Enhancing Reinforcement Learning with Error-Prone Language Models

Muhan Lin

CMU-RI-TR-25-12

April 25, 2025

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Katia Sycara, *chair*
Zackory Erickson
Renos Zabounidis

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

To my master's journey.

Abstract

The correct specification of reward models is a well-known challenge in reinforcement learning. Hand-crafted reward functions, which are usually sparse, often lead to inefficient or suboptimal policies, misalignment with user values, or difficulties in attributing credit or blame within multi-agent systems. Reinforcement learning from human feedback is a successful technique that can mitigate such issues by generating dense reward functions, but the collection of human feedback can be laborious. Recent works have solicited feedback from pre-trained large language models rather than humans to reduce or eliminate human effort. However, these approaches yield poor performance in the presence of hallucination and other errors.

For these challenges, this thesis proposes a simple yet effective method for soliciting and applying noisy LLM feedback via a potential-based reward shaping function in both single-agent and multi-agent settings. We theoretically show that inconsistent rankings – which approximate ranking errors – lead to uninformative rewards with our approach. Our method thus mitigates ranking errors while enabling the LLM to evaluate agents' individual contribution to the task and provide feedback via the reward function at every timestep. Our method empirically improves convergence speed and policy returns over commonly used baselines across single-agent and multi-agent benchmarks even with significant ranking errors.

Acknowledgments

I am deeply and sincerely grateful to my advisor, Dr. Katia Sycara, whose research insights and working style have profoundly influenced my understanding of reinforcement learning, multi-agent systems, critical thinking, expression skills and academic characteristics. I also extend my sincere appreciation to my committee members, Dr. Zachory Erickson and Renos Zabounidis, for their thoughtful feedback and support throughout my research.

My heartfelt gratitude goes to Dr. Joseph Campbell, formerly a postdoctoral researcher in our lab and now a professor at Purdue University, whose mentorship significantly advanced my research skills, my engineering skills, and my comprehensive understanding of my field. I am also indebted to Dr. Woojun Kim and Dr. Simon Stepputtis for their enlightening advice and valuable suggestions. I am always thankful for the precious support, dedication and partnership of my collaborators, Shuyang Shi and Dr. Yue (Sophie) Guo. The late nights rushing to meet paper submission deadlines together with all of you will forever be remembered.

Meanwhile, I would like to acknowledge the Honda Research Institute (HRI) etc. for funding our projects. Discussions with HRI collaborators Dr. Vaishnav Tadiparthi, Dr. Behdad Chalaki, and Dr. Hossein Nourkhiz Mahjoub greatly inspired my research.

Additionally, Dr. Jiaoyang Li, my instructor for the course *Multi-robot Planning*, has enriched my understanding of multi-agent systems, which I truly value. I also owe a debt of gratitude to my friends, group mates, and labmates—Banji Li, He Jiang, Jiawen Zhang, Nathan Ludlow, Weihao (Zack) Zeng, Xijia (Pony) Zhang, Yiwei Lyu and Yiran Tao—for their constant assistance. Finally, I extend my deepest appreciation to my family for their unwavering love and support, without which none of this would have been possible.

Funding

This work is sponsored by Honda Research 58629.1.1012949, AFOSR FA9550-18-1-0251, DARPA under ANSR grant FA8750-23-2-1015 and Prime Contract No. HR00112490409, as well as ONR under CAI grant N00014-23-1-2840.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Reinforcement Learning (RL) has gained significant attention as a powerful decision-making framework by learning optimal policies through interactions with the environment, while Multi-agent Reinforcement Learning (MARL) extends this capability for complex teamwork between multiple agents. RL has demonstrated remarkable achievements in various challenging domains, such as robotics [2, 31, 33], game-playing [40, 52], coordinating autonomous vehicles [39, 62] in traffic systems [11, 57] and managing resources in distributed networks.

However, the correct specification of task rewards is a well-known challenge in RL [27]. Complex tasks often necessitate complex, nuanced reward models, particularly as shaping terms may be required to guide exploration. However, hand-crafting these reward functions is difficult and often leads to a phenomenon known as *reward hacking*, wherein an agent learns to exploit a reward function for increased returns while yielding unexpected or undesired behavior [41]. Reward hacking is symptomatic of the broader challenge of *value alignment*, in which it is difficult for a human domain expert to fully and accurately specify the desired behavior of the learned policy in terms of a reward function.

Moreover, while single-agent RL uses the reward signal to directly reflect an agent's progress, MARL involves multiple agents whose actions collectively shape the team reward, making it difficult to measure each agent's contribution. As a result, determining how to assign credit for the team's success or failure, known as the credit assignment problem,

has become a fundamental challenge in MARL [14, 36]. Designing dense, credit-aware, and value-aligned reward functions remains the most direct strategy for tackling the credit assignment problem for MARL tasks. Prior methods, such as value decomposition [36, 42, 44], address this issue by expressing the team value as a (possibly nonlinear) combination of per-agent values. Despite their successes [53], such decompositions are less competitive in sparse reward settings where feedback is both infrequent and delayed [30].

In this study, we aim to eliminate the dependence on handcrafted dense and value-aligned reward functions by training agents with reward functions derived from data. A notable method in this domain is reinforcement learning from human feedback (RLHF), where policy trajectories are ranked by humans. These rankings are then used to learn a reward function which guides model training and facilitates value alignment. This process is extremely costly in terms of human effort, however, requiring a significant number of rankings to train accurate reward models [9].

We can avoid the need for humans-in-the-loop by instead generating rankings with pre-trained large language models (LLMs) in a process known as reinforcement learning with AI feedback (RLAIF) [3, 19, 24]. However, LLMs are well known to hallucinate and present false information as fact [63], which reduces the accuracy and reliability of the resulting rankings. This is often overcome through complex reward post-processing techniques, which may be task-specific and difficult to tune [20]. Meanwhile, it remains an open question as to whether AI-generated reward functions can properly attribute credits in the multi-agent case.

We extended these challenges into two problems which this thesis seeks to answer:

- How can we incorporate noisy LLM feedback into RL reward training while mitigating its negative impact?
- Can we further leverage noisy LLM feedback in MARL to assign credits by generating informative, agent-specific rewards based on natural language descriptions of tasks and goals?

By solving these problems, we can successfully extend RLAIF to function with small language models, which consume less computation resources but tend to produce noisy feedback. Meanwhile, the capability of RLAIF is also extended to MARL when realizing effective credit assignment.

1.2 Related Works

1.2.1 Constructing Value-aligned Dense Rewards based on Preference

Constructing rewards based on human feedback has a long history [49]. To efficiently use human domain knowledge and provide more generalizable rewards, human preference on episode segments [4, 10, 37] and human demonstrations [5] are distilled into models which serve as reward functions for RL. The method has witnessed great success in complex domains where rewards are difficult to specify such as training large language models (LLM) to align with human logic and common sense [35, 64].

One major drawback of RLHF is its requirement of exhaustive human participation to provide demonstrations and feedback. LLMs have shown deductive logic abilities comparable to humans in recent years [13], and are able to substitute humans in reward issuing [22, 24, 58, 60], or data collection and labeling for reward model training [20, 24]. While the former suffers from time and resource costs for training RL agents, the latter is becoming promising for training complex RL tasks [54].

An outstanding challenge with leveraging LLM-based feedback is that the performance of RLHF is dependent on the quality of feedback received [9]. Different LLMs have distinct probabilities of giving wrong feedback, thus leading to rewards of varying quality. Casper et al. [9] also suggests that comparison-based feedback may not be efficient and adequate to train reward models with noisy LLM outputs. In this work, we analyze the training performance of reinforcement learning agents across various LLMs, each of which produce different error distributions in feedback.

Another challenge is that of the reward formulation itself. Many works train a model distilling LLM or human preference and use it as the reward model [10, 20, 24, 54], but in practice, this needs post-processing on outputs of the reward model such as filtering [20], and normalization [10]. Our work posits that a reward function trained without complex post-processing and environment rewards would be more general and adaptable to various practical scenarios.

1.2.2 Credit Assignment in Multi-agent Reinforcement Learning

Credit assignment in multi-agent reinforcement learning remains a fundamental challenge, especially in environments with sparse team rewards. There are two main classes of traditional approaches for the credit assignment, value decomposition [12, 14, 36, 44] and slight modifications to known algorithms such as the gradient-descent decomposition [43].

There are also some works that combine the basic ideas of both method classes. [18] adapts the partial reward decoupling into MAPPO [59] to eliminate contributions from other irrelevant agents based on attention. The other work [56] utilizes transformer with PPO loss [38] adapted to the value decomposition idea.

The methods above have made progress in assigning credits, but their effectiveness diminishes with delayed or sparse rewards. For example, the work [30] shows the poor performance of QMIX [36] with sparse rewards. However, designing dense rewards to combat this challenge is difficult given the complexity of tasks [6, 21, 27]. Although social-influence-based rewarding calculates dense individual rewards [16], it requires teammates' behavior models, which often need additional training to estimate and update.

One general method of generating dense rewards, particularly in single-agent settings, is RLHF [10] and its extension, RLAIF [23]. However, these approaches are limited to single-agent environments and do not address the unique requirements and challenges that exist within the multi-agent counterparts, according to RLAIF. [61] shows one direction of generating dense rewards for credit assignment with LLM in multi-agent scenarios. Utilizing the coding capabilities of LLM, this method iteratively queries LLM to generate multiple reward functions with high density and refine the reward functions gradually in the training process. However, this method can suffer from LLM hallucination problems, which can cause misleading rewards due to inconsistent rankings or other ranking errors. Considering these problems, our method adapts the potential-based RLAIF in Chap.2, which can handle LLM hallucination with the multi-query approach, from the single-agent scenarios to multi-agent ones, and successfully handles the credit assignment problem.

1.3 Background

1.3.1 Reinforcement Learning

In reinforcement learning, an agent interacts with an environment and receives a reward for its current action at each time step, learning an optimal action policy to maximize the rewards over time. This procedure can be formulated as an infinite horizon discounted Markov Decision Process (MDP) [45].

At each discrete timestep t in this process, the agent observes environment state s_t and takes action a_t, leading to the next environment state s_{t+1} and a reward r_t. An MDP is represented as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma)$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{R} : \mathcal{S} \mapsto \mathbb{R}$ is a reward function, $\mathcal{T}(s, a, s') = P(s'|s, a)$ is a transition function, and γ is a discount factor. A stochastic policy $\pi(a|s) : \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ indicates the probability that the agent selects action a given the state s. The agent's goal is to learn π maximizing the expected discounted return through training, given an initial state distribution.

1.3.2 Multi-Agent Reinforcement Learning

We consider a fully cooperative Markov Game [34], which generalizes the Markov Decision Process (MDP) to multi-agent settings where multiple agents interact in a shared space and collaborate by maximizing a common reward function. A fully cooperative Markov Game is represented by the tuple $(N, S, \{A_i\}_{i=1}^N, P, R, \gamma)$, where N is the number of agents, S represents the set of global states, $\{A_i\}$ is the action space for each agent, and $P(s'|s, a_1, \ldots, a_N)$ describes the probability of transitioning from one state to another based on the joint actions of all agents. The agents share a reward function $R(s, a_1, a_2, \ldots, a_N)$, which assigns a common reward based on the state-action pairs. The objective is for the agents to collaboratively learn policies that maximize the cumulative discounted reward, where γ denotes the discount factor.

1.3.3 Value Decomposition

In multi-agent systems, value decomposition partitions a complex global value function into multiple local components, allowing each agent to independently learn its own value

function while collectively contributing to a shared objective. By optimizing each local component separately, the method preserves alignment with the overall target, thus facilitating efficient learning and coordination among agents.

1.3.4 Preference-based Reinforcement Learning

Our work is based on the framework of preference-based reinforcement learning, where the reward function is learned from preference labels over agent behaviors [10, 15, 25, 26]. For a pair of states (s_a, s_b), an annotator gives a preference label y that indicates which state is ranked higher, considering which state is closer to the given task goal. The preference label $y \in \{0, 1\}$, where 0 indicates s_a is ranked higher than s_b, and 1 indicates s_b is ranked higher than s_a. Given a parameterized state-score function σ_ψ, which is commonly called the potential function and usually equated with the reward model r_ψ, we compute the preference probability of a state pair with the standard Bradley-Terry model [7],

$$P_\psi[s_b \succ s_a] = \frac{\exp\left(\sigma_\psi(s_b)\right)}{\exp\left(\sigma_\psi(s_a)\right) + \exp\left(\sigma_\psi(s_b)\right)} = sigmoid(\sigma_\psi(s_b) - \sigma_\psi(s_a)), \tag{1.1}$$

where $s_b \succ s_a$ indicates s_b is ranked higher than the state s_a. With a preference dataset $D = (s_a^i, s_b^i, y^i)$, preference-based RL learns the state-score model σ_ψ by minimizing the cross-entropy loss, which aims to maximize the score difference between the high and low states:

$$\mathcal{L} = -\mathbb{E}_{(s_a, s_b, y)\sim\mathcal{D}}\Big[\mathbb{I}\{y = (s_a \succ s_b)\} \log P_\psi[s_a \succ s_b] + \mathbb{I}\{y = (s_b \succ s_a)\} \log P_\psi[s_b \succ s_a]\Big]. \tag{1.2}$$

This framework is used in both RLHF and RLAIF where rewards are issued directly from the state-score model and differ only in the choice of annotator, i.e., human or LLM.

Chapter 2

Potential-based Reward Structure for RL from LLM Feedback

This section introduces our work on the potential-based reward function, a simple and effective strategy for reinforcement learning in the face of unreliable LLM feedback [28]. The core idea underlying our approach is to issue uninformative rewards for states in which the LLM is uncertain. Thus, we avoid issuing potentially misleading rewards which allows us to train performant policies even in the face of significant ranking errors. Building off the insight that certainty in language models is expressed through output consistency [47], we show that potential-difference rewards learned over repeated rankings naturally reflect an LLM's uncertainty.

This chapter makes the following contributions: we 1) introduce a methodology for incorporating noisy LLM feedback into RL which out-performs prior SOTA; and 2) provide theoretical and empirical analysis showing that uncertain LLM outputs – as given by inconsistent responses – lead to uninformative rewards which improve convergence speed and policy returns in experiments. The codes of this work can be accessed here.

2.1 Methods

Despite the success of LLMs in few-shot task generalization, these models are imperfect and yield sub-optimal performance in many areas. One notable issue is the well-documented tendency of LLMs to hallucinate, which results in LLM-generated preference rankings frequently containing errors (see Table 2.1). These errors present major challenges for reinforcement learning from LLM feedback, as they result in noise in the learned score function. Under the standard RLHF formulation where rewards are directly issued from the score function [10], this can lead to inefficient exploration at best and, at worst, trap the agent in sub-optimal local minima.

2.1.1 Quantifying Feedback Error through Output Consistency

It has been shown that the certainty of LLM predictions can be quantified by issuing the same query multiple times and measuring the *consistency* of the predictions [32]. Specifically, the confidence of ranking s_a higher than s_b, $conf\{y = (s_a \succ s_b)\}$, is defined as $\frac{N(s_a \succ s_b)}{N_{query}(s_a, s_b)}$. where $N(s_a \succ s_b)$ denotes the number of times LLM ranks s_a higher than s_b, and $N_{query}(s_a, s_b)$ denotes the total number of queries on s_a and s_b. Confidence is a necessary condition to consider when evaluating LLM feedback quality, given that low confidence often causes considerable noise in feedback which manifests as incorrect rewards. Based on the definition of feedback confidence, we derive an equivalent form of the RLHF loss based on ranking confidence and consistency as follows:

$$\begin{aligned}
\mathcal{L} &= -\mathbb{E}_{(s_a,s_b,y)\sim\mathcal{D}}\Big[\mathbb{E}_{N_{query}}\Big[\mathbb{I}\{y = (s_a \succ s_b)\}\log P_\psi[s_a \succ s_b] \\
&\quad + \mathbb{I}\{y = (s_b \succ s_a)\}\log P_\psi[s_b \succ s_a]\Big]\Big] \\
&= -\mathbb{E}_{(s_a,s_b,y)\sim\mathcal{D}}\Big[conf\{y = (s_a \succ s_b)\}\log(sigmoid(\sigma_\psi(s_a) - \sigma_\psi(s_b))) \\
&\quad + conf\{y = (s_b \succ s_a)\}\log(sigmoid(\sigma_\psi(s_b) - \sigma_\psi(s_a)))\Big].
\end{aligned} \tag{2.1}$$

This loss function uses confidence-based weights to relate the scores between each state in ranked pairs. From this derivation, we see the following. 1) A more confident ranking

produces a larger score difference between the ranked states, i.e., the magnitude of the score difference is proportional to the confidence. Formally, if $conf\{y = (s_a \succ s_b)\} > conf\{y = (s_b \succ s_a)\}$ then $\sigma_\psi(s_a) - \sigma_\psi(s_b) > 0$. In the event the LLM is perfectly confident, $conf\{y = (s_a \succ s_b)\} = 1$, then the loss function will maximize $\sigma_\psi(s_a) - \sigma_\psi(s_b)$. 2) As the confidence *decreases*, then $|conf\{y = (s_a \succ s_b)\} - conf\{y = (s_b \succ s_a)\}|$ converges to 0. When the LLM is completely uncertain then $conf\{y = (s_a \succ s_b)\} = conf\{y = (s_b \succ s_a)\}$ and the loss function will minimize $|\sigma_\psi(s_a) - \sigma_\psi(s_b)|$, resulting in identical state scores such that $\sigma_\psi(s_a) = \sigma_\psi(s_b)$. The formal proof is given in Appendix A.

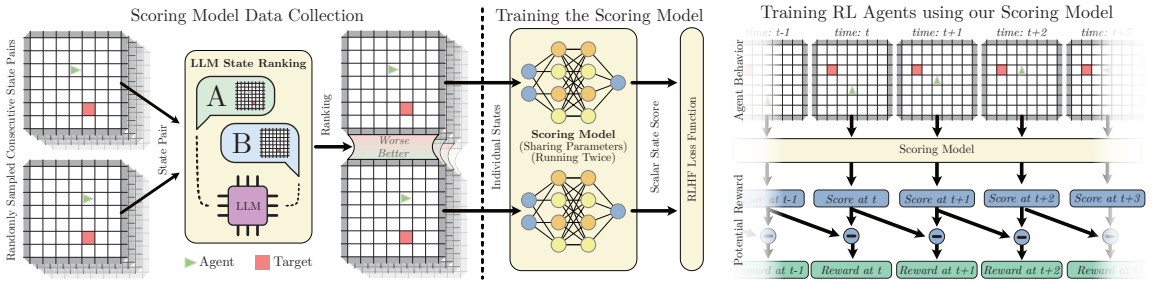
2.1.2 Potential-based Rewards for Learning with Noisy Feedback

The above observations stemming from Eq. A.1 motivate the form of our proposed method. Intuitively, when the LLM is completely uncertain when ranking s_a and s_b then the difference between their scores is zero. This is ideal, as **when the LLM is unable to issue an accurate ranking then we would like it to issue an uninformative reward**, i.e., a reward of zero. Our solution is to treat the state-score as a *potential function*, defining the reward as the difference between the scores of successive state pairs:

$$r(s_t) = \sigma_\psi(s_t) - \sigma_\psi(s_{t-1}). \tag{2.2}$$

Thus, the more uncertain an LLM's ranking is, the less informative the reward is. The potential in Eq. 2.2 is naturally shaped to a proper scale range, with positive rewards for actions that are beneficial and promising to the given task goal and negative rewards for detrimental actions. Large values correspond to more confident rankings, while small ones to less confident rankings. As such, our approach is particularly well-suited to RLAIF with smaller, specialized models which are often necessary in resource-constrained environments.

There is an additional benefit to this formulation. Prior works treat the state-score function as a reward function and directly issue rewards from it, which we call the "direct-reward" method. This often leads to training instability as the rewards may have significant differences in scale, which need to be corrected via post-processing techniques such as normalization and thresholding as well as extrinsic per-step reward penalties. However, the performance of post-processed direct rewards is highly sensitive to these hyper-parameters, as they are often task-specific. Our potential difference formulation helps alleviate this



(a) Training the scoring model from LLM-ranked state pairs.

(b) Agent training using the scoring model.

Figure 2.1: Our approach: (a) We train our scoring model with randomly sampled *consecutive* state pairs, which are ranked by an LLM with respect to task completion. The resulting dataset of ranked state pairs is utilized in an RLHF fashion to train a single scoring model, capable of providing a score for any novel state. (b) Using the scoring model, an RL agent is trained by scoring each state. Prior work uses this score as a reward; however, our approach utilizes the score differences as a potential reward.

issue as 1) uncertain states converge to the same score value so the impact of noisy rankings no longer needs to be mitigated through post-processing, and 2) per-step penalties can be discarded in favor of simple timestep-based discounting which are far less sensitive.

2.1.3 Algorithm

Our algorithm consists of the following four steps: 1) Randomly sample pairs of sequential states from the environment. 2) Query the LLM to rank states in each pair with respect to a natural language task description, e.g., “Go to the green goal”. The prompt contains a language task description, environment description, and in-context learning examples [55] as context to generate preference labels for states in each pair. 3) Train the state-score model σ_ψ with the loss function in Eq. 1.2. 4) Train a reinforcement learning agent with feedback from the state-score model.

2.2 Performance Analysis of Potential-Difference Rewards

We evaluate our approach in commonly-used discrete (Grid World) and continuous (MuJoCo) [8] benchmark environments. Throughout these experiments, we investigate the

effectiveness of our potential-based reward function a) as compared to using the score as a reward directly in both single and multi-query settings; and b) its sensitivity to inconsistency in state rankings.

2.2.1 Experiment Setup

Grid World. We examine three scenarios within Grid World [46]: **NoLock**, **Lock**, and **MultiLock**. The layouts are shown in Fig. 2.2. In each scenario, the agent (green triangle) must navigate to the target (green rectangle). There are one and two locked doors in the Lock and MultiLock variants, respectively, that block the agent's way to the goal. To unlock each door the agent must pick up the appropriate key and use it on the door. The agent, goal, and key positions are randomly initialized in every episode.

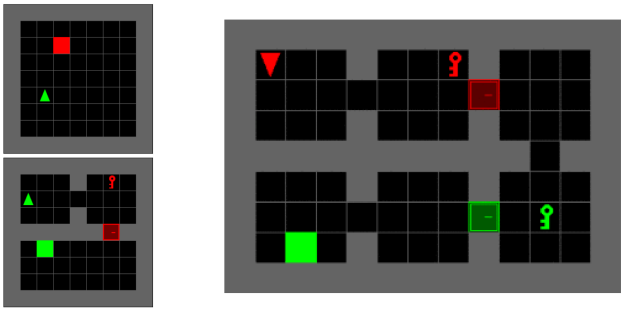


Figure 2.2: Grid world environments with NoLock (upper-left), Lock (lower-left), and MultiLock (right) variants from left to right.

MuJoCo. We examine a subset of robot control tasks selected from the simulated robotics benchmark MuJoCo [50]. We choose 3 tasks with increasing degrees of complexity: **Inverted Pendulum**, **Reacher**, and **Hopper**.

For each of these six environments, we compare our approach with the following baselines:

- **Direct reward** directly utilizes the trained state-score functions' score as reward; i.e., $r(s) = \sigma_\psi(s)$. Following Christiano et al. [10], the reward is normalized to zero-mean with a standard deviation of 1.
- **Default reward** utilizes the vanilla RL objective of each environment with human-specified reward functions. In grid world variants, the default reward is defined as 0 for failure and $1 - 0.99n/n_{max}$ otherwise when the agent reaches the goal, where

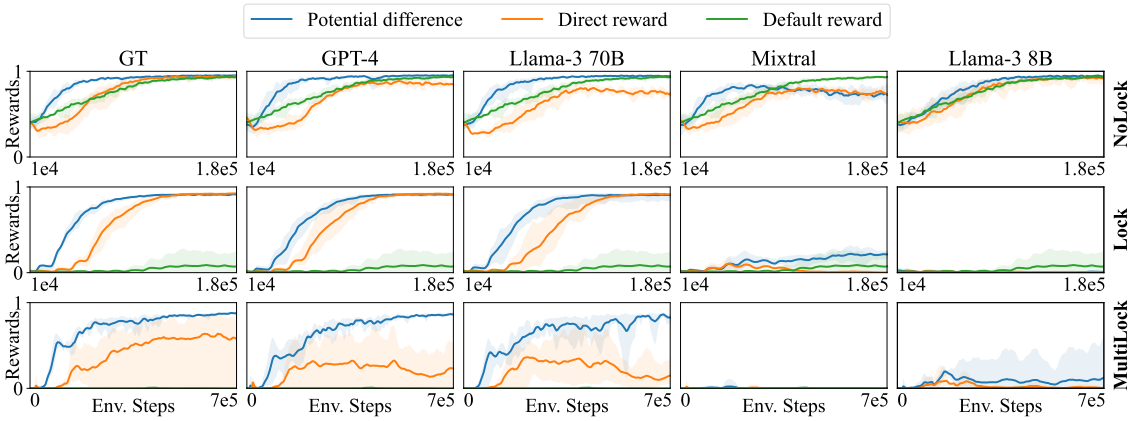


Figure 2.3: The average learning curves with reward functions trained from single LLM queries in the Grid World environments over 5 random seeds, with the return variance visualized as shaded areas.

n_{max} is the maximum time steps for each episode and n is the step count until success. For MuJoCo tasks, the default rewards are specified as those defined in OpenAI Gym [8].

In each environment, we randomly sample pairs of sequential states from the environment with replacement in order to generate rankings for training the state-score model used by both potential difference and direct reward. For single-query experiments, Grid World uses 2500, 3500, and 6000 samples for NoLock, Lock, and MultiLock respectively while MuJoCo uses 1000 samples for each environment.

Without loss of generality, we employ PPO as the underlying policy-training framework [38] and make the following assumptions: a) the environment is fully observable; and b) the agent has no knowledge of the task before training, i.e., is randomly initialized.

2.2.2 LLM Ranking Performance

We first quantify the performance of four different LLM models used in this work over each Grid World environment. After sampling pairs of sequential states as discussed in Sec. 3.2.1, we measure the accuracy of a) the LLM's rankings and b) the resulting state-score model with respect to ground truth rankings. The results, shown in Table 2.1, provide us with an approximate ordering of LLM ranking performance, where GPT-4 $>$ Llama-3 70B $>$ Mixtral $>$ Llama-3 8B.

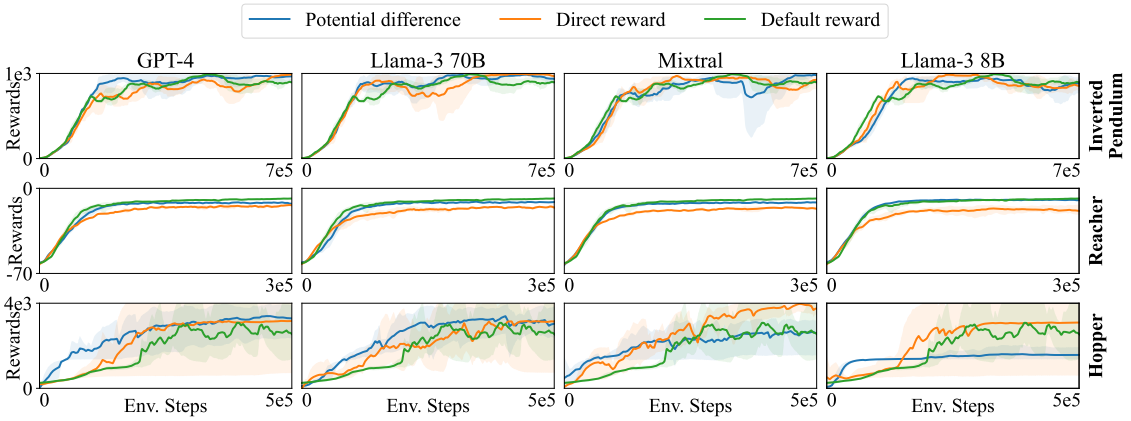


Figure 2.4: The average learning curves with reward functions trained from single LLM queries in the MuJoCo environments over 5 random seeds, with the return variance visualized as shaded areas.

2.2.3 Single-Query Evaluation

We next examine how our approach performs compared to the standard direct reward approach commonly utilized in RLHF. In each environment, we train our state score models with 4 LLMs: Mixtral [17], GPT 4 [1], and Llama-3 with 8B and 70B parameters [51]. For Grid World environments, we add an additional baseline in which rankings are generated using a ground truth heuristic function (GT) which serves as an upper bound for our methods.

The state score models are trained by minimizing the loss in Eq. 1.2. Then they are employed to train 5 RL policies with random seeds and initializations for each method. As a common approach to avoid reward hacking, a constant step penalty is applied to the produced rewards from both methods in all environments except for MuJoCo Reacher, which exploits a torque penalty as described in Brockman et al. [8]. The results, as well as the default reward performance, are shown in Fig. 2.3 and Fig. 2.4.

In Grid World environments, our method (in blue) of potential difference-based reward outperforms the direct reward method in most cases. When using GT, GPT-4, or Llama-3 70B rankings, our method converges the fastest and yields the highest final reward. For the environments of Lock and MultiLock (bottom two rows), the tasks are more challenging when using the default reward; however, our method remains on par, or outperforms the baseline with respect to convergence speed and final reward. However, when using LLMs

Env.	Mthd.	GT	Llama-3 8B	Mixtral	Llama-3 70B	GPT-4
No Lock	Rank	1.0	0.69	0.76	0.93	1.0
	Score	1.0	0.77	0.89	0.98	1.0
Lock	Rank	1.0	0.54	0.65	0.89	0.98
	Score	1.0	0.55	0.74	0.97	0.98
Multi Lock	Rank	1.0	0.58	0.60	0.90	0.99
	Score	1.0	0.66	0.66	0.96	0.99

Table 2.1: Ranking accuracy for each LLM across 1000 state pairs sampled from each environment. Rank indicates the direct ranking performance of the LLMs and Score indicates the ranking performance of the trained score models. Given that the ground-truth ranking are only accessible in grid world environments, we only show the ranking correctness of LLMs and state-score models in these three environments.

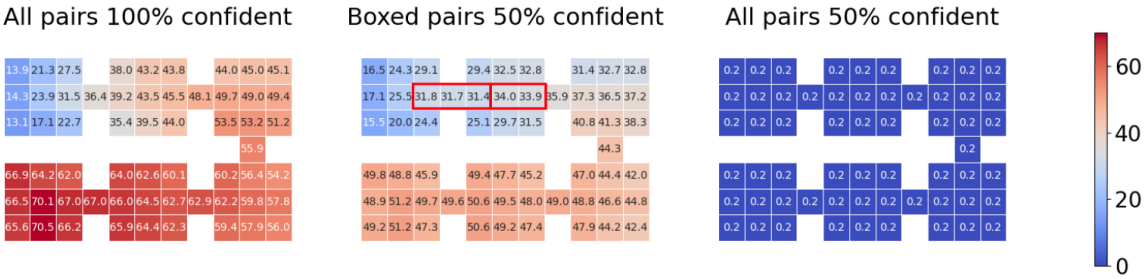


Figure 2.5: The heat maps showing that feedback inconsistency pushes rewards towards 0. Each grid in the map shows the score of the state where the agent is at this grid. The first heat map shows state scores trained with 100% confident rankings on all state pairs. The second heat map shows state scores trained with 100% confident ranking on all state pairs except 50% confident rankings on state pairs where the agent is in the red block. The third heat map shows state scores trained with 50% rankings on all state pairs.

which generate noisy outputs (i.e., Mixtral and Llama-3 8B), all methods fail to converge in the Lock and MultiLock environments. In Sec. 2.2.4, we detail our approach of using multiple queries, particularly for low-performing LLMs, to regain training performance using our potential-based reward function.

In MuJoCo environments, reported in Fig. 2.4, our potential-based reward method slightly outperforms (particularly in Hopper with GPT-4 and Llama-3 70B) or is on par with our baseline methods. Exceptions to this trend can be seen with low-performing LLMs (e.g., Llama-3 8b). Our method outperforms direct reward in Reacher and achieves a performance similar to the well-crafted default reward function, showing that potential-difference rewards are better. However, direct reward outperforms ours when using low-

performing LLMs, particularly Mixtral and Llama-3 8B. We attribute this to the challenge of designing appropriate prompts based on human intuition, i.e., we prompt LLMs to compare the hopper's speed in two consecutive states because the hopper should learn to move forward without falling down. However, these LLMs then encourage moving faster instead of simply moving forward. While this prompt could lead to a good reward for high-performing LLMs, low-performing LLMs could not handle such situations, and we hypothesize that this leads to sub-optimal training results.

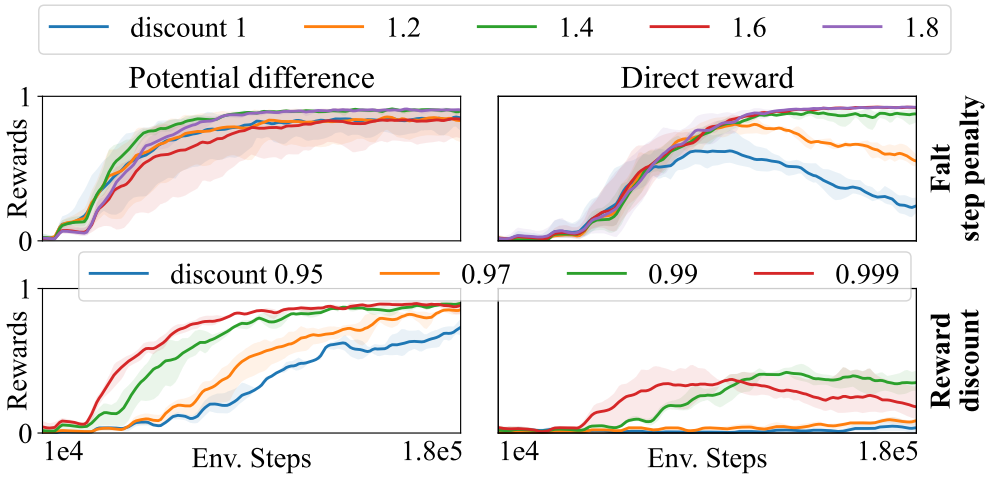


Figure 2.6: The average learning curves for rewards with multiple step penalties or discounts in the Grid World - Lock scenario, over 3 random seeds.

Hyper-Parameter Sensitivity Analysis

Since potential difference reward and direct reward suffer from reward hacking without post-processing, a step penalty is essential; however, choosing this value can be difficult. We conduct further experiments in the grid world Lock scenario to show that our method is less sensitive to step penalty than direct reward. Two penalty schemes with multiple parameters are tested: 1) **Flat Step Penalty**: A positive constant is subtracted at each time step. 2) **Reward Discount**: Reward for episode step t is discounted by γ_r^t, where $\gamma_r < 1$ is a positive constant. We use the state score model trained from ground truth human heuristics for comparison. Each parameter is tested to train 3 RL policies with random seeds and initialization. The results are shown in Fig. 2.6.

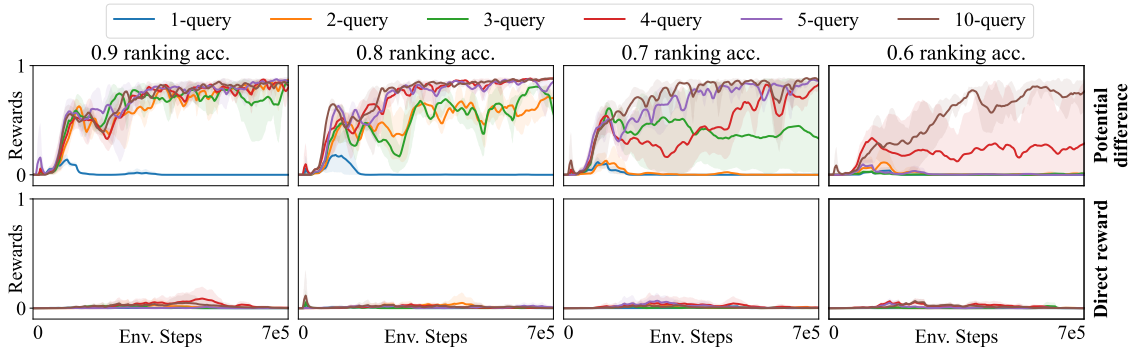


Figure 2.7: The average learning curves with rewards trained from synthetic multi-query ranking datasets in the Grid World - MultiLock scenario, over 3 random seeds.

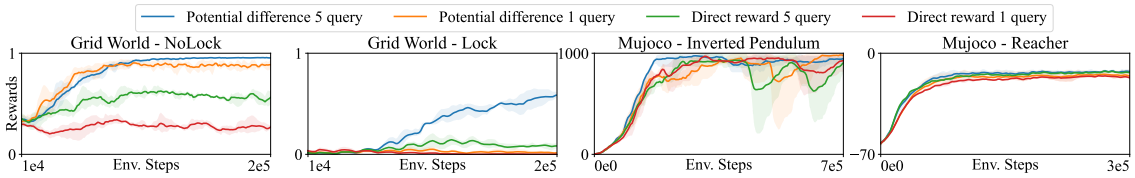


Figure 2.8: The average-performance comparison of 5-query variation of potential-difference rewards and direct rewards with 1800, 2200, 1000, 1000 state pairs ranked with Mixtral, over 5 random seeds.

Our method shows robustness to the choice of the flat step penalty, as the curves of penalty variances are less divergent. However, when using it as a direct reward, it can be seen that the performance is affected significantly with respect to the penalty, as many of them prevent the agent from converging. The results also show that our method can perform well by picking a commonly-used discount factor such as 0.99, avoiding the burden of extensive hyperparameter tuning. However, using it as a direct reward requires further hyperparameter tuning.

2.2.4 Multi-Query Evaluation

We introduce a multi-query approach that queries about ranking each state-transition case in the scoring-model training dataset a given number of times to address the rankings' inconsistency with lower-performing LLMs to push potential difference rewards toward zeros in the face of conflicting responses. In Fig. 2.5, we illustrate the heat maps of state scores trained with datasets of distinct consistency degrees, demonstrated in the Grid World

MultiLock environment. Each grid in the heat maps records the score the scoring model assigns for an agent at that location. The left sub-figure demonstrates the ideal case in which 100% correct rankings are utilized to train the scoring model, demonstrating a smooth gradient from the start room (top left corner) to the final room (bottom left corner) roughly following the correct path. However, if the scoring model is trained with 50% confidence on all state pairs (right sub-figure in Fig. 2.5), the score in any state becomes equal as no adjacent states are ranked higher with high confidence. This demonstrates our method's ability to disregard states, and thus not provide rewards when LLM rankings are inconsistent across multiple queries. Finally, when the ranking results for a subset of states are inconsistent, yet consistent for all other locations (see Fig. 2.5 center), the correct gradual change in score is maintained outside of the affected area. These results underline our method's capabilities with respect to the effects of pushing uncertain state scores toward zero while giving contrasting rewards to confident pairs, ultimately improving performance of our method with low-performing LLMs (see Sec. 2.2.4).

Synthetic Ranking Evaluation

To test what ranking accuracy of datasets is needed for the LLM with multi-query methods, and how many queries are required, we synthesized training datasets with specific ranking accuracy from 60% to 90% and simulated query times from 1 to 10. State-score models trained with these datasets output rewards when training RL policies, and their performance is shown in Fig. 2.7. The specific ranking correctness rates are controlled by introducing random ranking errors into the ground-truth ranking datasets. This approach is repeated on several copies of the ground-truth datasets to simulate the multi-query ranking results.

The result demonstrates that with more and more queries, the potential-difference reward gradually improves the training performance. Two or more queries may achieve fast policy training converging towards optimal if using ranking datasets with high feedback consistency to train state-score models. Notably, even for the datasets of only 60% ranking accuracy, which is close to random guessing, potential-difference rewards trained with enough queries can still increase the average policy training returns from 0 to an almost optimal level with 10 queries. This indicates that with enough queries, even the datasets with low-ranking accuracy can be boosted to function like those with high accuracy. This finding is consistent with our theoretic analysis and demonstrates considerable potential in

mitigating significant ranking errors.

LLM Ranking Evaluation

Observing that Mixtral has the highest inconsistency in ranking states and thus has the largest potential for improvement, we evaluate the 5-query variations of potential-difference rewards and direct rewards with ranking results from Mixtral to verify our claims. Different methods' RL policy training curves averaged over 5 random seeds are compared in Fig. 2.8. As hypothesized, the 5-query potential-difference rewards achieve faster policy training and result in the highest rewards in all experiments. The single-query potential-difference rewards also outperform the single-query direct rewards. The improvement is most significant in Grid World - Lock scenario.

2.3 Conclusion

In this chapter, we propose a simple method for incorporating noisy LLM feedback into reinforcement learning. The core idea is to learn a potential-based reward function over repeated LLM-generated preference rankings which issues uninformative rewards for states in which the LLM is uncertain. We show both theoretically and empirically that this results in a natural trade-off between reward sparsity and LLM confidence in a variety of discrete and continuous environments.

Chapter 3

LLM-guided Reward for Credit Assignment in MARL

This chapter presents our work on **L**LM-guided **C**redit **A**ssignment **(LCA)**, a novel framework that integrates LLMs into the MARL training process to facilitate credit assignment with sparse environment rewards [29]. Our approach retrieves information about the overall team objective and its key steps from existing team rewards and provides them to the LLM. The LLM generates preference rankings over each agent's actions so that actions that are more contributive from the perspective of the team's success are preferred. These rankings are used to train dense potential-based reward functions for each agent, simultaneously addressing both credit assignment and reward sparsity. Recalling that potential-based reward functions enhance RL by incorporating noisy LLM feedback, LCA, built on these functions, remains resilient to LLM ranking noise.

The main contributions of this chapter are summarized as follows:

1. We leverage LLMs to generate dense agent-specific rewards based on a natural language description of the team's goal, successfully handling the credit assignment.
2. We empirically show that our approach leads to higher policy returns and faster convergence speeds than baseline methods, even when rankings are generated from smaller, more error-prone LLMs.

3.1 Methods

Existing RLAIF approaches [28] do not lend themselves well to multi-agent settings when ranking joint state-actions. Consider a two-agent scenario in which the agents perform actions with conflicting contributions toward the team goal: one positive and one negative. The positive reward from a beneficial action that contributes to the team's success is canceled out by the negative reward from another agent. This results in an ambiguous state which is difficult for an LLM to rank when considering both agents, ultimately resulting in a sparse rather than dense reward function. In contrast, our LCA approach seeks to decompose the joint preference ranking into individual preference rankings for the purpose of learning individual reward functions, overcoming this issue.

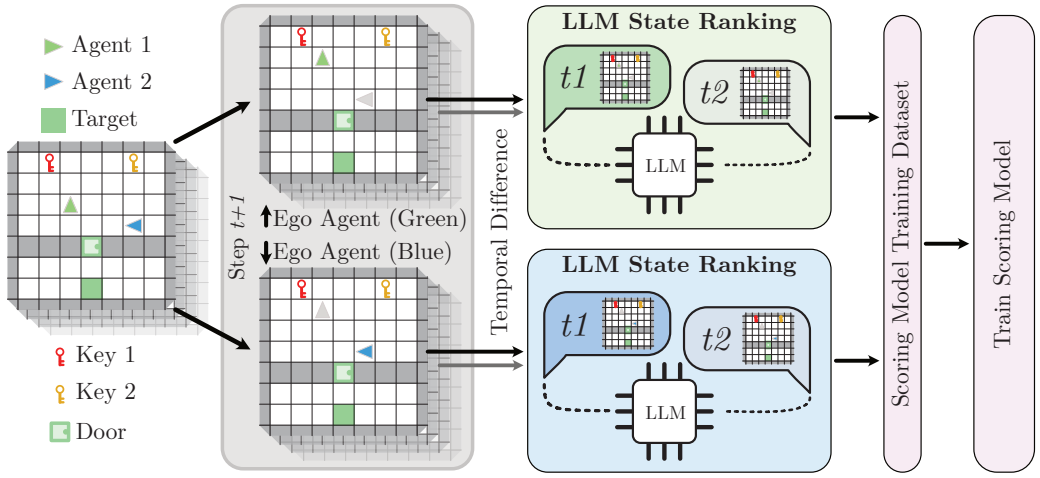


Figure 3.1: Overview of our method LCA: We first generate the agent-specific encodings of state observations, and then prompt an LLM to execute pairwise state ranking from each agent's perspective in the contexts of collaboration. Specifically, if ranking state pairs in Agent 1's perspective, Agent 1 will be encoded as the "ego" agent and other agents as "teammates" in the observation, allowing the LLM to differentiate them with the language-based observation description. The individual rewards trained with such agent-specific ranking results properly handle the credit assignment in MARL. We test our approach in the grid world and pistonball environments.

3.1.1 LLM-based Reward Decomposition

Describing Team Goals from Team Rewards

Given that not all environments provide explicit, natural language descriptions of states, goals, or sub-goals, this information can be inferred from the team reward structure by investigating a trajectory sampled beforehand. Without loss of generalization, we assume that there exists one team reward function, $r_t(s_i)$, from the environment, which is usually sparse (We assume it does not include step penalty and is not finely hand-crafted). Therefore, on a trajectory randomly sampled without a limit of max steps - which means it ends when the team task is completed - there are only a few states s_i where $r_t(s_i) \neq 0$. If $r_t(s_i) > 0$, s_i should be a key landmark of completing the team task. If $r_t(s_i) < 0$, it would be critical to avoid this state s_i. Therefore, the natural language description of such s_i following the order they appear on the sampled trajectory can provide LLM enough information about how the agent team should complete the task, which will be critical information for agent-specific state ranking.

Agent-specific State Ranking

We prompt an LLM to implicitly assign credits to each agent separately by ranking state pairs based on the agent's own actions from that agent's perspective. We first generate an agent-specific encoding o^i of the observation o of a state s by labeling the agent i itself as the "ego" and any other agent as the "teammate", allowing the LLM and state-scoring models to identify which agent they are evaluating. Given any state transition (s, a, s'), where $a = \langle a_1, \ldots, a_n \rangle$ and n is the number of agents, the LLM generates a preference label for agent i as:

$$y^i(s, a, s') = y^i(o^i, a_i, o'^i).$$

The LLM is then prompted to reason from agent i's perspective to determine whether the agent's action a_i between these two states, o^i and o'^i, is appropriate for collaboration. If agent i performs a correct action while another agent j performs an incorrect one—a scenario where single-agent-style RLHF struggles to generate a single ranking—this method assigns:

$$y^i(s, a, s') = (o'^i \succ o^i) = (s' \succ s),$$

and

$$y^j(s, a_j, s') = (o^j \succ o'^j) = (s \succ s').$$

LLM-Guided Individual Reward Training

Given that the LLM implicitly assigns credit by generating differentiated rankings for each agent i $\mathcal{D}^i = \{(s_a, s_b, y^i) | s_a, s_b \in \mathcal{S}\}$, these rankings can be used to train individual state-scoring models $\sigma^i(o^i)$. The loss function for each individual state-scoring model will be

$$\begin{aligned}\mathcal{L}^i &= -\mathbb{E}_{(s_a,s_b,y^i)\sim\mathcal{D}^i}\Big[\mathbb{I}\{y^i = (s_a \succ s_b)\}\log P^i_\psi[o^i_a \succ o^i_b] + \mathbb{I}\{y^i = (s_b \succ s_a)\}\log P^i_\psi[o^i_b \succ o^i_a]\Big], \\ &= -\mathbb{E}_{(s_a,s_b,y^i)\sim\mathcal{D}^i}\Big[conf\{y^i = (s_a \succ s_b)\}\log(sigmoid(\sigma^i_\psi(o^i_a) - \sigma^i_\psi(o^i_b)))+ \\ &conf\{y^i = (s_b \succ s_a)\}\log(sigmoid(\sigma^i_\psi(o^i_b) - \sigma^i_\psi(o^i_a)))\Big].\end{aligned} \tag{3.1}$$

The individual reward will be formulated as

$$r_i(s, a_i, s') = \sigma^i_\psi(o'^i) - \sigma^i_\psi(o^i) \tag{3.2}$$

except the case where the agent i stays still without taking an actual action and the reward will be 0.

This reward function generalizes potential-based rewards from single-agent to multi-agent settings, while maintaining the claims in the last chapter that the RLAIF loss encodes ranking confidence, and that inconsistent rankings, implying that the confidence of two possible ranking results over a state pair are closer, possible drive the individual reward towards zero with the loss function of the state-scoring model in Eq. 3.1. Intuitively, this means that the individual reward functions are robust to ranking errors stemming from high uncertainty when each state-action pair is ranked multiple times.

Additionally, it is unnecessary to train one reward function for each agent if agents are homogeneous with the same individual task. Since these agents take the same, exchangeable role in the team, for a transition (s_a, a, s_b) with encoded observation o^i_a, o^i_b for agent i, there must exist another transition (s'_a, a, s'_b) with encoded observation o'^j_a, o'^j_b for agent j such that $o^i_a = o'^j_a, o^i_b = o'^j_b$. The loss function for agent i's state-scoring model over the

preference dataset $\mathcal{D}^i$ can be written as

$$\begin{aligned}\mathcal{L}^i &= -\mathbb{E}_{(s_a,s_b,y^i)\sim\mathcal{D}^i}\Big[\mathbb{I}\{y^i=(o_a^i\succ o_b^i)\}\log P_\psi^i[o_a^i\succ o_b^i]+\mathbb{I}\{y^i=(o_b^i\succ o_a^i)\}\log P_\psi^i[o_b^i\succ o_a^i]\Big],\\ &= -\mathbb{E}_{(s'_a,s'_b,y^i)\sim\mathcal{D}^i}\Big[\mathbb{I}\{y^i=(o_a'^j\succ o_b'^j)\}\log P_\psi^i[o_a'^j\succ o_b'^j]+\mathbb{I}\{y^i=(o_b'^j\succ o_a'^j)\}\log P_\psi^i[o_b'^j\succ o_a'^j]\Big].\end{aligned} \tag{3.3}$$

If agent i and j share $y^i, \mathcal{D}^i, P_\psi^i$, which means they share the ranking dataset and the state-scoring model, $\mathcal{L}^i$ will be directly transformed to $\mathcal{L}^j$. Therefore, homogeneous agents with the same tasks can be grouped together and share the same reward function. The single reward function can handle the credit assignment among them and gives distinct individual rewards by taking differentiated observations in their own view over the current state. We only need to train different reward functions for heterogeneous agents or homogeneous ones with different pre-assigned tasks.

3.1.2 Prompt Designs for Agent-specific State Ranking Reflecting Collaboration

Although we decompose the joint state-action rankings into individual rankings, it does not mean the ranking for each agent is the same as it would be in a single-agent scenario. Although the LLM thinks in the “ego" agent’s view, it needs to think for the team rather than the “ego" agent itself so that the agent-specific ranking can evaluate the collaboration between the “ego" agent and the “teammate" agents and correctly assign credit for collaboration. This section introduces how to achieve this via prompt design.

During collaboration, each agent’s policy depends on the states and actions of other agents. We design our prompt to make this dependency understandable by LLMs. We consider two types of collaboration dependencies:

1. **Behavior dependence:** Teammates’ current state and latest action influence the ego’s current action choice.
2. **Task dependence:** The “ego" agent needs to change its task steps according to others’ task requirements.

The Two-Switch and Victim-Rubble environments introduced in the experiment section 3.2.1 are two examples corresponding to these collaboration dependencies. We introduce prompt designs for the above two dependency types separately with these two examples.

Prompt Design for Behavior Dependence

In the Two-Switch environment (see Sec. 3.2.1 for description), the optimal teamwork requires two agents to separately trigger each switch and unlock the door. Without specific guidance and inter-agent communication, it is natural that the “ego" agent will observe which switch its teammate is moving towards and then choose the other switch. However, if the teammate is undecided and fails to commit to a particular switch, this can lead to a deadlock as each agent adapts its goal based on the other agent’s goal inferred by the teammate’s latest action. Such behavior is undesirable as it can introduce non-stationarity into the environment, i.e. from the perspective of the “ego" agent the teammate’s behavior can rapidly change as its policy updates. In addition to destabilizing training, this kind of behavior dependency can create sub-optimal policies in which one agent is “lazy” and fails to contribute to the team’s success [30].

The agent-specific LLM-generated rankings produced by our approach are designed to address these issues. We instruct the LLM to **believe the teammates are acting with the optimal policy** when generating rankings. The resulting individual reward function will encourage the “ego" agent to pursue optimal actions under the assumption that the teammates will act similarly optimal. In this way, the agents avoid falling into both deadlocks and behaviors where they must compensate for “lazy” teammate behavior. To achieve this, besides offering the team target, key steps, the environment, and current states of all agents, we add the following constraint to our prompt:

Assuming the “teammate" agent will take the best action for the team at this step, *does the current action taken by the “ego" agent help ... from the view of team?*

With this prompt, the LLM understands that it should rank state pairs based on whether the “ego" has made the optimal decision, without being influenced by or hesitating over the teammate’s subsequent actions.

Prompt Design for Task Dependence

In the Victim-Rubble environment (see Sec. 3.2.1 for description), optimal teamwork requires two agents to adjust the order of their task steps in response to the needs of their teammate. Specifically, the green agent must prioritize which victims to heal and the orange

agent must prioritize which pieces of rubble to remove. For example, if the agents start in the center room, then the orange agent should prioritize removing the rubble in the right rooms as it blocks access to a victim which the green agent will need to heal. And depending on the relative location of the orange and green agents, the green agent may be more optimal by first healing the accessible victim in the left room while it waits for the orange agent to open up access to the blocked victim.

To achieve this level of collaboration, besides offering the description of the team target, key steps, the environment and the current states of agents, the prompt should first describe the dependency between different agents' tasks:

```
The green agent always prioritizes rescuing victims whose path is free of any rubble,
↪  waiting for the orange agent to remove rubbles and clear paths.
```

Then describe the current dependency constraints:

```
Rubble1: Chamber5 (8,1), **blocking the only passage** between Chamber5 and 3 from
↪  the side of Chamber5
Rubble2: Chamber5 (9,2), **blocking the only passage** reaching Chamber4 which
↪  contains one Victim
```

And also tell LLM which role the "ego" agent takes:

```
You are the orange agent at Chamber3 (4,3)
```

Combining these information, the LLM can identify the next rubble the orange agent should first remove. Part of the example response is as follows:

```
The next step for the orange agent should be to clear the path to Chamber4 so that
↪  the green agent can rescue the victim.
```

3.2 Experiments

We tested LCA in three multi-agent collaboration scenarios without inter-agent communication, outer access to policy models, or state transition models. Fig. 3.2 shows the layouts.

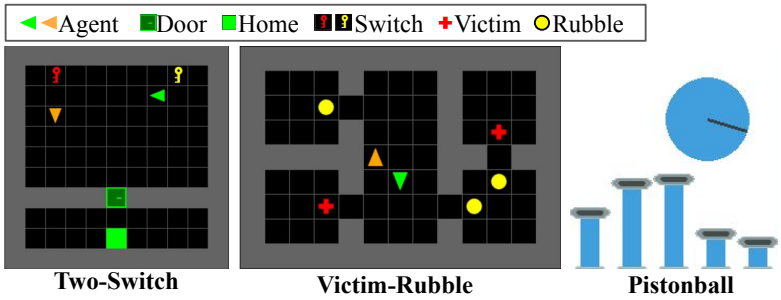


Figure 3.2: Grid world environments with Two-Switch (left), Victim-Rubble (middle) and Pistonball (right) variants from left to right.

3.2.1 Experiment Setup

Grid World. We examine two multi-agent collaboration scenarios within Grid World [46]: **Two-Switch** and **Victim-Rubble**.

In the Two-Switch variant, two agents (green and orange triangles) start from random positions in the upper room and at least one of them should navigate to the target (green rectangle) in the lower room. There is one locked door that blocks the agent's way to the goal and only opens when both switches have been triggered. To unlock the door, the agents must move to the switches, face the switches and trigger them. Therefore, agents are expected to distribute switches between each other and trigger each switch separately. This should be achieved by observing the other agent's position since agents cannot communicate.

In the Victim-Rubble variant, the green agent must heal all victims (red crosses) and the orange agent must clear all rubble (yellow circles) in the rooms. There is always one victim lying at the end of a long corridor (the upper-right one in the middle environment in Fig. 3.2) and there are always two pieces of rubble blocking the way to this victim. Additionally, there is always one piece of rubble blocking nothing and one victim to which the passage is free to pass through. To complete the task as fast as possible, the orange agent should learn to first clear the rubble blocking the passage, and the green agent should learn to first heal the accessible victim.

Pistonball. We also investigate the Pistonball environment from PettingZoo [48]. There are five pistons which are five independent agents in this environment, moving upwards and downwards. They aim to push the ball starting from the rightmost point of the environment

to reach the leftmost point with the least steps.

We compare our approach with the following baselines:

- **MAPPO with the default team reward** This is the vanilla case of MARL where no explicit credit assignment is done, utilizing the team's overall objective of each environment with human-specified reward functions given to all agents. In grid world variants, each agent receives a default reward of 0 for failure and 1 when the team completes the task. Additionally, both agents earn 1 if any agent either handles a switch, victim, or rubble, or arrives home. A step penalty of $-n/n_{max}$ is applied, where n is the step count and n_{max} is the episode's maximum time steps. In the Pistonball variant, all agents obtain the team reward of 1 when the ball reaches the leftmost point, and a step penalty of -0.1 for each step.
- **MAPPO with the default team reward plus individual rewards** Besides the team reward based on outcomes (success/failure), this baseline assigns credits in a naive way with default hand-crafted individual rewards. In the Two-Switch variant, the default individual reward is defined as 1 if the agent triggers a switch or arrives at the goal. In the Victim-Rubble variant, the individual reward is 1 for the orange agent if it removes a piece of rubble, and for the green agent if it heals one victim. There are no simple individual rewards in the Pistonball variant, which thus does not have this baseline.
- **QMIX and VDN with the default team reward** These two baselines decompose the team reward described above into individual Q values for credit assignment [36, 44]. We evaluate LCA against these two classical value-decomposition methods to show its effectiveness.

Team rewards often fail to discourage poor agent behaviors. While naive hand-crafted individual rewards can partially address this, their sparsity limits effectiveness. Our method's dense individual rewards are expected to significantly outperform these alternatives. Specifically speaking,

1) In Two-Switch: Team rewards grant all agents +1 when a switch is triggered, regardless of which agent triggers it. If the orange agent learns this first, the green agent may remain idle, letting the orange agent trigger both switches and still earning +2. This inefficiency increases team steps. Our rewards immediately penalize agents once they act improperly.

2) In Victim-Rubble: If the orange agent fails to clear the rubble blocking a passage or remains idle, the green agent can only save accessible victims. Both agents still earn +1 team reward for this action, despite reduced overall performance. Our rewards immediately penalize the orange agent once it stops moving toward the critical rubble.

3) In Pistonball: Team rewards penalize all pistons if the ball moves right, even if some act correctly. There are no straightforward individual rewards, unless with extensive tuning. Our dense rewards target only the piston directly responsible for the incorrect ball motion.

These challenging collaborative scenarios make the three environments ideal for testing our method against baseline approaches. Without loss of generality, we employ IPPO as the underlying policy-training framework [38] and assume the agent has no knowledge of the task before training, i.e., is randomly initialized.

We randomly sampled sequential state pairs to train state-scoring models and formulate potential-difference reward functions in each environment. Since the agents in the Two-Switch environment are homogeneous with the same individual tasks, a single state-scoring model is trained with 4400 state pairs in total for two agents. Similarly, a single state-scoring model is trained with 1000 state pairs in total for five agents in the Pistonball environment. Two state-scoring models are trained for the two heterogeneous agents in the Victim-Rubble variant and each takes 2000 state pairs.

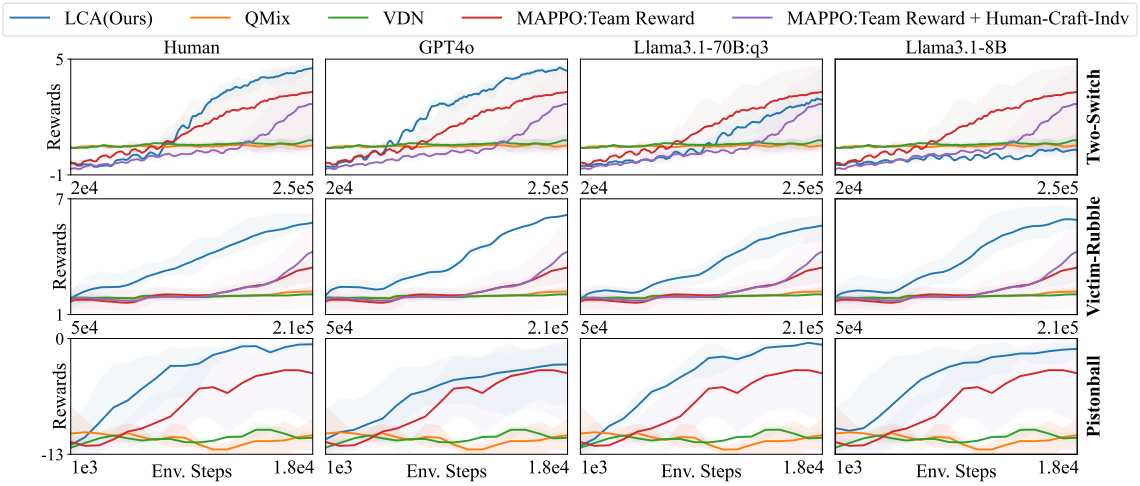


Figure 3.3: The average learning curves with reward functions trained from single LLM ranking per state pair in the Two-Switch, Victim-Rubble and Pistonball environments over 3 random seeds, with the return variance visualized as shaded areas. The training returns shown as the y axis are measured with vanilla individual rewards plus team rewards.

3.2.2 Single-Query Evaluation

We first evaluate the performance of our method using a single query to the LLM to rank each sampled state pair. In each environment, we train our state-scoring models with the human ranking-heuristic function, which serves as an estimated ground-truth ranking based on human heuristics, and evaluate them against 3 LLMs: GPT-4 [1], and two versions of Llama-3.1 [51]—one small and fast version with 70B parameters, referred to as q3_K_M, and another with 8B parameters. Then the potential-difference rewards based on state-scoring models above are employed to train 3 RL policies with random seeds and initializations for each method. The results, as well as the baseline performance, are shown in Fig. 3.3.

In the Two-Switch variant, our method with human heuristics and GPT4o achieves the optimal return ($5 - step\_penalty$) in 250k training steps with faster learning speed and less variance than baselines. In this single-query experiment, it is normal to observe that policies trained with the quantized Llama3.1-70B:q3 learn more slowly and the rewards from Llama-3.1 8B generating noisy outputs fail to train a useful policy according to the last chapter. They can be further improved with multiple ranking queries per state pair, particularly Llama 3.1-70B:q3, which outperforms the baselines with just two queries, as demonstrated in the next section on multi-query experiments.

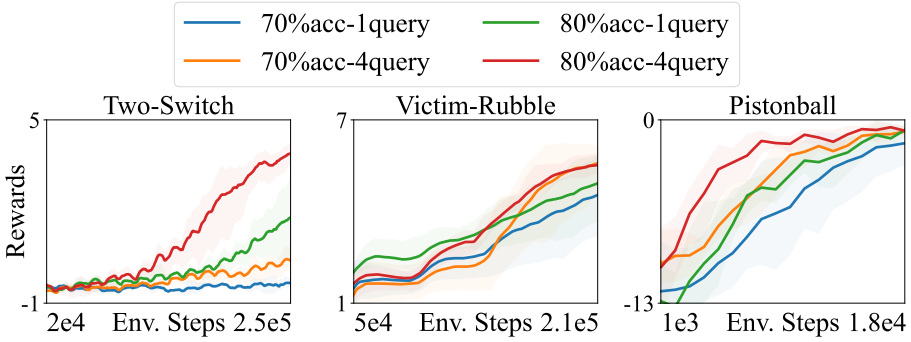


Figure 3.4: The learning curves with reward functions trained from four-query synthetic experiments over 3 random seeds.

In the Victim-Rubble variant, the default reward easily fails to reach a high return in 210k training steps while LCA with human, GPT4o, Llama3.1-70B:q3 and Llama3.1-8B rankings converges much faster and reaches the optimal reward ($7 - step\_penalty$). GPT4o-reward rollout over an episode in Appendix B shows that LCA effectively decomposes sparse

team rewards into dense informative individual rewards. However, the imperfect human ranking heuristic causes our method to learn slightly more slowly than with GPT4o. The human ranking heuristic in this environment forces the green agent to always first save the accessible victim and the orange agent to always first remove the rubbles blocking passages. However, on certain trajectories from suboptimal policies during training, the orange agent may encounter harmless rubble before clearing other rubble, making immediate removal more efficient than returning later. Llama3.1-70B:q3 can have similar ranking flaws. Such ranking flaws may lead to some local optimality and slightly slow down the training speed.

In the Pistonball variant, the baselines fail with the sparse vanilla team reward, while our method with human, Llama3.1-8B and -70B:q3 learn the optimal policy much faster with less variance in 18k training steps. Compared with other LLMs, GPT4o struggles a bit to understand the introduced physical mechanism, so it slows down the training process slightly but still trains some useful policies.

Due to the ϵ-greedy controller PPO methods do not adopt, QMIX and VDN can sometimes start training with a higher return, where $\epsilon = 1$, than IPPO-based LCA and MAPPO. However, sparse rewards cause QMIX and VDN to learn slowly and fail to reach significant returns within LCA's limited training steps, though they learn much faster after a few hundred thousand training steps exceeding LCA training time. We also tried LIIR [12] and encountered similar consequences, so we ignore its results here.

3.2.3 Multi-Query Evaluation

This section verifies that our method successfully inherits the robustness of potential-based rewards to noisy preference labels, extending the multi-query approach from single-agent scenarios to MARL. The multi-query approach is to query about ranking over each state pair in the ranking dataset multiple times to handle LLM-ranking inconsistencies for small but fast LLMs generating errors.

Synthetic Ranking Evaluation

To evaluate LCA's robustness, we synthesized ranking datasets with 70% and 80% accuracy and simulated ranking results with four queries per state pair across three environments. These rankings have correctness between 60% (near random guessing) and 90% (high accuracy), providing a comprehensive assessment of LCA's performance. The four-query

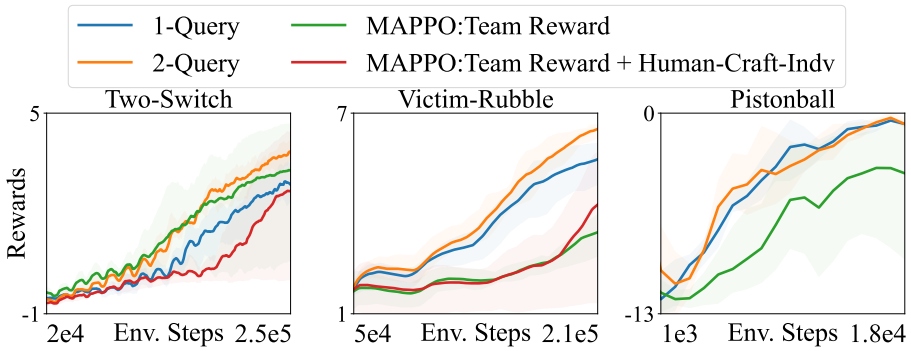


Figure 3.5: The learning curves with reward functions trained from two-query with Llama3.1-70B:q3 over 3 random seeds.

ranking datasets are synthesized based on four copies of the human-ranking datasets by randomly flipping a specific percentage of rankings. The data are used to train state-scoring models separately, based on which we obtain multi-query potential-based rewards. Fig. 3.4 shows the resulting policy learning curves averaged over 3 random seeds. We can see the four-query rankings significantly improve the training speed and returns in all environments, especially the Two-Switch variant. In this scenario, the four-query rankings of 80% correctness dramatically raise the training returns to the optimal. The policy with rewards from single-query rankings of 70% correctness fails, while the four-query rankings of 70% accuracy considerably improve the individual reward quality and train some useful policies.

LLM Two-Query Evaluation

As discussed above, the q3 version of the Llama3.1-70B is faster and more accessible than the full-sized version but generates more errors and has a flawed performance when training credit-aware individual rewards using a single query per state pair. This section shows that the learning speed can be accelerated with less variance and the training return can be raised to the optimal if using one more query to rank each state pair, as demonstrated by the learning curves averaged over 3 random seeds in Fig. 3.5. In the Pistonball environment, since the policy trained with single-query Llama3.1-70B:q3 rankings is already with the fastest learning speed, least variance and optimal training returns, the improvement from the multi-query approach is limited and the two-query variation remains on par with it.

3.3 Conclusion

In this chapter, LCA leverages LLMs to handle the critical challenge of credit assignment in MARL in environments with sparse rewards. This LCA method decomposes sparse team rewards into dense, agent-specific ones by using LLM to evaluate each agent's actions in the contexts of collaboration. The potential-based reward-shaping mechanism mitigates the impact of LLM hallucination, enhancing the robustness and reliability of our method. Our extensive experiments demonstrate that multi-agent collaboration policies trained with our LLM-guided individual rewards achieve faster convergence and higher policy returns compared to state-of-the-art MARL baselines. Experiments also show the resilience of LCA to ranking errors. Therefore, without significant performance degradation, LCA is applicable to smaller and more accessible language models.

Chapter 4

Conclusions

This thesis introduces a simple yet effective method for leveraging noisy feedback from LLM to train RL reward models in both single-agent and multi-agent settings. Our approach learns a potential-based reward function from repeated LLM-generated preference rankings. In this way, it can assign uninformative rewards to states where the LLM expresses uncertainty, as demonstrated by our theoretical analysis. In the multi-agent setting, our method preserves robustness to LLM noises while addressing the credit assignment problem. It generates dense, credit-aware, individual rewards from sparse team rewards by evaluating each agent's actions within collaborative scenarios using LLM feedback. We provide empirical evidence showing that our method leads to faster convergence and higher policy returns across a range of discrete and continuous multi-agent environments, despite substantial noises in LLM rankings.

This thesis extends RLAIF to handle single- and multi-agent tasks with small language models that are fast and lightweight but prone to producing inaccurate rankings. Unlike top-tier models such as GPT-4, which have a large number of parameters and require significant computational resources and time to generate sufficient preference samples across the state space, our approach offers a practical and economic alternative.

Furthermore, our method is theoretically compatible with visual and multimodal environments that possess richer state and action spaces and local observations, which can be ranked by LLMs or VLMs. Exploring these scenarios will be the focus of our future work. A limitation is that we currently assume that all sequential state pairs can be randomly sampled from the environment. While this is true in most simulated environments, this

assumption is violated in others such as the real world. In future work, we will adopt iterative algorithms which alternate training the policy and sampling state pairs for ranking.

Appendix A

Theoretical Proof: Inconsistent rankings lead to uninformative rewards

Lemma 1. *In the scope of RL based on LLM feedback, the confidence-based preference loss is equivalent to the standard preference loss used by state-score model training over multi-query ranking datasets.*

Proof. Given that the the confidence of ranking s_a higher than s_b, $conf\{y = (s_a \succ s_b)\}$, is defined as $\frac{N(s_a \succ s_b)}{N_{query}(s_a, s_b)}$. where $N(s_a \succ s_b)$ denotes the number of times LLM ranks s_a higher than s_b, and $N_{query}(s_a, s_b)$ denotes the total number of queries on s_a and s_b.

The standard preference loss over multiple-query ranking dataset $\mathcal{D}$ can be written as

$$\begin{aligned}
&\mathcal{L}_{\mathcal{D}} = -\mathbb{E}_{(s_a,s_b,y)\sim\mathcal{D}}\Bigg[\mathbb{E}_{N_{query}}\Big[\mathbb{I}\{y = (s_a \succ s_b)\}\log(sigmoid(\sigma_\psi(s_a) - \sigma_\psi(s_b)))+ \\
&\mathbb{I}\{y = (s_b \succ s_a)\}\log(sigmoid(\sigma_\psi(s_b) - \sigma_\psi(s_a)))\Big]\Bigg] \\
&= -\mathbb{E}_{(s_a,s_b,y)\sim\mathcal{D}}\Bigg[conf\{y = (s_a \succ s_b)\}\log(sigmoid(\sigma_\psi(s_a) - \sigma_\psi(s_b))) \\
&+ conf\{y = (s_b \succ s_a)\}\log(sigmoid(\sigma_\psi(s_b) - \sigma_\psi(s_a)))\Bigg].
\end{aligned} \tag{A.1}$$

□

Theorem 1. *As inconsistency of a ranking over two states increases, the scores of these*

two states converge to the same value.

Proof. Based on Equation A.1,

$$\mathcal{L}_{\mathcal{D}} = -\mathbb{E}_{(s_a,s_b,y)\sim\mathcal{D}}\Big[conf\{y=(s_a\succ s_b)\}\log(sigmoid(\sigma_\psi(s_a)-\sigma_\psi(s_b)))+ \\ (1-conf\{y=(s_a\succ s_b)\})\log(1-sigmoid(\sigma_\psi(s_a)-\sigma_\psi(s_b)))\Big]. \tag{A.2}$$

Take an arbitrary state pair (s_0, s_1) from $\mathcal{D}$. As inconsistency of the ranking over s_0 and s_1 increases, $conf\{y=(s_0\succ s_1)\}\to 0.5$. Denote $sigmoid(\sigma_\psi(s_0)-\sigma_\psi(s_1))$ as $p_{0,1}$, $\mathcal{L}$ over other states in $\mathcal{D}$ without s_0, s_1 as $\mathcal{L}_{\mathcal{D}\setminus\{s_0,s_1\}}$,

$$\lim_{conf\{y=(s_0\succ s_1)\}\to 0.5}\mathcal{L}_{\mathcal{D}} \to -\frac{1}{2|\mathcal{D}|}\log(p_{0,1}(1-p_{0,1}))+\mathcal{L}_{\mathcal{D}\setminus\{s_0,s_1\}} \geq \frac{1}{|\mathcal{D}|}\log 2+\mathcal{L}_{\mathcal{D}\setminus\{s_0,s_1\}}. \tag{A.3}$$

If and only if $p_{0,1}\to\frac{1}{2}$, where $\sigma_\psi(s_0)-\sigma_\psi(s_1)\to 0$, $\lim_{conf\{y=(s_0\succ s_1)\}\to 0.5}\mathcal{L}_{\mathcal{D}}\to\frac{1}{|\mathcal{D}|}\log 2+\mathcal{L}_{\mathcal{D}\setminus\{s_0,s_1\}}$, reaching the lower bound.

Therefore, when training the state-score model with this loss $\mathcal{L}$, the scores of any two states whose ranking confidence is close to 50% will be pushed to the same value. □

Appendix B

Individual-LCA-Reward Rollout over an Episode

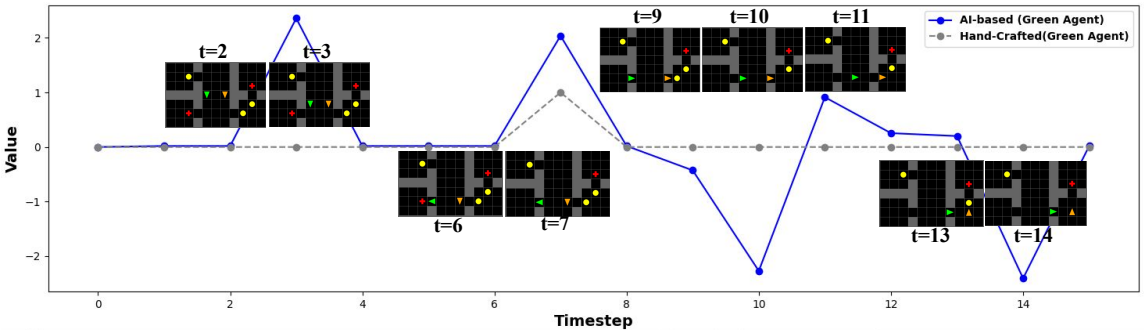


Figure B.1: Rolling out individual rewards (blue line) over states of an episode from time steps 0 to 15 in the Victim-Rubble environment. The individual rewards here are the potential-based rewards trained with single-query GPT4o rankings.

We plotted the green agent's individual rewards at states from a continuous episode in the Victim-Rubble environment. The individual rewards here are trained with single-query GPT4o rankings. Compared with the default sparse team reward (grey line), we can see that LCA successfully generates dense individual rewards evaluating individual actions in the contexts of collaboration. Besides giving positive rewards when the green agent makes significant progress (ie. saving victims) like simple hand-crafted reward functions do, LCA also rewards the green agent when it makes a critical turn or movement to the correct target

(t=3, 11 in Fig. B.1). Meanwhile, LCA individual rewards punish the green agent not only when it takes the wrong action, but also when its teammate makes significant progress but it does nothing special (t=10, 14). It seems that the LLM tends to push the green agent to make progress, effectively avoiding lazy agents.

Bibliography

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 2.2.3, 3.2.2

[2] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020. 1.1

[3] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022. 1.1

[4] Erdem Bıyık, Malayandi Palan, Nicholas C Landolfi, Dylan P Losey, and Dorsa Sadigh. Asking easy questions: A user-friendly approach to active reward learning. *arXiv preprint arXiv:1910.04365*, 2019. 1.2.1

[5] Erdem Bıyık, Dylan P Losey, Malayandi Palan, Nicholas C Landolfi, Gleb Shevchuk, and Dorsa Sadigh. Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences. *The International Journal of Robotics Research*, 41(1):45–67, 2022. 1.2.1

[6] Serena Booth, W Bradley Knox, Julie Shah, Scott Niekum, Peter Stone, and Alessandro Allievi. The perils of trial-and-error reward design: misdesign through overfitting and invalid task specifications. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 5920–5929, 2023. 1.2.2

[7] Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952. 1.3.4

[8] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. 2.2, 2.2.1, 2.2.3

[9] Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro

Freire, et al. Open problems and fundamental limitations of reinforcement learning from human feedback. *arXiv preprint arXiv:2307.15217*, 2023. 1.1, 1.2.1

[10] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017. 1.2.1, 1.2.2, 1.3.4, 2.1, 2.2.1

[11] Tianshu Chu, Jie Wang, Lara Codecà, and Zhaojian Li. Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE transactions on intelligent transportation systems*, 21(3):1086–1095, 2019. 1.1

[12] Yali Du, Lei Han, Meng Fang, Ji Liu, Tianhong Dai, and Dacheng Tao. Liir: Learning individual intrinsic reward in multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019. 1.2.2, 3.2.2

[13] Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. Guiding pretraining in reinforcement learning with large language models. In *International Conference on Machine Learning*, pages 8657–8677. PMLR, 2023. 1.2.1

[14] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018. 1.1, 1.2.2

[15] Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in atari. *Advances in neural information processing systems*, 31, 2018. 1.3.4

[16] Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro Ortega, DJ Strouse, Joel Z Leibo, and Nando De Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *International conference on machine learning*, pages 3040–3049. PMLR, 2019. 1.2.2

[17] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024. 2.2.3

[18] Aditya Kapoor, Benjamin Freed, Howie Choset, and Jeff Schneider. Assigning credit with partial reward decoupling in multi-agent proximal policy optimization, 2024. URL https://arxiv.org/abs/2408.04295. 1.2.2

[19] Sungdong Kim, Sanghwan Bae, Jamin Shin, Soyoung Kang, Donghyun Kwak, Kang Min Yoo, and Minjoon Seo. Aligning large language models through synthetic feedback. *arXiv preprint arXiv:2305.13735*, 2023. 1.1

[20] Martin Klissarov, Pierluca D'Oro, Shagun Sodhani, Roberta Raileanu, Pierre-Luc Bacon, Pascal Vincent, Amy Zhang, and Mikael Henaff. Motif: Intrinsic motivation

from artificial intelligence feedback. *arXiv preprint arXiv:2310.00166*, 2023. 1.1, 1.2.1

[21] W Bradley Knox, Alessandro Allievi, Holger Banzhaf, Felix Schmitt, and Peter Stone. Reward (mis) design for autonomous driving. *Artificial Intelligence*, 316:103829, 2023. 1.2.2

[22] Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. Reward design with language models. *arXiv preprint arXiv:2303.00001*, 2023. 1.2.1

[23] Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Ren Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, et al. Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai feedback. In *Forty-first International Conference on Machine Learning*. 1.2.2

[24] Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*, 2023. 1.1, 1.2.1

[25] Kimin Lee, Laura Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. *arXiv preprint arXiv:2106.05091*, 2021. 1.3.4

[26] Kimin Lee, Laura Smith, Anca Dragan, and Pieter Abbeel. B-pref: Benchmarking preference-based reinforcement learning. *arXiv preprint arXiv:2111.03026*, 2021. 1.3.4

[27] Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*, 2018. 1.1, 1.2.2

[28] Muhan Lin, Shuyang Shi, Yue Guo, Behdad Chalaki, Vaishnav Tadiparthi, Ehsan Moradi Pari, Simon Stepputtis, Joseph Campbell, and Katia Sycara. Navigating noisy feedback: Enhancing reinforcement learning with error-prone language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024. 2, 3.1

[29] Muhan Lin, Shuyang Shi, Yue Guo, Vaishnav Tadiparthi, Behdad Chalaki, Ehsan Moradi Pari, Simon Stepputtis, Woojun Kim, Joseph Campbell, and Katia Sycara. Speaking the language of teamwork: Llm-guided credit assignment in multi-agent reinforcement learning. *arXiv preprint arXiv:2502.03723*, 2025. 3

[30] Boyin Liu, Zhiqiang Pu, Yi Pan, Jianqiang Yi, Yanyan Liang, and Du Zhang. Lazy agents: a new perspective on solving sparse reward problem in multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 21937–21950. PMLR, 2023. 1.1, 1.2.2, 3.1.2

[31] Yao Lu, Karol Hausman, Yevgen Chebotar, Mengyuan Yan, Eric Jang, Alexander Herzog, Ted Xiao, Alex Irpan, Mohi Khansari, Dmitry Kalashnikov, et al. Aw-opt: Learning robotic skills with imitation andreinforcement at scale. In *Conference on Robot Learning*, pages 1078–1088. PMLR, 2022. 1.1

[32] Qing Lyu, Kumar Shridhar, Chaitanya Malaviya, Li Zhang, Yanai Elazar, Niket Tandon, Marianna Apidianaki, Mrinmaya Sachan, and Chris Callison-Burch. Calibrating large language models with sample consistency. *arXiv preprint arXiv:2402.13904*, 2024. 2.1.1

[33] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023. 1.1

[34] Laetitia Matignon, Guillaume J Laurent, and Nadine Le Fort-Piat. Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems. *The Knowledge Engineering Review*, 27(1):1–31, 2012. 1.3.2

[35] 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. 1.2.1

[36] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning, 2020. URL https://arxiv.org/abs/2003.08839. 1.1, 1.2.2, 3.2.1

[37] Dorsa Sadigh, Anca D Dragan, Shankar Sastry, and Sanjit A Seshia. *Active preference-based learning of reward functions*. 2017. 1.2.1

[38] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. 1.2.2, 2.2.1, 3.2.1

[39] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016. 1.1

[40] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017. 1.1

[41] Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. *Advances in Neural Information Processing Systems*, 35:9460–9471, 2022. 1.1

[42] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International conference on machine learning*, pages 5887–5896. PMLR, 2019. 1.1

[43] Jianyu Su, Stephen Adams, and Peter A. Beling. Value-decomposition multi-agent actor-critics, 2020. URL https://arxiv.org/abs/2007.12306. 1.2.2

[44] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017. 1.1, 1.2.2, 3.2.1

[45] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 1.3.1

[46] Gokul Swamy, Christoph Dann, Rahul Kidambi, Zhiwei Steven Wu, and Alekh Agarwal. A minimaximalist approach to reinforcement learning from human feedback. *arXiv preprint arXiv:2401.04056*, 2024. 2.2.1, 3.2.1

[47] Sree Harsha Tanneru, Chirag Agarwal, and Himabindu Lakkaraju. Quantifying uncertainty in natural language explanations of large language models. In *International Conference on Artificial Intelligence and Statistics*, pages 1072–1080. PMLR, 2024. 2

[48] J Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis S Santos, Clemens Dieffendahl, Caroline Horsch, Rodrigo Perez-Vicente, Niall Williams, Yashas Lokesh, and Praveen Ravi. Pettingzoo: Gym for multi-agent reinforcement learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 15032–15043. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/7ed2d3454c5eea71148b11d0c25104ff-Paper.pdf. 3.2.1

[49] Andrea Lockerd Thomaz, Cynthia Breazeal, et al. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *Aaai*, volume 6, pages 1000–1005. Boston, MA, 2006. 1.2.1

[50] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012. 2.2.1

[51] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 2.2.3, 3.2.2

[52] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian

Schrittwieser, et al. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017. 1.1

[53] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling multi-agent q-learning. *arXiv preprint arXiv:2008.01062*, 2020. 1.1

[54] Yufei Wang, Zhanyi Sun, Jesse Zhang, Zhou Xian, Erdem Biyik, David Held, and Zackory Erickson. Rl-vlm-f: Reinforcement learning from vision language foundation model feedback. *arXiv preprint arXiv:2402.03681*, 2024. 1.2.1

[55] 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. 2.1.3

[56] Muning Wen, Jakub Grudzien Kuba, Runji Lin, Weinan Zhang, Ying Wen, Jun Wang, and Yaodong Yang. Multi-agent reinforcement learning is a sequence modeling problem, 2022. URL https://arxiv.org/abs/2205.14953. 1.2.2

[57] Marco A Wiering et al. Multi-agent reinforcement learning for traffic light control. In *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*, pages 1151–1158, 2000. 1.1

[58] Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. Text2reward: Automated dense reward function generation for reinforcement learning. *arXiv preprint arXiv:2309.11489*, 2023. 1.2.1

[59] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative, multi-agent games, 2022. URL https://arxiv.org/abs/2103.01955. 1.2.2

[60] Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montse Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, et al. Language to rewards for robotic skill synthesis. *arXiv preprint arXiv:2306.08647*, 2023. 1.2.1

[61] Alex Zhang, Ananya Parashar, and Dwaipayan Saha. A simple framework for intrinsic reward-shaping for rl using llm feedback. 1.2.2

[62] Ruiqi Zhang, Jing Hou, Florian Walter, Shangding Gu, Jiayi Guan, Florian Röhrbein, Yali Du, Panpan Cai, Guang Chen, and Alois Knoll. Multi-agent reinforcement learning for autonomous driving: A survey. *arXiv preprint arXiv:2408.09675*, 2024. 1.1

[63] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. Siren's song in the ai ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*, 2023. 1.1

[64] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario

Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019. 1.2.1