly enhances SkillGen 's generalization in challenging settings. As shown in Table 12, when equipped with subgoal supervision, SkillGen consistently outperforms its subgoal-free counterpart across tasks and model backbones, with the largest gains observed in complex domains like ScienceWorld. For instance, on ScienceWorld with Qwen2.5-7B-Instruct, the success rate increases from 18.9% to 23.4%, and the progress rate improves from 29.1% to 46.7%. On GPT-4o-mini, AUPC rises substantially from 0.357 to 0.442—a 23.8% relative improvement. Similarly, on ALFWorld with GPT-4o-mini, subgoal-aware SkillGen nearly doubles the success rate from 14.2% to 29.8%. These results underscore the value of intermediate structure in improving reasoning efficiency, particularly in tasks that involve long-horizon dependencies or ambiguous subgoal transitions.

SkillGen achieves strong generalization even without subgoal supervision. Despite using only sampled successful trajectories without subgoal labels, SkillGen outperforms strong multi-shot prompting baselines like Synapse (3-shot) across all benchmarks. On ALFWorld with Qwen2.5-7B-Instruct, it boosts the success rate from 19.4% to 50.1%, a 2.6× improvement. On ScienceWorld, it nearly doubles the progress rate (15.3% → 29.1%) and triples the success rate (5.5% → 18.9%). Even in BabyAI with Qwen-Turbo, where improvements are more modest, AUPC increases from 0.191 to 0.221 (+16%). These gains highlight

Table 13: Sensitive Analysis on Sampling Scaling (Supplementary Results for Figure 6). Grounding Rate [%] (↑), Progress Rate [%] (↑), Success Rate [%] (↑), and AUPC [0, 1] (↑) on ScienceWorld. The best method for each LLM is in **bold**.

Method	Qwen2.5-7B-Instruct				Qwen-Turbo				GPT-4o-mini			
	GR	PR	SR	AUPC	GR	PR	SR	AUPC	GR	PR	SR	AUPC
ALFWorld												
SkillGen (sampling=3)	84.4	68.8	53.7	0.475	84.0	64.5	50.1	0.423	82.7	51.6	25.4	0.353
SkillGen (sampling=6)	**84.9**	**68.0**	**55.2**	**0.464**	85.6	**67.6**	53.8	**0.460**	**83.6**	**55.1**	**29.8**	**0.369**
SkillGen (sampling=9)	83.6	68.3	53.7	0.463	**86.5**	66.4	**54.4**	0.442	80.1	49.5	29.4	0.334
SkillGen (sampling=12)	84.4	66.4	51.4	0.455	85.2	65.2	50.7	0.442	79.6	49.1	23.1	0.322
BabyAI												
SkillGen (sampling=3)	66.5	45.2	26.8	0.128	69.8	50.9	35.7	0.212	88.1	55.2	41.0	0.219
SkillGen (sampling=6)	66.7	**50.0**	31.2	0.158	**73.9**	**59.4**	**45.5**	**0.254**	89.5	**57.6**	**41.1**	**0.248**
SkillGen (sampling=9)	67.8	46.2	30.4	0.162	72.8	52.8	39.3	0.218	**89.6**	56.9	41.1	0.228
SkillGen (sampling=12)	**68.4**	49.7	**33.0**	**0.186**	72.9	54.1	42.0	0.234	89.4	54.6	39.3	0.216
ScienceWorld												
SkillGen (sampling=3)	14.7	40.1	15.5	0.243	**14.6**	**42.9**	12.4	0.262	24.6	62.3	33.4	0.396
SkillGen (sampling=6)	16.1	**46.7**	**23.4**	**0.298**	13.3	37.7	11.1	**0.242**	**25.3**	**67.3**	**40.2**	**0.442**
SkillGen (sampling=9)	15.5	46.4	23.4	0.294	13	36.8	12.2	0.235	24.7	64.4	33.6	0.424
SkillGen (sampling=12)	**16.4**	45.4	23.3	0.285	12.4	38.5	**13.4**	0.247	24.5	65.3	39	0.435

Table 14: Sensitivity Analysis on Sampling Temperature. Grounding Rate [%] (↑), Progress Rate [%] (↑), Success Rate [%] (↑), and AUPC [0, 1] (↑). The best method for each setting is in **bold**.

Temperature	ALFWorld				BabyAI				ScienceWorld			
	GR	PR	SR	AUPC	GR	PR	SR	AUPC	GR	PR	SR	AUPC
0.3	80.7	63.8	47.8	0.447	65.8	46.1	26.8	0.134	**17.5**	41.1	14.6	0.232
0.7	**85.5**	**70.1**	**56.7**	**0.485**	64.3	44.4	28.6	0.136	14.9	39.1	14.3	0.230
1.0	84.9	68.0	55.2	0.464	**66.7**	**50.0**	**31.2**	**0.158**	16.1	**46.7**	**23.4**	**0.298**

the robustness of SkillGen 's interaction-derived skill extraction, which enables effective generalization without relying on explicit subgoal supervision.

E.5 Sensitive Analysis on Sampling Scaling

Supplementary Results for Figure 6. Table 13 reports a sensitivity analysis of the number of sampled paths used in SkillGen. Across all three environments, performance remains stable over a range of sampling sizes, with 6 samples often yielding the best trade-off between efficiency and effectiveness. For example, in ScienceWorld with GPT-4o-mini, sampling 6 paths achieves the highest Success Rate (40.2) and AUPC (0.442). These results suggest that moderate sampling is sufficient to capture high-utility skills, and that our method is robust to the sampling budget.

E.6 Sensitive Analysis on Sampling Temperature.

Sampling Temperature Sensitivity. Table 14 reports the effect of varying the sampling temperature when generating training trajectories for SkillGen. A higher temperature encourages more diverse trajectory sampling, which can uncover additional high-utility skills. On ALFWorld, increasing the temperature to 0.7

Table 15: Sensitive Analysis on Skill Retrieval (Supplementary Results for Figure 6). Grounding Rate [%] (↑), Progress Rate [%] (↑), Success Rate [%] (↑), and AUPC [0, 1] (↑). The best method for each LLM is in **bold**.

Method	Qwen2.5-7B-Instruct				Qwen-Turbo				GPT-4o-mini			
	GR	PR	SR	AUPC	GR	PR	SR	AUPC	GR	PR	SR	AUPC
ALFWorld												
SkillGen (s=1, k=1)	84.9	68.0	55.2	0.464	85.6	67.6	53.8	0.460	83.6	**55.1**	**29.8**	**0.369**
SkillGen (s=3, k=1)	84.0	64.0	48.4	0.444	**86.4**	66.9	56.6	0.448	83.8	52.5	25.4	0.354
SkillGen (s=1, k=3)	83.1	62.1	47.0	0.423	85.8	68.0	57.4	0.459	83.4	52.2	26.1	0.349
SkillGen (s=3, k=3)	**88.7**	**70.2**	**57.4**	**0.481**	83.8	**69.2**	**56.7**	**0.468**	**84.4**	52	26.9	0.361
BabyAI												
SkillGen (s=1, k=1)	66.7	**50.0**	**31.2**	0.158	**73.9**	**59.4**	**45.5**	0.254	89.5	57.6	41.1	**0.248**
SkillGen (s=3, k=1)	69.5	46.2	27.7	0.166	73.4	52.0	39.3	0.229	90.1	52.8	38.4	0.198
SkillGen (s=1, k=3)	66.6	46.1	28.6	0.156	72.8	57.7	44.7	**0.255**	89.4	**56.7**	**42.9**	0.222
SkillGen (s=3, k=3)	**73.9**	49.3	30.4	**0.173**	72.5	52.2	38.4	0.206	**90.6**	52.0	37.5	0.204
ScienceWorld												
SkillGen (s=1, k=1)	16.1	44.8	23.4	0.277	13.6	39.3	13.4	0.241	24.4	63.2	34.6	0.416
SkillGen (s=3, k=1)	14	39.6	14.4	0.245	**14.6**	**45.2**	14.8	0.299	24.4	66.5	34.7	0.437
SkillGen (s=1, k=3)	**16.1**	**46.7**	**23.4**	**0.298**	13.3	37.7	11.1	0.242	25.3	67.3	**40.2**	0.442
SkillGen (s=3, k=3)	15	42.4	20	0.282	13.9	40.8	**16.6**	**0.267**	**26.2**	**68.3**	38	**0.443**

significantly boosts performance, achieving the highest Success Rate (56.7) and AUPC (0.485). However, on BabyAI and ScienceWorld, the best results are obtained at T=1.0, indicating that increased diversity is especially beneficial in tasks requiring fine-grained reasoning or multi-step composition. These results suggest that controlled sampling diversity improves the coverage of decision-relevant behaviors and enhances downstream prompt quality, supporting more effective skill extraction and reuse.

E.7 Sensitive Analysis on Skill Retrieval

Supplementary Results for Figure 6. Table 15 presents a sensitivity analysis of *SkillGen* under different retrieval configurations, varying the number of sampled paths (s) and the number of top-k skills retrieved per step. Overall, the setting s=3, k=3 yields the strongest performance on ALFWorld and ScienceWorld, while s=1, k=1 performs best on BabyAI. These results indicate that focused retrieval is effective even with minimal samples, but leveraging multiple paths and top-k retrieval further enhances performance in more complex environmentsg.

E.8 Token Cost Comparison

In Table 16, we examine token efficiency, measured as the progress rate achieved per 1,000 tokens used (PR/kTok). This metric reflects how effectively each method translates language model computation into task progress. SkillGen consistently achieves the highest token efficiency across all benchmarks and models. On ALFWorld, SkillGen reaches 5.9 PR/kTok with Qwen-Turbo, outperforming Synapse (1-shot: 3.4, 3-shot: 2.4) and Trad (3.2), while using only 11.4k tokens per task, compared to Synapse's 25k–41k. A similar pattern appears on BabyAI, where SkillGen achieves 5.0 PR/kTok (Qwen-Turbo), compared to 3.4 for Synapse (1-shot) and 1.8 for Synapse (3-shot). The advantage is most pronounced on the challenging ScienceWorld domain. Here, SkillGen achieves 4.2 PR/kTok on GPT-4o-mini, which is 1.68× higher

Table 16: Token Efficiency and Task Progress Comparison. Progress Rate [%] (↑), Success Rate [%] (↑), average number of tokens used per task in thousands (Avg. kTok) (↓), and progress increase per 1,000 tokens (PR / kTok) [%] (↑). The best method for each LLM is in **bold**; the second-best method is underlined.

Method	Qwen2.5-7B-Instruct				Qwen-Turbo				GPT-4o-mini			
	PR	SR	Avg. kTok	PR / kTok	PR	SR	Avg. kTok	PR / kTok	PR	SR	Avg. kTok	PR / kTok
ALFWorld												
Leap	21.2	5.2	20.2	1.05	55.6	37.3	16.8	3.3	50.8	11.2	21.6	2.3
Synapse (1-shot)	41.6	17.1	22.6	1.84	54.7	35.8	16.0	3.4	48.8	14.8	17.2	2.8
Synapse (3-shot)	44.8	19.4	41.2	1.09	60.6	47.0	25.2	2.4	52.9	17.8	28.0	1.9
Trad	44.2	22.4	23.5	1.88	54.8	35.8	17.1	3.2	49.1	16.4	23.5	2.1
SkillGen	**68.0**	**55.2**	**12.5**	**5.45**	**67.6**	**53.8**	**11.4**	**5.9**	**55.1**	**29.8**	**15.7**	**3.5**
BabyAI												
Leap	46.3	27.7	31.2	1.5	52.9	38.4	26.0	2.0	43.8	29.4	30.1	1.5
Synapse (1-shot)	39.4	21.4	24.5	1.6	55.8	45.5	16.5	3.4	44.9	33.9	19.7	2.3
Synapse (3-shot)	44.6	28.6	42.8	1.0	50.8	38.4	28.2	1.8	49.5	38.4	32.0	1.5
Trad	36.9	19.7	28.8	1.3	46.9	34.8	22.1	2.1	40.9	30.4	23.4	1.7
SkillGen	**50.0**	**31.2**	**15.4**	**3.2**	**59.4**	**45.5**	**11.9**	**5.0**	**57.6**	**41.1**	**14.0**	**4.1**
ScienceWorld												
Leap	25.8	11.1	43.4	0.6	21.8	6.6	40.7	0.5	51.7	21.1	44.0	1.2
Synapse (1-shot)	15.4	4.4	22.6	0.7	16.0	3.3	20.4	0.8	52.8	25.6	21.0	2.5
Synapse (3-shot)	15.3	5.5	39.6	0.4	24.0	10.1	32.6	0.7	60.0	32.4	31.9	1.9
Trad	21.1	4.4	32.6	0.6	29.3	8.8	27.6	1.1	61.4	29.0	26.8	2.3
SkillGen	**46.7**	**23.4**	**15.9**	**2.9**	**37.7**	**11.1**	**16.1**	**2.3**	**67.3**	**40.2**	**16.1**	**4.2**

than the best-performing baseline, Synapse (1-shot, 2.5 PR/kTok), and far ahead of Synapse (3-shot, 1.9), despite their higher token consumption. Notably, increasing the number of in-context demonstrations leads to diminishing efficiency returns: although Synapse (3-shot) uses over 31k tokens per task, its PR/kTok consistently drops compared to the 1-shot variant.

These findings demonstrate that SkillGen not only excels in performance but delivers the best return on token usage, making it the most cost-effective solution among all evaluated methods.

E.9 Sampling Strategies for TD-Based Credit Assignment

To explore different sampling strategies for TD-based credit assignment, we consider not only uniform sampling but also a weighted alternative, where trajectory selection probabilities are derived from empirical progress deltas. For a given path $\tau = (a_0, a_1, \ldots, a_T)$, we compute its score as the cumulative sum of edge-level progress: $\text{score}(\tau) = \sum_{t=0}^{T-1} \bar{\Delta}(a_t, a_{t+1})$, where $\bar{\Delta}(a_t, a_{t+1})$ denotes the empirical mean of recorded progress deltas for transition (a_t, a_{t+1}). To convert scores into a probabilistic sampling distribution, we apply a numerically stable softmax: $p_i = \frac{\exp(\text{score}(\tau_i) - \max_j \text{score}(\tau_j))}{\sum_j \exp(\text{score}(\tau_j) - \max_k \text{score}(\tau_k))}$, allowing high-score paths to be prioritized while preserving nonzero probability for exploration. Sampled paths are then drawn without replacement from this weighted distribution.

We compare the performance of SkillGen under two sampling strategies—weighted sampling and uniform sampling. All results are conducted using a single retrieved skill (s=1) and top-1 antecedent and consequence (k=1). In Table 17, uniform sampling consistently outperforms weighted sampling across all three tasks in PR, SR, and AUPC. This finding suggests that while prioritizing high-scoring trajectories can potentially focus learning on informative regions of the search space, it may also amplify noise or overfit to progress signals—especially when empirical deltas are sparse or unevenly distributed. By treating all observed transitions equally, uniform sampling promotes broader trajectory diversity and more stable value propagation, which appears crucial for generalization and long-horizon task completion.

Table 17: Comparison of SkillGen with weighted vs. uniform sampling strategies under the Qwen-7B-Instruct model with $s = 1$, $k = 1$.

Method	AlFWorld				BabyAI				ScienceWorld			
	GR	PR	SR	AUPC	GR	PR	SR	AUPC	GR	PR	SR	AUPC
SkillGen (Weighted Sampling)	82.9	67	53	0.455	**66.9**	47.4	27.6	0.142	15.2	41.7	21.1	0.259
SkillGen (Uniform Sampling)	**84.9**	**68.0**	**55.2**	**0.464**	66.7	**50.0**	**31.2**	**0.158**	**16.1**	**44.8**	**23.4**	**0.277**

Table 18: Comparison between Act-style (*observation* $\rightarrow$ *action*) and React-style (*observation* $\rightarrow$ *thought* $\rightarrow$ *action*) prompting strategies for SkillGen in ScienceWorld using GPT-4o-mini.

Method	GPT-4o-mini			
	GR	PR	SR	AUPC
SkillGen (Act)	**25.3**	67.3	40.2	0.442
SkillGen (React)	23	**70.1**	**43.4**	**0.451**

E.10 Act / ReAct for SkillGen Inference

Our main experiments use a minimalist *Act-style* prompting framework (*observation* $\rightarrow$ *action*), which allows us to directly evaluate the decision-making abilities of LLMs without relying on additional components such as memory modules or external tools. This lightweight design makes the framework broadly applicable to most instruction-following models and helps us focus on the model's intrinsic inference capabilities, without interference from extra reasoning mechanisms.

To further examine the impact of explicit reasoning structure, we conduct a controlled comparison on the ScienceWorld dataset using `GPT-4o-mini`. In this setting, we evaluate SkillGen under the *React-style* prompting format, which extends the Act-style formulation (*observation* $\rightarrow$ *action*) with an intermediate reasoning step (*observation* $\rightarrow$ *thought* $\rightarrow$ *action*). To ensure a fair comparison, we limit the total number of action steps to 20 for both settings. For Act-style, each decision is made in a single inference step. For React-style, each decision comprises two inference steps—one for generating the thought and one for producing the final action—resulting in a total of 20 thoughts and 20 actions per task. As shown in Table 18, React-style prompting improves progress rate (70.1 vs. 67.3), success rate (43.4 vs. 40.2), and AUPC (0.451 vs. 0.442), while incurring a slight drop in goal completion (23.0 vs. 25.3). These results suggest that incorporating intermediate thoughts can enhance local decision quality and stabilize planning, although at the cost of increased prompt complexity.

F Analysis on Domain Knowledge Graphs

Tables 19–21 summarize the structural statistics of the action-centric domain graphs constructed from sampled trajectories using different base models—Qwen2.5-7B-Instruct, Qwen-Turbo, and GPT-4o-mini—across the three task domains: ALFWorld, BabyAI, and ScienceWorld. For each setting, we vary the number of sampled trajectories per task (#Sampling) from 3 to 12, and report the average number of nodes, edges, and the average node degree in the resulting graph.

We observe that increasing the number of sampled trajectories consistently leads to denser graphs, reflected by higher node counts and edge connectivity. This trend is especially pronounced in the ALFWorld domain, which exhibits the largest graph size and connectivity across all models. In contrast, BabyAI and ScienceWorld produce smaller graphs, but still show steady growth in structural complexity with more samples. Across models, GPT-4o-mini yields the most extensive graphs, suggesting that its behavior leads to more diverse action coverage during trajectory sampling. These statistics provide insight into the richness of

Table 19: Action-Centric Graphs Statistics of Qwen2.5-7B-Instruct.

Dataset	#Sampling	Avg #Nodes	Avg #Edges	Avg Degree
ALFWorld	3	17.46	34.0	3.8
	6	21.71	46.79	4.27
	9	24.62	59.17	4.89
	12	27.12	69.33	5.17
BabyAI	3	7.81	13.71	3.27
	6	9.05	18.54	3.93
	9	9.78	22.03	4.36
	12	10.12	24.1	4.63
ScienceWorld	3	6.92	10.68	2.59
	6	8.49	14.7	3.17
	9	9.13	17.73	3.6
	12	9.71	19.46	3.74

Table 20: Action-Centric Graphs Statistics of Qwen-Turbo.

Dataset	#Sampling	Avg #Nodes	Avg #Edges	Avg Degree
ALFWorld	3	28.67	60.71	4.27
	6	33.21	77.21	4.67
	9	35.04	89.08	5.04
	12	36.42	98.5	5.34
BabyAI	3	9.88	15.44	2.9
	6	11.69	22.36	3.61
	9	12.09	24.17	3.75
	12	12.22	25.29	3.86
ScienceWorld	3	8.52	12.35	2.41
	6	10.54	17.43	2.82
	9	12.11	21.38	3.03
	12	12.9	23.98	3.2

the induced action space and the granularity of skills extracted during graph-based knowledge construction.

G Prompt Examples

We present several examples to illustrate the prompt structure.The step-wise skill section displays only the skill retrieved for the final action.

SkillGen Prompt Template

{task_description}

Golden Segment (What to Imitate)
Here is a related action sequence, which may not be fully accurate, but help identify promising directions:
{golden_segment}

Step-wise Reusable Skills (Context-Aware Guidance)

Table 21: Action-Centric Graphs Statistics of GPT-4o-mini.

Dataset	#Sampling	Avg #Nodes	Avg #Edges	Avg Degree
ALFWorld	3	30.71	75.38	4.84
	6	39.29	105.08	5.31
	9	43.79	126.67	5.79
	12	46.04	138.25	5.99
BabyAI	3	9.08	15.95	3.03
	6	11.04	22.48	3.75
	9	11.46	23.88	3.9
	12	11.76	24.84	3.98
ScienceWorld	3	13.13	23.94	3.46
	6	16.22	33.6	3.94
	9	18.29	41.19	4.26
	12	20.27	47.52	4.43

These skills are relevant to the current context based on your most recent action.
They suggest promising steps to explore the environment: {step_wise_skills}

Instruction
You should use the following commands for help when your action cannot be understood: `check valid actions`.
You should use the following commands for help when your action cannot be understood: `inventory`.
Generate the **next best action** to reach the goal.
Goal: {goal}
{history_transitions}
Action:

SkillGen Prompt Example (ALFWorld)

Your task is to interact with a virtual household simulator to accomplish a specific task. With each interaction, you will receive an observation.Your role is to decide on an action based on the observation

Golden Segment (What to Imitate):
Here is a related demonstration, which may not be fully accurate, but help identify promising directions:
Goal: examine the alarmclock with the desklamp
You are in the middle of a room. Looking quickly around you, you see a bed 1, a desk 2, a desk 1, a drawer 6, a drawer 5, a drawer 4, a drawer 3, a drawer 2, a drawer 1, a garbagecan 1, a laundryhamper 1, a safe 1, a shelf 6, a shelf 5, a shelf 4, a shelf 3, a shelf 2, and a shelf 1.
Your task is to: examine the alarmclock with the desklamp.

```
ACTION: go to desk 1
ACTION: check valid actions
ACTION: use desklamp 1
ACTION: go to desk 2
ACTION: check valid actions
ACTION: take alarmclock 1 from desk 2
```

Step-wise Reusable Skills (Context-Aware Guidance):

These skills are relevant to the current context based on your most recent action. They suggest promising steps to explore the environment:
Skill 1: Centered on action `'go to desk'`

- **Common precursors:** `check valid actions`
- **Typical next steps:** `use desklamp`

Instruction:

- Use `check valid actions` if your action is not understood.
- Use `inventory` to see what you are holding.
- Generate the **next best action** to reach the goal.

Interaction History:
Goal: Examine the bowl with the desklamp

- `ACTION: go to desk 1`
  `OBSERVATION:` On the desk 1, you see a alarmclock 2, a alarmclock 1, a desklamp 1, a mug 1, and a pencil 1.
- `ACTION: check valid actions`
  `OBSERVATION:` Choose from: examine desk 1, go to bed 1, go to desk 2, go to drawer 1–6, go to garbagecan 1, laundryhamper 1, safe 1, shelf 1–6, inventory, look, take alarmclock 1/2, take mug 1, take pencil 1, use desklamp 1.
- `ACTION: go to desk 2`
  `OBSERVATION:` On the desk 2, you see a bowl 3, a bowl 2, and a mug 2.
- `ACTION: check valid actions`
  `OBSERVATION:` Choose from: examine desk 2, go to other objects, inventory, look, take bowl 2/3, take mug 2.
- `ACTION: take bowl 2 from desk 2`
  `OBSERVATION:` You pick up the bowl 2 from the desk 2.
- `ACTION: go to desk 1`
  `OBSERVATION:` On the desk 1, you see a alarmclock 2, a alarmclock 1, a desklamp 1, a mug 1, and a pencil 1.
- **ACTION: `use desklamp 1`** *Progress Rate: 1.0*

Note: This example is provided to illustrate the overall prompt structure; the step-wise skill section only displays the skill retrieved for the final action.

SkillGen Prompt Example (BabyAI)

You are placed in a room and you need to accomplish the given goal with actions.

Golden Segment (What to Imitate):
Here is a related action sequence, which may not be fully accurate, but help identify promising directions:
Goal: open the blue door
In front of you in this room, you can see several objects: There is a blue closed door 1 1 steps in front of you and 1 steps to your left. The room has walls around you. You are facing a wall 2 steps away. You are not carrying anything.
`ACTION: go to blue closed door 1`
`ACTION: toggle and go through blue closed door 1`
`ACTION: move forward`

Step-wise Reusable Skills (Context-Aware Guidance):
These skills are relevant to the current context based on your most recent action. They suggest promising steps to explore the environment:
Skill 1: Centered on action `'move forward'`

- **Common precursors:** `toggle and go through blue closed door`
- **Typical next steps:** `turn left`

Instruction:

- Use `check valid actions` if your action is not understood.
- Use `inventory` to see what you are holding.
- Generate the **next best action** to reach the goal.

Interaction History:
Goal: Open the red door, and open the blue door

- `ACTION: check valid actions`
  `OBSERVATION:` You can take the following actions: turn left, turn right, check available actions.
- `ACTION: go to blue closed door 1`
  `OBSERVATION:` The action is not recognized. Please check valid actions.
- `ACTION: turn left`
  `OBSERVATION:` There is a red closed door 1 4 steps in front of you and 3 steps to your left.
- `ACTION: go to red closed door 1`
  `OBSERVATION:` Red closed door 1 is right in front of you 1 step away.
- `ACTION: toggle and go through red closed door 1`
  `OBSERVATION:` You are facing a wall 5 steps away.
- `ACTION: move forward`
  `OBSERVATION:` You are facing a wall 4 steps away.

- `ACTION: turn left`
  `OBSERVATION:` There is a red open door 1 0 steps in front of you and 1 step to your left.
- `ACTION: go to blue closed door 1`
  `OBSERVATION:` The action is not recognized.
- `ACTION: go to blue closed door 1`
  `OBSERVATION:` The action is not recognized.
- `ACTION: check valid actions`
  `OBSERVATION:` You can take the following actions: turn left, turn right, go through red open door 1, go to red open door 1.
- `ACTION: go through red open door 1`
  `OBSERVATION:` There is a blue closed door 1 5 steps in front of you and 2 steps to your left.
- `ACTION: go to blue closed door 1`
  `OBSERVATION:` Blue closed door 1 is right in front of you 1 step away.
- `ACTION: toggle and go through blue closed door 1`
  `OBSERVATION:` You are facing a wall 5 steps away.
- `ACTION: move forward`
  `OBSERVATION:` You are facing a wall 4 steps away.
- **ACTION: `turn left`** *Progress Rate: 1.0*

  Note: This example is provided to illustrate the overall prompt structure; the step-wise skill section only displays the skill retrieved for the final action.

SkillGen Prompt Example (ScienceWorld)

You are an agent in a virtual science school environment, tasked to interact with various elements.

Golden Segment (What to Imitate):
Here is a related action sequence, which may not be fully accurate, but help identify promising directions:
Goal: Find butterfly egg in the outside. First, focus on the thing. Then, move it to the blue box in the bedroom.
`This room is called the hallway. In it, you see:` air, a painting, doors to the art studio, bedroom, greenhouse, kitchen, living room, and workshop (all open).
`Inventory:` an orange.

```
ACTION: go to greenhouse
ACTION: look around
ACTION: look around
ACTION: go to outside
ACTION: look around
ACTION: focus on butterfly egg
ACTION: pick up butterfly egg
ACTION: move butterfly egg to blue box
```

Step-wise Reusable Skills (Context-Aware Guidance):

These skills are relevant to the current context based on your most recent action. They suggest promising steps to explore the environment:
Skill 1: Centered on action `'look around'`

- **Common precursors:** `go to outside`
- **Typical next steps:** `move butterfly egg to blue box`

Instruction:

- Use `check valid actions` if your action is not understood.
- Use `inventory` to inspect your current items.
- Generate the **next best action** to complete the goal.

Interaction History:
Goal: Find a dove egg in the outside. First, focus on the thing. Then, move it to the red box in the bathroom.

- `OBSERVATION (greenhouse):` air, multiple flower pots, jug, shovel, sink, open doors to hallway and outside
  `Inventory:` an orange
- `ACTION: go to outside`
  `OBSERVATION:` You move to the outside.
- `ACTION: look around`
  `OBSERVATION:` You see air, an axe, blue jay egg, butterfly egg, dove egg, fire pit, fountain, ground, wood, and open doors to foundry, greenhouse, kitchen.
- `ACTION: focus on dove egg`
  `OBSERVATION:` You focus on the dove egg.
- `ACTION: pick up dove egg`
  `OBSERVATION:` You move the dove egg to the inventory.
- `ACTION: go to bathroom`
  `OBSERVATION:` No known action matches that input.
- `ACTION: go to hallway`
  `OBSERVATION:` No known action matches that input.
- `ACTION: go to kitchen`
  `OBSERVATION:` You move to the kitchen.
- `ACTION: look around`
  `OBSERVATION:` air, chair, counter, cupboard, freezer, fridge, jar of sodium chloride, lighter, oven, painting, sink, soap, stopwatch, stove, table, thermometer; open doors to bathroom, hallway, and outside.

- `ACTION: go to bathroom`
  `OBSERVATION:` You move to the bathroom.
- `ACTION: look around`
  `OBSERVATION:` air, bathtub, glass cup, painting, red box (empty), sink, toilet (with water and a closed drain)
- **ACTION:** `move dove egg to red box` *Progress Rate: 1.0*

Note: This example is provided to illustrate the overall prompt structure; the step-wise skill section only displays the skill retrieved for the final action.

H Case Study

Here, we present two cases to illustrate how skill-enhanced prompting guides LLMs through structured, multi-step decision-making.

Skill knowledge, distilled from past interactions, captures reusable patterns of behavior that are both interpretable and broadly transferable across tasks. Case A presents a manipulation-centered task in ALFWorld, where the agent must retrieve and place two peppershakers into a drawer. The trajectory demonstrates how skill-enhanced prompting enables modular and recoverable decision-making in object-centric settings. Early in the task, the agent execute "go to countertop" to explore items. Upon discovering the first peppershaker, the agent reuses the structured routine of "take object," followed by "go to drawer," "open drawer," and "put object"—a sequence grounded in manipulation skills that apply broadly across object types and locations. When the first peppershaker is placed in the drawer, the agent invokes the same search pattern on another countertop, demonstrating flexible reuse of skills under uncertainty. Throughout the trajectory, the agent follows interpretable subgoal transitions aligned with learned affordances, such as only placing items after opening drawers. The agent's ability to recover from failure, recompose subroutines, and generalize across instances of "peppershaker" and "drawer" reflects the power of skills as mid-level priors. This example further affirms that skill-driven prompting enables compositional manipulation planning, supporting both action precision and long-horizon success in LLM-based sequential decision making.

Case A: Put Two Peppershakers in Drawer

Goal: put two peppershaker in drawer
Progress Rate: 1.0

OBS: You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a cabinet 5, a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 3, a countertop 2, a countertop 1, a drawer 3, a drawer 2, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a shelf 3, a shelf 2, a shelf 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.
Your task is to: put two peppershaker in drawer.
SKILL Centered on action 'the beginning of the task'

- Typical next steps after this action:
  - After 'the beginning of the task', agents usually continue with 'go to shelf'.

ACTION: go to shelf 1 **PROGRESS RATE:** 0.0

OBS: On the shelf 1, you see a statue 1.
SKILL Centered on action 'go to shelf'

- Common precursors to this action:
  - Agents often perform 'the beginning of the task' before 'go to shelf'.
- Typical next steps after this action:
  - After 'go to shelf', agents usually continue with 'go to countertop'.

ACTION: go to countertop 1 **PROGRESS RATE:** 0.0
OBS: On the countertop 1, you see a apple 1, a creditcard 2, a egg 1, and a tomato 1.
SKILL Centered on action 'go to countertop'

- Common precursors to this action:
  - Agents often perform 'the beginning of the task' before 'go to countertop'.
- Typical next steps after this action:
  - After 'go to countertop', agents usually continue with 'take soapbottle from countertop'.

ACTION: go to countertop 2 **PROGRESS RATE:** 0.25
OBS: On the countertop 2, you see a bread 1, a butterknife 1, a cellphone 1, a cup 3, a knife 1, a pan 1, a papertowelroll 1, a peppershaker 1, a saltshaker 1, and a spatula 1.
ACTION: take peppershaker 1 from countertop 2 **PROGRESS RATE:** 0.5
OBS: You pick up the peppershaker 1 from the countertop 2.
SKILL Centered on action 'take peppershaker from countertop'

- Common precursors to this action:
  - Agents often perform 'go to countertop' before 'take peppershaker from countertop'.
- Typical next steps after this action:
  - After 'take peppershaker from countertop', agents usually continue with 'go to countertop'.

ACTION: go to drawer 1 **PROGRESS RATE:** 0.5
OBS: The drawer 1 is closed.
SKILL Centered on action 'go to drawer'

- Common precursors to this action:
  - Agents often perform 'go to countertop' before 'go to drawer'.
- Typical next steps after this action:
  - After 'go to drawer', agents usually continue with 'open drawer'.

ACTION: open drawer 1 **PROGRESS RATE:** 0.5
OBS: You open the drawer 1. The drawer 1 is open. In it, you see a spoon 1.
SKILL Centered on action 'open drawer'

- Common precursors to this action:
  - Agents often perform 'go to drawer' before 'open drawer'.
- Typical next steps after this action:
  - After 'open drawer', agents usually continue with 'put peppershaker in/on drawer'.

ACTION: put peppershaker 1 in/on drawer 1 **PROGRESS RATE:** 0.75
OBS: You put the peppershaker 1 in/on the drawer 1.
SKILL Centered on action 'put peppershaker in/on drawer'

- Common precursors to this action:
  - Agents often perform 'open drawer' before 'put peppershaker in/on drawer'.
- Typical next steps after this action:
  - After 'put peppershaker in/on drawer', agents usually continue with 'go to countertop'.

ACTION: go to countertop 2 **PROGRESS RATE:** 0.75
OBS: On the countertop 2, you see a bread 1, a butterknife 1, a cellphone 1, a cup 3, a knife 1, a pan 1, a papertowelroll 1, a saltshaker 1, and a spatula 1.
ACTION: take peppershaker 2 from countertop 2 **PROGRESS RATE:** 0.75
OBS: Nothing happens.
ACTION: go to countertop 3 **PROGRESS RATE:** 0.75
OBS: On the countertop 3, you see a butterknife 2, a creditcard 3, a dishsponge 3, a houseplant 1, a knife 2, a lettuce 1, a peppershaker 2, a spatula 3, a vase 3, a vase 2, and a vase 1.
ACTION: take peppershaker 2 from countertop 3 **PROGRESS RATE:** 0.75
OBS: You pick up the peppershaker 2 from the countertop 3.
ACTION: go to drawer 1 **PROGRESS RATE:** 0.75
OBS: The drawer 1 is open. In it, you see a peppershaker 1, and a spoon 1.
SKILL Centered on action 'go to drawer'

- Common precursors to this action:
  - Agents often perform 'go to countertop' before 'go to drawer'.
- Typical next steps after this action:
  - After 'go to drawer', agents usually continue with 'open drawer'.

ACTION: put peppershaker 2 in/on drawer 1 **PROGRESS RATE:** 1.0

Case B showcases how skill-enhanced prompting enables structured and efficient multi-step decision-making in ScienceWorld, a more complex environment. At the task's outset, the agent leverages the skill suggesting "go to hallway" as a common first step, thereby avoiding unproductive exploration in the workshop. As it navigates toward the target, skills centered on "look around" and "go to greenhouse" guide the agent through meaningful spatial transitions. These skills are domain-general: they help not only in tasks involving egg retrieval, but also in any setting requiring room exploration. Upon encountering the blue jay egg, the agent applies a skill pattern originally associated with the "butterfly egg" task—namely, a two-step focus-then-pick-up strategy—demonstrating the generalization of interaction structures across object types. Later,

skills associated with "go to kitchen" and "look around" support the correct identification of the delivery location. Finally, although no explicit skill specifies "move X to Y," the agent successfully executes the goal action by generalizing from previously learned manipulation patterns. This example illustrates how skill-driven scaffolding provides trajectory templates, narrows the action space, and promotes compositional generalization, ultimately allowing the agent to complete the task with a final progress rate of 1.0. This compositional reuse of mid-level routines highlights how skills operate as modular and interpretable units of behavior, enabling robust generalization and precise control for LLMs.

Case B: Blue Jay Egg → Purple Box

Goal: Your task is to find a blue jay egg in the outside. First, focus on the thing. Then, move it to the purple box in the kitchen.
Progress Rate: 1.0

OBS: This room is called the workshop. In it, you see: the agent, a substance called air, a table. On the table is: a battery, a black wire, a blue wire, a orange wire, a red light bulb, which is off, a switch, which is off, a violet light bulb, which is off, a yellow light bulb, which is off. An ultra low temperature freezer. The ultra low temperature freezer door is closed.
You also see: A door to the hallway (that is open).
Inventory: an orange
SKILL: Centered on action 'the beginning of the task'

- Typical next steps after this action:
  - After 'the beginning of the task', agents usually continue with 'go to hallway'.
  - After 'the beginning of the task', agents usually continue with 'check valid actions'.
  - After 'the beginning of the task', agents usually continue with 'go to greenhouse'.

ACTION: go to hallway **PROGRESS RATE:** 0.0

OBS: You move to the hallway.
SKILL: Centered on action 'go to hallway'

- Common precursors to this action:
  - Agents often perform 'the beginning of the task' before 'go to hallway'.
  - Agents often perform 'look around' before 'go to hallway'.
  - Agents often perform 'go to living room' before 'go to hallway'.
- Typical next steps after this action:
  - After 'go to hallway', agents usually continue with 'go to bedroom'.
  - After 'go to hallway', agents usually continue with 'look around'.

ACTION: look around **PROGRESS RATE:** 0.0

OBS: This room is called the hallway. In it, you see: the agent, a substance called air, a painting.
You also see: A door to the art studio (that is open), A door to the bedroom (that is open), A door to the greenhouse (that is open), A door to the kitchen (that is open), A door to the living room (that is open), A door to the workshop (that is open)
SKILL: Centered on action 'look around'

- Common precursors to this action:
  - Agents often perform 'the beginning of the task' before 'look around'.
  - Agents often perform 'pick up flower pot' before 'look around'.
  - Agents often perform 'check valid actions' before 'look around'.
- Typical next steps after this action:
  - After 'look around', agents usually continue with 'go to hallway'.
  - After 'look around', agents usually continue with 'focus on butterfly egg'.
  - After 'look around', agents usually continue with 'pick up shovel'.

ACTION: go to greenhouse **PROGRESS RATE:** 0.0

OBS: You move to the greenhouse.
SKILL: Centered on action 'go to greenhouse'

- Common precursors to this action:
  - Agents often perform 'the beginning of the task' before 'go to greenhouse'.
  - Agents often perform 'look around' before 'go to greenhouse'.
- Typical next steps after this action:
  - After 'go to greenhouse', agents usually continue with 'look around'.

ACTION: look around **PROGRESS RATE:** 0.0

OBS: This room is called the greenhouse. In it, you see: the agent, a substance called air, a bee hive (closed), flower pots, a jug, a shovel, a sink (off).
You also see: A door to the hallway (that is open), A door to the outside (that is open)
SKILL: Centered on action 'look around'

- Common precursors to this action:
  - Agents often perform 'the beginning of the task' before 'look around'.
  - Agents often perform 'pick up flower pot' before 'look around'.
  - Agents often perform 'check valid actions' before 'look around'.
- Typical next steps after this action:
  - After 'look around', agents usually continue with 'go to hallway'.
  - After 'look around', agents usually continue with 'focus on butterfly egg'.
  - After 'look around', agents usually continue with 'pick up shovel'.

ACTION: go to outside **PROGRESS RATE:** 0.2

OBS: You move to the outside.
SKILL: Centered on action 'go to outside'

- Common precursors to this action:

- Agents often perform 'the beginning of the task' before 'go to outside'.
- Agents often perform 'check valid actions' before 'go to outside'.
- Agents often perform 'look around' before 'go to outside'.

- Typical next steps after this action:
  - After 'go to outside', agents usually continue with 'look around'.

ACTION: look around **PROGRESS RATE:** 0.2

OBS: This outside location is called the outside. Here you see: the agent, a substance called air, an axe, a blue jay egg, a butterfly egg, a dove egg, a fire pit (empty), a fountain (with water), the ground, a substance called wood.
You also see: A door to the foundry (open), A door to the greenhouse (open), A door to the kitchen (open)
SKILL: Centered on action 'look around'

- Common precursors to this action:
  - Agents often perform 'the beginning of the task' before 'look around'.
  - Agents often perform 'pick up flower pot' before 'look around'.
  - Agents often perform 'check valid actions' before 'look around'.

- Typical next steps after this action:
  - After 'look around', agents usually continue with 'go to hallway'.
  - After 'look around', agents usually continue with 'focus on butterfly egg'.
  - After 'look around', agents usually continue with 'pick up shovel'.

ACTION: focus on blue jay egg **PROGRESS RATE:** 0.4

OBS: You focus on the blue jay egg.
SKILL: Centered on action 'focus on butterfly egg'

- Common precursors to this action:
  - Agents often perform 'look around' before 'focus on butterfly egg'.

- Typical next steps after this action:
  - After 'focus on butterfly egg', agents usually continue with 'pick up butterfly egg'.

ACTION: pick up blue jay egg **PROGRESS RATE:** 0.6

OBS: You move the blue jay to the inventory.
ACTION: go to kitchen **PROGRESS RATE:** 0.8

OBS: You move to the kitchen.
SKILL: Centered on action 'go to kitchen'

- Typical next steps after this action:
  - After 'go to kitchen', agents usually continue with 'look around'.

ACTION: look around **PROGRESS RATE:** 0.8

OBS: The kitchen contains: various furniture, a bowl with fruit, a purple box (empty), stove, oven, thermometer (10°C), etc.
You also see: A door to the bathroom (open), hallway (open), outside (open)
SKILL: Centered on action 'look around'

- Common precursors to this action:
  - Agents often perform 'the beginning of the task' before 'look around'.
  - Agents often perform 'pick up flower pot' before 'look around'.
  - Agents often perform 'check valid actions' before 'look around'.
- Typical next steps after this action:
  - After 'look around', agents usually continue with 'go to hallway'.
  - After 'look around', agents usually continue with 'focus on butterfly egg'.
  - After 'look around', agents usually continue with 'pick up shovel'.

ACTION: move blue jay egg to purple box **PROGRESS RATE:** 1.0