A Reward Analysis of Reinforcement Learning from Large Language Model Feedback

Muhan Lin[*]
Carnegie Mellon University

Shuyang Shi
Carnegie Mellon University

Yue Guo
Carnegie Mellon University

Behdad Chalaki
Honda Research Institute USA

Vaishnav Tadiparthi
Honda Research Institute USA

Simon Stepputtis
Carnegie Mellon University

Joseph Campbell
Carnegie Mellon University

Katia Sycara
Carnegie Mellon University

Abstract

The correct specification of reward models is a well-known challenge in reinforcement learning. Hand-crafted reward functions often lead to inefficient or suboptimal policies and may not be aligned with user values. Reinforcement learning from human feedback is a successful technique that can mitigate such issues, however, 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. In this paper, we study the advantages and limitations of reinforcement learning from large language model feedback and propose a simple yet effective method for soliciting and applying feedback as a potential-based shaping function. We theoretically and empirically show that our approach results in higher policy returns compared to prior works, even with significant ranking errors, and eliminates the need for complex post-processing of the reward function.

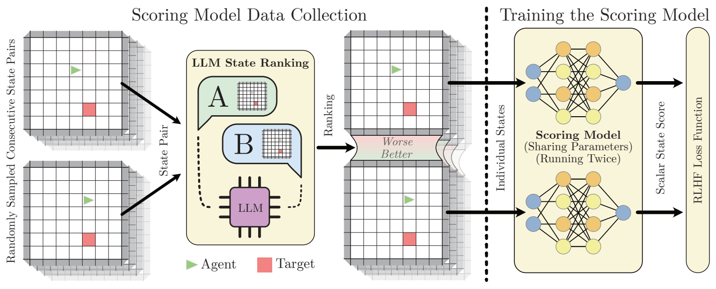
1 Introduction

The correct specification of task rewards is a well-known challenge in reinforcement learning (RL) (Leike et al., 2018). Complex tasks often necessitate complex, nuanced reward models, particularly as shaping terms may be required to guide exploration. However, handcrafting 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 (Skalse et al., 2022). 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.

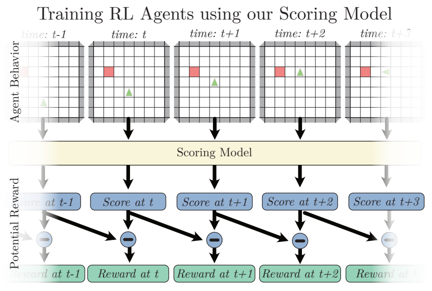
In this study, we aim to eliminate the dependence on handcrafted reward functions by training agents with reward functions derived from data. A notable method in this domain is reinforcement learning from human feedback (RLHF), where policy trajectories are ranked by humans. These rankings are then used 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 (Casper et al., 2023).

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

[*]Corresponding author: `muhanlin@cs.cmu.edu`


Scoring Model Data Collection
Training the Scoring Model
Randomly Sampled Consecutive State Pairs
State Pair
LLM State Ranking
A
B
LLM
Ranking
Worse
Better
Agent
Target
Individual States
Scoring Model (Sharing Parameters) (Running Twice)
Scalar State Score
RLHF Loss Function

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



(b) Agent training using the scoring model.

Figure 1: Our approach: (a) We randomly sample *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 score 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.

(RLAIF) (Lee et al., 2023; Bai et al., 2022; Kim et al., 2023). However, LLMs are well known to hallucinate and present false information as fact (Zhang et al., 2023), 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 (Klissarov et al., 2023).

In this work, we propose a simple and effective strategy for reinforcement learning in the face of unreliable LLM feedback. 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 allow 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 (Tanneru et al., 2024), we show that rewards issued from a potential-based scoring function learned over repeated rankings naturally reflect an LLM's uncertainty. Our contributions are as follow, 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.

2 Related Works

Constructing rewards based on human feedback has a long history (Thomaz et al., 2006). To efficiently use human domain knowledge and provide more generalizable rewards, human preference on episode segments (Sadigh et al., 2017; Christiano et al., 2017; Bıyık et al., 2019) and human demonstrations (Bıyık et al., 2022) 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 (Ziegler et al., 2019; Ouyang et al., 2022).

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 (Du et al., 2023), and are able to substitute humans in reward issuing (Kwon et al., 2023; Yu et al., 2023; Lee et al., 2023; Xie et al., 2023), or data collection and labeling for reward model training (Lee et al., 2023; Klissarov et al., 2023). While the former suffers from time and resource costs for training RL agents, the latter is becoming promising for training complex RL tasks (Wang et al., 2024).

An outstanding challenge with leveraging LLM-based feedback is that the performance of RLHF is dependent on the quality of feedback received (Casper et al., 2023). Different LLMs have distinct probabilities of giving wrong feedback, thus leading to rewards of varying quality. Casper et al. (2023) 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 (Christiano et al., 2017; Wang et al., 2024; Klissarov et al., 2023; Lee et al., 2023), but in practice, this needs post-processing on outputs of the reward model such as filtering (Klissarov et al., 2023), and normalization (Christiano et al., 2017). 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.

3 Background

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) (Sutton & Barto, 2018).

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.

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 (Christiano et al., 2017; Ibarz et al., 2018; Lee et al., 2021a;b). 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 parameterized reward model r_ψ, we compute the preference probability of a state pair with the standard Bradley-Terry model (Bradley & Terry, 1952),

$$P_\psi[s_b \succ s_a] = \frac{\exp(\sigma_\psi(s_b))}{\exp(\sigma_\psi(s_a)) + \exp(\sigma_\psi(s_b))} = sigmoid(\sigma_\psi(s_a) - \sigma_\psi(s_b)), \tag{1}$$

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:

$$\begin{aligned}\mathcal{L} = -\mathbb{E}_{(s_a, s_b, y)\sim\mathcal{D}}\Big[&\mathbb{I}\{y = (s_a \succ s_b)\} \log P_\psi[s_a \succ s_b] \\ &+ \mathbb{I}\{y = (s_b \succ s_a)\} \log P_\psi[s_b \succ s_a]\Big].\end{aligned} \tag{2}$$

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.

4 Methodology

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 1 in Appendix B). These errors present major challenges for reinforcement learning from LLM feedback, as they result in noises in the learned score function. Under the standard RLHF formulation where rewards are directly issued from the score function (Christiano et al., 2017), this can lead to inefficient exploration at best and, at worst, trap the agent in sub-optimal local minima.

4.1 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 (Lyu et al., 2024). 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 necessary for evaluating LLM feedback, given that low confidence causes considerable noises in feedback which manifests as incorrect rewards.

Based on the definition of feedback confidence, we can derive an equivalent form of the RLHF loss in Eq. 2 based on ranking confidence and consistency as follows:

$$\begin{aligned}\mathcal{L} = -\mathbb{E}_{(s_a,s_b,y)\sim\mathcal{D}}\Big[\mathbb{E}_{N_{query}}\Big[&\mathbb{I}\{y = (s_a \succ s_b)\}\log P_\psi[s_a \succ s_b] \\ &+ \mathbb{I}\{y = (s_b \succ s_a)\}\log P_\psi[s_b \succ s_a]\Big]\Big] \\ = -\mathbb{E}_{(s_a,s_b,y)\sim\mathcal{D}}\Big[&conf\{y = (s_a \succ s_b)\}\log(sigmoid(\sigma_\psi(s_a) - \sigma_\psi(s_b)))+ \\ &conf\{y = (s_b \succ s_a)\}\log(sigmoid(\sigma_\psi(s_b) - \sigma_\psi(s_a)))\Big].\end{aligned} \tag{3}$$

This loss function uses confidence-based weights to relate the scores between each state in ranked pairs. From this derivation, we can 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*, $|conf\{y = (s_a \succ s_b)\} - conf\{y = (s_b \succ s_a)\}|$ converges to 0. If the LLM is completely uncertain, $conf\{y = (s_a \succ s_b)\} = conf\{y = (s_b \succ s_a)\}$ and the loss function will minimize $|\sigma_\psi(s_a) - \sigma_\psi(s_b)|$, resulting in identical state scores such that $\sigma_\psi(s_a) = \sigma_\psi(s_b)$, supported by empirical evidence in Appendix A.

5 Potential-based Rewards for Learning with Noisy Feedback

The above observations stemming from Eq. 3 motivate the form of our proposed method. Intuitively, if 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* and we define the reward to be the difference between the scores of successive state pairs:

$$r(s_t) = \sigma_\psi(s_t) - \sigma_\psi(s_{t-1}). \tag{4}$$

Thus, the more uncertain an LLM's ranking is, the less informative the reward is. The potential in Eq. 4 is naturally shaped to a proper scale range, with positive rewards for actions 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 needs 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 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) less sensitive to scales of flat step penalties.

5.1 Algorithm

Our algorithm (Fig. 1) 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 (Wei et al., 2022) 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. 2. 4) Train a reinforcement learning agent with feedback from the state-score model.

6 Performance Analysis of Potential-Difference Rewards

We evaluate our approach in commonly-used discrete (grid world) and continuous (MuJoCo) (Brockman et al., 2016) 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.

6.1 Experiment Setup

Grid world. We examine three scenarios within grid world: **NoLock**, **Lock**, and **MultiLock**. The layouts are shown in Fig. 2. 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 correct key and use it on the door. The agent, goal, and key positions are randomly initialized in every episode.

MuJoCo. We examine a subset of robot control tasks selected from the simulated robotics benchmark MuJoCo (Todorov et al., 2012). 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 two baselines: 1) **Direct reward:** Following Christiano et al. (2017), the reward is normalized to zero-mean with a standard deviation of 1. 2) **Default reward:** 0 for failure, and $1 - \frac{0.99n}{n_{\max}}$ when the agent reaches the goal. $n_{\max}$ is the maximum time steps for each episode. n denotes the step count on success.

To train state-score models, grid world uses 2500, 3500, and 6000 state pairs 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 (Schulman et al., 2017) 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.

6.2 Single-Query Evaluation

We first 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 (Jiang et al., 2024), GPT 4 (Achiam et al., 2023), and Llama-3 with 8B and 70B parameters

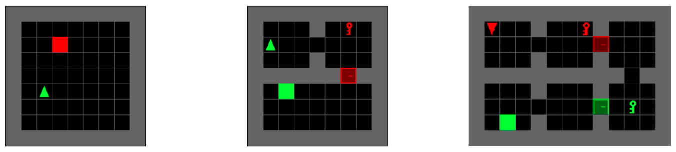


Figure 2: Grid world NoLock, Lock, MultiLock environments from left to right

(Touvron et al., 2023). For grid world environments, we add an additional baseline where rankings are generated with a ground truth heuristic function (GT) serving as an upper bound for our methods. Compared with GT, the ranking performance of the LLMs and state-score models trained with them are shown in Table 1 (Appendix B), where GPT-4 > Llama-3 70B > Mixtral > Llama-3 8B. The state-score models are employed to train 5 RL policies with random seeds and initializations.

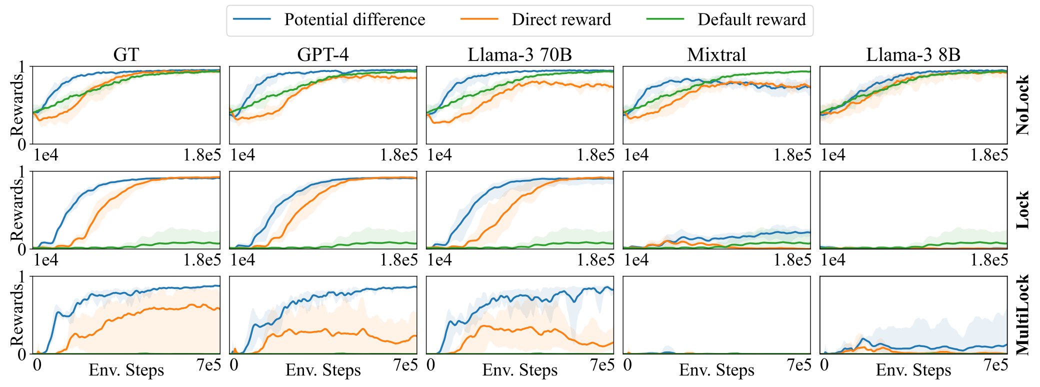


Figure 3: Learning curves for single-query rewards in the grid world environments

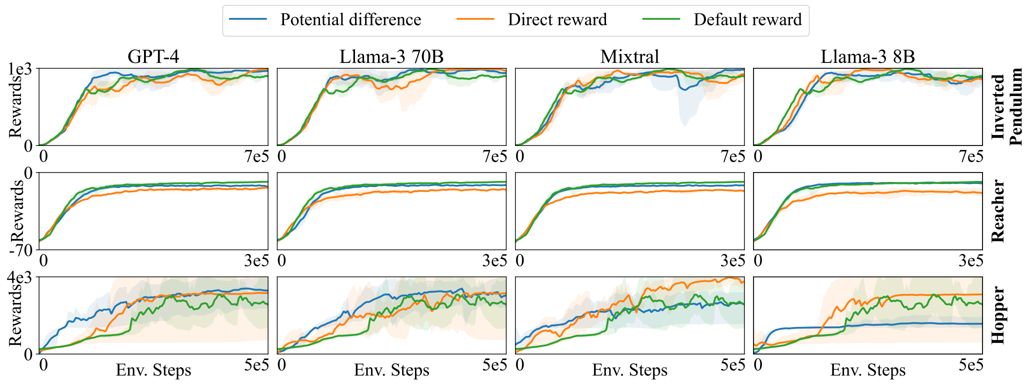


Figure 4: Learning curves for single-query rewards in the MuJoCo environments

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. (2016). The results, as well as the default reward performance, are shown in Fig. 3 and Fig. 4.

In grid world environments, when using GT, GPT-4, or Llama-3 70B rankings of high quality, our method converges the fastest and yields the highest final rewards compared with two baselines. When using LLMs which generate noisy outputs (i.e., Mixtral and Llama-3 8B), our method still outperforms the direct rewards, but all methods fail to converge in the Lock and MultiLock environments.

In Sec. 6.3, we detail the multi-query variation of our approach, particularly for low-performing LLMs, to regain training performance.

In MuJoCo environments, reported in Fig. 4, our potential-difference rewards slightly outperform (particularly in Hopper with GPT-4 and Llama-3 70B) or are on par with our baselines. Exceptions to this trend can be seen with low-performing LLMs (e.g., Llama-3 8b). Our method outperforms direct rewards 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, probably due to the challenge of designing appropriate prompts for low-performance LLMs with human intuition, i.e., comparing the hopper's speed in two consecutive states because the hopper should learn to move forward without falling down. While this prompt could lead to good rewards with high-performing LLMs, low-performing LLMs could not handle such rankings. We hypothesize that this leads to sub-optimal training results.

6.2.1 Hyper-Parameter Sensitivity Analysis

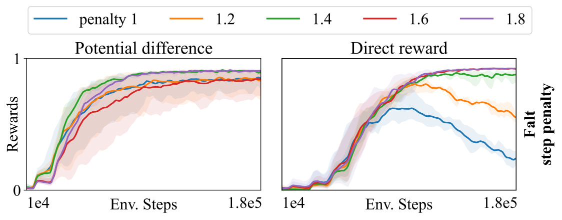


Figure 5: Learning curves of rewards with multiple flat step penalties/discounts in the Lock scenario.

Since potential-difference rewards and direct rewards 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. **Flat Step Penalty** is tested: A positive constant is subtracted at each time step. We use the state-score model trained from GT for comparison. Each parameter is tested by training 3 RL policies with random seeds and initialization. The results are shown in Fig. 5. Our method shows robustness to the choice of the flat step penalty, as the curves of penalty variances are less divergent. However, when using direct rewards, the performance is affected significantly with respect to the penalty, as many of them prevent the agent from converging.

6.3 Multi-Query Evaluation

This section demonstrates that the multi-query variation of potential-difference rewards can handle feedback noises and regain training performance given LLMs with considerable ranking noises.

6.3.1 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. 6. 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 increasing queries, the potential-difference rewards gradually improve the training performance. Two or more queries may achieve fast policy training converging towards the optimal if there are few ranking noises. Notably, even for the datasets of only 60%

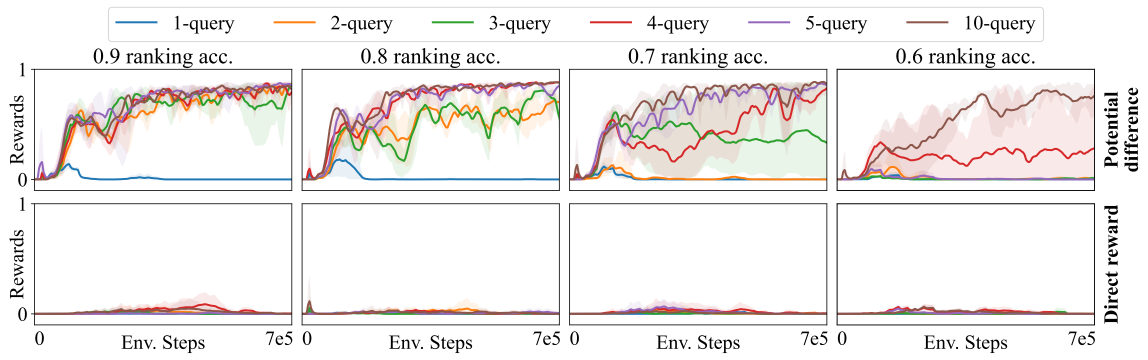


Figure 6: Muli-query experiments with synthetic datasets in the grid world - MultiLock scenario

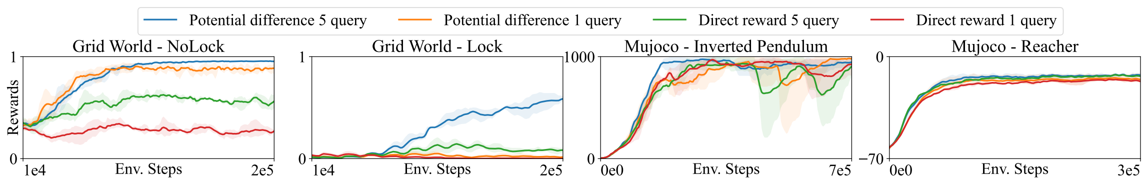


Figure 7: The performance comparison of 5-query variations of potential-difference rewards and direct rewards with 1800, 2200, 1000, 1000 state pairs ranked with Mixtral.

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. 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.

6.3.2 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 are compared in Fig. 7. As hypothesized, the 5-query potential-difference rewards achieve faster policy training and result in the highest training returns in all experiments. The single-query potential-difference rewards have the second-best performance, outperforming or being on par with both the single-query and 5-query direct rewards in most environments. The improvement is most significant in the Lock scenario.

7 Conclusions

In this work, we propose a simple method for incorporating noisy LLM feedback into reinforcement learning. Our approach is based on learning a potential-based score function over repeated LLM-generated preference rankings which issues uninformative rewards for states in which the LLM is uncertain. This results in a natural trade-off between reward sparsity and LLM confidence which we demonstrate both theoretically and empirically in a variety of discrete and continuous environments.

Acknowledgments

This work is sponsored by Honda Research 58629.1.1012949, AFOSR FA9550-18-1-0251 and DARPA FA8750-23-2-1015. The authors would also like to thank Dr. Woojun Kim for his assistance during discussion of this research.

References

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.

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.

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.

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.

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.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

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.

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.

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*, pp. 8657–8677. PMLR, 2023.

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.

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.

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.

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.

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

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.

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*, 2021a.

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

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.

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.

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.

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

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

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

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

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*, pp. 1072–1080. PMLR, 2024.

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, pp. 1000–1005. Boston, MA, 2006.

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*, pp. 5026–5033. IEEE, 2012.

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.

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.

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.

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.

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.

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.

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.

A Empirical Evidence that Ranking Inconsistency Pushes Potential-Difference Rewards to Zeros

In Fig. 8, 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. 8), 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, if 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. 8 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. 6.3.1).

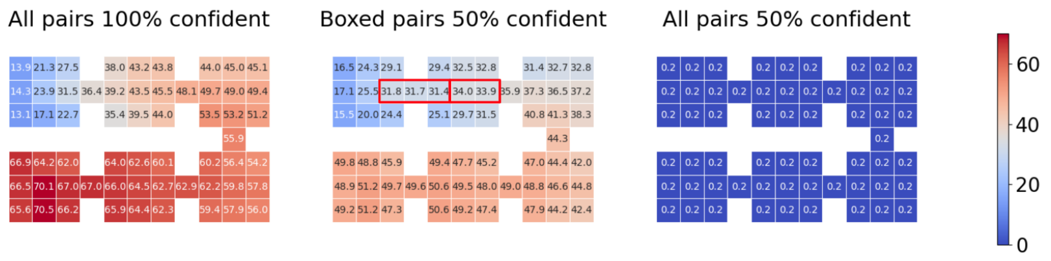


Figure 8: The heat maps showing that feedback inconsistency pushes rewards towards 0. Each grid in the map shows the score of the state where the agent is at this grid. The first heat map shows state scores trained with 100% confident rankings on all state pairs. The second heat map show state scores trained with 100% confident ranking on all state pairs except 50% confident rankings on state pairs in the red block. The third heat map shows state scores trained with 50% rankings on all state pairs.

B LLM Ranking Performance

Mthd.	Llama-3 8B	GT	Mixtral	Llama-3 70B	GPT4
NoLock(Rank)	1.0	0.69	0.76	0.93	1.0
NoLock(Score)	1.0	0.77	0.89	0.98	1.0
Lock(Rank)	1.0	0.54	0.65	0.89	0.98
Lock(Score)	1.0	0.55	0.74	0.97	0.98
MultiLock(Rank)	1.0	0.58	0.60	0.90	0.99
MultiLock(Score)	1.0	0.66	0.66	0.96	0.99

Table 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 is only accessible in grid world environments, we only show the ranking correctness of LLMs and state-score models in these three environments.

C Reward-Function Scale Analysis

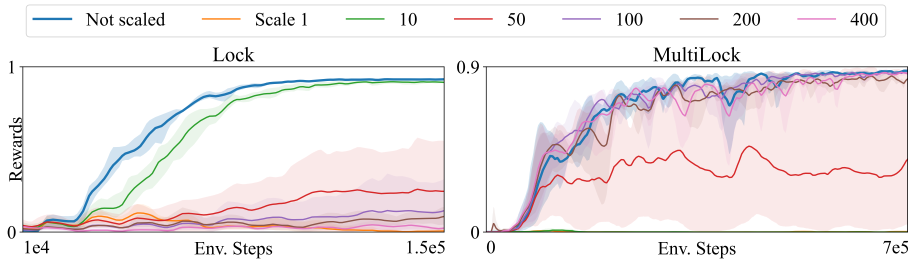


Figure 9: Policy training curves for state-score models with different scales using tanh.

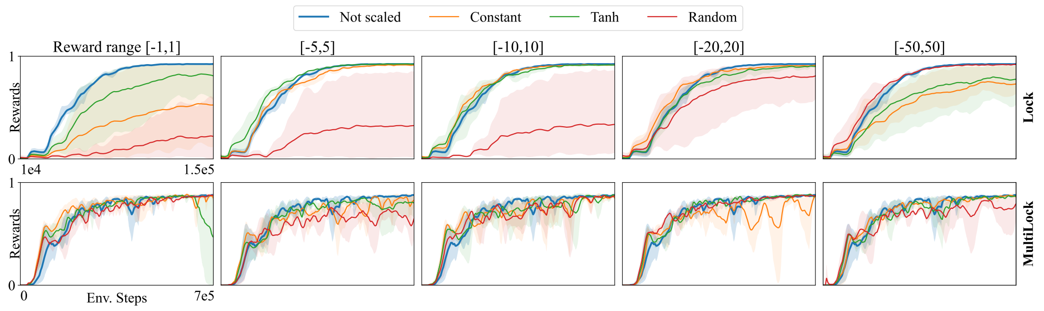


Figure 10: Policy training curves for rewards scaled to different ranges with different functions

The magnitude of the reward signal is crucial for shaping effective reward functions for RL agents' learning. Large reward signals can boost learning with strong incentives, but would lead to detrimental outcomes if they are not representative of the task's objective. While small reward signals could potentially offer more stability and smaller variance, they might lack sufficient impacts to drive the learning process. To further study the strengths and drawbacks of potential-difference rewards through scaling, we deploy two scaling methods: (1) insert an output layer of `tanh` with a scaling factor to the state-score model, and then compute the score difference as rewards, (2) post-process the potential-difference rewards without changing the state-score model. The experiments are conducted with models trained from Llama3 70B data in the Lock and MultiLock scenarios.

The result of the first method is shown in Fig. 9. For the Lock scenario, all 6 scaling factors performed worse than our potential-difference rewards without any scaling. While in the MultiLock scenario, large scaling factors can produce performance no better than potential-difference rewards. The possible reason is that the learned state scores are naturally optimized to represent the preferences over the state space. The scaling method actually exerts range constraints on the state-score model, making additional difficulty to learning.

The result of the second method is shown in Fig. 10, testing three scaling functions to filter potential-difference rewards. Given a scale range $(-s, s)$ with s in $\{1, 5, 10, 20, 50\}$, the scale-constant method maps potential-difference rewards to $\{-s, s\}$, the scale-tanh method maps to $(-s, s)$ by filtering potential-difference rewards with `tanh`, and the scale-random method filters potential-difference rewards with a random scaling function $f(r) : r \mapsto (-s, s)$. None of the post-processing methods tend to outperform the potential-difference rewards. Beyond this, the random scaling is shown to produce significantly worse average returns in 6 of the 10 experiment settings, while Scale-constant and Scale-tanh produce less bad results. These results emphasize the effectiveness of both the scale and shape of the potential-difference rewards.

D Example LLM Prompts

A typical LLM description prompt consists of three parts: environment and task description, an example based on the Chain of Thoughts, and the question. Take a prompt for the MultiLock scenario as an example.

Environment and Task Description

- Layout: The environment consists of an 11x7 grid divided into 6 chambers. Chamber1 occupies the top-left 3x3 section, Chamber2 the top-middle 3x3 section, Chamber3 the top-right 3x3 section, Chamber4 the bottom-right 3x3 section, Chamber5 the bottom-middle 3x3 section, and Chamber6 the bottom-left 3x3 section. A door at Chamber2 connects Chamber2 and Chamber3. Another door at Chamber4 connects Chamber4 and Chamber5. There is one Agent moving to a clinic in Chamber6 in this environment. The agent starts in Chamber1, and it must go to Chamber2 first, then to Chamber3 from Chamber2, then to Chamber4 from Chamber3, then to Chamber5 from Chamber4, and then go to Chamber6 from Chamber5. Every time the Agent can only move for one grid in one of four directions, up/down/left/right.

- Coordinate System: In this 11x7 grid, points are labeled (x, y), with (0, 0) at the bottom left and (11, 7) at the top right. Therefore, the coordinates span from (0,4) to (3,7) in Chamber1. The coordinates span from (4,4) to (7,7) in Chamber2. The coordinates span from (8,3) to (11,7) in Chamber3. The coordinates span from (7,0) to (11,3) in Chamber4. The coordinates span from (3,0) to (7,3) in Chamber5. The coordinates span from (0,0) to (3,3) in Chamber6.

Example

Q:
State[a]:
Agent: Chamber1 (2,4)
Passage to Chamber2: Chamber1 (3,5) right
Door at Chamber2 (7,5) to Chamber3: locked
Key in Chamber2: (5,6)
Passage down to Chamber4: Chamber4 (9,3)
Door at Chamber4 (7,1) to Chamber5: locked
Key in Chamber4: (10,2)
Passage to Chamber6: Chamber5 (3,1) left
Clinic: Chamber6 (1,1)
The agent does not carry any key. It needs a key.
Agent action: move up

Does the action taken by the Agent in State[a] help it progress toward the Clinic? Explain with Manhattan distance.

A: Let's think step by step.

Example

First, what action should the Agent take to progress toward the Clinic in State[a]? Given the Agent in Chamber1, to reach the Clinic in Chamber6, the Agent must first try entering Chamber2. To enter Chamber2 from Chamber1, the Agent must first pass the passage at Chamber1 (3,5).
Then, did the Agent do so? The mahanttan distance between the Agent and the passage is |2-3|+|4-5|=1+1, which is two.
The Agent takes an action to "move up", which means it will move from (2, 4) to (2, 5).
Which chamber is (2,5) in?
Still in Chamber1. But the mahanttan distance between the Agent and the passage becomes |2-3|+|5-5|=1+0, which is one, so the Agent is indeed one step closer to this passage.
Therefore, the action taken by the Agent in State[a] indeed helps it progress toward the Clinic. The answer is Yes.

Question

State[b]:
Agent: Chamber1 (0,6)
Passage to Chamber2: Chamber1 (3,5) right
Door at Chamber2 (7,5) to Chamber3: locked
Key in Chamber2: (5,6)
Passage down to Chamber4: Chamber4 (9,3)
Door at Chamber4 (7,1) to Chamber5: locked
Key in Chamber4: (10,2)
Passage to Chamber6: Chamber5 (3,1) left
Clinic: Chamber6 (1,1)
The agent does not carry any key. It needs a key.
Agent action: move down

Does the action taken by the Agent in State[b] help it progress toward the Clinic? Explain with Manhattan distance.