Reinforcement Learning from Human Text Feedback: Learning a Reward Model from Human Text Input

Belen Martin-Urcelay [1] **Andreas Krause** [2] **Giorgia Ramponi** [3]

Abstract

We explore the use of human-generated text inputs to model rewards in Reinforcement Learning with Human Feedback (RLHF). Human text contains rich and nuanced information, yet most previous work relies on preference feedback or restricts the text structure. We propose using Large Language Models (LLMs) as a way of harnessing the information from natural text to train a reward model efficiently. Our empirical evaluations demonstrate the advantages of this approach in both tabular and continuous reinforcement learning tasks. The results show that even with minimal human interactions, integrating text feedback with LLMs enables our method to approximate the reward function accurately, leading to significant performance improvements.

1. Introduction

Reinforcement Learning (RL) (Sutton & Barto, 2018) is a powerful framework for solving complex decision-making problems by training agents to maximize cumulative rewards through interactions with an environment. RL has achieved remarkable success in a variety of domains, from games (Mnih et al., 2015) and robotics (Kaufmann et al., 2023) to healthcare (Yu et al., 2021) and finance (Pendharkar, 2022). Central to the RL paradigm is the concept of a reward function, which provides the agent with feedback on its actions and guides its learning process.

However, defining an appropriate reward function in real-world applications is a significant challenge (Hadfield-Menell et al., 2017). In many cases, crafting a precise and comprehensive reward function that captures all aspects of desired behavior is difficult or impractical. Moreover, the reward function might need frequent adjustments as task requirements evolve, adding further complexity. This limitation hinders the deployment of RL in many practical scenarios where the specification of a reward function is ambiguous or subjective.

To address these challenges, preference-based Reinforcement Learning from Human Feedback (RLHF) (Christiano et al., 2017b) has emerged as a promising approach. Instead of relying solely on predefined reward functions, RLHF leverages human preferences to shape the learning process. By asking humans to compare and rank different trajectories, RLHF infers a reward function that aligns more closely with human values and intentions. This approach has shown success in areas where human judgment is critical for defining task performance.

In this paper, we propose a novel algorithm for reinforcement learning without explicit rewards. Our approach involves learning a reward function through human textual comments on the presented trajectories. Unlike traditional preference-based methods that rely on binary or ranked choices, our method utilizes natural language feedback to provide richer and more nuanced information about the trajectories. This allows the agent to understand the context and subtleties of human preferences more effectively. Our approach is based on the intuition that human text feedback contains much more information than pairwise comparisons. By harnessing this information through large language models (LLMs), we can train a more sophisticated and accurate reward model.

Our contributions are as follows:

- We introduce a new framework for reinforcement learning that derives a reward function from human textual feedback on trajectories.
- We develop an algorithm that integrates natural language processing with reinforcement learning to interpret and utilize human comments. We leverage the capabilities of LLMs to extract and encode the rich information contained in human textual feedback.
- We demonstrate the effectiveness of our approach

[1]Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, U.S.A. [2]Department of Computer Science, ETH Zurich, Switzerland [3]Department of Computer Science, University of Zurich, Switzerland. Correspondence to: Anonymus <Anonymus>.

Proceedings of the 41st ICML Workshop on Models of Human Feedback for AI Alignment, Vienna, Austria. PMLR 235, 2024.
Copyright 2024 by the author(s).

through experiments in various simulated environments, showing improved performance over traditional RL methods with predefined reward functions.

By harnessing human insights through textual comments, our method aims to bridge the gap between human preferences and machine learning, paving the way for more adaptable and human-aligned RL systems.

2. Related Work

2.1. Preference Based Reinforcement Learning

Traditional Reinforcement Learning (RL) relies on explicit reward functions to drive the learning process. When the reward function is not known or difficult to construct, one may collect human feedback to model the reward. In Preference Based Reinforcement Learning (PbRL), human oracles provide their preferences between pairs of trajectories or actions. These preferences are used to train a reward model, enabling the deployment of standard RL algorithms to find the optimal policy (Busa-Fekete et al., 2014; Christiano et al., 2017a).

2.2. Learning from Natural Human Feedback

A natural way for humans to interact and express their intentions is through text. Consequently, there is much interest in leveraging natural language in RL. One common strategy is to map natural language instructions to trajectories or features. To achieve this mapping, previous works limit the instructions to a finite set (Goyal et al., 2019; Bahdanau et al., 2019; Nguyen et al., 2021; Lin et al., 2022), or force a specific sentence structure, e.g., "Go to X" (Fu et al., 2019). These restrictions simplify the mapping process but also limit the flexibility of the language used.

Another approach, which allows for more general language, employs Neural Networks (NN) to map from natural language to rewards (MacGlashan et al., 2015; Tung et al., 2018; Narasimhan et al., 2018; Yang et al., 2021). These networks receive two inputs: a language description of the goal and the observed trajectory. The NNs are trained to compute how close a given trajectory is to the described goal. The output is used as a reward for standard RL algorithms. While this approach achieves good performance, it requires a large labeled dataset to train the NNs.

To avoid curating extensive datasets, recent research focuses on exploiting pre-trained models. Pre-trained Visual-Language Models (VLM), such as CLIP (Radford et al., 2021), encode images and text in a shared latent space such that semantically close instances are embedded close together. Leveraging this property, a scalar reward is computed by the dot product between the embeddings of the image observations and the text description of the goal (Mahmoudieh et al., 2022; Rocamonde et al., 2024). We may also compute a preference reward by prompting the VLM to describe two image observations, and then querying for the description most aligned with the goal (Wang et al., 2024). To obtain a good zero-shot performance, previous works finetune the VLM for each specific environment (Mahmoudieh et al., 2022), or they edit the environments to resemble photorealistic scenarios (Rocamonde et al., 2024). Unfortunately, current VLM struggle to distinguish subtle differences (e.g. standing with crossed arms vs. standing with arms on hips) or to generalize to novel environments (e.g. humanoid with round feet). These limitations may lead to failures in complex tasks (Rocamonde et al., 2024).

Another line of research focuses on pre-trained valence analyzers (Hutto & Gilbert, 2014) to translate text feedback into a sentiment score (Sumers et al., 2021). The scores drive the Bayesian updates describing the expected reward of each feature. At each iteration, the features are updated according to the sentiment score of the whole text. A single sentiment score may not capture all nuances in human text, e.g. "*the start is good, but the end is bad*". We believe a more fine-grained interpretation of the feedback would be beneficial.

Similarly, Kwon et al. (2023) leverage a pretrained Large Language Model (LLM) as a proxy for the reward signal. The authors prompt an LLM with a language description of the task together with an episode's outcome and ask whether the outcomes satisfy the objective. Complex episodes and goals may not be readily interpretable by current LLMs. Additionally, the binary nature of the feedback limits the information gained per query.

3. Preliminaries and Problem Setting

In this paper, we consider an *agent* who interacts with an *environment* aiming to maximize an expected reward. We describe the interactions between the agent and the environment as an episodic Markov Decision Process without reward function (MDP\R) (Puterman, 2014). Formally, an episodic MDP\R is a tuple $\mathcal{M} := (\mathcal{S}, \mathcal{A}, \mathbb{P}, T)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the set of actions that the agent can perform in the environment, $\mathbb{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ captures the transition probabilities, mapping state-action pairs to a probability distribution of the next state over $\mathcal{S}$, and T is the time horizon. The reward function, which maps action pairs to a reward $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, is unknown to the agent. Instead, the agent learns a reward model $\widehat{r} : \mathcal{S} \to \mathbb{R}$ based on human feedback.

At each step, the agent performs an action according to a deterministic *policy* $\pi : \mathcal{S} \to \mathcal{A}$. The goal is to learn the policy π^* that maximizes the expected return from the

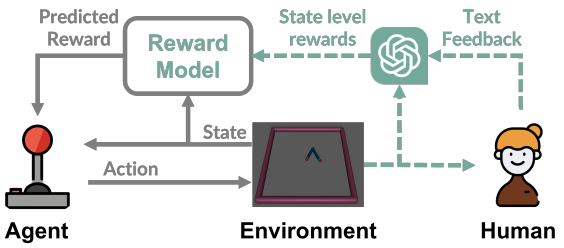


Figure 1. Block diagram of RLHF learning a reward model from human text input. We use an LLM to translate human evaluations in the form of natural language into state level rewards. These labeled states are used to train a reward model, which in turn, is employed with standard RL algorithms to train the agent.

current state $\mathbf{s}$,

$$V_r^t(\mathbf{s}) = \max_{\mathbf{a}\in\mathcal{S}} r(\mathbf{s},\mathbf{a}) + \sum_{\mathbf{s}'\in\mathcal{S}} \mathbb{P}(\mathbf{s}'|\mathbf{s},\mathbf{a}) V_r^{t-1}(\mathbf{s}'), \qquad (1)$$

where $V_r^0(\mathbf{s}) = \max_{\mathbf{a}\in\mathcal{S}} r(\mathbf{s},\mathbf{a})$ and $t \in [0; T-1]$ represents the number of timesteps left until the trajectory finishes. We denote the state-action pairs visited when following π^* as the optimal trajectory $\boldsymbol{\tau}^* = \{\mathbf{s}_t^*, \mathbf{a}_t^*\}_{t=0}^T$.

As the agent does not have direct access to the reward function r, the agent follows instead the policy $\widehat{\pi}$ that maximizes value function $V_{\widehat{r}}$ derived from the estimated reward. Our primary goal is to minimize the performance gap between this learned policy $\widehat{\pi}$ and optimal policy π^*. We denote this difference in expected rewards as *imitation gap*, which we formally define as

$$\mathbb{E}\left[\sum_{t=0}^{T} r\left(\mathbf{s}_t^*, \mathbf{a}_t^*\right) - r\left(\widehat{\mathbf{s}}_t, \widehat{\mathbf{a}}_t\right)\right],$$

where $\widehat{\boldsymbol{\tau}} = \{\widehat{\mathbf{s}}_t, \widehat{\mathbf{a}}_t\}_{t=0}^T$ represents the trajectory when following $\widehat{\pi}$.

4. Algorithm

The proposed algorithm consists of two phases, which are executed iteratively. First, we leverage human feedback to train a reward model. Second, we train an agent to perform a policy that maximizes the reward as estimated by the model. The algorithm, which we call Reinforcement Learning from Human Text Feedback (RLHTF), is outlined in Algorithm 1. Next, we describe each phase in detail.

4.1. Learning a Reward Model

Our key insight is that human text feedback contains much more information than pairwise comparisons, and this information may be harnessed through LLMs to train a reward model.

The dashed lines in Figure 1 depict how the reward model is trained. At each interaction, the human evaluator observes and evaluates the trajectory given by the current policy. For example, the human evaluation may include criticisms of specific states, or suggestions for alternative, unexplored states.

The human evaluation, together with information about the environment and the trajectory, is fed into an LLM. The prompt is further described in Appendix B. The LLM then processes this information and identifies which states the human feedback refers to, and with what connotation. Namely, the LLM generates a set of states with their corresponding labels (positive or negative). This set is used as a training dataset to train the reward model.

In the tabular setting, the agent tracks the reward probability distribution for every state, which is modeled as a beta distribution. Specifically, we initialize the reward distributions as $\beta(0.5, 0.5)$ to introduce a bias towards binary rewards: 0 (negative) or 1 (positive). For analytical tractability, we model the observed labels with the conjugate prior, i.e., as Bernoulli distributions. This way, the posterior of a given state probability distribution $\beta(a, b)$ simply becomes $\beta(a, b+1)$ when receiving a positive label, or $\beta(a+1, b)$ if the label is negative.

To extend RLHTF beyond the tabular setting, we approximate the reward function as a NN. At each iteration, we expand the training dataset with the state-reward pairs outputted by the LLM. Then, we employ Stochastic Gradient Descent (SGD) to finetune the NN with the expanded training dataset. This supervised learning process utilizes a cross-entropy loss function.

4.2. Learning a Policy

The reward model learns to measure how close the trajectory is to the human intentions. Consequently, to find the optimal policy the agent may directly query the reward model, instead of the human evaluators. This approach significantly reduces time, energy, and monetary costs during policy learning. The interactions are described by the solid arrows in Figure 1 and line 8 of Algorithm 1.

In the tabular setting, we employ Q-learning (Watkins & Dayan, 1992) to guide the agent. Specifically, the agent uses a greedy policy, selecting actions that maximize the expected future rewards as in (1). In contrast, for the continuous setting, we model the policy with a NN. We train the network using the REINFORCE algorithm (Williams, 1992), where the reward signal is given by the reward model. Following standard practices from Reinforcement Learning from Human Feedback (RLHF), we compute several REINFORCE epochs before querying the human evaluators and updating the reward model again.

Algorithm 1 RLHTF

Input: number of human iterations N
Initialize Policy π_0 and reward model $\widehat{r}_0$.
for $i = 0$ **to** N **do**
  Record trajectory following policy: $\mathbf{t}_i \leftarrow \pi_i$
  Query human for feedback: $\mathbf{f}_i \leftarrow \mathbf{t}_i$
  Translate feedback to state-reward pairs with LLM: $\mathbf{s}_i, \mathbf{r}_i \leftarrow \mathbf{f}_i, \mathbf{t}_i$
  Update reward model: $\widehat{r}_{i+1} \leftarrow \widehat{r}_i, \{\mathbf{s}_t, \mathbf{r}_t\}_{t=0}^{i}$
  Update policy: $\pi_{i+1} \leftarrow \pi_i, \widehat{r}_{i+1}$
end for

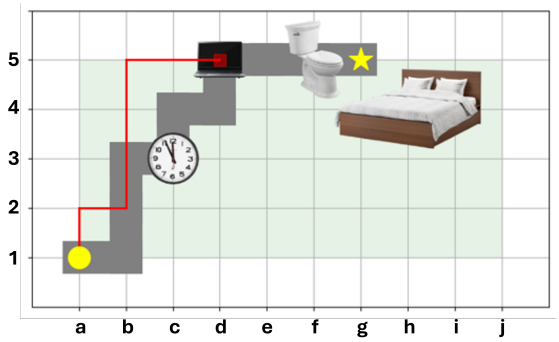
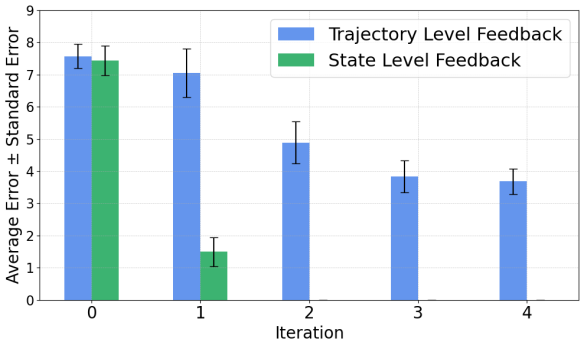
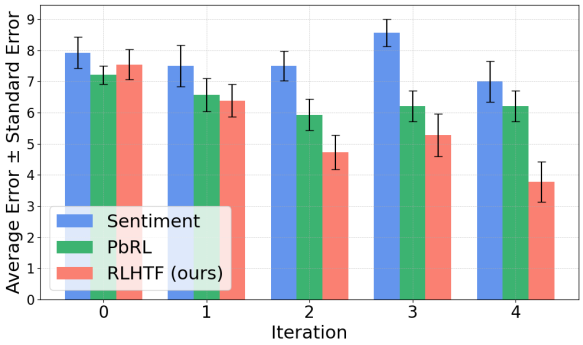


Figure 2. Gridworld environment. The aim of the agent is to follow the a specific path (in grey) from the start (yellow circle) to the end (yellow star). The agent's trajectory (in red) currently deviates from this path.



(a) True feedback from environment.



(b) Human feedback.

Figure 3. Performance comparison of algorithms in the Gridworld.

5. Experiments

We empirically evaluate RLHTF against three RL baselines:

- **Sentiment**: We measure the sentiment of the human text feedback, and apply it as a reward for all states in the trajectory (Sumers et al., 2021).
- **PbRL**: We query human evaluators for pairwise comparisons between trajectories (Christiano et al., 2017a).
- **True**: The agent receives the true reward from the environment.

The details of the implementation may be found in Appendix A.

5.1. Gridworld

We first test our approach in a tabular setting. Particularly we apply the algorithm to the Gridworld shown in Figure 2. An agent aims to follow a specific path, which is known to the human evaluators but unknown to the agent. Human evaluators observe the agent's trajectory and provide feedback to guide the agent. To provide context, we randomly include landmarks in the environment, e.g.: a clock or a bed. We use YOLOv8 (Jocher et al., 2022) to detect these landmarks and their locations. This enables the LLM to translate feedback such as "*The behavior to the left of the PC is bad*" in Figure 2 to a negative reward at position 'c5'.

Figure 3 shows the error evolution as the agent interacts with the environment, where the error is defined as the number of steps in which the agent deviates from the ideal path. Figure 3a show the performance when the agent receives the true feedback from the environment. We consider two scenarios: under the first scenario, the agent receives the accumulated reward for the entire trajectory, this is used to equally update all the states visited; under the second scenario, the environment provides state-level rewards, explicitly indicating whether each of the observed states are part of the ideal trajectory or not. We observe, that the agent rapidly improves its performance under both scenarios, but the state-level feedback is significantly more effective. The agent with state-level feedback identifies the correct path after only two iterations. This highlights the advantage of having fine-grained feedback in reinforcement learning, where detailed information about each state's contribution leads to faster learning.

A main advantage of RLHTF is the algorithm leverages LLM to perform reward attribution. In contrast, previous

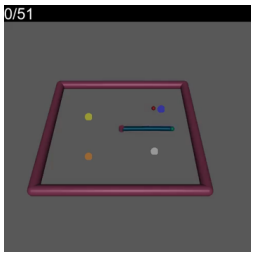


Figure 4. Modified reacher environment. The aim of the robotic arm is to stay close to the red ball. We add circles and a timestamp to aid human evaluation of specific states.

works update all states in a trajectory (Sumers et al., 2021) or all differing states between two trajectories (Christiano et al., 2017a) with a single reward. Figure 3b compares the performance of RLHTF against these other strategies for learning from human feedback. We observe that breaking down the feedback into fine-grained rewards with an LLM greatly benefits the performance. The error decreases faster with RLHTF than the state of the art *Sentiment* or *PbRL* algorithms. After only receiving 4 instances of human feedback, RLHTF on average reduces the error by half.

5.2. Reacher Environment

We also test RLHTF in a continuous environment of the physics simulator MuJoCo (Todorov et al., 2012). Namely, we apply RLHTF to *Reacher*, a two-jointed robot arm. The aim is to apply appropriate torques to the hinges so that the robot's fingertip reaches a target. The human evaluators watch and criticise a video of the robot arm moving under the current policy. To aid human evaluators in their assessment, we incorporate a timestamp and some visual landmarks (colored circles) into the environment. These aids the human evaluator to refer to specific locations (*"Go to the left of the blue circle."*) or moments in the trajectory (*"The movement from frame 4 to 8 is wrong."*). Figure 4 shows the modified reacher environment.

The standard observation space contains information about the goal (target location) and the performance (distance to target). We remove this information before inputting the observation to the LLM, as the performance should be solely deduced from the human feedback. This filtering is further detailed in Appendix A.2.

The LLM processes three inputs: The human feedback, the landmark locations and the sequence of filtered states making up the video. The LLM's task is to interpret these inputs to deduce what states are described as positive or negative. Figure 5 shows how the state-reward pairs are generated. The LLM outputs pairs of filtered states and binary labels. Each label indicates whether the corresponding state is positive or negative. If a filtered state is returned partially, missing elements are filled in by randomly sampling from the set of observed trajectory states.

We suggest that feedback in natural language is more informative than the traditional approach of selecting the preference among two trajectories, and thus RLHTF requires less interactions with humans to achieve an accurate reward model. To verify the hypothesis, we compare the evolution of the reward model with RLHTF versus PbRL. Figure 6 shows the reward predictions for different positions of the arm, with darker colors indicating lower predicted rewards. The blue star marks the target location. On the left, Figure 6a displays the reward model at random initialization. Figures 6b, 6c and 6d at the top show the reward model evolution as more text feedback is gathered. Whereas Figures 6e, 6f and 6g on the bottom show the reward model evolution with pairwise comparison feedback. After just one interaction, both feedback types significantly enhance the reward model. However, text feedback offers a more precise model. By the tenth interaction, Figure 6d shows that only a small area around the target receives a high reward with RLHTF, whereas Figure 6g shows that PbRL results in a reward model more uncertain about the target location, giving high reward to large areas of the environment. In conclusion, Figure 6 suggests that natural language feedback enables the reward model to more quickly and accurately identify the target.

We also compare the agent's performance, in terms of average distance to the target, when there is a budget of 10 human interactions. Figure 7 shows how the reward evolves as the agent learns. Although RLHTF is not as effective as directly observing the true reward, RLHTF performs much better in regimes with low feedback than PbRL. In fact, in our experiments RLHTF increases the reward by 40% with only 10 human inputs, while the reward even decreases with PbRL when the agent only has access to 10 comparison.

6. Discussion

Our work leverages LLMs to extract state level rewards from human feedback. This approach tackles the challenge of reward attribution and garners richer information from human interactions beyond mere binary comparisons. However, we study just one way of interactivity with humans, in which the human evaluates the current agent's trajectory, in future work we could prompt humans in different manners, for instance, asking humans to directly describe the goal (Li et al., 2023). Moreover, in the current implementation of RLHTF, the human only observes a greedy realization of the agent's trajectory. As a next step, we will incorporate strategies balancing exploration and exploitation, such as the Upper Confidence Bound (UCB) algorithm (Lai & Robbins, 1985).

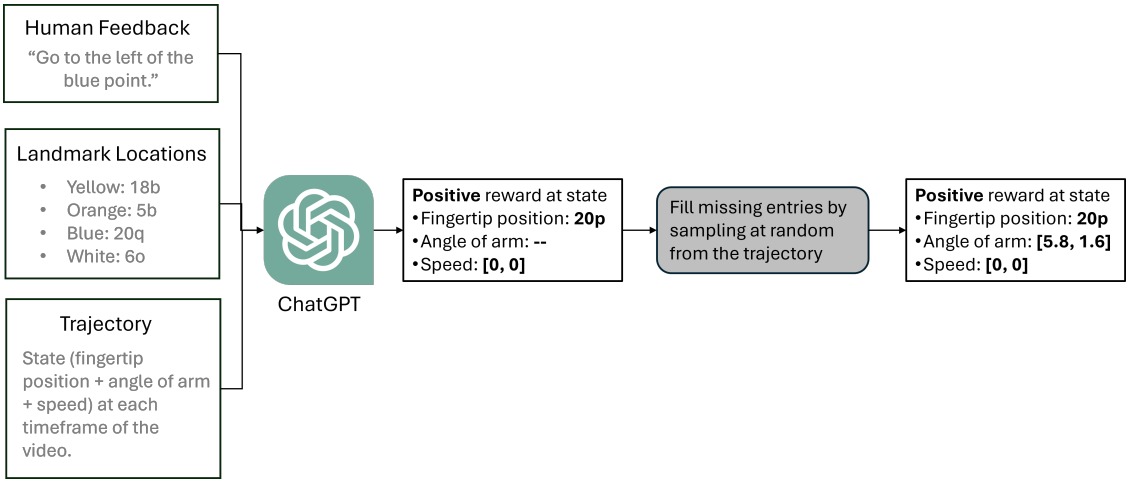
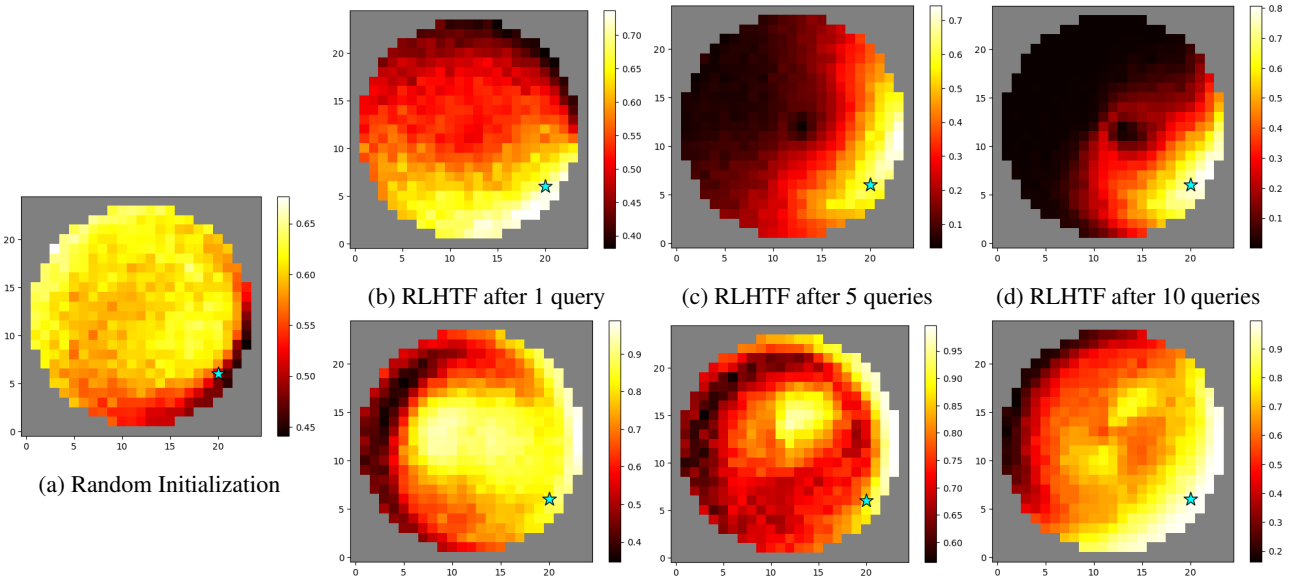


Figure 5. Example of Step 6 in Algorithm 1.



(a) Random Initialization (b) RLHTF after 1 query (c) RLHTF after 5 queries (d) RLHTF after 10 queries

(e) PbRL after 1 query (f) PbRL after 5 queries (g) PbRL after 10 queries

Figure 6. Reward model visualization for RLHTF vs PbRL. The top row illustrates the evolution of the reward estimations as more text feedback is gathered, while the bottom row shows the corresponding evolution with preference feedback. Darker colors indicate lower predicted rewards and the blue star marks the target location. Notably, RLHTF quickly converges to a more accurate reward model with fewer interactions than PbRL, as evidenced by the more localized high-reward region around the target.

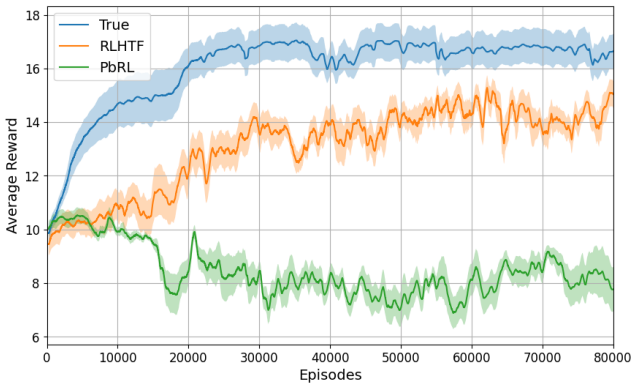


Figure 7. Reward evolution with episodes of the REINFORCE algorithm. We compare the performance for RLHTF with 10 text feedback, PbRL with 10 trajectory comparisons, and true environment reward.

Our preliminary experiments in Section 5 show the potential of our approach. Nonetheless, the scope of these experiments remains limited. Moving forward, we will apply RLHTF to other domains and we will perform extensive experiments with a wider range of human evaluators.

By addressing these aspects, we aim to enhance the robustness, adaptability, and practical utility of our approach in integrating human text with reinforcement learning.

Acknowledgements

We extend our sincere gratitude to the reviewers for their valuable feedback and insights. This work was supported by the Rafael del Pino Foundation. We also acknowledge the contributions of the Learning and Adaptive Systems Group.

References

Bahdanau, D., Hosseini, A., Hill, F., Kohli, P., Leike, J., Hughes, E., and Grefenstette, E. Learning to understand goal specifications by modelling reward. In *Proc. of International Conference on Learning Representations*, 2019.

Busa-Fekete, R., Szörényi, B., Weng, P., Cheng, W., and Hüllermeier, E. Preference-based reinforcement learning: Evolutionary direct policy search using a preference-based racing algorithm. In *Machine Learning*, volume 97, pp. 327–351. Kluwer Academic Publishers, 10 2014. doi: 10.1007/s10994-014-5458-8.

Christiano, P. F., Leike, J., Brown, T. B., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. In *Proc. of Advances in Neural Information Processing Systems*, volume 2017-Decem, pp. 4300–4308, 2017a.

Christiano, P. F., Leike, J., Brown, T. B., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pp. 4299–4307, 2017b.

Fu, J., Korattikara, A., Levine, S., and Guadarrama, S. From language to goals: Inverse reinforcement learning for vision-based instruction following. In *Proc. of International Conference on Learning Representations*, pp. 1–14, 2019.

Goyal, P., Niekum, S., and Mooney, R. J. Using natural language for reward shaping in reinforcement learning. In *Proc. of International Joint Conference on Artificial Intelligence*, volume 2019-Augus, pp. 2385–2391, 2019. ISBN 9780999241141. doi: 10.24963/ijcai.2019/331.

Hadfield-Menell, D., Russell, S. J., Abbeel, P., and Dragan, A. D. Inverse reward design. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 6765–6774, 2017.

Hutto, C. J. and Gilbert, E. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Prof. of Conference on Artificial Intelligence*, pp. 216–225, 2014. URL http://sentic.net/.

Jocher, G., Chaurasia, A., Stoken, A., Borovec, J., NanoCode012, Kwon, Y., Michael, K., TaoXie, Fang, J., imyhxy, Lorna, Yifu, Z., Wong, C., V, A., Montes, D., Wang, Z., Fati, C., Nadar, J., Laughing, UnglvKitDe, Sonck, V., tkianai, yxNONG, Skalski, P., Hogan, A., Nair, D., Strobel, M., and Jain, M. ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation, November 2022. URL https://doi.org/10.5281/zenodo.7347926.

Kaufmann, E., Bauersfeld, L., Loquercio, A., Müller, M., Koltun, V., and Scaramuzza, D. Champion-level drone racing using deep reinforcement learning. *Nature*, 620 (7976):982–987, 2023.

Kwon, M., Xie, S. M., Bullard, K., and Sadigh, D. Reward design with language models. In *ICLR 2023*, 2023. URL http://arxiv.org/abs/2303.00001.

Lai, T. L. and Robbins, H. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985. ISSN 10902074. doi: 10.1016/0196-8858(85)90002-8.

Li, B. Z., Tamkin, A., Goodman, N., and Andreas, J. Eliciting Human Preferences with Language Models. 2023. URL http://arxiv.org/abs/2310.11589.

Lin, J., Fried, D., Klein, D., and Dragan, A. Inferring rewards from language in context. In *Proc. of the Association for Computational Linguistics*, volume 1,

pp. 8546–8560, 2022. ISBN 9781955917216. doi: 10.18653/v1/2022.acl-long.585.

MacGlashan, J., Babeş-Vroman, M., DesJardins, M., Littman, M. L., Muresan, S., Squire, S., Tellex, S., Arumugam, D., and Yang, L. Grounding english commands to reward functions. *Robotics: Science and Systems*, 11, 2015. ISSN 2330765X. doi: 10.15607/RSS.2015.XI.018.

Mahmoudieh, P., Pathak, D., and Darrell, T. Zero-shot reward specification via grounded natural language. *Proc. of Machine Learning Research*, 162:14743–14752, 2022. ISSN 26403498.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Narasimhan, K., Barzilay, R., and Jaakkola, T. Grounding language for transfer in deep reinforcement learning. *Artificial Intelligence Research*, 63:849–874, 2018. ISSN 10769757. doi: 10.1613/jair.1.11263.

Nguyen, K., Misra, D., Schapire, R., Dudík, M., and Shafto, P. Interactive learning from activity description. *Proceedings of Machine Learning Research*, 139:8096–8108, 2021. ISSN 26403498.

Pendharkar, P. C. Reinforcement learning in financial market applications: a survey and research agenda. *Artificial Intelligence Review*, 55(2):1247–1306, 2022.

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision. In *Proc. of International Conference on Machine Learning*, 2 2021. URL http://arxiv.org/abs/2103.00020.

Rocamonde, J., Montesinos, V., Nava, E., Perez, E., and Lindner, D. Vision-language models are zero-shot reward models for reinforcement learning. In *Proc. of International Conference on Learning Representations*, 10 2024. URL http://arxiv.org/abs/2310.12921.

Sumers, T. R., Ho, M. K., Hawkins, R. D., Narasimhan, K., and Griffiths, T. L. Learning rewards from linguistic feedback. In *Proc, of Conference on Artificial Intelligence*, volume 7, pp. 6002–6010, 2021. ISBN 9781713835974. doi: 10.1609/aaai.v35i7.16749.

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

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *Proc. of International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Tung, H. Y., Harley, A. W., Huang, L. K., and Fragkiadaki, K. Reward learning from narrated demonstrations. In *Proc. of Conference on Computer Vision and Pattern Recognition*, pp. 7004–7013, 2018. ISBN 9781538664209. doi: 10.1109/CVPR.2018.00732.

Wang, Y., Sun, Z., Zhang, J., Xian, Z., Biyik, E., Held, D., and Erickson, Z. RL-VLM-F: Reinforcement Learning from Vision Language Foundation Model Feedback. In *Proc. of International Conference on Machine Learning*, 2024. URL http://arxiv.org/abs/2402.03681https://rlvlmf2024.github.io/.

Watkins, C. J. C. H. and Dayan, P. Q-learning. *Machine Learning*, 8(3-4):279–292, may 1992. ISSN 0885-6125. doi: 10.1007/BF00992698.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. In *Proc. of Advances in Neural Information Processing Systems*, 1 2022. URL http://arxiv.org/abs/2201.11903.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992. ISSN 0885-6125. doi: 10.1007/bf00992696.

Yang, T. Y., Hu, M., Chow, Y., Ramadge, P. J., and Narasimhan, K. Safe reinforcement learning with natural language constraints. In *Proc. of Advances in Neural Information Processing Systems*, volume 17, pp. 13794–13808, 2021. ISBN 9781713845393.

Yu, C., Liu, J., Nemati, S., and Gregg, W. M. Reinforcement learning in healthcare: A survey. *arXiv preprint arXiv:1908.08796*, 2021.

Default Observation	
cosine of the angle of the first arm	1.3184
cosine of the angle of the second arm	1.3184
sine of the angle of the first arm	0.9996
sine of the angle of the second arm	0.9995
x-coordinate of the target	-0.0288
y-coordinate of the target	-0.0307
angular velocity of the first arm	-0.1320
angular velocity of the second arm	0.1214
x-value of $\text{position}_{\text{fingertip}}$ - $\text{position}_{\text{target}}$	3.9565
y-value of $\text{position}_{\text{fingertip}}$ - $\text{position}_{\text{target}}$	3.9607

Filtered State	
Angles of arms in degrees	[5.8, 11.6]
Fingertip position in algebraic notation	x15
Angular speed in rads/sec	[0.62, 3.73]

Figure 8. Preprocessing of state observation.

A. Experiment Details

A.1. Gridworld

The agent takes 10 steps. At each timestep the possible actions are either to move right or up. The agents movement are restricted to a 5x10 grid. When the agent is in a border and performs an action that would take the agent outside of the allowed region, the agent doesn't move.

We randomly add landmarks to the gridworld to aid with the evaluation. Then, the landmarks are detected with YOLOv8 model trained on the Microsoft COCO dataset, and its positions are fed to the LLM for context. Namely, we use *gpt-4-1106-preview* to translate human feedback into state-level reward.

We conducted the experiments with the assitance of volunteer Ph.D. students who acted as evaluators. We tested the Sentiment and PbRL algorithms for 14 trials each and RLHTF for 18 trials.

A.2. Reacher Environment

Each trajectory in the reacher environment spans $T = 50$ frames. The default observations for each frame include the target location and the distance between the arm and target. However, to proof that the LLM is capable of deducing the performance solely from the human feedback, we filter this information from our prompt. The LLM obtains a filtered state that includes the fingertip's location, arm's joint angles, and angular velocities. The filtering and preprocessing of the observation are detailed in Figure 8.

We use the LLM *gpt-4o* to translate the human feedback to state-reward pairs. To conduct a fair comparison with PbRL, we reconstruct the filtered states output by the LLM back into the complete original observation space. Using this full observations with their corresponding binary labels, we train a reward model in a supervised learning framework. The architecture of this reward model is a neural network (NN) with one hidden layer consisting of 32 nodes, employing a RELU activation function characterized by a leaky parameter $\alpha = 0.01$. In the case of PbRL, three such NNs are initialized at random to create an ensemble. Before asking the human for a preference, we sample random trajectories and then chose the pair whose preference is most uncertainty, specifically, the pair for which there is the most disagreement among the ensemble's predictive outcomes. During policy training, the agent observes the average reward from the ensemble.

We perform the experiments four times for every algorithm under consideration. A different target location was set for each experiment. The human feedback was provided by one of the authors of this paper. The shaded area in Figure 6 shows the standard error between the four experiments. Note that the plotted lines have been smoothed using a convolution operation with a window size of 500 episodes, which helps to reduce noise and provide a clearer trend of the data.

B. Prompt Engineering

There are four key elements in the system instructions which lead to a good performance. First, we define the coordinates in the grid as *chess coordinates*, i.e., as a letter and number pair indicating the column and row respectively. Second, we follow

An agent is trying to learn and follow a specific path in a {grid_height}x{grid_width} grid map.

Your job is to translate the feedback of the current trajectory into feedback types, locations in the map, and a label.

- If the feedback type is imperative, compute what locations in the map the instructions are referring to and label them as either 'good' (go to) or 'bad' (avoid).
- If the feedback type is evaluative, determine what locations in the map are being referred to by the feedback and whether the feedback is positive or negative.
- If the feedback type is descriptive, compute what new locations the agent should have visited, and label them as positive.

CoT

Use the getReward function to only return a JSON file with the specified shape enclosed in double quotes.

Output format

For example: if the user's input is {example_feedback}, then the output should be {example_output}.

Another example: if the user's input is {example_feedback2}, then the output should be {example_output2}

Few Shot

Figure 9. System prompt for Gridworld.

chain of thought (CoT) prompting (Wei et al., 2022). As feedback is context dependent, we ask the LLM to categorize the feedback into imperative, evaluative or descriptive as an intermediate step. Third, in accordance with few shot prompting, we provide demonstrations to steer the model to better performance. Lastly, we use function calling to force ChatGPT's output to have a prespecified json format. The desired output format for the Gridworld and Reacher enviroments are described in Figures 11 and 14 respectively. While the full system instructions are detailed in Figures 9, 10, 12 and 13.

```
grid_height = 5
grid_width = 10
max_col_letter = ‘j’

example_feedback = '''
        {"feedback": "it should not go below the bed",
        "landmarks": {"clock": ["2f"], "bed": ["2c", "2d"]},
        "trajectory": ["1b", "1c", "1d", "1e", "1f", "1g", "2g", "3g", "4g", "4h"]}
        '''
example_output = {'locations': ['1c', '1d'], 'label': 'NEG', 'feedback_type': 'imperative'}


example_feedback2 = '''
        {"feedback": "the last couple steps are good",
        "landmarks": {"clock": ["2f"], "bed": ["2c", "2d"]},
        "trajectory": ["1b", "1c", "1d", "1e", "1f", "1g", "2g", "3g", "4g", "4h"]}
        '''
example_output2 = {'locations': ['4g', '4h'], 'label': 'POS', 'feedback_type': 'evaluative'}
```

Figure 10. Parameters for system prompt in Gridworld.

```
"name": "getReward",
"parameters": {
        "type": "object",
        "properties": {
          "locations": {
            "type": "array",
            "items": {
              "type": "string",
              "pattern": [1-{grid_height}][a-{max_col_letter}]
            },
            "description": ("Locations in the grid referring to feedback with row numbers and a lowercase letter "
                    "for columns. The rows are numbered from bottom to top (1 is the lowest row, increasing "
                    "as you move upward), and columns are labeled from left to right (a-j). For example, '1b' "
                    "refers to the lowest row in the second column from the left.")
          },
          "label": {
            "type": "string",
            "enum": ["POS", "NEG"],
            "description": "The feedback's connotation, positive or negative."
          },
          "feedback_type": {
            "type": "string",
            "enum": ["imperative", "evaluative", "descriptive"],
            "description": """
            Imperative: Feedback includes instructions on what locations are good or should be avoided.
            Evaluative: Feedback is an assessment of the current trajectory.
            Descriptive: Feedback is about modifications for an improved trajectory.
            """
          }
        },
        "required": ["locations", "label"]
    }
  }
```

Figure 11. Function calling to force output format in Gridworld.

You will return state-reward pairs for a two-joined robotic arm named 'Reacher-v4'.

The arm aims to reach targets with its end effector (fingertip), and you must assess what states are good or bad based on human observers' feedback.

As input, you will receive:

1. The natural language comment by a human observer who has seen the simulation.

2. The location of landmarks which are circles of different colors.

3. Trajectory of a simulation of the robot trying to reach a target. Each timestep is described by:

    a) The fingertip position - letter representing column (left 'a' to right 'z') and number representing row (down '0' to up '26')

    b) A 2 item list of angles in degrees corresponding to the first and second joint respectively.

    c) A value of 0 in the second joint means the arm is fully bent, while it is completely straight when it is -180 or 180.

    d) A 2 item list with the angular speed on the first and second joint respectively.

For each set of observer comments, use the provided trajectory and landmarks to determine successful and unsuccessful states.

Follow this steps

1. Classify each section (sentence or linked group of sentences) in the feedback text as:

  a) Goal description: It describes where the target is, or where the fingertip should go (e.g.: You should go to the pink dot).

  b) Trajectory feedback: It criticizes the simulation observed (e.g.: The first two steps are wrong).

  c) Trajectory suggestion: It describes ways to improve upon states in the simulation observed (e.g.: Go a bit to the right of the state at time 23).

2. Generate state reward pairs depending on the feedback type

  a) For Goal description: Provide a "reward": +1, "angular_speed": [0, 0] at the location of the described target position.

  b) For Trajectory feedback:

    2.1. Determine whether the feedback has a positive ("reward": +1) or a bad ("reward": -1) connotation.

    2.2. Determine what state or states of the simulation it is referring to, and get the fingertip position, angles and angular speed of those locations

  c) For Trajectory suggestion:

    2.1. Determine what state or states of the simulation it is referring to, and get the fingertip position, angles and angular speed of those locations

    2.2. Correct the states as suggested by the feedback and pair with a "reward": +1

Use the getReward function to only return a JSON file with the specified shape.

CoT

Output format

Figure 12. System prompt for Reacher environment Part 1.

Few Shot

Example inputs with expected outputs are provided below for guidance:

Example 1:

Input:

\{'feedback': 'The last half is very bad, it should go a bit higher than the blue dot',

'landmarks': \{'yellow': 'b3', 'blue': 'l6', 'white': 'm7', 'orange': 'c23'\},

'fingertip_position': ['k15', 'k14', 'k14', 'k14', 'k13', 'l12'],

'angle': [[30.7, -63.9], [33.2, -68.2], [35.7, -72.6], [38.3, -76.9], [40.8, -81.3], [43.2, -85.7]],

'angular_speed': [[0.1, -0.06], [0.12, -0.12], [0.13, -0.17], [0.15, -0.23], [0.16, -0.29], [0.18, -0.34]] \}

Expected Output:

\{"referred_steps": [

\{"fingertip_position": 'k14', "angle": [38.3, -76.9], "angular_speed": [0.15, -0.23], "reward": -1\},

\{"fingertip_position": 'k13', "angle": [40.8, -81.3], "angular_speed": [0.16, -0.29], "reward": -1\},

\{"fingertip_position": 'l12', "angle": [43.2, -85.7], "angular_speed": [0.18, -0.34], "reward": -1\},

\{"fingertip_position": 'l7', "angular_speed": [0, 0], "reward": 1\},

\{"fingertip_position": 'l8', "angular_speed": [0, 0], "reward": 1\}]\}

Example 2:

Input:

\{'feedback': 'The fifth step but slower is good. Stop at the last point. The goal is to go to the white point.',

'landmarks': \{'yellow': 'u8', 'purple': 'k12', 'white': 'w16'\},

'fingertip_position': ['k15', 'k14', 'k14', 'k14', 'k13', 'l12'],

'angle': [[12.2, 34.2], [11.3, 39.6], [10.0, 45.2], [8.2, 50.9], [5.9, 56.6], [3.1, 62.3],],

'angular_speed': [[0.1, -0.06], [0.12, -0.12], [0.13, -0.17], [0.15, -0.23], [0.16, -0.29], [0.18, -0.34]] \}

Expected Output:

\{"referred_steps": [

\{"fingertip_position": 'k13', "angle": [5.9, 56.6], "angular_speed": [0.08, -0.15], "reward": 1\},

\{"fingertip_position": 'l12', "angle": [3.1, 62.3], "angular_speed": [0, 0], "reward": 1\},

\{"fingertip_position": 'w16', "angle": [13.0, 10.1], "angular_speed": [0, 0], "reward": 1\},]\}

Figure 13. System prompt for Reacher environment Part 2.

```
_pattern = '^26|[1-3]?[0-9]{1,2}[a-z](?:[a-k])?$'
FUNCTION_STRUCTURE = {
  "name": "getReward",
  "parameters": {
    "type": "object",
    "properties": {
      "referred_steps": {
        "type": "array",
        "items": {
          "type": "object",
          "properties": {
            "fingertip_position": {
              "type": "string",
              "pattern": _pattern,
              "description": ("Location of the fingertip with row numbers and a lowercase letter "
                              "for columns. The rows are numbered from bottom to top (1 is the lowest row, increasing "
                              "as you move upward), and columns are labeled from left to right (a, b, ..., z). "
                              "For example, '1b' refers to the lowest row in the second column from the left.")
            },
            "angle": {
              "type": "array",
              "items": {
                "type": "number",
                "format": "float",
                "minimum": -180,
                "maximum": 180,
                "description": "Angle in degrees for the first and second joint of the arm."
              },
              "minItems": 2,
              "maxItems": 2,
              "description": "A 2-item list of angles in degrees corresponding to the first and second joint
respectively."
            },
            "angular_velocity": {
              "type": "array",
              "items": {
                "type": "number"
              },
              "minItems": 2,
              "maxItems": 2,
              "description": ("Vector of two elements corresponding to the angular velocity of the first and second
arm respectively."
                              "If feedback refers to a location with positive reward, but without specifying the speed, set the
angular speed to [0, 0].")
            },
            "reward": {
              "type": "integer",
              "enum": [-1, 1],
              "description": "The reward value, which can be +1 for good performance or -1 for bad performance."
            }
          },
          "required": ["fingertip_position", "reward"],
          "minProperties": 2,
          "description": ("Dictionary containing information about the state and its reward. "
                          "Must have the 'reward' and 'fingertip_position' keys and optionally other keys: "
                          "'angle', or 'angular_velocity'.")
        },
        "description": "List of states described by the feedback along with their reward implied by the feedback."
      }
    },
    "required": ["referred_steps"]
  }
}
```

Figure 14. Function calling to force output format in Reacher environment.