Reinforcing LLM Agents via Policy Optimization with Action Decomposition

Muning Wen[1], **Ziyu Wan**[1], **Jun Wang**[2], **Weinan Zhang**[1,†], **Ying Wen**[1,†],
[1]Shanghai Jiao Tong University, [2]University College London

Abstract

Language models as intelligent agents push the boundaries of sequential decision-making agents but struggle with limited knowledge of environmental dynamics and exponentially huge action space. Recent efforts like GLAM and TWOSOME manually constrain the action space to a restricted subset and employ reinforcement learning to align agents' knowledge with specific environments. However, they overlook fine-grained credit assignments for intra-action tokens, which is essential for efficient language agent optimization, and rely on human's prior knowledge to restrict action space. This paper proposes decomposing language agent optimization from the action level to the token level, offering finer supervision for each intra-action token and manageable optimization complexity in environments with unrestricted action spaces. Beginning with the simplification of flattening all actions, we theoretically explore the discrepancies between action-level optimization and this naive token-level optimization. We then derive the Bellman backup with Action Decomposition (BAD) to integrate credit assignments for both intra-action and inter-action tokens, effectively eliminating the discrepancies. Implementing BAD within the PPO algorithm, we introduce Policy Optimization with Action Decomposition (POAD). POAD benefits from a finer-grained credit assignment process and lower optimization complexity, leading to enhanced learning efficiency and generalization abilities in aligning language agents with interactive environments. We validate POAD across diverse testbeds, with results affirming the advantages of our approach and the correctness of our theoretical analysis[1].

1 Introduction

Large language models (LLMs) have demonstrated promising capabilities of solving various tasks, from instructions following to complex reasoning and real-world interaction [1–3]. This growing task-solving ability underscores their potential as intelligent language agents in interactive environments [4, 5]. However, despite in-context language generation aiding comprehension of environmental states and action spaces in sequential decision-making tasks, misalignment issues [4, 5] such as generating invalid actions and lacking knowledge of environmental dynamics hinder these agents' ability to complete decision-making tasks robustly and efficiently.

Recent advances [4–8] have showcased that the aforementioned challenges can be alleviated in a trial-and-error learning style, namely Reinforcement Learning (RL) [9]. Representatively, GLAM [4] and TWOSOME [5] treat the language actions, i.e. token sequences outputted by a language model, as whole units, and optimize actions' likelihood, calculated as the products of conditional probabilities of intra-action tokens. Leveraging the chain rule of probability, they build the bridge between optimizing actions and optimizing tokens, aligning the training process of language models as

[1†]Corresponding to Ying Wen <ying.wen@sjtu.edu.cn>, Weinan Zhang <wnzhang@sjtu.edu.cn>. The source code could be accessed directly with this link `https://github.com/morning9393/ADRL`.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-3297

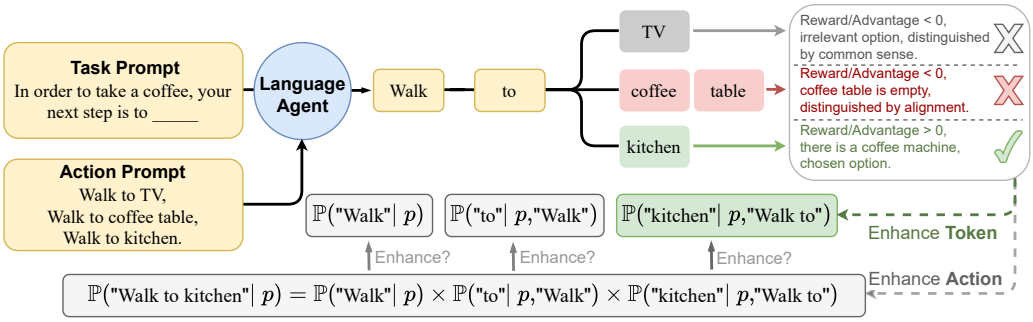


Figure 1: A Case to demonstrate: (a) the necessity of aligning language agents with environments to exclude the wrong option, since the agent does not initially know that *"coffee table is empty"*. (b) Action-level optimization is uncertain to what extent the key tokens, i.e. $\mathbb{P}(\text{"kitchen"}|p, \text{"Walk to"})$, will be enhanced when optimizing the joint probability $\mathbb{P}(\text{"Walk to kitchen"}|p)$.

next-token predictors with the RL objective of maximizing actions' utilities. However, they still suffer from limitations in optimization and exploration efficiency, due to the uncertainty of credit assignment to intra-action tokens. As shown in Figure 1, when optimizing the distribution over three candidate actions that only differ from the last token, action-level policy optimization strategies in previous works cannot ensure that the probability of the key tokens, i.e. $\mathbb{P}(\text{"kitchen"}|p, \text{"Walk to"})$ here, will be enhanced precisely when optimizing the joint probability $\mathbb{P}(\text{"Walk to kitchen"}|p)$. Furthermore, optimizing at the action level poses the challenge of overlarge optimization complexity due to exponentially growing action spaces, leading GLAM and TWOSOME to manually constrain action spaces. But they remain incapable in environments whose action spaces cannot be restricted.

A natural attempt to solve these problems is to incorporate the token generation process in each decision step as part of the sequential decision-making process [10–12]. This approach resolves the credit assignment problem and reduces the growth of optimization complexity from multiplicative to additive as the number of intra-action tokens increases, by updating each token's output distribution separately with fine-grained signals from value backups. Empirical success in many single-step tasks, e.g. question answering [13] and alignment [10, 11], have demonstrated its effectiveness. However, the multi-step nature of sequential decision-making tasks leads to extra difficulties in coordinating the credit assignment process across actions and their constituent tokens. In such scenarios, embedding the intra-action token generation process into the original Markov Decision Process (MDP) [14] results in a new MDP inconsistent with the original one. Intuitively speaking, it introduces an undesired assumption of "*in a sentence, tokens appearing later are more important for conveying meaning than those appearing earlier.*" This assumption, however, is unrealistic due to the nature of linguistic actions. Such a significant discrepancy has been ignored in previous research and has not been tackled properly for a long time.

In this paper, we provide a comprehensive theoretical analysis of the discrepancy and derive the Bellman backup with Action Decomposition (BAD), which guarantees theoretical consistency with optimizing the original MDP. While being possible to seamlessly integrate with a variety of traditional RL methods, in this work, we apply BAD to Proximal Policy Optimization (PPO) [15], resulting in the formulation of Policy Optimization with Action Decomposition (POAD). Benefiting from the finer-grained supervision afforded by BAD, POAD mitigates the uncertainty in the credit assignment process described in Figure 1, thereby enjoying better interpretability, lower optimization complexity, and higher training efficiency. Meanwhile, it theoretically maintains consistency between the token-level training process for language models and the RL objective of maximizing actions' utilities. We justify our claims by evaluating POAD in both classical sequential decision-making environments with limited action space, i.e Overcooked and VirtualHome [5], and a self-constructed data science coding environment featuring an unrestricted action space, i.e. DataSciCoding; results verify POAD's advantages in performance and efficiency over baseline methods, highlighting the significance of BAD. Moreover, we empirically demonstrate that language agents trained with POAD exhibit excellent generalization ability across unseen tasks, without compromising the inherent functionalities of language models. Finally, ablation studies confirm the correctness of our theoretical insights.

2 Related Works

Language-based Decision-Making Agents. Leveraging LLMs' powerful capability and plenty of common knowledge, recent efforts successfully adapt LLMs in decision-making tasks as a policy model in interactive environments. In robotics, LLMs have been employed as high-level planners of control policies [16–18]. Similarly, LLMs work particularly well in text-based environments [19, 20]. ReAct [21] combines chain-of-thought reasoning [2] with acting-based prompt, efficiently solving hard decision-making tasks. Self-Refine [22] and Reflexion [23] further improve language agents' efficiency and robustness via online adaptation through the self-reflection process. In this work, we also apply language agents in sequential decision-making scenarios, i.e. interactive environments.

Fine-tuning Language Models with RL. A body of literature explores the prospects of leveraging strategic planning methodologies to enhance the performance of language agents [7, 24, 25, 6]. Besides, RL has also been widely applied in fine-tuning LLMs [10, 11, 7, 24, 25, 6]. Particularly, proximal policy optimization (PPO) [15] is the most commonly used RL method for reinforcement learning from human feedback (RLHF), proposing a breakthrough in aligning LLMs with human preference [10, 11]. In classical sequential decision-making scenarios, to align the objective of token-level optimization with action-level optimization, GLAM [4] and TWOSOME [5] estimate the probability of possible actions with the products of the conditional probability of tokens composing the actions and update the action as a whole. In this work, instead of treating an action as a whole, we attempt to decompose actions and explicitly assign precise credit to each intra-action token, while ensuring that its optimality is consistent with updates at the action level. While a concurrent work, ArCHer [26], also targets token-level supervision for LLMs in interactive environments, it employs a hierarchical RL framework, using a Q-network for action-level credit approximation and REINFORCE [27] for token-level backpropagation. However, ArCHer's use of multiple value networks (Q, V, and optional baseline networks) demands extensive manual tuning and can introduce cumulative bias and variance, potentially affecting stability.

Sequential Decomposition in RL. Recent years have witnessed an increasing trend of decomposing high-dimension actions to leverage the powerful modeling abilities of sequential models like Transformer [28] in RL problems [29–34]. While Kuba et al. [35, 36] proposing a sequential decomposition method to provide finer-grained supervision for multi-agent joint actions, Multi-agent Transformer [31] inherits this idea and solves multi-agent RL problems with Transformer. More recently, Q-transformer [34] managed to decompose the Q-functions for high-dimensional actions by representing each action dimension as separate tokens. For language agents, the language generation process inherently conforms to the pattern of sequential decomposition, which offers a promising avenue for providing finer-grained supervision to intra-action tokens.

3 Preliminaries

3.1 Language-augmented RL

In this work, we assume a textual RL setting that frames sequential decision-making problems with linguistic inputs and outputs as a language-augmented Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{V}, \mathcal{S}, \mathcal{A}, \mathcal{T}, R, \gamma)$ [37, 4]. Given $\mathcal{V}$ the vocabulary and $w \in \mathcal{V}$ the tokens, $\mathcal{A} \subset \mathcal{V}^N$, $\mathcal{S} \subset \mathcal{V}^N$ are the action space and state space respectively, i.e. actions and states are sequences of tokens. $\mathcal{T} : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ is the state transition function. $R : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function that only responds to complete actions, and γ is the discounted factor that typically less than 1. At time step $t \in \mathbb{N}$, a language agent receives a textual state $s_t \in \mathcal{S}$ from an interactive environment as input and generates an action $a_t \in \mathcal{A}$ in an auto-regressive manner, i.e. $a_t = (w_t^1, \ldots, w_t^{|a_t|})$ where $|a_t|$ denotes the number of tokens in the action string and $\{w_t^i\}_{i=1}^{|a_t|}$ are tokens in it. Then, textual action a_t will be grounded to a specific API call or command in the environment [38]. After execution, the language agent receives a reward $r_t = R(s_t, a_t)$ along with the next state s_{t+1}, based on the transition function $\mathcal{T}$. Following this process with trajectories of a maximum timestep T, the agents earn a discounted cumulative return of $R^\gamma = \sum_{t=0}^{T} \gamma^t r_t$, which is aimed to be maximized by RL algorithms.

 https://doi.org/10.52202/079017-3297

3.2 Action-level Policy Optimization

We begin by briefly reviewing the process of action-level policy optimization which is widely adopted in several state-of-the-art methods that align language agents with environments via RL algorithms [4, 5, 39]. It facilitates seamless integration between any textual environment and conventional RL algorithms and thus is an ideal starting point for our analysis.

The possibly achieved episodic return following policy π given action and state is usually evaluated by state-action value function $Q_\pi(s, a)$ or state value function $V_\pi(s)$. Then, a language agent updates its policy π according to credits calculated on the value functions, defined as

$$Q_\pi(s, a) \triangleq \mathbb{E}_{s_{1:T}\sim\mathcal{T}, a_{1:T}\sim\pi}\big[R^\gamma | s_0 = s, a_0 = a\big], \tag{1}$$

$$V_\pi(s) \triangleq \mathbb{E}_{s_{1:T}\sim\mathcal{T}, a_{0:T}\sim\pi}\big[R^\gamma | s_0 = s\big], \tag{2}$$

where $a = (w^1, \ldots, w^{|a_t|}) = w^{1:|a|}$. While Ahn et al. [16] builds the connection between the likelihoods of actions and tokens through the chain rule of probability as

$$\pi(a|s) = \prod_{j=1}^{|a|} \pi(w^j | s, w^{1:j-1}), \tag{3}$$

recent approaches like GLAM [4] and TWOSOME [5] leverage similar ideas and optimize action-level likelihoods with RL methods directly. When considering optimizing for $\pi(a|s)$, Equations 1 and 2 are aligned with the definition of the value function in traditional RL settings, allowing them to be updated with traditional Bellman backup [9]

$$Q_\pi(s_t, a_t) \leftarrow R(s_t, a_t) + \gamma \max_{a_{t+1}} Q_\pi(s_{t+1}, a_{t+1}), \tag{4}$$

$$V_\pi(s_t) \leftarrow R(s_t, a_t) + \gamma V_\pi(s_{t+1}). \tag{5}$$

Moreover, it is noteworthy that Equation 3 calculates the likelihood of action a in an exponentially growing language space as $|a|$ increases, i.e. $|\mathcal{A}| = |\mathcal{V}|^{|a|}$. Exploration and optimization in such a huge action space are typically intractable for RL methods. Therefore, in the settings of GLAM and TWOSOME, the feasible action space is significantly restricted and smaller than the entire language space, i.e. $|\mathcal{A}| \ll |\mathcal{V}|^{|a|}$. Taking TWOSOME as an example, it optimizes the likelihood of action a concerning the feasible action space with Equation 6 to mask out invalid outputs.

$$\pi_{\text{TWOSOME}}(a|s) = \frac{\exp(\log \pi(a|s)/L(a))}{\sum_{a'\in\mathcal{A}} \exp(\log \pi(a'|s)/L(a'))}. \tag{6}$$

$L(a)$ indicates the number of tokens or words in the action prompt, utilized as a normalization term to mitigate the effects of varying action lengths.

Underpinned by Equation 3, GLAM and TWOSOME ensure the consistency between token-level optimization for language models and action-level optimization in an RL manner, without the need to explicitly assign credits for intra-action tokens. However, the jointness of the objective causes difficulties associated with the uncertainty in the credit assignment process [40, 41]—as shown in Figure 1, after assigning credit to an action, it's unsure whether key tokens in this action have been identified, and how much they are influenced. Thus, conducting RL training at the action level introduces uncertainty, which may lead to an inefficient learning process for the language agent.

4 From Actions to Tokens: Naive Token-level Policy Optimization

4.1 Naive Token-level Policy Optimization

To address the unclear credit assignment issue described in Figure 1 and Section 3.2, our target is to provide finer-grained supervision for each token during update while maintaining consistency in the optimality with action-level optimization, i.e. maximizing agents' cumulative returns. For arbitrary subsets of actions $w^{1:j}$ with $j \leq |a|$, we define token value functions for supervising policy update as

$$Q_\pi(s, w^{1:j-1}, w^j) \triangleq \mathbb{E}\big[R^\gamma | s_0 = s, w_0^{1:j-1} = w^{1:j-1}, w_0^j = w^j\big], \tag{7}$$

$$V_\pi(s, w^{1:j-1}) \triangleq \mathbb{E}\big[R^\gamma | s_0 = s, w_0^{1:j-1} = w^{1:j-1}\big]. \tag{8}$$

A natural approach to approximate $Q_\pi(s, w^{1:j-1}, w^j)$ and $V_\pi(s, w^{1:j-1})$ is conceptualizing the token generation process as part of the MDP, where each token is treated as a micro action. This enables

back-propagating credit among all tokens to furnish detailed supervision. Such an idea is borrowed from the modeling process of RLHF in general language tasks[10, 42]. In this way, the token-level Bellman backup corresponding to Equations 4 and 5 can be expressed as

$$Q_\pi(s_t, w_t^{1:j-1}, w_t^j) \leftarrow \begin{cases} 0 + \gamma_w \max_{w_t^{j+1}} Q_\pi(s_t, w_t^{1:j}, w_t^{j+1}), & \text{if } j < |a_t| \\ R(s_t, a_t) + \gamma_a \max_{w_{t+1}^1} Q_\pi(s_{t+1}, w_{t+1}^1), & \text{if } j = |a_t| \end{cases}, \tag{9}$$

$$V_\pi(s_t, w_t^{1:j}) \leftarrow \begin{cases} 0 + \gamma_w V_\pi(s_t, w_t^{1:j+1}), & \text{if } j < |a_t| \\ R(s_t, a_t) + \gamma_a V_\pi(s_{t+1}, \emptyset), & \text{if } j = |a_t| \end{cases}. \tag{10}$$

To facilitate subsequent theoretical analysis, we separate the discount factor γ into intra-action γ_w and inter-action γ_a, despite their numerical equivalence here. The above backups can be interpreted as applying RL algorithms on a modified reward function, which maintains action-level feedback while introducing extra 0 feedback for tokens within an action, except for the last token. Intuitively, this approach seems feasible and decomposes the action-level reward signal $R(s_t, a_t)$ to intra-action tokens, thus alleviating the uncertainty in credit assignment [31] and reducing optimization complexity. However, it must be noted that the optimization of Equations 9 and 10 are inconsistent with those of Equations 4 and 5 for sequential decision-making tasks since it introduces a new MDP $\bar{\mathcal{M}}$ that diverges from the origin one. We will analyze this discrepancy in the following section.

4.2 The Discrepancy

To maintain the consistency between action-level updates and token-level updates, we should ensure that optimizing over tokens gives the same optimality as optimizing the whole action, which is analogous to the multi-dimensional action optimization setting in traditional RL[34]. That is, given the deterministic nature of linguistic action generation, and an optimal polity π^* after sufficient training following the token-level Bellman backups, ideal optimal token value functions should satisfy $Q_{\pi^*}(s_t, w_t^{1:j-1}, w_t^j) = Q_{\pi^*}(s_t, a_t)$ and $V_{\pi^*}(s_t, w_t^{1:j}) = V_{\pi^*}(s_t)$ for $\forall j < |a_t|$. To quantify the discrepancy between action-level optimization and token-level optimization with the shape of Equations 9 and 10, we expand the value backup process over each token starting from arbitrary $j < |a_t|$ as the following equations (For derivation see Appendix A).

$$Q_{\pi^*}(s_t, w_t^{1:j-1}, w_t^j) = \underbrace{R(s_t, a_t) + \gamma_a \max_{a_t} Q_{\pi^*}(s_{t+1}, a_{t+1})}_{Q_{\pi^*}(s_t, a_t)} \tag{11}$$
$$- \underbrace{\left[(1 - \gamma_w^{|a_t|-j}) R(s_t, a_t) + \gamma_a (1 - \gamma_w^{|a_t|+|a_{t+1}|-j-1}) \max_{a_{t+1}} Q_{\pi^*}(s_{t+1}, a_{t+1})\right]}_{\text{Discrepancy between Equation 4 and 9}},$$

$$V_{\pi^*}(s_t, w_t^{1:j}) = \underbrace{R(s_t, a_t) + \gamma_a V_{\pi^*}(s_{t+1})}_{V_{\pi^*}(s_t)} - \underbrace{\left[(1 - \gamma_w^{|a_t|-j}) R(s_t, a_t) + \gamma_a (1 - \gamma_w^{|a_t|-j}) V_{\pi^*}(s_{t+1})\right]}_{\text{Discrepancy between Equation 5 and 10}}, \tag{12}$$

With Equation 11 and 12, we observe following significant insights:

(i) *The discrepancy of the Bellman optimality between action-level optimization and token-level optimization diminishes as $\gamma_w \in [0, 1]$ increases, achieving consistency when $\gamma_w = 1$.*

(ii) *Given $\gamma_w < 1$, the discrepancy increases as the number of intra-action tokens, $|a_t|$, increases.*

Based on the first insight, we will derive our main approach, namely the Bellman backup with Action-Decomposition, in the next section. The second insight can be intuitively understood as follows: The diverged MDP $\bar{\mathcal{M}}$ attaches the assumption that "*words later in a sentence are more expressive than earlier ones*", which is unrealistic for representing the semantics of linguistic actions.

 https://doi.org/10.52202/079017-3297

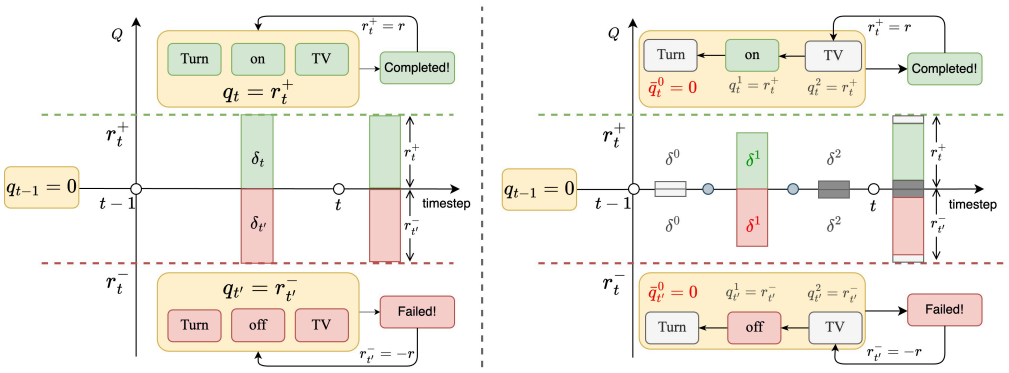


Figure 2: Visual comparison of the differences between action-level Bellman backup (left) and our BAD (right), given the goal *turn on the TV*, where q is the action or token value estimations, $\delta_t = q_t - q_{t-1}$ or $\delta^j = q^j - q^{j-1}$ represent the credit assigned to corresponding actions and tokens respectively for policy update, e.g. the advantage value[43]. To facilitate understanding, a step-by-step breakdown of the right figure is provided in Appendix L.

5 Action-Decomposition Reinforcement Learning

5.1 Bellman Backup with Action-Decomposition

According to the first insight, we can modify Equations 9 and 10, proposing the *Bellman backup with Action-Decomposition (BAD)* as

$$Q_\pi(s_t, w_t^{1:j-1}, w_t^j) \leftarrow \begin{cases} \max_{w_t^{j+1}} Q_\pi(s_t, w_t^{1:j}, w_t^{j+1}), & \text{if } j < |a_t| \\ R(s_t, a_t) + \gamma \max_{w_{t+1}^1} Q_\pi(s_{t+1}, w_{t+1}^1), & \text{if } j = |a_t| \end{cases}, \tag{13}$$

$$V_\pi(s_t, w_t^{1:j}) \leftarrow \begin{cases} V_\pi(s_t, w_t^{1:j+1}), & \text{if } j < |a_t| \\ R(s_t, a_t) + \gamma V_\pi(s_{t+1}, \emptyset), & \text{if } j = |a_t| \end{cases}. \tag{14}$$

(For proof of optimization consistency see Appendix B.) Training language agents with BAD provides finer-grained supervision for credit backpropagation, eliminating uncertainty in credit assignment and thus enjoying better interpretability and efficiency of the RL training process. In addition, it theoretically ensures consistency between the token-level training process for language models and the RL objective of maximizing actions' utilities. Figure 2 visually compares the differences between action-level Bellman backup and our BAD and demonstrates how BAD precisely assigns credit to each token.

Another advantage of BAD is the ability to decompose the optimization in an intractable action space of size $O(|V|^{|a|})$, into $|a|$ times optimizations in token spaces of size $O(|V|)$, reducing the complexity of RL problem with language agent to $O(|a| \times |V|)$, thus rendering the problem more manageable. Moreover, BAD can be seamlessly integrated into various existing RL methods, including off-policy algorithms e.g. DQN[44], on-policy ones like Actor-Critic[45] and PPO[15]. Furthermore, we have also provided a version of the Soft Q-function in Appendix B.3 to support various entropy-regularized RL algorithms like SAC[46, 47].

5.2 Policy Optimization with Action Decomposition

In this section, we integrate the BAD into the widely used PPO and propose a specific method called *Policy Optimization with Action Decomposition (POAD)*, while the integration with other algorithms will be reserved for future works. POAD sequentially decomposes the policy update granularity from the action level to the token level. To approximate the token value function, we introduce a critic network with parameters ϕ whose objective is to minimize the empirical Bellman error of tokens by

$$L(\theta) = \frac{1}{T}\sum_{t=0}^{T-1}\left[\frac{1}{|a_t|}\left(\underbrace{\left[R(s_t,a_t) + \gamma V_\theta(s_{t+1},\emptyset) - V_\theta(s_t, w_t^{1:|a_t|})\right]^2}_{\text{Inter-action credit assignment}} + \underbrace{\sum_{j=1}^{|a_t|-1}\left[V_\theta(s_t, w_t^{1:j+1}) - V_\theta(s_t, w_t^{1:j})\right]^2}_{\text{Intra-action credit assignment}}\right)\right], \tag{15}$$

where $\bar{\theta}$ represents the non-differentiable parameter of the target network, updated at intervals. The policy network's parameters are denoted as ϕ, and optimization follows the clipping PPO objective.

$$L(\phi) = -\frac{1}{T}\sum_{t=0}^{T-1}\frac{1}{|a_t|}\sum_{j=1}^{|a_t|}\left[\min\left(\text{ratio}_t^j(\phi)\hat{A}_t^j, \text{clip}(\text{ratio}_t^j(\phi), 1\pm\epsilon)\hat{A}_t^j\right)\right], \tag{16}$$

$$\text{ratio}_t^j(\phi) = \frac{\pi_\phi(w_t^j|s_t, w_t^{1:j-1})}{\pi_{\phi_{old}}(w_t^j|s_t, w_t^{1:j-1})}, \text{and } \hat{A}_t^j = \hat{A}(w_t^j|s_t, w_t^{1:j-1}),$$

where $\hat{A}_t^j$ is an estimate of the advantage value for each token with the *generalized advantage estimation* (GAE) [48]. To capture more details about POAD, we draw a pseudo-code in Appendix D.

6 Experiments

In this section, we show the superiority of POAD in performance, efficiency, and generalization abilities with different testbeds. Moreover, we conduct meaningful ablations on γ_a and γ_w to verify the theoretical analysis in Section 4.2. Finally, we examine models trained with POAD and baseline methods to investigate their impact on the model's original language abilities. For in-depth analysis, we conduct a case study in Appendix F to validate the effectiveness of BAD in terms of token-level credit assignment. We deploy LLaMA2-7B [49] for Overcooked and VirtualHome, and CodeLLaMA-7b [50] for DataSciCoding, fine-tuned with Low Rank Adaptation (LoRA) [51] with 1 Nvidia A100 GPU.

6.1 Environmental Setup

We first evaluate our method on two classical sequential decision-making environments with limited action space: Overcooked [5] and VirtualHome [5], where the action space consists of approximately 10 possible actions per state, each includes 5-10 tokens. Then we evaluate our method in a data science coding and debugging environment with unrestricted action space: DataSciCoding, where agents generate actions (up to 128 tokens) freely. More detailed descriptions can be found in Appendix E.

Overcooked and VirtualHome. Overcooked challenges agents to prepare dishes such as *tomato salad* and *tomato-lettuce salad* in a 7x7 grid kitchen using linguistic actions like *Chop, Get-Tomato, and Go-Cutting-Board*, with rewards for correct deliveries and penalties for incorrect ones or time wastage. Meanwhile, VirtualHome simulates household tasks like reheating *pancakes* in Food Preparation and organizing an evening of Entertainment, with actions such as *walk to the living room* and *turn on the TV*, rewarding agents only upon task completion in a partially observable setting.

DataSciCoding. We develop DataSciCoding to automate data science coding tasks with unrestricted action space, currently adopting 3 Kaggle datasets and 3 OpenML datasets [52] with details in Appendix E.1. In each task, agents aim to implement the most effective classifier with the scikit-learn module, striving to achieve the highest possible ROC AUC score [53] on test sets. As the prompts provided to the agents contain no detailed information about task datasets, agents are required to interactively modify and debug their code based on feedback from the runtime environment until it works, thus aligning the task datasets. Agents receive ROC AUC scores $\in [0, 1]$ as rewards for workable codes and -1 as penalties for run failed. Adopting the same evaluation metrics as CAAFE [54], for each dataset and code, we evaluate 5 repetitions, each with a random $50\% - 50\%$ train-test split [55], and record the average ROC AUC score across these splits.

6.2 Baseline Methods

For Overcooked and VirtualHome, we compare POAD's performance with Naive Token-Level Policy Optimization (NTPO) mentioned in Section 4.1, i.e. integrating Equation 10 with $\gamma_w = \gamma_a$ into PPO, and TWOSOME [5]—the current state-of-the-art method on Overcooked and VirtualHome. Besides, we also incorporate ArCHer [26] as a baseline in VirtualHome with comparative analysis, since it is positioned as an intermediate between NTPO and POAD for token-level credit assignment, theoretically enjoying reduced discrepancy compared to NTPO. We demonstrate the difference in the backup processes between POAD and these baselines as well as the optimality after convergence in Appendix C

 https://doi.org/10.52202/079017-3297

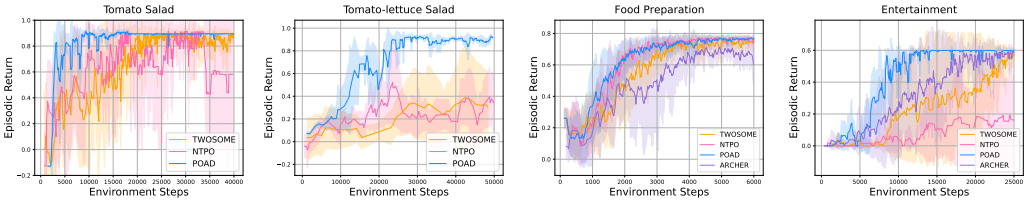


Figure 3: Performance comparisons on Overcooked (first two) and VirtualHome (last two).

For the DataSciCoding environment, in case TWOSOME is inapplicable due to the unrestricted action space, we examine POAD's performance along with NTPO. Meanwhile, we also compare POAD with CAAFE [54]—a novel AutoML framework in which humans collaborate with large language models for data science. In CAAFE, humans implement machine learning models, e.g. classifiers, while large language models generate the code for feature engineering. CAAFE represents the current state-of-the-art performance with collaboration between humans and powerful closed-source LLMs such as GPT-3.5 and GPT-4 on the given task datasets we used.

6.3 Main Results

6.3.1 Classical Sequential Decision-Making Tasks with Restricted Action Space

As shown in Figure 3, where curves are averaged over 3 seeds with the shadow showing their standard deviation, the performance drop of NTPO when comparing with POAD verifies the existence and negative impact of the discrepancy analyzed in Section 4.2. While POAD can achieve the same (or even better) convergence performance compared to TWOSOME which further verifies the consistency between token-level optimization with BAD and action-level optimization. In addition, the training curves of POAD are more stable (enjoying smaller standard deviations) than TWOSOME and converge much faster than all other baselines, indicating POAD's stability and efficiency by integrating our BAD. In complex Entertainment tasks, ArCHer is second only to POAD, aligning with our theoretical expectations. However, in tasks such as Food Preparation where the methods' performance gap was less pronounced, ArCHer performed poorly, potentially due to instability in its system that involved multiple value networks.

6.3.2 Data Science Coding Tasks with Unrestricted Action Space

According to the training curves in Figure 4, our method POAD significantly outperforms NTPO both in the convergence speed and the final score. Compared to the results on sequential decision-making tasks in Section 6.3.1, the performance gap between POAD and NTPO on DataSciCoding is consistently larger, due to the much longer action length $|a_t|$, at most 128 tokens for each action in DataSciCoding. Such results are consistent with our second insights in Section 4.2 and empirically highlight the importance of a proper intra-action credit assignment, which, our Bellman Backup with Action Decomposition (BAD) is designed for.

In Figure 4, POAD-Best means the performance of the best code discovered during POAD's training process with CodeLLaMA-7B. We compare it with the best performance achieved by the state-of-the-art AutoML framework CAAFE [54] with GPT-4 model. In this experiment, we aim to prove that even small-scale language models can also provide better outcomes than large-scale models, what is needed is just a stable and efficient training algorithm POAD and only 2-3 hours on Nvidia A100 (Details of wall-time on each task are shown in Appendix H). A more detailed Comparison between POAD and CAAFE with both GPT-3.5 and GPT-4 can be found in Appendix G.

6.4 Open-Vocabulary Task Generalization

LLMs' open-vocabulary feature enables language agents to transfer their learned skills into unseen similar tasks, expanding the capabilities of decision-making agents. We compare the generalization performance of language agents trained by POAD, TWOSOME and NTPO in the *Food Preparation* task with the original base model LLaMA2-7B. Table 1 shows that token-level policy optimization methods achieve better generalization performance in unseen tasks. And our POAD outperforms the other baselines in seven of eight tasks.

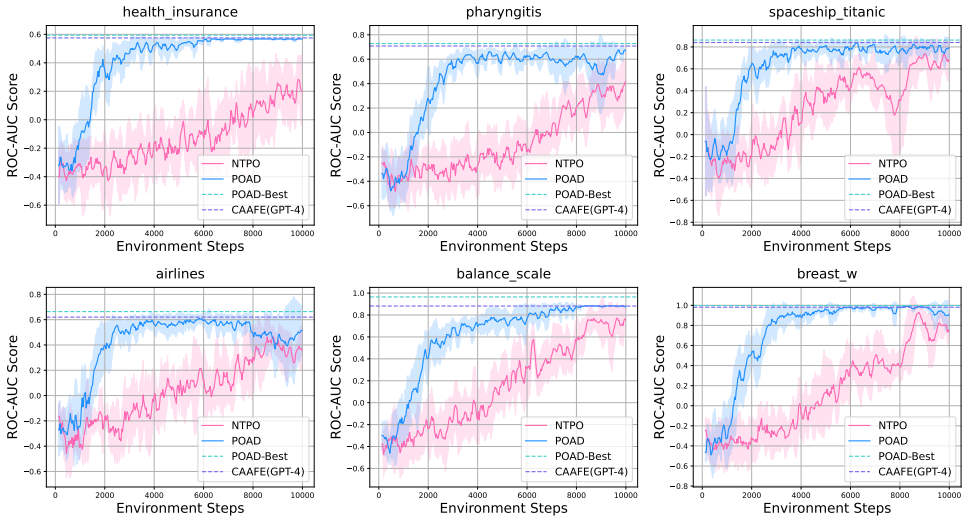


Figure 4: While TWOSOME does not support open action space tasks, we compare the average performance between POAD and NTPO on the DataSciCoding benchmarks, as well as POAD-Best the performance of best code explored by POAD during the training phase and CAAFE with GPT-4.

Table 1: Comparison of generalization performance on eight unseen tasks, with episodic returns averaged over 100 episodes. In these tasks, we replace the *pancake* in the original Food Preparation task with other foods like *cheese*, *hamburger*, *apple pie* and *pizza*, or replace the (*pancake*, *microwave*) at the same time with (*dishes*, *dishwasher*) or (*clothes*, *washing machine*) for greater differences.

Methods	Cheese	Hamburger	Apple Pie	Pizza	Washing Plate	Laundry
LLaMA2-7B	0.1351	0.1342	0.1656	0.1409	0.0527	0.0344
TWOSOME	0.7119	0.7058	0.7304	0.7047	0.7031	0.6038
NTPO	0.7428	0.7476	0.7141	0.7355	**0.7491**	0.5687
POAD	**0.7553**	**0.7602**	**0.7650**	**0.7625**	0.7075	**0.7014**

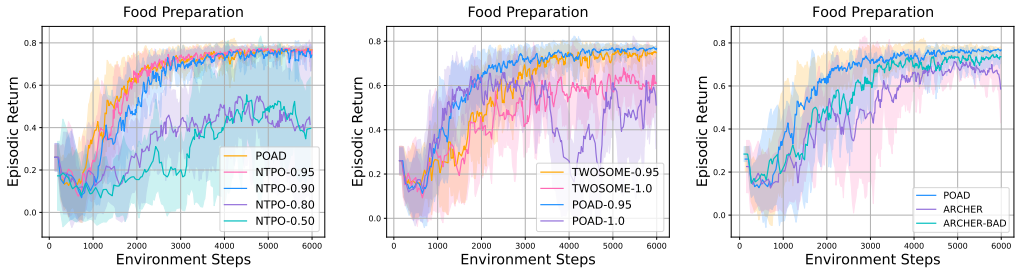


Figure 5: Ablation on $\gamma_a \in \{1.0, 0.95\}$ for both TWOSOME and POAD (left), and $\gamma_w \in \{0.95, 0.9, 0.8, 0.5\}$ for NTPO while keeping the $\gamma_a = 0.95$ unchanged (middle). In the left figure, Setting $\gamma_a = 1.0$ led to decreased performance and convergence for TWOSOME and POAD, validating necessity of $\gamma_a < 1.0$. While in the right figure, the increasingly larger performance gap between POAD and NTPO, as γ_w decreases, verifies the theoretical analysis in Section 4.2. The right one shows the performance change after applying theoretical insights to enhance ArCHer's performance, i.e. ARCHER-BAD with $\gamma_w = 1.0$.

6.5 Ablations: the Impact of γ_a and γ_w to the Discrepancy

To verify our analysis in Section 4.2, we conduct ablations on γ_w to further investigate how large the discrepancy would be by using different values of γ_w in NTPO. Therefore, we deploy NTPO on the *Food Preparation* task with $\gamma_w \in \{0.95, 0.9, 0.8, 0.5\}$, while we keep $\gamma_a = 0.95$ which is consistent with our main experiments. Besides, we also show the performance with $\gamma_a \in \{1.0, 0.95\}$ to show the necessity of setting γ_a strictly less than 1.0, and thus the necessity to separate inter-action tokens and intra-action tokens. Further, our theoretical analysis is also compatible with ArCHer, motivating us to apply insights in Section 4.2 to enhance ArCHer's performance.

https://doi.org/10.52202/079017-3297

The left side of Figure 5 demonstrates an increasingly larger performance gap between POAD ($\gamma_w = 1$) and NTPO as γ_w decreases. These results empirically showcase the discrepancy between naive token-level credit assignment and our BAD, which is consistent with our first theoretical insight in Section 4.2. Moreover, in the middle of Figure 5, for both TWOSOME and our proposed method POAD, setting $\gamma_a = 1.0$ performs much worse than $\gamma_a = 0.95$, indicating that the discrepancy between NTPO and POAD can not be solved by simply setting $\gamma_a = \gamma_w = 1.0$. Furthermore, the right one in Figure 2 shows improved results that validate the effectiveness of applying our insights to ArCHer. However, due to the inherent challenges of excessive network complexity and difficult hyper-parameter tuning, ArCHer-BAD still falls short of matching POAD's performance.

6.6 Impact on Original Language Ability

To investigate the impacts of online RL fine-tuning on LLMs' original capabilities, we evaluate the models trained by POAD, TWOSOME and NTPO on widely used NLP benchmarks[56] which are also reported in Tan et al. [5] and Touvron et al. [49]. These models are trained in *Food Preparation*. Table 2 demonstrates these models' zero-shot performance, compared with the original LLaMA2-7B model. The results show no significant losses of general ability like natural language understanding after aligning with the embodied environment, even sometimes bringing minor improvements.

Table 2: Zero-shot performance on Language Model Evaluation Harness [56], with details in Appendix J.

Methods	ARC_C	HellaSwag	PIQA	MMLU
LLaMA2-7B	0.44	0.57	0.78	0.41
TWOSOME	0.44	0.58	0.78	0.41
NTPO	0.44	0.58	0.78	0.41
POAD	0.45	0.59	0.78	0.41

7 Conclusion

In this work, we propose the Bellman backup with Action Decomposition (BAD), theoretically eliminating the discrepancy between naive token-level policy optimization and action-level policy optimization for language agents. Integrating BAD with PPO, we propose our method of Policy Optimization with Action Decomposition (POAD), providing finer-grained supervision for each intra-action token and ensuring theoretical consistency between the token-level training nature of language models and the RL objective of maximizing actions' utilities. Empirical experiments and thorough ablations showcase the effectiveness of BAD as well as the superiority of POAD in learning efficiency and generalization abilities, over strong action-level baseline TWOSOME.

Limitation and Future Work. The existing limitation of POAD is on the requirement for a quantitative reward function, which is not easily attainable in some environments. To mitigate this, we envisage integrating POAD with self-rewarding [57, 58] or hindsight relabeling [59].

Social Impact. The advancements in RL for language agents can significantly enhance decision-making processes in various domains such as healthcare, finance, and autonomous systems. Improved decision-making can lead to better outcomes, increased efficiency, and reduced errors. However, we acknowledge that when optimizing agents using our method, language agents may potentially resort to unscrupulous means to maximize rewards, which could lead to potentially harmful results. Thus, we advocate for a more comprehensive consideration when designing the reward function, or combining it with safety-constrained RL methods to mitigate these risks.

Acknowledgment

The SJTU team is partially supported by National Key R&D Program of China (2022ZD0114804), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102) and National Natural Science Foundation of China (62322603, 62076161, 62106141). Muning Wen is supported by Wu Wen Jun Honorary Scholarship, AI Institute, Shanghai Jiao Tong University.

References

[1] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

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

[3] Xidong Feng, Yicheng Luo, Ziyan Wang, Hongrui Tang, Mengyue Yang, Kun Shao, David Mguni, Yali Du, and Jun Wang. Chessgpt: Bridging policy learning and language modeling. *Advances in Neural Information Processing Systems*, 36, 2024.

[4] Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. Grounding large language models in interactive environments with online reinforcement learning. In *International Conference on Machine Learning*, pages 3676–3713. PMLR, 2023.

[5] Weihao Tan, Wentao Zhang, Shanqi Liu, Longtao Zheng, Xinrun Wang, and Bo An. True knowledge comes from practice: Aligning large language models with embodied environments via reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=hILVmJ4Uvu`.

[6] Xidong Feng, Ziyu Wan, Muning Wen, Ying Wen, Weinan Zhang, and Jun Wang. Alphazero-like tree-search can guide large language model decoding and training. *arXiv preprint arXiv:2309.17179*, 2023.

[7] 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, 2024.

[8] Fengshuo Bai, Hongming Zhang, Tianyang Tao, Zhiheng Wu, Yanna Wang, and Bo Xu. Picor: Multi-task deep reinforcement learning with policy correction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(6):6728–6736, Jun. 2023. doi: 10.1609/aaai.v37i6.25825. URL `https://ojs.aaai.org/index.php/AAAI/article/view/25825`.

[9] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

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

[11] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

[12] Rafael Rafailov, Joey Hejna, Ryan Park, and Chelsea Finn. From r to q* : Your language model is secretly a q-function. *arXiv preprint arXiv:2404.12358*, 2024.

[13] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

[14] Frédérick Garcia and Emmanuel Rachelson. Markov decision processes. *Markov Decision Processes in Artificial Intelligence*, pages 1–38, 2013.

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

[16] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

 https://doi.org/10.52202/079017-3297

[17] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. In *6th Annual Conference on Robot Learning*, 2022.

[18] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

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

[20] 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*, 2023.

[21] 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*, 2022.

[22] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[23] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[24] 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, 2024.

[25] Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*, 2023.

[26] Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. Archer: Training language model agents via hierarchical multi-turn rl. In *Forty-first International Conference on Machine Learning*, 2024.

[27] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[29] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[30] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

[31] Muning Wen, Jakub Kuba, Runji Lin, Weinan Zhang, Ying Wen, Jun Wang, and Yaodong Yang. Multi-agent reinforcement learning is a sequence modeling problem. *Advances in Neural Information Processing Systems*, 35:16509–16521, 2022.

[32] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[33] Ying Wen, Ziyu Wan, Ming Zhou, Shufang Hou, Zhe Cao, Chenyang Le, Jingxiao Chen, Zheng Tian, Weinan Zhang, and Jun Wang. On realization of intelligent decision-making in the real world: A foundation decision model perspective. *arXiv preprint arXiv:2212.12669*, 2022.

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

[35] Jakub Grudzien Kuba, Muning Wen, Linghui Meng, Haifeng Zhang, David Mguni, Jun Wang, Yaodong Yang, et al. Settling the variance of multi-agent policy gradients. *Advances in Neural Information Processing Systems*, 34:13458–13470, 2021.

[36] Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2021.

[37] Martijn Van Otterlo and Marco Wiering. Reinforcement learning and markov decision processes. In *Reinforcement learning: State-of-the-art*, pages 3–42. Springer, 2012.

[38] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36, 2024.

[39] Filippos Christianos, Georgios Papoudakis, Matthieu Zimmer, Thomas Coste, Zhihao Wu, Jingxuan Chen, Khyati Khandelwal, James Doran, Xidong Feng, Jiacheng Liu, et al. Pangu-agent: A fine-tunable generalist agent with structured reasoning. *arXiv preprint arXiv:2312.14878*, 2023.

[40] Zhipeng Chen, Kun Zhou, Wayne Xin Zhao, Junchen Wan, Fuzheng Zhang, Di Zhang, and Ji-Rong Wen. Improving large language models via fine-grained reinforcement learning with minimum editing constraint. *arXiv preprint arXiv:2401.06081*, 2024.

[41] Gregor Bachmann and Vaishnavh Nagarajan. The pitfalls of next-token prediction. *arXiv preprint arXiv:2403.06963*, 2024.

[42] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

[43] Mohammad Babaeizadeh, Iuri Frosio, Stephen Tyree, Jason Clemons, and Jan Kautz. Reinforcement learning through asynchronous advantage actor-critic on a gpu. *arXiv preprint arXiv:1611.06256*, 2016.

[44] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[45] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

[46] John Schulman, Xi Chen, and Pieter Abbeel. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*, 2017.

[47] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[48] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

 https://doi.org/10.52202/079017-3297

[49] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[50] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code, 2023.

[51] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[52] Joaquin Vanschoren, Jan N Van Rijn, Bernd Bischl, and Luis Torgo. Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

[53] Sarang Narkhede. Understanding auc-roc curve. *Towards data science*, 26(1):220–227, 2018.

[54] Noah Hollmann, Samuel Müller, and Frank Hutter. Large language models for automated data science: Introducing caafe for context-aware automated feature engineering. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[55] Xavier Bouthillier, Pierre Delaunay, Mirko Bronzi, Assya Trofimov, Brennan Nichyporuk, Justin Szeto, Nazanin Mohammadi Sepahvand, Edward Raff, Kanika Madan, Vikram Voleti, et al. Accounting for variance in machine learning benchmarks. *Proceedings of Machine Learning and Systems*, 3:747–769, 2021.

[56] Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, et al. A framework for few-shot language model evaluation. *Version v0. 0.1. Sept*, page 8, 2021.

[57] Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2024.

[58] Runze Liu, Fengshuo Bai, Yali Du, and Yaodong Yang. Meta-reward-net: Implicitly differentiable reward learning for preference-based reinforcement learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 22270–22284. Curran Associates, Inc., 2022. URL `https://proceedings.neurips.cc/paper_files/paper/2022/file/8be9c134bb193d8bd3827d4df8488228-Paper-Conference.pdf`.

[59] Alexander Li, Lerrel Pinto, and Pieter Abbeel. Generalized hindsight for reinforcement learning. *Advances in neural information processing systems*, 33:7754–7767, 2020.

A Derivation for the Discrepancy

A.1 Q-Function

The optimization over each token following Equation 9, starting from arbitrary $j < |a_t|$, could be expanded as

$$
\begin{aligned}
Q_{\pi^*}(s_t, w_t^{1:j-1}, w_t^j) &= \gamma_w \max_{w_t^{j+1}} Q_{\pi^*}(s_t, w_t^{1:j}, w_t^{j+1}) \\
&= \gamma_w \max_{w_t^{j+1}} \left[\gamma_w \max_{w_t^{j+2}} Q_{\pi^*}(s_t, w_t^{1:j}, w_t^{j+1}, w_t^{j+2}) \right] \\
&= \gamma_w^{|a_t|-j} R(s_t, a_t) + \gamma_a \gamma_w^{|a_t|-j} \max_{w_{t+1}^1} Q_{\pi^*}(s_{t+1}, w_{t+1}^1) \\
&= \gamma_w^{|a_t|-j} \left[R(s_t, a_t) + \gamma_a \max_{w_{t+1}^1} \left(\gamma_w \max_{w_{t+1}^2} Q_{\pi^*}(s_{t+1}, w_{t+1}^1, w_{t+1}^2) \right) \right] \\
&= \gamma_w^{|a_t|-j} \left[R(s_t, a_t) + \gamma_a \gamma_w \left(\max_{w_{t+1}^{1:2}} Q_{\pi^*}(s_{t+1}, w_{t+1}^1, w_{t+1}^2) \right) \right] \\
&= \gamma_w^{|a_t|-j} \left[R(s_t, a_t) + \gamma_a \gamma_w^{|a_{t+1}|-1} \max_{w_{t+1}^{1:|a_{t+1}|}} Q_{\pi^*}(s_{t+1}, w_{t+1}^{1:|a_{t+1}|}) \right] \\
&= \gamma_w^{|a_t|-j} \left[R(s_t, a_t) + \gamma_a \gamma_w^{|a_{t+1}|-1} \max_{a_{t+1}} Q_{\pi^*}(s_{t+1}, a_{t+1}) \right] \\
&= \underbrace{R(s_t, a_t) + \gamma_a \max_{a_{t+1}} Q_{\pi^*}(s_{t+1}, a_{t+1})}_{Q_{\pi^*}(s_t, a_t)} \\
&\quad - \underbrace{\left[(1 - \gamma_w^{|a_t|-j}) R(s_t, a_t) + \gamma_a (1 - \gamma_w^{|a_t|+|a_{t+1}|-j-1}) \max_{a_{t+1}} Q_{\pi^*}(s_{t+1}, a_{t+1}) \right]}_{\text{Discrepancy between Equation 4 and 9}}.
\end{aligned} \tag{17}
$$

A.2 V-Function

Similar to Appendix A.1, the optimization over each token following Equation 10, starting from arbitrary $j < |a_t|$, could be expanded as

$$
\begin{aligned}
V_{\pi^*}(s_t, w_t^{1:j-1}) &= \gamma_w V_{\pi^*}(s_t, w_t^{1:j}) \\
&= \gamma_w \left[\gamma_w V_{\pi^*}(s_t, w_t^{1:j+1}) \right] \\
&= \gamma_w^{|a_t|-j} R(s_t, a_t) + \gamma_a \gamma_w^{|a_t|-j} V_{\pi^*}(s_{t+1}) \\
&= \underbrace{R(s_t, a_t) + \gamma_a V_{\pi^*}(s_{t+1})}_{V_{\pi^*}(s_t)} - \underbrace{\left[(1 - \gamma_w^{|a_t|-j}) R(s_t, a_t) + \gamma_a (1 - \gamma_w^{|a_t|-j}) V_{\pi^*}(s_{t+1}) \right]}_{\text{Discrepancy between Equation 5 and 10}}.
\end{aligned} \tag{18}
$$

B Proof of Optimization Consistency

To show that optimizing value functions with BAD still ensures the same optimality with action-level optimization, we show that optimizing the value functions for each token is equivalent to optimizing for the full action.

https://doi.org/10.52202/079017-3297

B.1 Q Function

The optimization over each token with the optimal policy π^* following Equation 13, starting from arbitrary $j < |a_t|$, could be expanded as

$$\begin{aligned}
Q_{\pi^*}(s_t, w_t^{1:j-1}, w_t^j) &= \max_{w_t^{j+1}} Q_{\pi^*}(s_t, w_t^{1:j}, w_t^{j+1}) \\
&= \max_{w_t^{j+1}} \left[\max_{w_t^{j+2}} Q_{\pi^*}(s_t, w_t^{1:j}, w_t^{j+1}, w_t^{j+2}) \right] \\
&= R(s_t, a_t) + \gamma \max_{w_{t+1}^1} Q_{\pi^*}(s_{t+1}, w_{t+1}^1) \\
&= R(s_t, a_t) + \gamma \max_{w_{t+1}^1} \left(\max_{w_{t+1}^2} Q_{\pi^*}(s_{t+1}, w_{t+1}^1, w_{t+1}^2) \right) \\
&= R(s_t, a_t) + \gamma \left(\max_{w_{t+1}^{1:2}} Q_{\pi^*}(s_{t+1}, w_{t+1}^1, w_{t+1}^2) \right) \\
&= R(s_t, a_t) + \gamma \max_{w_{t+1}^{1:|a_{t+1}|}} Q_{\pi^*}(s_{t+1}, w_{t+1}^{1:|a_{t+1}|}) \\
&= R(s_t, a_t) + \gamma \max_{a_t} Q_{\pi^*}(s_{t+1}, a_{t+1}) \\
&= Q_{\pi^*}(s_t, a_t).
\end{aligned} \tag{19}$$

B.2 V Function

The optimization over each token following Equation 14, starting from arbitrary $j < |a_t|$, could be expanded as

$$\begin{aligned}
V_{\pi^*}(s_t, w_t^{1:j-1}) &= V_{\pi^*}(s_t, w_t^{1:j}) \\
&= V_{\pi^*}(s_t, w_t^{1:j+1}) \\
&= R(s_t, a_t) + \gamma V_{\pi^*}(s_{t+1}). \\
&= V_{\pi^*}(s_t),
\end{aligned} \tag{20}$$

where $V_{\pi^*}(s_t, w_t^{1:j-1})$ and $V_{\pi^*}(s_t, w_t^{1:|a_t|})$ are equivalent to $\mathbb{E}_{w_t^j \sim \pi^*}[Q(s_t, w_t^{1:j-1}, w_t^j)]$ and $\mathbb{E}_{w_{t+1}^1 \sim \pi^*}[Q(s_{t+1}, w_{t+1}^1)]$.

B.3 Soft Q function in Entropy-Regularized RL

As an extension of BAD, we also provide the soft Bellman backup with Action-Decomposition (sBAD), to support Entropy-Regularized methods like SAC (by setting the reference policy $\bar{\pi}$ to a uniform distribution), as

$$Q_\pi(s_t, w_t^{1:j-1}, w_t^j) = \begin{cases} \mathbb{E}_{w_t^{j+1} \sim \pi}[Q_\pi(s_t, w_t^{1:j}, w_t^{j+1})] - \beta D_{KL}[\pi || \bar{\pi}](s_t, w_t^{1:j}), & \text{if } j < |a_t| \\ R(s_t, a_t) + \gamma(\mathbb{E}_{w_{t+1}^1 \sim \pi}[Q_\pi(s_{t+1}, w_{t+1}^1)] - \beta D_{KL}[\pi || \bar{\pi}](s_{t+1})), & \text{if } j = |a_t| \end{cases}. \tag{21}$$

We show that optimizing the soft Q-function with sBAD at the token level is consistent with optimizing for the full action. We adopt $\mathrm{KL}(a|s) = D_{KL}[\pi^*||\bar{\pi}](s)$, where the $\mathrm{KL}(a|s)$ indicate it is a action level KL divergence, $\mathrm{KL}(w|s)$ indicate a token level KL divergence. Given an optimal stochastic policy π^*, the vanilla soft Bellman backup for full actions is:

$$\begin{aligned}
\mathbb{E}_{a_t \sim \pi^*}[Q(s_t, a_t)] &= \mathbb{E}_{a_t \sim \pi^*}\Big[R(s_t, a_t) + \gamma(\mathbb{E}_{a_{t+1} \sim \pi^*}[Q(s_t, a_t)] - \beta \mathrm{KL}(a_{t+1}|s_{t+1}))\Big] \\
&= \mathbb{E}_{a_t \sim \pi^*}[R(s_t, a_t)] + \gamma \mathbb{E}_{a_t, a_{t+1} \sim \pi^*}[Q(s_{t+1}, a_{t+1})] - \gamma\beta \mathbb{E}_{a_t \sim \pi^*}[\mathrm{KL}(a_{t+1}|s_{t+1})].
\end{aligned} \tag{22}$$

Following our sBAD, the update over each token within an action ($j < |a|$) is:

$$\begin{aligned}
\mathbb{E}_{w_t^1 \sim \pi^*}[Q(s_t, w_t^1)] &= \mathbb{E}_{w_t^1 \sim \pi^*}[\mathbb{E}_{w_t^2 \sim \pi^*}[Q(s_t, w_t^1, w_t^2)] - \beta \mathrm{KL}(w_t^2 | s_t, w_t^1)] \\
&= \mathbb{E}_{w_t^1, w_t^2 \sim \pi^*}[Q(s_t, w_t^1, w_t^2)] - \beta \mathbb{E}_{w_t^1 \sim \pi^*}[\mathrm{KL}(w_t^2 | s_t, w_t^1)] \\
&= \mathbb{E}_{w_t^1, \ldots, w_t^{|a|} \sim \pi^*}[Q(s_t, w_t^{1:|a_t|-1}, w_t^{|a_t|})] \\
&- \beta \mathbb{E}_{w_t^1, \ldots, w_t^{|a_t|-1} \sim \pi^*}[\sum_{j=1}^{|a_t|-1} \mathrm{KL}(w_t^{j+1} | s_t, w_t^{1:j})] \\
&= \mathbb{E}_{a_t \sim \pi^*}[Q(s_t, a_t)] \\
&- \beta \mathbb{E}_{w_t^1, \ldots, w_t^{|a_t|-1} \sim \pi^*}[\sum_{j=1}^{|a_t|-1} \mathrm{KL}(w_t^{j+1} | s_t, w_t^{1:j})]
\end{aligned} \tag{23}$$

Minus $\beta \mathrm{KL}(w_t^1 | s_t)$ from both sides, we have:

$$\mathbb{E}_{w_t^1 \sim \pi^*(s_t)}[Q(s_t, w_t^1)] - \beta \mathrm{KL}(w_t^1 | s_t) = \mathbb{E}_{a_t \sim \pi^*}[Q(s_t, a_t)] - \beta \mathrm{KL}(a_t | s_t), \tag{24}$$

where $\mathrm{KL}(a_t | s_t) = \sum_{j=1}^{|a_t|} \mathrm{KL}(w_t^j | s_t, w_t^{1:j-1})$.

Then, the update across the current action a_t and the next action a_{t+1} with our sBAD:

$$\begin{aligned}
\mathbb{E}_{a_t \sim \pi^*}[Q(s_t, a_t)] &= \mathbb{E}_{a_t \sim \pi^*}[R(s_t, a_t)] + \gamma(\mathbb{E}_{a_t \sim \pi^*, w_{t+1}^1 \sim \pi^*}[Q(s_{t+1}, w_{t+1}^1)] \\
&- \beta \mathbb{E}_{a_t \sim \pi^*}[\mathrm{KL}(w_{t+1}^1 | s_{t+1})]) \\
&= \mathbb{E}_{a_t \sim \pi^*}[R(s_t, a_t)] + \gamma \mathbb{E}_{a_t, a_{t+1} \sim \pi^*}[Q(s_{t+1}, a_{t+1})] \\
&- \gamma \beta \mathbb{E}_{a_t \sim \pi^*}[\mathrm{KL}(a_{t+1} | s_{t+1})].
\end{aligned} \tag{25}$$

Equation 25 enjoys the same shape as Equation 22, thus optimizing the soft Q function following sBAD is equivalent to the action-level optimization. We prove the consistency between sBAD and the traditional soft Bellman updates over full actions.

C Comparison of the Optimality between Three Backup Approaches

We provided a visual comparison of the differences between four different backup approaches with the optimal policy after convergence in Figure 6.

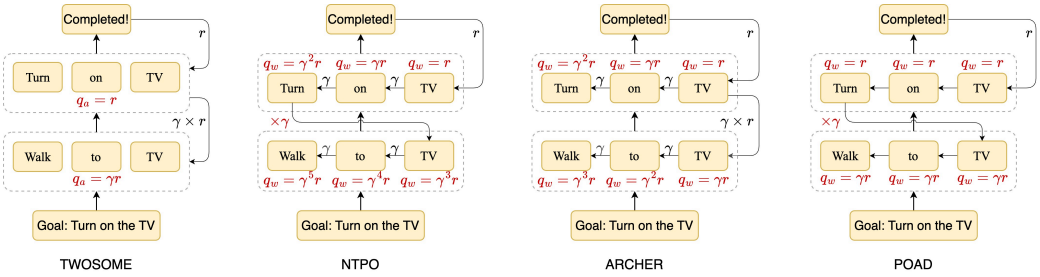


Figure 6: Visual comparison of the differences between four different backup approaches with the optimal policy after convergence. We show that optimizing the Q-function for each token with BAD (POAD) is equivalent to backup for the full action (TWOSOME), while others are not.

https://doi.org/10.52202/079017-3297

D Pseudo Code of POAD

Algorithm 1 Policy Optimization with Action Decomposition

Input: LLM ρ, Learning rate α, MDP $\mathcal{M} = (\mathcal{V}, \mathcal{S}, \mathcal{A}, \mathcal{T}, R, \gamma)$
Initialize: $\pi_\theta : \{\rho, \theta\}; V_\phi : \{\rho, \phi\}; \bar{\theta} \leftarrow \theta$
Initialize: Data buffer $\mathcal{D} \leftarrow \emptyset$
for each epoch **do**
  for $t = 0$ **to** $T - 1$ **do**
    Collect s_t.
    $a_t \sim \pi_\phi(\cdot|s_t)$.
    $s_{t+1} \sim \mathcal{T}(\cdot|s_t, a_t)$.
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, R(s_t, a_t), s_{t+1})\}$.
  end for
  Sample a mini-batch $\mathcal{B}$ from $\mathcal{D}$.
  for each time step t in $\mathcal{B}$ **do**
    for each token j in a_t **do**
      if $j < |a_t|$ **then**
        $v_{targ} \leftarrow V_{\bar{\theta}}(s_t, w_t^{1:j+1})$
      else if $j = |a_t|$ **then**
        $v_{targ} \leftarrow R(s_t, a_t) + \gamma V_{\bar{\theta}}(s_{t+1}, \emptyset)$
      end if
      $v \leftarrow V_\theta(s_t, w_t^{1:j})$
      Estimate $\hat{A}_t^j$ with GAE or $v_{targ} - v$
    end for
  end for
  $\theta \leftarrow \theta - \alpha \nabla_\theta \mathbb{E}_\mathcal{B}[(v - v_{targ})^2]$
  $\phi \leftarrow \phi - \alpha \nabla_\phi \mathbb{E}_\mathcal{B}[L(\phi)]$
  $\bar{\theta} \leftarrow \theta$
end for

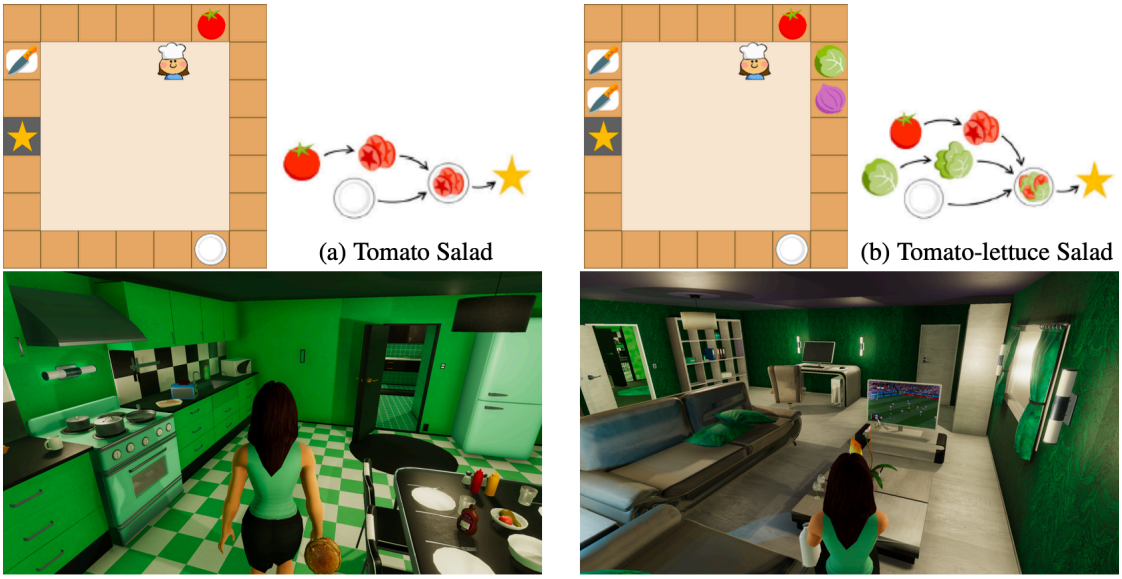
E Detailed Description of Experimental Environments

Figure 7 visually shows the Overcooked and VirtualHome tasks.

Overcooked is proposed as a typical deicision-making environment in Tan et al. [5]. There are two tasks in which a language agent is aiming to make and serve a *tomato salad* and *tomato-lettuce salad* respectively, with the provided ingredients and tools in a 7×7 grid world as a kitchen. To solve the tasks, the agent needs to explore and learn the correct order to cook the dish with the provided macro-actions, such as *Chop, Get-Tomato, and Go-Cutting-Board*. The environment is partially observable. The agent only observes the objects within 5×5 square centered on the agent. The reward involves +0.2 for chopping a correct ingredient, +1 terminal reward for delivering the correct dish, -0.1 for delivering any wrong item, and -0.001 for every time step.

VirtualHome is a simulated physical household environment proposed in Tan et al. [5]. In this environment, an agent is placed in a fully-furnished household with various objects. There are two tasks in the environment, the first one is to find the cold *pancake* on the table and heat it with the *microwave* in the kitchen. In the generalization tasks, we replace the *pancake* with other foods like *hamburger* and *pizza*, or replace the (*pancake*, *microwave*) at the same time with (*dishes*, *dishwasher*) or (*clothes*, *washing machine*), to evaluate agents' generalization abilities for similar but unseen tasks. The second one is to plan to have some entertainment while watching TV, for example picking up chips and milk in the kitchen, bringing them to the living room, turning on the TV, sitting on the sofa and enjoying. In order to solve the tasks, the agent need to use macro-actions to interact with the environment such as *walk to the living room*, *turn on the TV* and *sit on the sofa*. The environment is partially observable. Both tasks adopt a sparse reward setting, only when the task is finished, will the agent receive +1 reward.

DataSciCoding is an environment we developed for data science coding tasks with open action space. We adopt 3 Kaggle datasets and 3 OpenML datasets [52] with details in Appendix E.1. In each task,



(a) Tomato Salad (b) Tomato-lettuce Salad

(c) Food Preparation (d) Entertainment

Figure 7: Visual demonstrations of Overcooked and VirtualHome tasks [5].

agents aim to build the most effective classifier with the scikit-learn module, striving to achieve the highest possible ROC AUC score [53] on test sets. As the prompts provided to the agents contain no detailed information about task datasets, agents are required to align their knowledge to different datasets, that is, they must continuously modify and debug their code based on feedback from the runtime environment until it works. Therefore, it is a sequential decision-making process. Through this interaction, they align their approach to the specific task dataset. When the codes are executed successfully, a done signal will occur immediately, along with an ROC AUC score $\in [0, 1]$. If the codes run wrong, the agent receives a constant -1.0 as a penalty. Adopting the same evaluation metrics as Hollmann et al. [54], for each dataset, we evaluate 5 repetitions, each with a random $50\% - 50\%$ train-test split [55], and record the average ROC AUC score across these splits.

E.1 Details of Datasets Used in DataSciCoding Tasks

These datasets are collected from Kaggle and OpenML respectively, with pruning approaches consistent with Hollmann et al. [54].

Table 3: Details of task datasets in DataSciCoding, where [K] denotes Kaggle.

Data set	#Features	#Samples	#Classes
Pharyngitis [K]	19	512	2
Health Insurance [K]	13	2000	2
Spaceship Titanic [K]	13	2000	2
Airlines	7	2000	2
Balance Scale	4	125	3
Breast-w	9	69	2

F Case Study in Token-Level Credit Assignment

In this section, we conduct a case study for in-depth analysis to validate the effectiveness of BAD in terms of token-level credit assignment. We selected the last three states from the Food Prepa-

https://doi.org/10.52202/079017-3297

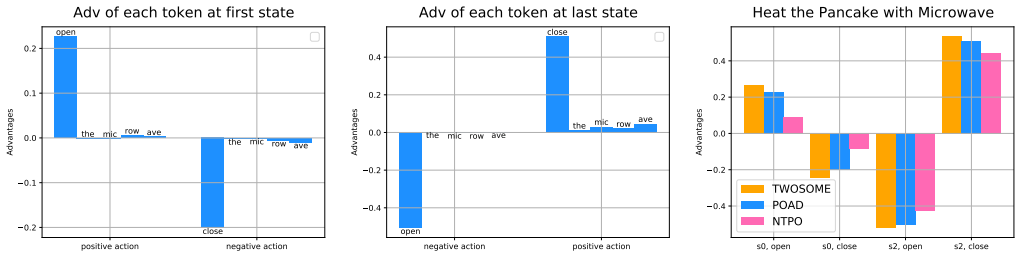


Figure 8: Case Study: Demonstration of the token-level credit assignment learned by the BAD at two states (left two). And a comparison of the volume of credit assignment for key tokens between different methods (right one), where TWOSOME indicates the credit assigned to entire actions instead of specific tokens.

ration task, which form a complete subtask: *Heat the Pancake with Microwave*. This serves as a simple yet practical case for our analysis, with transitions: $\mathcal{T}(s_0, \text{“open the microwave”}) \rightarrow s_1$, $\mathcal{T}(s_0, \text{“close the microwave”}) \rightarrow s_0$, $\mathcal{T}(s_1, \text{“put the pancake into the microwave”}) \rightarrow s_2$, $\mathcal{T}(s_2, \text{“open the microwave”}) \rightarrow s_2$, $\mathcal{T}(s_2, \text{“close the microwave”}) \rightarrow$ success with reward 1.0. According to these transitions, the optimal trajectory to complete this subtask is (*open the microwave, put the pancake into the microwave, close the microwave*). Additionally, the maximum step length for the task is 5; if the task is not completed within 5 steps, it is considered a failure, resulting in a reward signal of -1.0.

Based on this, we first sample 1,000 examples using a random policy, then train a critic to convergence with three different backups: action-level Bellman backup (TWOSOME), naive token-level Bellman backup (NTPO), and BAD (POAD). In Figure 3, we recorded the credit assignment results with each critic for each token in positive and negative actions under the first and last states.

The first two in Figure 8 show the credit assigned to each token in a symmetric (positive, negative) action pair by the BAD-learned critic in two different states, results confirm that the BAD critic can effectively assign most of the credit to the key tokens while rarely influencing other irrelevant tokens. The right one illustrates the volume of credits assigned to key tokens by POAD and NTPO, compared with the credit assigned to the entire action by TWOSOME, showing that POAD enjoys a far smaller discrepancy than NTPO.

G Detailed Comparison between POAD and CAAFE

Table 4 shows the performance of the best code discovered during POAD's training process with CodeLLaMA-7B, comparing with CAAFE with GPT-3.5 and GPT-4.

Table 4: The performance of the best code discovered during POAD's training process with CodeLLaMA-7B, comparing with CAAFE with GPT-3.5 and GPT-4. [K] denotes that the dataset is collected from Kaggle, while others are collected from OpenML.

Task	POAD-Best	CAAFE(GPT-3.5)	CAAFE(GPT-4)
health-insurance[K]	**0.5939**±0.01	0.5745±0.02	0.5748±0.02
pharyngitis[K]	**0.7282**±0.01	0.6976±0.03	0.7078±0.04
spaceship-titanic[K]	**0.8628**±0.01	0.8383±0.02	0.8405±0.02
airlines	**0.664**±0.01	0.619±0.04	0.6203±0.04
balance-scale	**0.9651**±0.03	0.844±0.31	0.882±0.26
breast-w	**0.9981**±0.01	0.9809±0.02	0.9809±0.02

H Wall-time of POAD on DataSciCoding Environment

Table 5 shows the average wall-time of training LLaMA2-7b with LoRA and POAD on all DataSciDoing tasks.

Table 5: Average wall-time for POAD training with each dataset with 1 * Nvidia A100.

Task	Wall-Time	Environmental Steps
health-insurance[K]	2h 34m	10k
pharyngitis[K]	2h 11m	10k
spaceship-titanic[K]	3h 5m	10k
airlines	2h 56m	10k
balance-scale	1h 43m	10k
breast-w	2h 6m	10k

I Details in Generalization Experiments

Table 6: Comparison of generalization performance on eight unseen tasks, with episodic returns averaged over 100 episodes (The fewer timesteps consumed, the higher the returns). In these tasks, we replace the *pancake* in the original Food Preparation task with other foods like *cheese*, *hamburger*, *apple pie* and *pizza*, or replace the (*pancake*, *microwave*) at the same time with (*dishes*, *dishwasher*) or (*clothes*, *washing machine*) for greater differences. In parentheses is the success rate of completing the task within 50 timesteps.

Methods	Cheese	Hamburger	Apple Pie	Pizza	Washing Plate	Laundry
LLaMA2-7B	0.1351(0.55)	0.1342(0.55)	0.1656(0.61)	0.1409(0.55)	0.0527(0.27)	0.0344(0.15)
TWOSOME	0.7119(1.0)	0.7058(1.0)	0.7304(1.0)	0.7047(1.0)	0.7031(1.0)	0.6038(1.0)
NTPO	0.7428(1.0)	0.7476(1.0)	0.7141(1.0)	0.7355(1.0)	**0.7491(1.0)**	0.5687(1.0)
POAD	**0.7553(1.0)**	**0.7602(1.0)**	**0.7650(1.0)**	**0.7625(1.0)**	0.7075(1.0)	**0.7014(1.0)**

J Evaluation on NLP Benchmarks

To investigate the impacts of fine-tuning different RL methods on the LLM's original abilities, we evaluate the models trained by POAD, NTPO and TWOSOME on widely used NLP benchmarks[56]. All models are trained in the Food Preparation task. The four benchmarks are ARC_Challenge, HellaSwag, PIQA and MMLU, with results on ARC_Challenge, HellaSwag and PIQA are displayed in Table 7 and the results of MMLU are displayed in Table 8. All results are calculated following the default configurations in Gao et al. [56].

Table 7: Zero-shot performance on Common Sense Reasoning tasks

Methods	ARC_C	HellaSwag	PIQA
LLaMA2-7B	0.4352	0.5713	0.7807
TWOSOME	0.4445	0.5819	0.7786
NTPO	0.4428	0.5825	0.7824
POAD	0.4471	0.5856	0.7797

https://doi.org/10.52202/079017-3297

Table 8: Details of Zero-shot performance on Massive Multitask Language Understanding tasks

Tasks	LLaMA2-7B	TWOSOME	NTPO	POAD
abstract_algebra	0.3100	0.2900	0.3300	0.2900
anatomy	0.4296	0.4296	0.4444	0.4593
astronomy	0.4474	0.4145	0.4342	0.4079
business_ethics	0.4300	0.4300	0.4400	0.4500
clinical_knowledge	0.4868	0.4566	0.4792	0.4604
college_biology	0.4236	0.4097	0.4236	0.4306
college_chemistry	0.2900	0.3000	0.2700	0.2700
college_computer_science	0.2900	0.3000	0.3100	0.3000
college_mathematics	0.3700	0.3900	0.4000	0.3900
college_medicine	0.4451	0.4104	0.4220	0.4393
college_physics	0.2451	0.1961	0.2157	0.2157
computer_security	0.5400	0.5100	0.5300	0.5100
conceptual_physics	0.3532	0.3660	0.3489	0.3489
econometrics	0.2193	0.2105	0.2018	0.1930
electrical_engineering	0.3724	0.3586	0.3931	0.3655
elementary_mathematics	0.2460	0.2619	0.2672	0.2593
formal_logic	0.3413	0.3571	0.3175	0.3175
global_facts	0.3100	0.2900	0.3400	0.2800
high_school_biology	0.4516	0.4548	0.4452	0.4548
high_school_chemistry	0.3300	0.3054	0.3350	0.3153
high_school_computer_science	0.3800	0.4000	0.4100	0.4100
high_school_european_history	0.6000	0.5879	0.5636	0.5879
high_school_geography	0.4343	0.4545	0.4444	0.4495
high_school_government_and_politics	0.5389	0.5181	0.5233	0.5181
high_school_macroeconomics	0.3769	0.3718	0.3821	0.3821
high_school_mathematics	0.2704	0.2593	0.2481	0.2630
high_school_microeconomics	0.3739	0.3739	0.3655	0.3487
high_school_physics	0.2781	0.3179	0.2848	0.2914
high_school_psychology	0.5248	0.5376	0.5138	0.5174
high_school_statistics	0.2731	0.2824	0.2778	0.2546
high_school_us_history	0.5196	0.5441	0.5343	0.5049
high_school_world_history	0.5612	0.5823	0.5696	0.5696
human_aging	0.4350	0.4484	0.4395	0.4260
human_sexuality	0.5649	0.5267	0.5344	0.5191
international_law	0.5620	0.5785	0.5620	0.5702
jurisprudence	0.4722	0.4722	0.4630	0.4537
logical_fallacies	0.4785	0.4663	0.4663	0.4847
machine_learning	0.3929	0.3839	0.3571	0.3571
management	0.4466	0.4272	0.4272	0.4175
marketing	0.6026	0.6282	0.6239	0.5940
medical_genetics	0.4900	0.5000	0.5000	0.4800
miscellaneous	0.5428	0.5428	0.5581	0.5504
moral_disputes	0.4480	0.4249	0.4451	0.4364
moral_scenarios	0.2402	0.2413	0.2559	0.2425
nutrition	0.4771	0.4869	0.4739	0.4739
philosophy	0.4887	0.4695	0.4952	0.4823
prehistory	0.4660	0.4568	0.4784	0.4691
professional_accounting	0.3546	0.3511	0.3617	0.3511
professional_law	0.3096	0.3181	0.3136	0.3123
professional_medicine	0.4118	0.4228	0.3713	0.4044
professional_psychology	0.4199	0.4248	0.4167	0.4085
public_relations	0.4182	0.4636	0.4273	0.4364
security_studies	0.4735	0.4694	0.4571	0.4857
sociology	0.6020	0.6119	0.6269	0.6169
us_foreign_policy	0.6700	0.6400	0.6500	0.6400
virology	0.4217	0.4337	0.4157	0.4036
world_religions	0.6140	0.6023	0.5965	0.5965

K Hyper-Parameters Settings of Experiments

During experiments, the implementations of TWOSOME are consistent with their official repositories, reproducing the original best-performing status. We first list the hyper-parameter candidates used for grid search in Table 9. Then, we show the hyper-parameters that achieve the best performance for each method and environment in Table 10-17.

Table 9: Hyper-Parameters candidates for grid search in Overcooked, VirtualHome, and DataSciCoding environments.

hyper-parameters	candidates
critic lr	1e-3,5e-4,1e-4,5e-5,1e-5,
actor lr	5e-4,1e-4,5e-5,1e-5,5e-6,1e-6,5e-7
ppo epochs	1,5
num mini batch	2,4,8,16,32
gamma	0.99,0.95
entropy coef	0.01,0.001,0.0001
max grad norm	10,0.5

Table 10: Hyper-parameters used for POAD in Overcooked tasks.

hyper-parameters	value	hyper-parameters	value	hyper-parameters	value
critic lr	1e-5	actor lr	5e-7	ppo epochs	5
batch size	128	num mini batch	2	gamma	0.99
rollout threads	4	entropy coef	0.01	max grad norm	0.5
PPO clip	0.2	value coef	0.5	KL threshold	0.02

Table 11: Hyper-parameters used for NTPO in Overcooked tasks.

hyper-parameters	value	hyper-parameters	value	hyper-parameters	value
critic lr	1e-5	actor lr	5e-7	ppo epochs	5
batch size	128	num mini batch	4	gamma	0.99
rollout threads	4	entropy coef	0.01	max grad norm	0.5
PPO clip	0.2	value coef	0.5	KL threshold	0.02

Table 12: Hyper-parameters used for TWOSOME in Overcooked tasks.

hyper-parameters	value	hyper-parameters	value	hyper-parameters	value
critic lr	1e-5	actor lr	5e-7	ppo epochs	1
batch size	128	num mini batch	4	gamma	0.99
rollout threads	4	entropy coef	0.01	max grad norm	0.5
PPO clip	0.2	value coef	0.5	KL threshold	0.02

https://doi.org/10.52202/079017-3297

Table 13: Hyper-parameters used for POAD in VirtualHome tasks.

hyper-parameters	value	hyper-parameters	value	hyper-parameters	value
critic lr	5e-5	actor lr	1e-6	ppo epochs	5
batch size	128	num mini batch	2	gamma	0.95
rollout threads	4	entropy coef	0.0001	max grad norm	0.5
PPO clip	0.2	value coef	0.5	KL threshold	0.02

Table 14: Hyper-parameters used for NTPO in VirtualHome tasks.

hyper-parameters	value	hyper-parameters	value	hyper-parameters	value
critic lr	5e-5	actor lr	1e-6	ppo epochs	5
batch size	128	num mini batch	2	gamma	0.95
rollout threads	4	entropy coef	0.01	max grad norm	0.5
PPO clip	0.2	value coef	0.5	KL threshold	0.02

Table 15: Hyper-parameters used for TWOSOME in VirtualHome tasks.

hyper-parameters	value	hyper-parameters	value	hyper-parameters	value
critic lr	5e-5	actor lr	1e-6	ppo epochs	1
batch size	128	num mini batch	4	gamma	0.95
rollout threads	4	entropy coef	0.01	max grad norm	0.5
PPO clip	0.2	value coef	0.5	KL threshold	0.02

Table 16: Hyper-parameters used for POAD in DataSciCoding tasks.

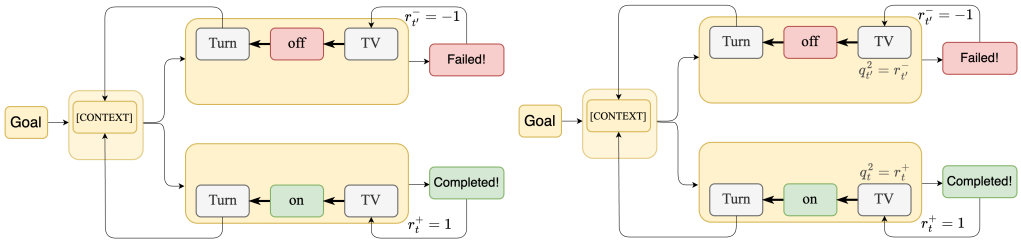
hyper-parameters	value	hyper-parameters	value	hyper-parameters	value
critic lr	5e-5	actor lr	1e-6	ppo epochs	1
batch size	128	num mini batch	4	gamma	0.95
rollout threads	4	entropy coef	0.01	max grad norm	0.5
PPO clip	0.2	value coef	0.5	KL threshold	0.02

Table 17: Hyper-parameters used for NTPO in DataSciCoding tasks.

hyper-parameters	value	hyper-parameters	value	hyper-parameters	value
critic lr	5e-5	actor lr	1e-6	ppo epochs	1
batch size	128	num mini batch	4	gamma	0.95
rollout threads	4	entropy coef	0.01	max grad norm	0.5
PPO clip	0.2	value coef	0.5	KL threshold	0.02

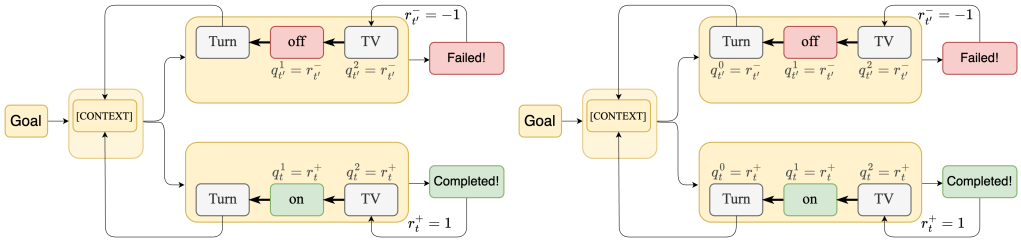
L Step-by-Step Breakdown of BAD

In this section, to facilitate the understanding of how BAD precisely assigns credit to each token, a step-by-step breakdown of the credit assignment process with BAD is shown in Figures 9-13



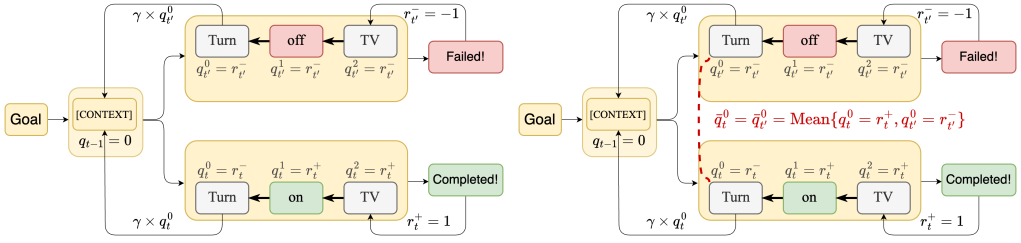
(a) Step 1 (b) Step 2

Figure 9: Step 1: Receiving feedback r_t^+ and $r_{t'}^-$ for positive and negative trajectories. Step 2: Propagating r_t^+ and $r_{t'}^-$ to $Q(\text{“TV”}|o_t, \text{“Turn on”})$ and $Q(\text{“TV”}|o_t, \text{“Turn off”})$.



(a) Step 3 (b) Step 4

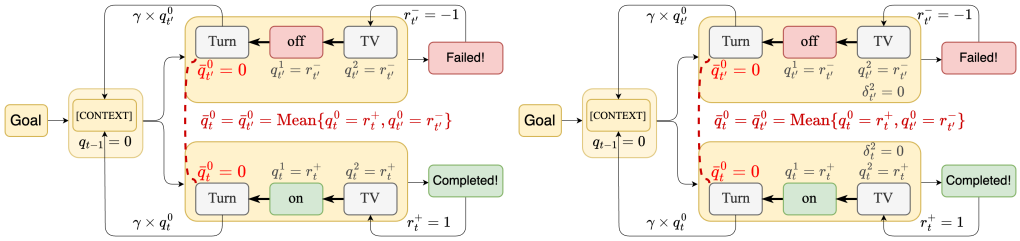
Figure 10: Step 3 and 4: Back-propagating r_t^+ and $r_{t'}^-$ to $Q(\text{“on”}|o_t, \text{“Turn”})$ and $Q(\text{“off”}|o_t, \text{“Turn”})$, and then to $Q(\text{“Turn”}|o_t)$ in both trajectories continuously.



(a) Step 5 (b) Step 6

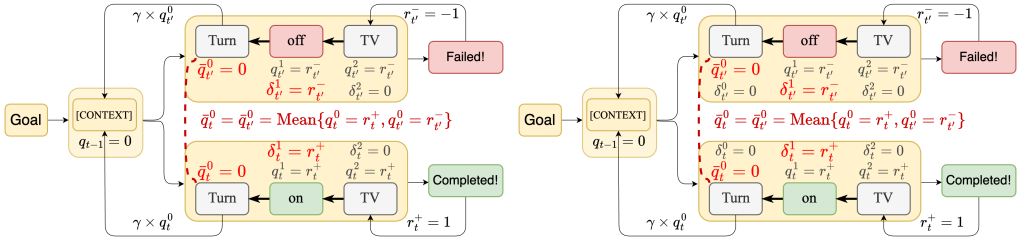
Figure 11: Step 5: Back-propagating to the previous action with $q_{t-1}^{|a_{t-1}|} = \text{Mean}\{\gamma \times q_t^0, \gamma \times q_{t'}^0\}$ since both trajectories share the same pre-action. Step 6: Similarly, $Q(\text{“Turn”}|o_t)$ is also shared among two trajectories, thus $Q(\text{“Turn”}|o_t) = \text{Mean}\{q_t^0, q_{t'}^0\} = 0$ as well.

https://doi.org/10.52202/079017-3297



(a) Step 7 (b) Step 8

Figure 12: Step 7: Modifying $\bar{q}_t^0$ and $\bar{q}_{t'}^0$ in both trajectories with the same $Q(\text{“Turn”}|o_t)$. Step 8: Starting to back-propagate again for credits assigned to each token for optimization with $\delta^j = q^j - q^{j-1}$ for $j > 0$; this process is similar to the calculation for advantage values.



(a) Step 9 (b) Step 10

Figure 13: Step 9: Calculating the credits for key tokens, where $\delta_t^1 = q_t^1 - q_t^0$ and $\delta_{t'}^1 = q_{t'}^1 - q_{t'}^0$. Step 10: Calculating the credits for tokens $j = 0$ with $\delta_t^0 = \bar{q}_t^0 - q_{t-1}^{|a_{t-1}|}$ and $\delta_{t'}^0 = \bar{q}_{t'}^0 - q_{t-1}^{|a_{t-1}|}$. Till now, we precisely emphasized key tokens while keeping irrelevant tokens with 0 credits.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Section 1

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 7

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

https://doi.org/10.52202/079017-3297

Justification: Section 3.1, 4, 5 and Appendix A, B

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 6.1 and Appendix K

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Code is included in the zip file with hyper-parameters in Appendix K

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 6.1 and Appendix E

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Secton 6

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

 https://doi.org/10.52202/079017-3297

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 6

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This work has no ethical violation.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: This work has no societal impact since it is theoretical research.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Section 6.1

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

 https://doi.org/10.52202/079017-3297

Answer: [NA]

Justification: This paper does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.