

# Achieving AI Alignment with Unreliable Supervision

*Shivam Singhal  
Cassidy Laidlaw  
Anca Dragan*



Electrical Engineering and Computer Sciences  
University of California, Berkeley

Technical Report No. UCB/Eecs-2024-148

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2024/Eecs-2024-148.html>

July 11, 2024

Copyright © 2024, by the author(s).  
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

### Acknowledgement

I would like to give a heartfelt thank you to Cassidy Laidlaw for all of his amazing mentorship. He has always made me feel welcomed into the research community, and he has always valued my ideas and believed in me—I truly couldn't have asked for a better person to work with. I would also like to thank Smitha Milli for inspiring me to partake in research and for introducing me to the important field of AI safety as a CHAI intern. Additionally, I would like to acknowledge Professor Anca Dragan for all of her insightful guidance and my labmates at the InterACT lab for their friendship. Last but definitely not least, I would like to thank my family for all of their constant love and support. Words cannot describe how much they mean to me, and without them, I wouldn't be here.

---

# Achieving AI Alignment with Unreliable Supervision

Shivam Singhal

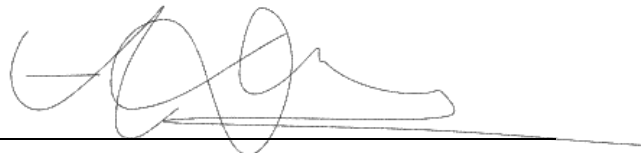
---

## Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences,  
University of California at Berkeley, in partial satisfaction of the requirements for the  
degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

### Committee:



---

Professor Anca Dragan  
Research Advisor

5/13/24

---

(Date)

\*\*\*\*\*



---

Professor Sergey Levine  
Second Reader

5/13/24

---

(Date)

# Achieving AI Alignment with Unreliable Supervision

Copyright 2024  
by  
Shivam Singhal

Abstract

Achieving AI Alignment with Unreliable Supervision

by

Shivam Singhal

Master's of Science in Computer Science

University of California, Berkeley

Professor Anca Dragan, Chair

As AI designers, our aim is to develop AI agents that are not only capable, but are also able to *understand the internal preferences of the humans with whom they are interacting in order to accomplish the correct goals*. However, humans are fundamentally complicated: we have dynamic, sometimes unknown or conflicting desiderata that are difficult to encode directly or learn, and we do not always exhibit optimal behavior that we would like AI to replicate. As a result, we are significantly lacking in our ability to *robustly model human preferences, while accounting for their suboptimality*. With these misspecified human models, AI systems can not properly infer the goals we would like them to accomplish or be aligned with the values we would like them to have. AI has become increasingly skilled at making complex decisions, but in general, the practical deployment of AI in its misaligned state remains *extremely dangerous* since there aren't any real guarantees about its activity. Thus, in this thesis, we explore two avenues for achieving AI alignment despite our limitations. In particular, we propose a new regularization regime to prevent AI agents from hacking their specified rewards, and we present two new modeling strategies that we can use to learn from unreliable human feedback.

To my parents, Ajay and Anju, and sister, Arpita

Words can't express how much you mean to me. Thank you for your constant love and support. Without you, I wouldn't be here.

# Contents

<b>Contents</b>	<b>ii</b>
<b>List of Figures</b>	<b>iv</b>
<b>List of Tables</b>	<b>vi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 The Challenge of Value Alignment . . . . .	1
1.2 Value Misalignment Mitigation when Goals are Misspecified . . . . .	2
1.3 Value Alignment when Unreliable Feedback is Provided . . . . .	2
<b>2 Preventing Reward Hacking with Occupancy Measure Regularization</b>	<b>4</b>
2.1 Introduction . . . . .	4
2.2 Related work . . . . .	6
2.3 Action distribution vs. occupancy measure regularization . . . . .	7
2.4 Occupancy-regularized policy optimization (ORPO) . . . . .	11
2.5 Experiments . . . . .	14
2.6 Conclusion . . . . .	17
<b>3 Scalable Oversight by Accounting for Unreliable Feedback</b>	<b>19</b>
3.1 Introduction . . . . .	19
3.2 Related Work . . . . .	22
3.3 Reward Learning with Unreliable Feedback . . . . .	23
3.4 Designing Metrics that Capture Annotation Difficulty . . . . .	25
3.5 Conclusion . . . . .	31
<b>Bibliography</b>	<b>32</b>
<b>A Project 1: Reward Hacking Mitigation</b>	<b>47</b>
A.1 Proofs . . . . .	47
A.2 Additional results . . . . .	56
A.3 Environment details . . . . .	62
A.4 Experiment details . . . . .	64

A.5	Elaborated Related Work . . . . .	66
<b>B</b>	<b>Project 2: Learning from Unreliable Preferences</b>	<b>69</b>
B.1	Difficult Dataset Creation and Survey Collection . . . . .	69
B.2	Reward Model Training . . . . .	73
B.3	Defining Difficulty Metrics . . . . .	75



# List of Figures

- 2.1 The MDP above, similar to that used in the proof of Proposition 2.3.1, demonstrates one drawback of using divergence between policies’ action distributions for regularization. The agent stays in state  $s_1$ , where it receives no reward, until it takes action  $a_2$ , after which it remains in state  $s_2$  forever and receives 1 reward per timestep. The plot shows the return  $J(\pi, R)$  for a policy  $\pi$  when  $\gamma = 0.99$  as a function of the policy’s action distribution at  $s_1$ . While  $\pi$  and  $\pi_{\text{safe}}$  (shown on the plot as dotted lines) are close in action distribution space, they achieve very different returns. Meanwhile, the optimal policy  $\pi^*$  is far from  $\pi_{\text{safe}}$  in action distribution space. Propositions 2.3.2 and A.1.2 show that occupancy measure divergences do not have these drawbacks. . . . . 8
- 2.2 This adaption of the tomato-watering AI Safety Gridworld [77] provides an intuitive example of why OM divergence is superior to AD divergence for regularizing to a safe policy. The robot agent can move up, down, left, right, or stay in place. The true reward function  $R$  only rewards watering tomatoes, while the proxy reward function  $\tilde{R}$  also highly rewards reaching the sprinkler.
- AD divergence** The top row shows three policies for this environment: a desired policy that achieves the highest true reward, a safe policy that achieves lower true reward, and a reward hacking policy that exploits the sprinkler state to achieve high proxy reward but low true reward. The AD KL divergences between the policies, shown over the arrows connecting them, suggest that the reward hacking policy is actually closer to the safe policy than the desired policy is. Thus, if we regularize to the safe policy using action distribution KL divergence, we would be more likely to find a policy that hacks the proxy reward, rather than one like the left policy, which we prefer.
- OM divergence** The bottom row shows the occupancy measures for each policy in the top row, and the arrows between the columns show the total variation distance  $\|\mu - \mu'\|_1$ . The desired policy on the left is closer to the safe policy than the reward hacking policy is in OM divergence. This is because both the desired and safe policies spend most of their time actually watering tomatoes, while the reward hacking policy mainly visits the sprinkler state. Thus, if we trained a policy regularized with occupancy measure divergence, we could hope to find a policy like the desired one on the left and avoid a reward hacking policy like the one on the right. . . . . 10

2.3	The top row of plots in the figure shows the true rewards of policies trained with three types of regularization to $\pi_{\text{safe}}$ for several values of $\lambda/R_{\text{average}}$ : action distribution, state-only OM, and state-action OM. The bottom two rows of plots in the plot show the KL divergence between the action distributions and occupancy measures of the learned and safe policies for each coefficient. For all plots and tables, we give the median across 5 seeds with error bars indicating the standard deviation. We find that reward hacking consistently occurs in each environment without regularization or with very small regularization coefficients $\lambda$ . As $\lambda$ is increased to moderate values, the learned policy stops reward hacking and often improves upon the safe policy. At high regularization coefficients, the learned policy approaches the safe policy. . . . .	12
3.1	Consider a preference learning dataset that contains one easy question and one difficult question. Assuming the annotator prefers correct responses, the responses to Question 1 are easy to judge because the question is based on common knowledge, and therefore, the annotator is able to correctly specify that they prefer Response A. On the other hand, Question 2 is much more difficult because it requires domain-specific expertise, and as a result, the annotator struggles with it and is forced to rely on unrelated facts (i.e., that gold is expensive) to make a judgement, which is ultimately factually incorrect. The traditional reward learning paradigm views the feedback given for each of these questions as being equivalent in quality. Our proposal is to account for how unreliable the annotator’s feedback is expected to be. In this case, our approach effectively up-weights the feedback given on Question 1 and down-weights the the preference specified for Question 2 since it isn’t reliable. . . . .	21
A.1	AUROC curves for OM and AD-based reward hacking predictors . . . . .	56
A.2	This plot is similar to the one shown in Figure 2.3, except instead of regularizing towards safe policies, we are regularizing away from reward hacking policies. . .	60
A.3	Here, the gray squares represent walls, and the white squares represent open spaces where the agent can travel. . . . .	63
A.4	Here, the green cars are controlled by the human driver model IDM controller, and the blue cars are controlled by RL. . . . .	63
B.1	These are the introductory remarks that we showed to survey participants. . . .	74
B.2	An example of the screening questions shown to participants . . . . .	75
B.3	An example of the questions shown to participants for evaluation. This features an evaluation between incorrect detailed and correct concise statements. . . . .	76

# List of Tables

2.1	The top three rows of the table give the performance gap recovered (PGR, see Section 2.5) when using the optimal coefficient $\lambda^*$ for each type of regularization. The middle three rows show the PGR attained when using the coefficient $\lambda_{\text{drop}}$ which decreases AD or OM divergence the most compared to a slightly smaller coefficient. The two rows show baselines: a policy trained on the proxy reward without regularization (exhibiting reward hacking) and a policy trained with the proxy reward with early stopping when the highest true reward is achieved. The latter baseline is impossible in practice because when true reward is unknown but is given as an additional comparison. The median and standard deviation across 5 random seeds are reported. . . . .	14
2.2	We find that, compared to AD divergence, OM divergence is a much better predictor of whether reward hacking is occurring during training according to its area under the ROC curve (AUROC). This validates that OM divergence is a more successful regularizer because it more accurately identifies when reward hacking is happening. See Figure A.1 for full AUROC curves. . . . .	16
2.3	The true rewards achieved by regularizing away from reward hacking policies in the four environments. OM regularization prevents reward hacking in all four environments, while AD regularization fails to improve on $\pi_{\text{hacking}}$ in the tomato and traffic environments.	17
3.1	We find that our LLM-based scores place a much higher weight on factual correctness compared to regular reward learning, but they do not place more weight on length as a feature. We also do not see that there is much of a difference between the two approaches in terms of the weights being placed on the features.	28
3.2	We calculate the FLRD metric for all difficulty metrics. Our proposed LLM autograder performs the best, suggesting that incorporating our designed difficulty metrics will allow for reward models that place more weight on important but harder-to-evaluate features, like factual correctness. . . . .	28

A.1	The top three rows of the table give the median true reward when using the optimal coefficient $\lambda^*$ for each type of regularization. The middle three rows show the true reward attained when using the coefficient $\lambda_{\text{drop}}$ which decreases AD or OM divergence the most compared to a slightly smaller coefficient. The bottom four rows show the true rewards for the baselines: the safe policy $\pi_{\text{safe}}$ , a policy trained on the proxy reward without regularization (exhibiting reward hacking), a policy trained with the proxy reward with early stopping when the highest true reward is achieved, and a policy trained on the true reward. The latter two baselines are impossible in practice because when true reward is unknown but are given as additional comparisons. The median and standard deviation across 5 random seeds are reported. . . . .	57
A.2	Results of regularizing towards a safe policy in the Tomato environment . . . . .	58
A.3	Results of regularizing towards a safe policy in the Traffic environment . . . . .	58
A.4	Results of regularizing towards a safe policy in the Glucose environment . . . . .	59
A.5	Results of regularizing towards a safe policy in the Pandemic environment . . . . .	59
A.6	Results of regularizing away from a reward hacking policy in the Tomato environment . . . . .	60
A.7	Results of regularizing away from a reward hacking policy in the Traffic environment . . . . .	61
A.8	Results of regularizing away from a reward hacking policy in the Glucose environment . . . . .	61
A.9	Results of regularizing away from a reward hacking policy in the Pandemic environment . . . . .	62
A.10	PPO/ORPO hyperparameters. . . . .	66
A.11	ORPO-specific hyperparameters. . . . .	66
B.1	We fit logistic regression models between generated difficulty scores and whether or not people made correct evaluations. We were interested in seeing whether annotators got more difficult questions incorrect more often. . . . .	81

## Acknowledgments

I would first like to give a heartfelt thank you to Cassidy Laidlaw for all of his amazing mentorship. He has always valued my ideas and believed in me, even when I don't, and I have really learned a lot from him over the years. I truly couldn't have asked for a better person to work with. I would also like to thank my previous mentor Smitha Milli for inspiring me to partake in research and introducing me to the wonderful community at CHAI as an intern.

Additionally, I would like to acknowledge my advisor Professor Anca Dragan for all of her insightful guidance. Despite her extremely busy schedule, she always took the time to meet with me and give advice. I will always fondly remember her excitement and enthusiasm for AI safety and alignment. I would also like to thank my labmates at the InterACT lab for being there whenever I needed help and for always making me feel welcome. They helped cultivate a really warm, friendly, and collaborative atmosphere that I am confident no other lab at BAIR has.

Last but definitely not least, I would like to thank my parents and sister for all of their consistent support. They are always there for me, no matter what happens, and they are always happy to put up with my craziness. Without them, I couldn't have achieved this or any other milestone.

# Chapter 1

## Introduction

### 1.1 The Challenge of Value Alignment

AI system designers are tasked with creating technology that caters to their users, but all of us are nuanced individuals with equally nuanced goals that we would like to achieve. With all of this complexity to take into account, a difficult question arises: *how can we design AI systems that do what humans want?*. Two common approaches to address this question are to manually specify our desiderata or to train a model that learns our preferences from our feedback or demonstrated behavior [24, 39, 78].

Ideally, we would be able to communicate our goals to our AI agents. However, this is no easy task since we are not skilled at conveying all of our wants and needs. Indeed, this is a challenge that dates back to good old King Midas, who wanted everything he touched to turn to gold without explicitly stating that he would want his loved ones or food to not be transformed in the process. Let's more concretely consider the example of recommender systems, a ubiquitous form of AI which have a loosely defined goal of maximizing the value that users attain from their time spent on the online platforms. Due to the ambiguous nature of optimizing users' well-being, designers will utilize some proxy, such as click-through rates, engagement time, and other types of implicit feedback they receive from users, but it is likely that these variables are not well-correlated with how satisfied users are with their experience [128].

In addition to being unable to mathematically quantify our requirements, our actions are not always reflective of the behavior we would like our AI to emulate [89], and our preferences are not always consistent or static [15]. With these misspecified human models, AI systems can not properly infer the goals we would like them to accomplish or be aligned with the values we would like them to have [46]. AI has become increasingly capable at complex decision-making, but in general, the practical deployment of AI in its misaligned state remains *extremely dangerous* since there aren't any real guarantees about its activity, especially in safety-critical scenarios like in hospitals or on the roads [109]. In this thesis, we explore two facets of the alignment problem: how we can effectively deal with value

misalignment when our goals are misspecified, and how we can better achieve value alignment when we are unable to provide reliable preferences.

## 1.2 Value Misalignment Mitigation when Goals are Misspecified

Reward hacking occurs when an agent performs very well with respect to a “proxy” reward function (which may be hand-specified or learned), but poorly with respect to the unknown true reward. Since ensuring good alignment between the proxy and true reward is extremely difficult, one approach to prevent reward hacking is optimizing the proxy conservatively. Prior work has particularly focused on forcing the learned policy to behave similarly to a “safe” policy by penalizing the KL divergence between their action distributions (AD). However, AD regularization doesn’t always work well since a small change in action distribution at a single state can lead to potentially calamitous outcomes, while large changes might not be indicative of any dangerous activity. Our insight is that when it is reward hacking, the agent visits drastically different states from those reached by the safe policy, causing large deviations in state *occupancy measure* (OM). Thus, in Chapter 2, we propose regularizing based on the OM divergence between policies instead of AD divergence to prevent reward hacking. We theoretically establish that OM regularization can more effectively avoid large drops in true reward. Then, we empirically demonstrate in a variety of realistic environments that OM divergence is superior to AD divergence for preventing reward hacking by regularizing towards a safe policy. Furthermore, we show that occupancy measure divergence can also regularize learned policies *away* from reward hacking behavior.

## 1.3 Value Alignment when Unreliable Feedback is Provided

Reward functions learned from human feedback serve as the training objective for RLHF, the current state-of-the-art approach for aligning large language models (LLMs) to our values; however, in practice, these reward models fail to robustly capture our desiderata. For instance, they often place more weight on the length of the output or agreement with the user and less on important features like factual correctness. A major reason behind these shortcomings of learned reward functions is the fact that human annotator feedback on which the models are trained is *unreliable*. Due to knowledge gaps, limited resources, cognitive biases, or other factors, annotators may not be able to accurately judge the model’s outputs, and thus, their feedback may not be reliably aligned with their true preferences. Current proposals to address the challenges posed by unreliable feedback include providing annotators with an AI assistant during evaluation, only asking them questions that they can easily answer, and relying primarily on AI feedback with limited human supervision (e.g., constitutional

AI). However, it remains unclear how practical and scalable these approaches are. In Chapter 3, we identify a complementary strategy that can easily be incorporated into existing alignment methods (e.g., RLHF, DPO, etc.): explicitly modeling the annotators' knowledge and judgment in order to better learn from unreliable feedback. In particular, we propose an adjustment to the Bradley-Terry model used in preference learning that accounts for how well an annotator's feedback is expected to match their true values or preferences. We test our approach in a setting where annotators are likely to provide unreliable feedback, and we find that it results in preference models that assign higher value to important characteristics, like factuality, than existing methods.



## Chapter 2

# Preventing Reward Hacking with Occupancy Measure Regularization

**Acknowledgements:** This chapter was co-written by the author and Cassidy Laidlaw.

### 2.1 Introduction

A major challenge for the designers of goal-oriented AI systems is specifying a reward function that robustly captures their goals and values. Manually hand-specifying reward functions is difficult due to the ambiguities and complex variables underlying real-world scenarios [51]. An alternative is to learn reward functions from human data [111, 54], but these often fail to generalize outside the distribution of behavior seen during training [88, 129]. Thus, a learned or hand-specified reward function is often just a *proxy* for the true reward underlying the system designer’s intent. Misalignment between the two objectives can lead to *reward hacking*: a learned policy performs well according to the proxy reward function, but not according to the true reward function [110, 1, 98, 120]. A reward hacking policy’s behavior is often undesirable and can be especially catastrophic when deployed in safety-critical scenarios, such as autonomous driving [66, 132, 61]. Unfortunately, reward hacking is a common phenomenon [65], which has problematic implications in the real world [85, 26, 94, 89, 103, 31, 60].

One method to prevent reward hacking is to avoid fully optimizing the proxy reward function by using constraints or regularization. In particular, prior work has regularized the chosen actions of a learning policy to be similar to those of a known safe policy [143]. A *safe policy* is any policy that has reasonable (although potentially quite suboptimal) performance and does not reward hack; safe policies can be hard-coded or learned from human data. For example, RLHF for LLMs generally optimizes the learned reward plus a term that penalizes divergence from the pre-trained language model’s output [35, 97]. Intuitively, this kind of regularization pushes the learned policy away from “unusual” behaviors for which the reward function may be misaligned.

The goal of optimizing a policy with regularization is to achieve higher true reward than the safe policy while avoiding reward hacking. To do so effectively, we must choose a regularization regime that is simultaneously strong enough to prevent the learned policy from reward hacking, while also sufficiently lenient to ensure the learned policy outperforms the safe policy. We argue that in many cases, regularizing based on the *action distributions* (AD) of policies makes it impossible to achieve this goal. This is because small shifts in action distribution can lead to large differences in outcomes, but large shifts in action distributions may not cause any difference in outcome. As an example, imagine an autonomous car driving alongside a steep cliff on a coastal highway. Suppose we have access to a safe policy that drives slowly and avoids falling off the cliff. However, the car is optimizing a proxy reward function that prioritizes quickly reaching the destination, but not necessarily staying on the road. Since only slightly increasing the probability of some unsafe action (e.g., making a sharp right turn) can lead to disaster, we will need to apply heavy regularization if we try to regularize the car’s AD to the safe policy. In turn, this heavy regularization will prevent even minor deviations in action distribution, making it near-impossible to improve upon the safe policy.

If action distribution divergences are poor regularizers for reward hacking, what can we do instead? In our car example, while a single catastrophic action doesn’t change the action distribution much, it does drastically change the *distribution over states* visited by the car. The learned policy will have a high probability of reaching states where the car is off the cliff and crashed, while the safe policy never reaches such states. Our proposal follows naturally from this observation: to avoid reward hacking, regularize based on divergence from the safe policy’s *occupancy measure* rather than action distribution. A policy’s occupancy measure (OM) is the distribution of states or state-action pairs seen by a policy when it interacts with its environment. While algorithms based on occupancy measures have been widely used for imitation learning [44], offline RL [72], and efficient exploration [41], using OM divergence to prevent reward hacking remains unexplored.

We show that OM-based regularization is superior to AD regularization for preventing reward hacking in both theory and practice. Theoretically, we show that there is a bound on the difference in return of two policies under *any* reward function based on the divergence between their occupancy measures. Thus, constraining the OM divergence from a safe policy can prevent the large drop in true reward associated with reward hacking, even when the true reward function is unknown. In contrast, only much weaker guarantees can be established for AD divergence.

Empirically, we derive an algorithm called **Occupancy-Regularized Policy Optimization** (ORPO) that can be easily incorporated into deep RL algorithms like Proximal Policy Optimization (PPO) [113]. ORPO approximates the occupancy measure divergence between policies using a discriminator network. We use ORPO to optimize policies trained with misaligned proxy reward functions in multiple reward hacking benchmark environments [98] and compare it to AD regularization. The results of our experiments demonstrate that training with occupancy measure regularization leads to better performance under the unseen true reward function in all environments. In contrast, we find that it is difficult to tune

AD regularization in some environments to both prevent reward hacking and meaningfully improve over the safe policy. To explain why this is the case, we show that, when compared with AD divergence, OM divergence from the safe policy is a much more accurate predictor of whether the learned policy is reward hacking. When a safe policy is unavailable, an alternative method to prevent reward hacking is to encourage a learned policy to have behavior that is as *different* from a reward hacking policy as possible. Our experiments show that optimizing for the proxy reward plus OM divergence from a reward hacking policy is also effective at avoiding reward hacking.

Our main contributions can be summarized as follows:

1. We show theoretically that occupancy measure regularization is superior to action distribution regularization for preventing reward hacking because constraining OM divergence effectively prevents large drops in the unknown true reward function.
2. We present the ORPO algorithm to implement OM regularization and show that it outperforms AD regularization in realistic environments.
3. We demonstrate that OM regularization can also be effectively used to regularize away from reward hacking.

## 2.2 Related work

While there have been separate lines of work investigating reward hacking and exploring the use of occupancy measures for other applications, to the best of our knowledge, we are the first to specifically study applying occupancy measures to the problem of reward hacking.

### **Reward hacking:**

Some prior works establish theoretical models of reward hacking as a special case of Goodhart’s Law [37, 78, 64, 120, 93]. Krakovna [65] provides a list of many examples of reward hacking. Pan, Bhatia, and Steinhardt [98] categorize types of reward misspecification and relate optimization power to reward hacking.

### **Safe reinforcement learning:**

Regularizing policies to be similar to an offline policy based on their action distribution KL divergences was first proposed by Stiennon et al. [122] and has since been widely employed in the context of optimizing LLMs using reinforcement learning from human feedback (RLHF) [97, 5, 35]. KL regularization for RLHF has been further studied by Vieillard et al. [135], Gao, Schulman, and Hilton [33], and Korbak, Perez, and Buckley [63]. Some alternative approaches to avoid reward hacking include quantilizers [126], “mild” optimization [127], and impact regularization [133]. While constrained RL can prevent the misbehavior of agents that optimize flawed reward functions [28, 22, 145, 108], it simply shifts the difficulty of designing a reward function to specifying a set of constraints and weights. Other proposals

to address the reward specification problem attempt to infer the true reward function based on the given proxy reward function, environment context, and/or feedback from humans [39, 107, 74].

### Applications of occupancy measures:

Occupancy measures have been used for many purposes in sequential decision making, such as model-based RL [142] and GAIL, a robust imitation learning algorithm [44]. Kang, Jie, and Feng [56] combines GAIL with a reward function to efficiently explore using human data. Another line of work aims to find a policy with the highest-entropy occupancy measure for the purpose of exploring the state space [41, 76, 92]. Many offline RL algorithms also use occupancy measure-based regularization to ensure that the learned policy remains within the training data distribution [87, 42, 21, 106, 140]. The DICE family of RL algorithms [91, 144, 73, 72] and dual RL [118] use the occupancy regularization via duality trick, but their aim is to improve RL performance rather than to prevent reward hacking.

### Our contribution:

The above works leverage occupancy measures and some derive algorithms that are similar to our proposed ORPO algorithm. However, unlike previous work, we use OM-based regularization to *prevent reward hacking*, which to our knowledge is a novel application. We view our contribution as demonstrating that occupancy measure regularization is superior to action distribution regularization for this purpose. We do not explore the myriad ways that OM regularization could be incorporated into RL to prevent reward hacking. Instead, we focus our experiments on the simple and general ORPO algorithm. We leave to future work further investigation of alternate OM-based regularization algorithms for preventing reward hacking.

## 2.3 Action distribution vs. occupancy measure regularization

We begin by theoretically and conceptually motivating why occupancy measure regularization should be superior to action distribution regularization for preventing reward hacking. We present our theoretical analysis in the setting of an infinite-horizon Markov decision process (MDP). An agent takes actions  $a \in \mathcal{A}$  to transition between states  $s \in \mathcal{S}$  over a series of timesteps  $t = 0, 1, 2, \dots$ . The first state  $s_0$  is sampled from an initial distribution  $\mu_0(s)$ , and when an agent takes action  $a_t$  in  $s_t$  at time  $t$ , the next state  $s_{t+1}$  is reached at timestep  $t + 1$  with transition probability  $p(s_{t+1} | s_t, a_t)$ . The agent aims to optimize a reward function  $R : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ , and rewards are accumulated over time with discount factor  $\gamma \in [0, 1)$ . A policy  $\pi$  maps each state  $s$  to a distribution over actions to take at that state  $\pi(a | s)$ . We define the (normalized) *return* of a policy  $\pi$  under a reward function  $R$  as

$$J(\pi, R) = (1 - \gamma) \mathbb{E}_\pi [\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)]$$

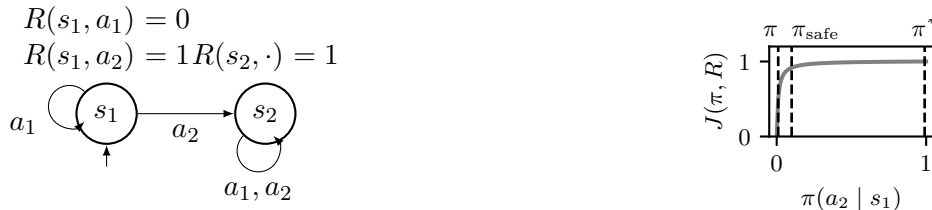


Figure 2.1: The MDP above, similar to that used in the proof of Proposition 2.3.1, demonstrates one drawback of using divergence between policies’ action distributions for regularization. The agent stays in state  $s_1$ , where it receives no reward, until it takes action  $a_2$ , after which it remains in state  $s_2$  forever and receives 1 reward per timestep. The plot shows the return  $J(\pi, R)$  for a policy  $\pi$  when  $\gamma = 0.99$  as a function of the policy’s action distribution at  $s_1$ . While  $\pi$  and  $\pi_{\text{safe}}$  (shown on the plot as dotted lines) are close in action distribution space, they achieve very different returns. Meanwhile, the optimal policy  $\pi^*$  is far from  $\pi_{\text{safe}}$  in action distribution space. Propositions 2.3.2 and A.1.2 show that occupancy measure divergences do not have these drawbacks.

where  $\mathbb{E}_\pi$  refers to the expectation under the distribution of states and actions induced by running the policy  $\pi$  in the environment. The normalizing factor  $1 - \gamma$  guarantees that  $J(\pi, R) \in [0, 1]$  always.

We define the *state-action occupancy measure*  $\mu_\pi$  of a policy  $\pi$  as the expected discounted number of times the agent will be in a particular state and take a specific action:

$$\mu_\pi(s, a) = (1 - \gamma) \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t \mathbf{1}\{s_t = s \wedge a_t = a\} \right].$$

Intuitively, the occupancy measure represents the distribution of states and actions visited by the policy over time. The standard approach to solving an MDP is to find a policy  $\pi$  that maximizes its return:

$$\text{maximize } J(\pi, R). \tag{2.1}$$

However, as we discussed in section 2.1, an AI system designer most likely does not have access to a reward function that perfectly encapsulates their preferences. Instead, the designer might optimize  $\pi$  using a learned or hand-specified *proxy* reward function  $\tilde{R}$  which is misaligned with the *true* reward function  $R$ . Blindly maximizing the proxy reward function could lead to reward hacking.

**The drawbacks of action distribution regularization:** One approach to preventing reward hacking is to optimize the policy’s return with respect to the proxy  $\tilde{R}$  plus a regularization term that penalizes the KL divergence of the policy’s action distribution (AD) from a *safe policy*  $\pi_{\text{safe}}$ . This is equivalent to solving the following constrained optimization problem:

$$\begin{aligned} &\text{maximize } J(\pi, \tilde{R}) \quad \text{s.t.} \\ &(1 - \gamma) \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t D_{\text{KL}}(\pi(\cdot | s_t) \| \pi_{\text{safe}}(\cdot | s_t)) \right] \leq \epsilon. \end{aligned} \tag{2.2}$$

Intuitively, the aim of the AD constraint in (2.2) is to prevent the unusual behavior associated with reward hacking policies by constraining  $\pi$  to take similar actions to  $\pi_{\text{safe}}$ .

While AD regularization is simple and easy to implement, this method also has serious drawbacks. In particular, the following proposition shows that in some cases small changes in action distribution from a safe policy can induce large drops in true reward, but large changes in AD are necessary to improve on the safe policy.

**Proposition 2.3.1.** *Fix  $c_1 > 0$  and  $\delta > 0$  arbitrarily small, and  $c_2 \geq 0$  arbitrarily large. Then there is an MDP, true reward function  $R$ , and safe policy  $\pi_{\text{safe}}$  where both of the following hold:*

1. *There is a policy  $\pi$  where the action distribution KL divergence satisfies*

$$(1 - \gamma) \mathbb{E}_{\pi} \left[ \sum_{t=0}^{\infty} \gamma^t D_{KL}(\pi(\cdot | s_t) \parallel \pi_{\text{safe}}(\cdot | s_t)) \right] \leq c_1$$

*but  $J(\pi_{\text{safe}}, R) - J(\pi, R) \geq 1 - \delta$ .*

2. *Any optimal policy  $\pi^* \in \arg \max_{\pi} J(\pi, R)$  satisfies*

$$(1 - \gamma) \mathbb{E}_{\pi^*} \left[ \sum_{t=0}^{\infty} \gamma^t D_{KL}(\pi^*(\cdot | s_t) \parallel \pi_{\text{safe}}(\cdot | s_t)) \right] \geq c_2.$$

All proofs are given in Appendix A.1. The first part of Proposition 2.3.1 states that in the worst case, a policy with AD divergence from the safe policy below some arbitrarily small threshold  $c_1$  can induce a drop in return under the true reward function  $R$  that is almost as large as the entire possible range of returns. Thus, we must set the AD divergence constraint very small (i.e.,  $\epsilon \ll c_1$ ) to prevent reward hacking. The second part of Proposition 2.3.1 shows that in the same MDP, it is necessary to change the action distribution by an arbitrarily large divergence  $c_2$  to improve on the safe policy and reach an optimal policy. Thus, if we set  $\epsilon \ll c_1$  to prevent reward hacking, it will not allow for the large changes to the action distribution that are necessary to improve over  $\pi_{\text{safe}}$ . See Figure 2.1 for a graphical illustration of the results in Proposition 2.3.1.

While the MDP in Proposition 2.3.1 is a particularly bad case for AD regularization, we argue that realistic environments often have the same issues. In many safety-critical environments, even slightly increasing the probability of taking an unsafe action can greatly reduce true reward, as posited in part 1 of the proposition. Furthermore, safe policies are often not robust out-of-distribution (OOD), e.g., an imitation learned policy might take unusual actions in states outside the training distribution. Since a single unusual action can lead to an OOD state in which the safe policy is no longer a meaningful regularization target, this also means small AD divergence can lead to large drops in reward.

**The benefits of occupancy measure regularization:** Due to the drawback of action distribution regularization, we propose preventing reward hacking by regularizing the divergence between the occupancy measures of the learned and safe policies:

$$\text{maximize } J(\pi, \tilde{R}) \quad \text{s.t.} \quad \|\mu_{\pi} - \mu_{\pi_{\text{safe}}}\|_1 \leq \epsilon. \quad (2.3)$$

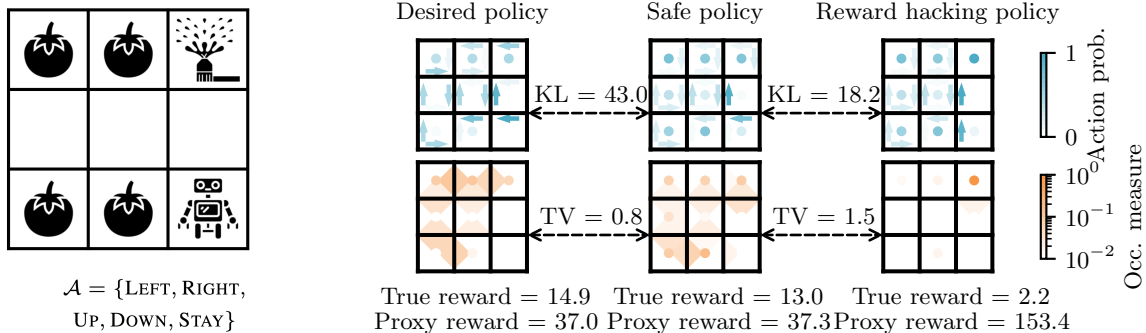


Figure 2.2: This adaption of the tomato-watering AI Safety Gridworld [77] provides an intuitive example of why OM divergence is superior to AD divergence for regularizing to a safe policy. The robot agent can move up, down, left, right, or stay in place. The true reward function  $R$  only rewards watering tomatoes, while the proxy reward function  $\tilde{R}$  also highly rewards reaching the sprinkler. **AD divergence** The top row shows three policies for this environment: a desired policy that achieves the highest true reward, a safe policy that achieves lower true reward, and a reward hacking policy that exploits the sprinkler state to achieve high proxy reward but low true reward. The AD KL divergences between the policies, shown over the arrows connecting them, suggest that the reward hacking policy is actually closer to the safe policy than the desired policy is. Thus, if we regularize to the safe policy using action distribution KL divergence, we would be more likely to find a policy that hacks the proxy reward, rather than one like the left policy, which we prefer. **OM divergence** The bottom row shows the occupancy measures for each policy in the top row, and the arrows between the columns show the total variation distance  $\|\mu - \mu\|_1$ . The desired policy on the left is closer to the safe policy than the reward hacking policy is in OM divergence. This is because both the desired and safe policies spend most of their time actually watering tomatoes, while the reward hacking policy mainly visits the sprinkler state. Thus, if we trained a policy regularized with occupancy measure divergence, we could hope to find a policy like the desired one on the left and avoid a reward hacking policy like the one on the right.

In (2.3), we use the total variation (TV) between the occupancy measures, defined as

$$\|\mu_\pi - \mu_{\pi_{\text{safe}}}\|_1 = \sum_{(s,a) \in \mathcal{S} \times \mathcal{A}} |\mu_\pi(s, a) - \mu_{\pi_{\text{safe}}}(s, a)|.$$

Why should using the occupancy measure divergence to regularize perform better than using the divergence between action distributions? Ideally, unlike action distribution divergence, there should be a closer relationship between the rewards of two policies and their occupancy measure divergence. In fact, it is possible to show that the difference in returns between two policies for *any* reward function can be bounded by their occupancy measures divergence:

**Proposition 2.3.2.** *For any MDP, reward function  $R$ , and pair of policies  $\pi, \pi_{\text{safe}}$ , we have*

$$|J(\pi_{\text{safe}}, R) - J(\pi, R)| \leq \|\mu_\pi - \mu_{\pi_{\text{safe}}}\|_1. \quad (2.4)$$

Results equivalent to Proposition 2.3.2 have been shown by Xu, Li, and Yu [141] among others, but this result has not been applied before in the context of reward hacking. For completeness we give a proof with our notation in Appendix A.1.

Proposition 2.3.2 suggests that OM regularization can effectively prevent the large drops in true reward associated with reward hacking, even when the true reward is unknown. Suppose the returns of all reward hacking policies  $\pi_{\text{hacking}}$  satisfy  $J_R(\pi_{\text{hacking}}) < J_R(\pi_{\text{safe}}) - C$ , i.e., reward hacking policies have true reward that is smaller than that of the safe policy by more than  $C$ . Then, setting the OM divergence constraint  $\epsilon = C$  in (2.3) will prevent reward hacking, since any policy within the constraint must satisfy  $J_R(\pi) \geq J_R(\pi_{\text{safe}}) - C$  by Proposition 2.3.2.  $C$  is often large in practice since reward hacking induces a large drop in true reward. Thus, we can use a large constraint bound  $\epsilon$  in (2.3) that allows improvement over the safe policy while still preventing reward hacking.

Although it is possible to prove a similar bound to (2.4) using action distribution divergence, it has a  $\frac{1}{1-\gamma}$  prefactor [141], meaning that a constraint on AD divergence must be set  $1-\gamma$  times the equivalent OM constraint to obtain an equivalent guarantee about the true reward. Thus, in environments with high discount factors—i.e., most realistic environments—the constraint must be set to a value too small to allow meaningful improvement over  $\pi_{\text{safe}}$ .

**An illustrative example:** See Figure 2.2 for an example of why OM regularization outperforms AD regularization. While in this example it is particularly obvious that OM regularization should work better, we find in Section 2.5 that OM outperforms AD in more realistic environments too.

**Why does AD regularization work for LLMs?:** Despite the drawbacks of action distribution regularization in theory, it has performed well in practice when used as part of RLHF for large language models [122, 33]. In Appendix A.1, we show that for current implementations of RLHF, action distribution and OM-based regularization are actually equivalent. Thus, RLHF is essentially already using occupancy measure regularization. However, this is only true under certain strict assumptions which are satisfied almost exclusively in the current RLHF-for-LLMs paradigm. For more general environments, there can be significant differences between action distribution and OM-based regularization, as clearly demonstrated by our experiments.

## 2.4 Occupancy-regularized policy optimization (ORPO)

In the previous sections, we showed theoretical evidence that regularizing RL by constraining OM divergence is superior to constraining AD divergence. We now introduce an algorithm, Occupancy-regularized policy optimization (ORPO), to feasibly approximate the occupancy measure divergence between the learned and safe policies for the purpose of regularizing deep RL agents.

While our theory uses the TV distance between occupancy measures, we find that the KL divergence is more stable to calculate in practice. Since Pinsker’s inequality and the



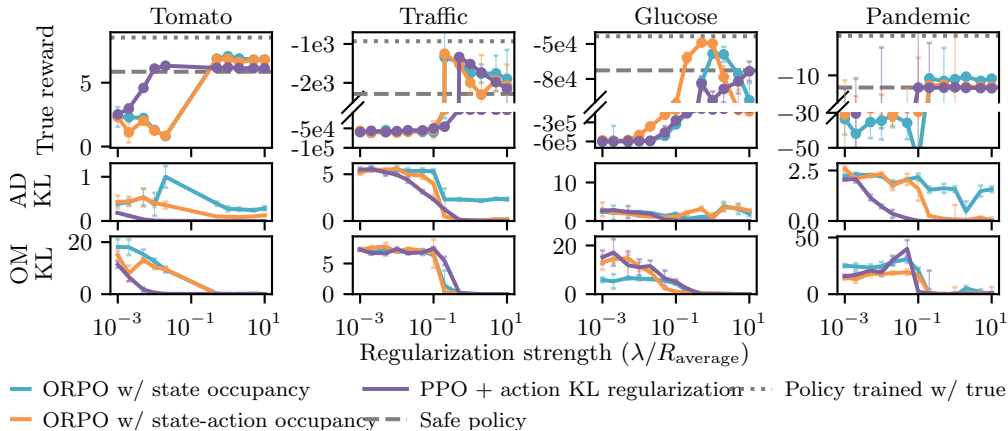


Figure 2.3: The top row of plots in the figure shows the true rewards of policies trained with three types of regularization to  $\pi_{\text{safe}}$  for several values of  $\lambda/R_{\text{average}}$ : action distribution, state-only OM, and state-action OM. The bottom two rows of plots in the plot show the KL divergence between the action distributions and occupancy measures of the learned and safe policies for each coefficient. For all plots and tables, we give the median across 5 seeds with error bars indicating the standard deviation. We find that reward hacking consistently occurs in each environment without regularization or with very small regularization coefficients  $\lambda$ . As  $\lambda$  is increased to moderate values, the learned policy stops reward hacking and often improves upon the safe policy. At high regularization coefficients, the learned policy approaches the safe policy.

Bretagnolle-Huber inequality show that TV distance is upper-bounded in terms of KL divergence, our theoretical guarantees remain valid in the case of OM KL [14]. Our objective from (2.3) can be reformulated with the KL divergence in place of the TV distance and a Lagrangian relaxation in place of the hard constraint:

$$\text{maximize } J(\pi, \tilde{R}) - \lambda D_{\text{KL}}(\mu_{\pi} \parallel \mu_{\pi_{\text{safe}}}). \quad (2.5)$$

DICE-based RL [91, 73] and dual RL [118] also optimize an objective similar to (2.5), but those algorithms are aimed at solving different challenges than preventing reward hacking. We leave the exploration of their use for preventing reward hacking to future work and instead focus here on comparing AD and OM-based regularization more generally.

We optimize (2.5) using a gradient-based method. The gradient of the first term is estimated using PPO, a popular policy gradient method [113]. However, calculating the occupancy measure divergence for the second term is intractable to do in closed form since it requires the enumeration of *all* possible state-action pairs, an impossible task in deep RL. Thus, we approximate the KL divergence between the occupancy measures of policies by training a *discriminator network*, a technique that has previously been used for generative adversarial networks (GANs) [36] and GAIL [44]. The discriminator network  $d : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$  assigns a score  $d(s, a) \in \mathbb{R}$  to any state-action pair  $(s, a) \in \mathcal{S} \times \mathcal{A}$ , and it is trained on a mixture of data from both the learned policy  $\pi$  and safe policy  $\pi_{\text{safe}}$ . The objective used to train  $d$  incentivizes low scores for state-action pairs from  $\pi_{\text{safe}}$  and high scores for state-action

pairs from  $\pi$ :

$$d = \arg \min_d \sum_{t=0}^{\infty} \left( \mathbb{E}_{\pi}[\gamma^t \log(1 + e^{-d(s_t, a_t)})] + \mathbb{E}_{\pi_{\text{safe}}}[\gamma^t \log(1 + e^{d(s_t, a_t)})] \right). \quad (2.6)$$

Huszár [50] proves that if the loss function in (2.6) is minimized, then the expected discriminator scores for state-action pairs drawn from the learned policy distribution will approximately equal the KL divergence between the occupancy measures of the two policies:

$$D_{\text{KL}}(\mu_{\pi}(s, a) \parallel \mu_{\pi_{\text{safe}}}(s, a)) \approx (1 - \gamma) \mathbb{E}_{\pi} \left[ \sum_{t=0}^{\infty} \gamma^t d(s_t, a_t) \right].$$

Applying the definitions of the learned policy’s returns and the KL divergence between the policies’ occupancy measures, we can now rewrite our ORPO objective:

$$\text{maximize } \mathbb{E}_{\pi} \left[ \sum_{t=0}^{\infty} \gamma^t \left( \tilde{R}(s_t, a_t) - \lambda d(s_t, a_t) \right) \right]. \quad (2.7)$$

Note that (2.7) is identical to the normal RL objective with a reward function  $R'(s, a) = \tilde{R}(s, a) - \lambda d(s, a)$ . Thus, once the discriminator has been trained, we add the discriminator scores to the proxy reward function and use the combined values to update  $\pi$  with PPO. The training process for ORPO consists of iterating between two phases: one in which data from both the safe and learned policies is used to train the discriminator to minimize (2.6), and one in which data from the learned policy is used to train the PPO agent with the augmented reward function in (2.7).

**Regularization with state-only occupancy measure:** While we have thus far considered the state-action occupancy measure of a policy  $\mu_{\pi}(s, a)$ , it sometimes makes more sense to regularize based on the state-only occupancy measure  $\mu_{\pi}(s) = (1 - \gamma) \mathbb{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t \mathbf{1}\{s_t = s\}]$ . In particular, when the reward function  $R(s)$  is a function of only the state, it is simple to establish similar guarantees to Proposition 2.3.2 based on just the state occupancy measure. We can implement this within ORPO by only providing the state as input to the discriminator. Intuitively, regularizing with state OM divergence in environments where the reward function only depends on the state avoids over-applying regularization when it is unnecessary. See Appendix A.1 for more details.

**Regularizing away from reward hacking policies:** While there is a natural safe policy for many environments, it may not always be possible to define one. In such cases, it may be possible to instead regularize *away* from reward hacking behavior. That is, suppose training without any regularization in some environment results in a policy  $\pi_{\text{hacking}}$  that exhibits reward hacking. Then, we can train a second policy with the following objective:

$$\text{maximize } J(\pi, \tilde{R}) + \lambda D_{\text{KL}}(\mu_{\pi} \parallel \mu_{\pi_{\text{hacking}}}).$$

Unlike in (2.3), we encourage the occupancy measure of  $\pi$  to be as far as possible from  $\pi_{\text{hacking}}$ . This will hopefully prevent  $\pi$  from exhibiting the same reward hacking behavior as  $\pi_{\text{hacking}}$ . It is trivial to modify ORPO to optimize this objective by flipping the sign of the discriminator term in (2.7).

Method	Environment			
	Tomato	Traffic	Glucose	Pandemic
Action dist. regularization ( $\lambda^*$ )	$0.12 \pm 0.01$	$0.70 \pm 0.04$	$-0.03 \pm 0.29$	$0.01 \pm 0.01$
State occupancy regularization ( $\lambda^*$ )	<b><math>0.44 \pm 0.04</math></b>	$0.61 \pm 0.12$	$0.47 \pm 0.12$	<b><math>0.21 \pm 0.08</math></b>
State-action occupancy regularization ( $\lambda^*$ )	$0.38 \pm 0.06$	<b><math>0.76 \pm 0.09</math></b>	<b><math>0.83 \pm 0.01</math></b>	$0.05 \pm 0.02$
Action dist. regularization ( $\lambda_{\text{drop}}$ )	$-0.47 \pm 0.07$	$-39.00 \pm 4.34$	$-13.82 \pm 3.47$	$-1.20 \pm 0.62$
State occupancy regularization ( $\lambda_{\text{drop}}$ )	<b><math>0.39 \pm 0.05</math></b>	$0.62 \pm 18.06$	<b><math>-2.93 \pm 0.86</math></b>	<b><math>0.20 \pm 0.08</math></b>
State-action occupancy regularization ( $\lambda_{\text{drop}}$ )	$0.34 \pm 0.02$	<b><math>0.76 \pm 0.09</math></b>	$-3.73 \pm 0.26$	$0.04 \pm 0.09$
No regularization	$-1.30 \pm 0.10$	$-39.65 \pm 4.68$	$-18.12 \pm 0.49$	$-1.91 \pm 0.73$
Early stopping (best case)	$0.36 \pm 0.07$	$0.03 \pm 0.09$	$-0.20 \pm 0.78$	$0.32 \pm 0.44$

Table 2.1: The top three rows of the table give the performance gap recovered (PGR, see Section 2.5) when using the optimal coefficient  $\lambda^*$  for each type of regularization. The middle three rows show the PGR attained when using the coefficient  $\lambda_{\text{drop}}$  which decreases AD or OM divergence the most compared to a slightly smaller coefficient. The two rows show baselines: a policy trained on the proxy reward without regularization (exhibiting reward hacking) and a policy trained with the proxy reward with early stopping when the highest true reward is achieved. The latter baseline is impossible in practice because when true reward is unknown but is given as an additional comparison. The median and standard deviation across 5 random seeds are reported.

## 2.5 Experiments

We now use ORPO to compare the empirical performance of occupancy measure and action distribution regularization in four environments, described below. We chose the first for illustrative purposes, and the following three because they are reward hacking benchmark environments from Pan, Bhatia, and Steinhardt [98].

**Tomato gridworld:** Like in Figure 2.2, the tomato environment contains a sprinkler state where the agent perceives all tomatoes as being watered and thus receives high proxy reward but no true reward. We train a safe policy using the true reward function, and then add a 10% chance of taking a random action to ensure there is room to improve upon it.

**Flow traffic simulator:** The traffic environment simulates a group of human-controlled and RL-controlled vehicles on an on-ramp attempting to merge into traffic on a highway [138]. The true reward prioritizes a small mean commute time, while the proxy reward is the average velocity of all cars. When reward hacking, the RL controlled vehicle on the on-ramp stops indefinitely and lets cars continue forward at high speeds on the highway, which maximizes the proxy reward but increases the commute times of cars on the on-ramp infinitely. As the safe policy for the traffic environment we used the Intelligent Driver Model (IDM), a standard approximation of human driving behavior [131]. In practice, safe policies are often learned via imitation learning, so to simulate this we generate data from the IDM controller and train a behavioral cloning (BC) policy using the generated data.

**SimGlucose:** The SimGlucose blood glucose monitoring environment is an extension of the FDA-approved glucose monitoring simulator proposed by Man et al. [86] for Type 1 Diabetes patients [30]. The RL agent controls the insulin administered to a simulated patient in order to maintain healthy glucose levels. The true reward is a standard measure of health risk for the patient, but the proxy reward is misaligned and prioritizes the monetary cost of the treatment. Optimizing for a cost-based proxy has caused major disparities in access to healthcare on the basis of race [94]. As the safe baseline policy, we train a BC policy based on data generated by a PID controller with parameters tuned by the original designers of the simulator [121].

**COVID-19 simulator:** The pandemic environment simulates a population’s infection dynamics using the SEIR model [90, 62]. The RL agent chooses the level of lockdown restrictions placed on the population by observing the results of testing. The proxy reward function omits the political cost associated with certain decisions. Our safe policy is trained via BC on a combination of hand-specified and real-world strategies employed by governments during the pandemic, which were also used by Kompella et al. [62] as baselines.

## Regularizing towards a safe policy

We train RL policies in each environment with action distribution regularization and OM regularization to the environment’s safe policy, varying the regularization coefficient  $\lambda$  across a wide range. Since the scale of the reward functions varies between environments, we normalize  $\lambda$  by the typical per-timestep reward for each environment, which we denote  $R_{\text{average}}$ . In the environments that we studied,  $\lambda/R_{\text{average}}$  values between  $10^{-3}$  and  $10^1$  seemed to work best. See Appendix A.4 for all experimental details.

The results of our experiments are shown in Table 2.1 and Figure 2.3. We report the performance gap recovered (PGR) for each learned policy, defined as

$$\text{PGR}(\pi) = \frac{J(\pi, R) - J(\pi_{\text{safe}}, R)}{J(\pi_{\text{true}}, R) - J(\pi_{\text{safe}}, R)}$$

where  $\pi_{\text{true}}$  is a policy trained on the true reward function. The PGR normalizes the true reward such that 0 represents no improvement over the safe policy and 1 represents equal performance as training with the true reward function. In each environment, we find that OM regularization with the optimal coefficient ( $\lambda^*$ ) outperforms action distribution regularization. OM regularization consistently allows improvement over the performance of  $\pi_{\text{safe}}$  while preventing reward hacking, recovering 20-80% of the gap between the safe policy and a policy trained on the true reward function. Meanwhile, action distribution regularization fails to improve significantly on the safe policy in the glucose and pandemic environments with a PGR of 1% or less.

Comparing performance with the optimal regularization coefficients  $\lambda^*$  is unrealistic since, in practice, system designers must choose the coefficient without access to the unknown true

Environment	AUROC for predicting reward hacking	
	OM KL	AD KL
Tomato	<b>1.00</b>	0.89
Traffic	<b>1.00</b>	0.98
Glucose	<b>0.99</b>	0.79
Pandemic	<b>0.94</b>	0.82

Table 2.2: We find that, compared to AD divergence, OM divergence is a much better predictor of whether reward hacking is occurring during training according to its area under the ROC curve (AUROC). This validates that OM divergence is a more successful regularizer because it more accurately identifies when reward hacking is happening. See Figure A.1 for full AUROC curves.

reward function. Observing changes in the policies’ divergences as  $\lambda$  is varied can help designers choose the right coefficient. In particular, we find that the optimal regularization coefficient is often the coefficient at which the regularized divergence drops the most compared to a slightly smaller coefficient, which we denote as  $\lambda_{\text{drop}}$ . In the middle three rows of Table 2.1, we compare the PGR of policies trained with  $\lambda_{\text{drop}}$ . Regularizing based on OM divergence with  $\lambda_{\text{drop}}$  generally achieves PGR close to those obtained with  $\lambda^*$ , despite being chosen without access to the true reward function.

We find that both state-only and state-action occupancy measure regularization achieve similar performance. Generally, state-only occupancy measures perform better in environments whose true reward functions depend primarily on the state, reflecting the intuition of our theory in Appendix A.1. In practice, we recommend experimenting with both OM regularizers.

In addition to comparing OM and AD regularization, we also test *early stopping*, which has been proposed by Karwowski et al. [57] as another method for preventing reward hacking. We consider the best possible case for early stopping: we train policies on the proxy reward function and then evaluate the policy from the iteration with the highest true reward. While this best-case approach is infeasible in practice since the true reward is unknown, we still find that OM regularization is superior to early stopping in all environments except for the pandemic simulator.

### Explaining the superior performance of OM regularization

In Section 2.3, we hypothesized that OM regularization is superior to action distribution regularization because there is a stronger relationship between OM divergence and the difference in returns of two policies under any reward function; therefore, OM divergence should better measure when there is a large difference in true rewards between the safe and learned policies, indicating reward hacking. We empirically test this hypothesis by comparing how well both action distribution and OM divergence predict if reward hacking is occurring during RL. In particular, we divide all of our training runs into ten segments, and for each segment

Regularization	Environment			
	Tomato	Traffic ( $\times 10^3$ )	Glucose ( $\times 10^3$ )	Pandemic
AD	$1.98 \pm 1.49$	$-58.23 \pm 2.95$	<b><math>-10.37 \pm 0.20</math></b>	<b><math>-8.35 \pm 1.94</math></b>
State OM	$5.32 \pm 0.22$	$-1.10 \pm 0.04$	$-186.14 \pm 17.98$	$-14.28 \pm 0.40$
State-Act. OM	<b><math>5.59 \pm 0.32</math></b>	<b><math>-1.07 \pm 0.01</math></b>	$-93.15 \pm 29.54$	$-14.23 \pm 5.02$
No regularization ( $\pi_{\text{hacking}}$ )	$2.35 \pm 0.14$	$-57.38 \pm 3.53$	$-599.02 \pm 1.58$	$-29.57 \pm 6.86$

Table 2.3: The true rewards achieved by regularizing away from reward hacking policies in the four environments. OM regularization prevents reward hacking in all four environments, while AD regularization fails to improve on  $\pi_{\text{hacking}}$  in the tomato and traffic environments.

record (i) whether the agent is reward hacking, (ii) the action distribution divergence from  $\pi_{\text{safe}}$ , and (iii) the OM divergence from  $\pi_{\text{safe}}$ . For (i), we define reward hacking as achieving higher proxy reward but lower true reward than  $\pi_{\text{safe}}$ . Then, we calculate how accurately each type of divergence can predict whether reward hacking is occurring across all training run segments. The results of this experiment are shown in Table 2.2. We find that in all environments, OM divergence is a better classifier of reward hacking behavior, validating our hypothesis as to why it is a better regularizer for preventing reward hacking.

## Regularizing away from reward hacking behavior

We experiment with regularizing away from reward hacking policies using both AD and OM regularization. We obtain a  $\pi_{\text{hacking}}$  for each environment by training on the proxy reward without regularization, and we regularize away from  $\pi_{\text{hacking}}$  using a range of values of  $\lambda$ . The results are presented in Table 2.3. OM KL regularization consistently avoids reward hacking and, in some cases, even outperforms the safe policies. On the other hand, the AD regularized policies’ true reward is dangerously close to that of  $\pi_{\text{hacking}}$  in some of the environments, indicating that it is unable to consistently prevent reward hacking.

## 2.6 Conclusion

We have presented theoretical and empirical evidence that occupancy measure regularization can more effectively prevent reward hacking than action distribution regularization when training with a misaligned proxy reward function. To address the practical challenges of the OM-based approach, we introduced an algorithm called ORPO, which uses an adversarially trained discriminator to approximate the KL divergence between the occupancy measures of policies. While OM regularization is not a perfect solution, our results are a step towards

a better understanding of methods for preventing reward hacking.

**Open questions:** Since OM divergence is more difficult to compute and optimize than AD divergence, future work could explore better approximators of OM divergence and ways of integrating occupancy measure regularization into the training process. Another open question is how to choose regularization coefficients without access to the true reward function.

**Impact:** Reward hacking in the real world has already led to significant disparities on the basis of race, gender, and other distinguishing factors in the realms of healthcare [94, 103], policing [85, 26, 31], and online platforms like recommender systems [89, 60]. As AI systems' objectives become more complex and they are used in increasingly important societal roles, reward hacking will continue to become both more common and more consequential. In particular, as RLHF-based assistants are trained with tool-use or simulated human interaction, AD and OM regularization will no longer be equivalent, and OM regularization may be needed to avoid reward hacking. Thus, we hope that our results contribute to the goal of ensuring that future AI systems are safe and beneficial.

## Chapter 3

# Scalable Oversight by Accounting for Unreliable Feedback

**Acknowledgements:** This chapter was co-written by the author and Cassidy Laidlaw.

### 3.1 Introduction

Human supervision has been the key to aligning widely deployed large language models (LLMs) to our complex, hard-to-define values [5, 95]. In particular, techniques like reinforcement learning from human feedback (RLHF) rely on a reward function that is learned from annotator-provided pairwise preference comparisons between different LLM-generated responses [24]. Then, pre-trained base LLMs are fine-tuned by optimizing for these rewards either explicitly using RL algorithms such as PPO, i.e., RLHF [5, 97, 130], or implicitly using various other techniques, e.g., DPO [104], ORPO [45]. While these alignment approaches have rendered LLMs capable of achieving impressive performance on tasks that are both in and out of their training distribution [43, 59], they have also made LLMs prone to potentially dangerous behaviors: fine-tuned LLMs are more likely than base models to produce sycophantic text in which they simply agree to whatever the user is saying [102, 117]; and they will easily hallucinate and produce text that is not factually correct [95, 79]. In fact, the literature has even shown that RLHF tends to lead LLMs to prioritize generating longer outputs [119, 20, 100]. Furthermore, models to which RLHF has been applied are more likely to imitate the persuasion and manipulation tactics that are employed by humans, outputting text in a confident tone even when incorrect [38, 125].

A significant factor contributing to these failure modes of LLMs is the *unreliable feedback provided by annotators*. Specifically, humans often struggle to provide annotations that accurately reflect their true values. This causes reward models (RMs) to disproportionately value more obvious output features, such as length and assertiveness, and underweight features that are more difficult to evaluate, such as factual correctness [48]. Human annotators have decaying attention spans and are likely to make trivial errors due to time constraints



and lack of interest, which is also affected in part by the survey setup (e.g., the amount they are paid, time required, task complexity and language, etc.) [96, 99, 5, 49]. Additionally, annotators are not all-knowing, particularly when it comes to domain-specific tasks [47, 3]. They are tasked with specifying their preferences even if they do not have all the relevant details to make an informed judgement, and this type of partial observability in preference learning is known to lead to undesirable behavior [70]. The preferences of human annotators is also likely to be driven by various cognitive biases that are invoked by the questions asked or the choice comparisons presented [32, 27].

These challenges of human annotation will be especially exacerbated as models produce content that is increasingly difficult to judge. For example, summaries of large passages are difficult to evaluate for fidelity because they require reading the entire source passage [112, 123]. This leads to the problem of **scalable oversight** [1, 8]: *how can we use suboptimal human annotators to oversee increasingly capable AI systems?* To address this issue, a few potential solutions have been proposed. Human annotators can either be assisted by or completely replaced by AI agents [23, 4]. In addition, annotators can simply be asked to make evaluations about easier questions and hope that the model will generalize to more difficult settings [12, 40]. However, all of these approaches are still active areas of research, and it is uncertain whether or not they will facilitate the learning of more robust RMs [17, 2].

Currently, preference learning relies on the Bradley-Terry model [9, 105, 24], which assumes humans are Boltzmann rational [84, 146, 55]—when people express their preferences, their likelihood of choosing a particular option is proportional to the exponentiated value or reward they associate with it. However, this model fails to account for *how difficult* it is for human annotators to accurately judge which option best aligns with their preferences. For example, consider the two preference comparisons in Figure 3.1: each consists of comparing correct and incorrect answers to a science question. Suppose the annotator assigns equal value to both incorrect answers and equal value to both correct answers. In this case, Boltzmann rationality would assume that an annotator would be equally likely to choose the correct answer for both questions. However, the first question is easy while the second requires more obscure knowledge. Thus, intuitively, it seems like an annotator is more likely to choose the correct response for question 1 than for question 2—an effect which the Bradley-Terry model is unable to capture. Since preference learning is based around Bradley-Terry, this results in preference learning treating both annotations as equally reliable sources of information about the annotator’s preferences.

Our insight is that we can fix this problem by *explicitly modeling the bounded rationality of the annotators that provide preferences*. We define **annotator difficulty** for each sample in a preference comparison dataset along three axes: whether or not the annotator will have enough knowledge to make a choice, whether or not they will have the cognitive resources (e.g., time, reasoning capacity, etc.) to make a judgement, and whether or not the annotator will be impacted by biases that impede the decision-making process. We propose the incorporation of a term into preference learning models that takes into account the variable difficulty that annotators experience when evaluating different examples, and we suggest a

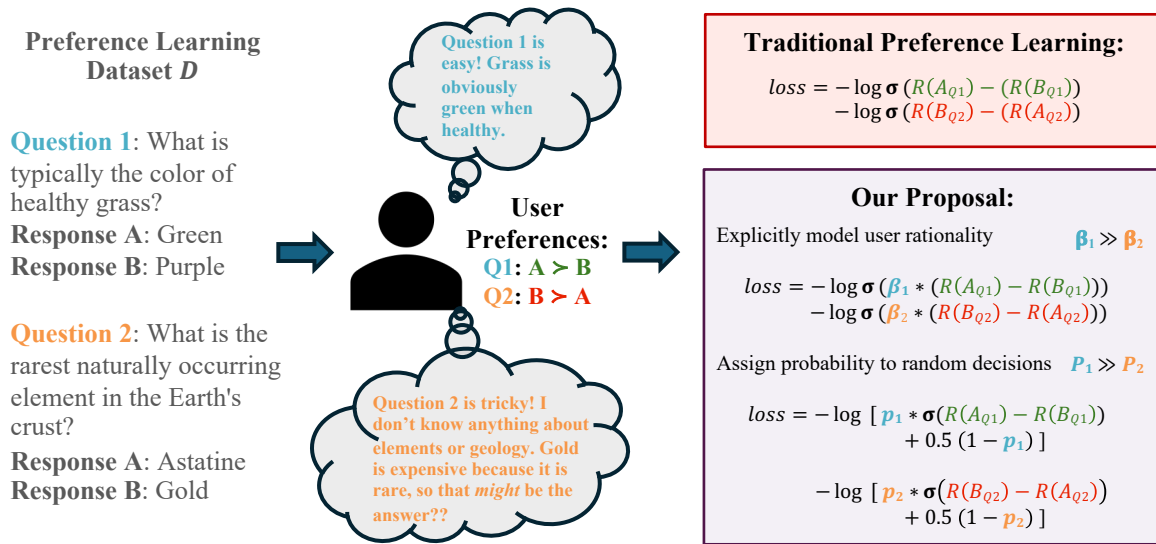


Figure 3.1: Consider a preference learning dataset that contains one easy question and one difficult question. Assuming the annotator prefers correct responses, the responses to Question 1 are easy to judge because the question is based on common knowledge, and therefore, the annotator is able to correctly specify that they prefer Response A. On the other hand, Question 2 is much more difficult because it requires domain-specific expertise, and as a result, the annotator struggles with it and is forced to rely on unrelated facts (i.e., that gold is expensive) to make a judgement, which is ultimately factually incorrect. The traditional reward learning paradigm views the feedback given for each of these questions as being equivalent in quality. Our proposal is to account for how unreliable the annotator’s feedback is expected to be. In this case, our approach effectively up-weights the feedback given on Question 1 and down-weights the the preference specified for Question 2 since it isn’t reliable.

practical method with which these difficulty scores can be specified based on our defined criteria.

To evaluate our method, we study an RLHF setting in which human feedback is unreliable. First, we construct a preference learning dataset that contains questions based on common misconceptions, and for each question, we generate responses that vary in length, factual correctness, or both. Then, we confirm that annotators rely on text length and assertiveness to make choices, especially for difficult questions [48], and we find that reward models trained on this flawed feedback tend to weight length more than correctness. Next, we explore how to explicitly account for the reliability of human feedback. To determine the difficulty of annotating each comparison, we first consider easily attainable measures, such as annotator confidence and time spent per-question. However, we find that these metrics are not good indicators of annotator reliability, and incorporating them into preference

learning does not have any significant effect on the weights placed on length or correctness by the resulting RMs. We then design an LLM prompting-based autograder to judge when annotators might find it difficult to provide feedback that aligns with their true preferences, and we find that our suggested prompting regime is able to elicit difficulty scores from LLMs that match when annotators tend to get evaluations correct.

Our contributions can be summarized as follows:

- We collect a dataset that can be used to evaluate a reward model’s ability to learn from unreliable feedback.
- We find that reward models trained on unreliable human feedback tend to place a higher weight on features that annotators use as proxies during their evaluations, such as length, under-valuing other desirable features, such as factual correctness.
- Incorporating a notion of evaluation difficulty into the training process results in better reward functions that assign greater weight to features that humans value but are harder to evaluate, such as factual correctness.
- We present an LLM-based autograder that is able to evaluate examples from preference learning datasets and generate scores that capture how difficult annotators would find it to provide an accurate preference.

## 3.2 Related Work

While the idea of modeling human rationality to adjust preference learning has been explored primarily in a theoretical fashion or in other settings, to the best of our knowledge, we are the first to empirically study this methodology for LLMs.

**The challenges with human annotation:** As discussed in Section 3.1, human annotators face various challenges when evaluating examples from preference learning datasets. Hosking, Blunsom, and Bartolo [48] systematically study human annotator responses on surveys and find that annotators’ judgements are skewed by the use of assertive or complex language towards factually incorrect responses. Singhal et al. [119] and Park et al. [100] identify the fact that RMs learned during preference learning can be mostly optimized if the length of the generated text is simply maximized.

**Scalable oversight proposals:** Amodei et al. [1] introduce the idea of scalable oversight—the ability to provide reliable supervision over examples that are beyond the scope of human understanding. In the context of RLHF for LLMs, several approaches to reconcile with the limitations of annotators are currently being considered by the research community:

- Annotators can simply be asked easier questions [137, 12], where difficult questions are filtered out from the evaluation set based on human or model-based difficulty measures,

and the goal is that what is learned from human supervision over easy questions will generalize to harder questions of the same variety [114, 11, 40, 124]. While initial results demonstrate the promise of easy-to-hard generalization, it remains unclear if completely omitting the signal learned from human supervision over hard examples will facilitate the learning of robust RMs.

- The other major proposal that is currently being explored is that of incorporating AI systems into the preference learning process, either to assist humans in their evaluations [23, 53, 78, 139] or to entirely replace the need for human supervision with AI feedback (i.e., RLAIIF) [4, 71]. RLAIIF pipelines have been found to be quite suboptimal in performance, however [116]. Additionally, humans may not agree with AI-generated judgements [71], and the goodness of these judgements is fundamentally tied to whether or not the AI assistant providing assistance or preferences is itself aligned (e.g., they can still generate manipulative language to affect humans as studied by Carroll et al. [16]).

**Learning from unreliable feedback:** Chan, Critch, and Dragan [18], Lindner and El-Assady [83], and Hong, Bhatia, and Dragan [46] identify the fact that modeling human irrationality can better inform the reward learning process and point out that modeling humans as Boltzmann rational leads to potentially less aligned RMs being learned. Some work in the literature has studied how to best use unreliable demonstrations in reinforcement learning [58, 67, 19, 10], and Lee et al. [75] benchmarks the impact of irrational preferences on various RL algorithms. In addition, some prior work has focused on primarily theoretically studying the effect of modeling human rationality in the Bradley-Terry model for various applications like actively querying a human in the loop [34] and addressing the expertise problem [29, 6]. Furthermore, Lang et al. [70] mathematically model what happens when human feedback is limited due to partial observability. In the context of RLHF for LLMs, Chen et al. [20] propose learning multiple rewards for different features, and Park et al. [100] suggest disentangling features like text length from factual correctness in the loss function.

**Other open challenges with RLHF:** Casper et al. [17] provide a comprehensive overview of the current challenges with RLHF, discussing the limitations of human annotators, reward modeling, and policy optimization. Lambert, Gilbert, and Zick [69] emphasize the need to study reward models to ensure the alignment of LLMs to our preferences.

### 3.3 Reward Learning with Unreliable Feedback

RLHF and other alignment methods aim to optimize an AI system according to the true underlying preferences of human users, denoted as the true reward  $R$ ; however, in practice  $R$  is unknown and needs to be learned. The established pipeline for learning from annotator feedback involves three main steps: collecting preference comparisons between example text generations, learning a reward model  $\hat{R}$  using this feedback, and optimizing the learned

reward function. Specifically, annotators are tasked with deciding between two statements or trajectories,  $a$  and  $b$  where the responses have been generated by some base LLM [24]. They are supposed to choose the statement that best represents the behavior that they would like an AI chatbot to emulate. The preference learning dataset  $D$  consists of  $(a_+, a_-)$  tuples where  $a_+$  is preferred and  $a_-$  is rejected by the annotator.

## Traditional Reward Learning

Under the current preference learning paradigm, humans are modeled as being Boltzmann rational [55], which implies that as annotators providing preference comparisons, their probability of choosing a particular option is supposed to be correlated, albeit somewhat noisily, with the exponentiated value that they associate with it. In other terms, the probability that an annotator prefers statement  $a$  to statement  $b$ ,  $P(a \succ b)$ , is assumed to follow the Bradley-Terry model [84, 146]:

$$P_R(a \succ b) = \frac{\exp(\beta * R(a))}{\exp(\beta * R(a)) + \exp(\beta * R(b))} \quad (3.1)$$

where  $\beta$  is a parameter that is supposed to be assigned a value based on how noisy the decision-making process is.  $\hat{R}$  is trained by minimizing the following loss function, equivalent to forming a maximum-likelihood estimate of  $R$  under the Bradley-Terry model:

$$\text{loss}(\hat{R}) = - \sum_{(a_+, a_-) \in D} \log P_{\hat{R}}(a_+ \succ a_-) \quad (3.2)$$

Intuitively, this loss aims to maximize the difference in reward assigned to statements that have been chosen by annotators and statements that have been rejected by annotators.

## Explicitly Modeling Annotator Rationality

As shown in Figure 3.1, our proposal is to explicitly model the difficulty that annotators experience when giving preferences due to various factors, such as lack of knowledge or cognitive biases. Specifically, we propose two ways in which this information can be incorporated into the preference learning setup:

- Approach 1: We can use difficulty scores directly as the rationality parameter  $\beta$  that is already a part of the Bradley Terry model.
- Approach 2: We can use difficulty scores to assign some probability mass to the event that the user randomly picks between the two alternatives rather than choosing based on their preferences.

**Adjusting  $\beta$ :** If we adjust the Bradley-Terry model’s  $\beta$  parameter directly, RMs should be trained to minimize the loss in Equation 3.3.

$$\text{loss}(\hat{R}) = \sum_{(a_+, a_-) \in D} -\log \sigma(\beta_a(\hat{R}(a_+) - \hat{R}(a_-))) \quad (3.3)$$

Here,  $\beta_a \in [0, \infty)$  is the unique value that is assigned to the response pair  $\{a_+, a_-\}$  based on the corresponding difficulty that annotators experience during evaluation. Since higher  $\beta$  values will ensure that the RM learns to highly value the features that are being preferred by the provided feedback, high  $\beta$  values should be assigned to preference comparisons where we are certain that we will receive reliable feedback from annotators. On the other hand, low  $\beta$  values will place less weight or essentially filter out preferences and should thus be applied to samples where we expect to receive unreliable annotator feedback.

While the existence of  $\beta$  has been noted in the preference learning literature, prior work has essentially ignored it, assigning it a value of 1 [24, 52] or another fixed value for all provided preferences [115, 13, 55, 75]. Additionally, this approach is somewhat similar to that of simply asking annotators easy questions; however, in that method, feedback for difficult questions is simply thrown out, implicitly assigning them a  $\beta$  value of 0. We believe that defining  $\beta$  values along a continuous spectrum as we propose is valuable because it allows for varying amounts of reward signal to be learned based on the degree of unreliability expected in the annotation—it doesn’t completely eliminate the signal that can be learned from these samples.

**Assigning probability mass to random preferences:** When annotators provide unreliable feedback on a particular example, it’s unclear which of the two preferences between which they are choosing better aligns with their true intentions. Thus, for samples that receive increasingly unreliable feedback, we can also simply assign higher probability to a random preference. Unlike Approach 1, which adjusts the rewards directly, this approach will involve training a RM that minimizes the loss function in Equation 3.4

$$\text{loss}(\hat{R}) = \sum_{(a_+, a_-) \in D} -\log \left[ p_a * \sigma(\hat{R}(a_+) - \hat{R}(a_-)) + (1 - p_a) * 0.5 \right] \quad (3.4)$$

Here,  $p_a \in [0, 1]$  is the unique probability value that is assigned to each response pair  $\{a_+, a_-\}$  based on how likely it is that the corresponding annotator-provided feedback will be reliable. The more difficult an evaluation is, the lower  $p$  should be.

### 3.4 Designing Metrics that Capture Annotation Difficulty

Given how hard it is to model human behavior accurately, how can we reliably define when annotators will have difficulty choosing an alternative that aligns with their preferences? In

this section, we explore different ways in which difficulty measures can be specified and how we can then incorporate these difficulty scores into our proposed approach. First, we examine what would happen if we were to train reward models using feedback that is perfectly reliable (i.e., annotators always chose the factually correct answer when possible). Afterwards, we study difficulty measures that are easily attainable when collecting preference comparison survey data—metrics that are provided either explicitly or implicitly by annotators themselves. Next, we explore better and more feasible ways in which difficulty information can be gathered about preference comparison pairs by employing prompting strategies on popular LLMs, such as Meta’s Large Language Model Meta AI (Llama) and OpenAI’s GPT models, that have been pre-trained and fine-tuned on large amounts of data that likely captures different facets of human behavior. Lastly, we compare our method to another comparable scalable oversight method, training only on easy questions.

**Difficult Dataset Design:** To study the effect of unreliable feedback on reward learning, we first needed to construct a setting where annotators would be highly likely to be unreliable. For this purpose, we built a dataset by taking questions from TruthfulQA, an existing LLM-evaluation benchmark that consists of misconceptions across various subject categories, such as health and finance, along with various incorrect and correct answers [82]. These questions are based on commonly-held falsehoods, so they are already quite difficult for the average annotator; they might require advanced knowledge, or they might invoke cognitive biases due to previously-held beliefs.

We further complicated the evaluation process for annotators by leveraging the fact that they often make decisions using simply the length of statements, especially when the questions being asked are already difficult [48]. Specifically, to develop our preference comparison pairs, we paired responses that varied both in their factual correctness and in their length and assertiveness. We chose the lengths and correctness of each pair of responses such that the two features would be anti-correlated: that is, statements that were correct were more likely to be concise, whereas statements that were incorrect were more likely to be detailed and confident in tone. Subsequently, we recruited annotators using CloudResearch Connect, a platform similar to Mechanical Turk and used their annotations to train reward models. We believe that our collected dataset can be beneficial in the future for evaluating RMs on their ability to learn from unreliable feedback. More details about our dataset creation and survey collection are available in Appendix B.1, and more information about our reward model training procedure can be found in Appendix B.2.

**Evaluation Criteria:** To evaluate the trained reward models, we used our test set, which consisted of questions that were not included during training and corresponding answer statements that varied in factuality and correctness. Afterwards, we bootstrap sampled questions and their corresponding statements from the test set, and we fit linear regression models, using binary variables representing whether or not the statements were correct and whether or not statements were detailed to predict the reward that was assigned to a particular statement. We repeated this process 100 times, and we took the median values of

the weights assigned by the models to the features to get a robust estimate of how highly the reward model valued the factual correctness and length. We denote the value that the reward models assign to factuality as  $V_F$ , and we denote the value that the reward models assign to length as  $V_L$ . We report all of these regression coefficients for comparison in Table 3.1.

Our goal was to train reward models that are able to place more weight on factual correctness but not place much more weight on length in comparison to a baseline model that has been trained using traditional preference learning by minimizing the loss in Equation 3.2. To quantify this, we define the Factuality-Length Ratio Difference (FLRD) metric which captures when the importance placed by an RM on correctness increases more than the change in importance placed by an RM on length:

$$\text{FLRD}(\text{R}) = \frac{V_{F, \text{R}}}{V_{F, \text{baseline}}} - \frac{V_{L, \text{R}}}{V_{L, \text{baseline}}} \quad (3.5)$$

When the FLRD is greater than 0, this implies that the trained RM applies more weight to correctness relative to the weight that it applies to length when comparing to the baseline regular reward learning model. Conversely, when the FLRD is less than 0, this implies that the trained RM applies more highly values length than correctness when comparing to the baseline model. These metrics are reported in Table 3.2.

Furthermore, we also consider how well the difficulty scores can explain the preferences that we observed during our data collection. To do this, we fit logistic regression models between the difficulty scores and whether or not people tended to get a question correct during our survey collection. We report these results across the various difficulty metrics we considered in Appendix B.3.

We now break down the various metrics that we considered by category below.

### Artificially annotated dataset

We first trained RMs in the practically impossible setting of perfectly reliable annotations (i.e., annotators always choose the correct answer whenever possible). In particular, we used the same questions from our training set, but we artificially annotated them to pick the correct statement when the two statements in the preference comparison pair had opposite factual correctness, or pick randomly when statements with the same factual correctness were paired together (since there is no objectively correct choice between a concise statement and a detailed statement). Because our training set contains several more correct and concise statements, and we have synthetically annotated our dataset to always pick the correct answer, concise responses were over-represented amongst the statements that were preferred, and thus, it makes sense that the regression coefficient corresponding to length is so negative. Additionally, the FLRD metric for the RM trained on this artificially annotated dataset gives us an upper-bound on what we can expect from reward models trained using the settings that we are using (e.g., the underlying LLM, hyperparameters, etc.). In practice, it is impossible to get this quality of annotations without paying an exorbitant amount of money for expert



Preference learning method	Regression weights	
	$V_L$	$V_F$
Normal PL	1.08	0.26
Artificial Labels	-0.35	0.25
Approach 1 (Confidence)	1.30	0.05
Approach 2 (Confidence)	1.21	-0.08
Approach 1 (Time)	1.14	0.27
Approach 2 (Time)	1.06	0.27
Approach 1 (Clicks)	1.18	0.17
Approach 2 (Clicks)	1.13	0.14
Approach 1 (LLM)	1.78	0.43
Approach 2 (LLM)	1.59	0.51
Easy Qs (diff. $\leq 0$ )	1.07	-0.20
Easy Qs (diff. $\leq 0.5$ )	0.92	-0.20

Table 3.1: We find that our LLM-based scores place a much higher weight on factual correctness compared to regular reward learning, but they do not place more weight on length as a feature. We also do not see that there is much of a difference between the two approaches in terms of the weights being placed on the features.

PL Method	FLRD
Normal PL	0.00
Artificial Labels	1.28
Approach 1 (Confidence)	-1.01
Approach 2 (Confidence)	-1.43
Approach 1 (Time)	-0.02
Approach 2 (Time)	0.06
Approach 1 (Clicks)	-0.44
Approach 2 (Clicks)	-0.51
Approach 1 (LLM)	0.01
Approach 2 (LLM)	<b>0.49</b>
Easy Qs (diff. $\leq 0$ )	-1.76
Easy Qs (diff. $\leq 0.5$ )	-1.62

Table 3.2: We calculate the FLRD metric for all difficulty metrics. Our proposed LLM autograder performs the best, suggesting that incorporating our designed difficulty metrics will allow for reward models that place more weight on important but harder-to-evaluate features, like factual correctness.

annotation, which is why we consider different difficulty metrics to incorporate into our proposed method.

### **Annotator-specific hardness metrics**

During data collection, we can easily gather various annotator-specific information that can be revealing of their behavior. When we collected data on our difficult questions dataset, we asked annotators to not just specify their preferences as binary variables, but specify their preferences on a scale that is reflective of their confidence. Intuitively, it would make sense that these values align well with when annotators find a decision difficult to make—annotators would be less confident about judgements that were difficult for them to make. However, we actually discovered that this isn't the case. In particular, annotators tend to over-estimate their confidence, confidently making incorrect choices. We found this out by fitting our simple logistic regression model between the inverse of the confidence scores (i.e., the less confident an annotator was, the more difficult an evaluation was) and whether or not annotators picked the correct response between pairs of correct and incorrect statements. We additionally trained RMs by incorporating this information and minimizing the loss functions in Equations 3.3 and 3.4, and we found that incorporating this metric actually resulted in models that were placing far less weight on correctness compared to the baseline model trained under the traditional reward learning paradigm, which makes sense given that confidence isn't a good predictor of when annotators got a question correct.

We also considered other annotator-related values that could implicitly be indicative of when they found an evaluation difficult to make. Most survey platforms, such as Qualtrics which is what we used, allow for survey-designers to collect information about the number of times that respondents click on a page and the amount of time spent answering a question. Intuitively, these could potentially serve as difficulty metrics because if a person clicks on a page several times, they might be changing their answer multiple times as they are uncertain about the choice that they picked, or if a person spends more time answering a question compared to others, this might be because they need to think more carefully about this evaluation. Unfortunately, we found that incorporating this information also did not result in models that were better than the baseline.

It's worth noting that when specifying the rationality parameter  $\beta$  or the probability of getting unreliable feedback  $p$  for our proposed models, we assumed a linear relationship between the difficulty metrics and the specified parameter values. As we have seen throughout the cognitive science literature, this relationship may not necessarily hold true, so we would like to explore this further in the future.

### **LLM-generated metrics**

Since easily-specifiable difficulty scores did not result in reward models that were better than those trained using the traditional reward learning loss, we aimed to specify difficulty scores that will ideally result in better reward models but are also practically attainable.

Given some of the recent success of LLMs as cognitive agents [7], we attempted to see if we can elicit difficulty scores that train better reward models by using various prompting strategies on fine-tuned LLMs. In particular, we tried using OpenAI’s GPT models [95] and Meta’s Llama 3 Instruct models [130], and we experimented with several different versions of zero-shot prompts, few-shot prompts, and chain-of-thought (CoT) prompts [136]. By fitting logistic regression models between whether or not the annotators in our study chose the correct answer and the various generated difficulty scores that we considered, we found that scores that were generated by prompting OpenAI’s GPT-3.5 with one of our CoT autograders were well-aligned with when people tended to get questions incorrect. We provide more information about our specific prompting regimes in Appendix B.3.

When exploring these difficulty metrics, we also considered whether the assigned difficulty scores are simply inversely related to the  $\beta$  values and probabilities  $p$  that we use in our approaches. While for the annotator specified metrics and ground truth metrics, it might make more sense that this linear relationship exists between difficulty and the rationality parameters, it might not be the case that LLMs are generating scores that are also linearly related to annotator rationality. Thus, we tried implementing various schemes to relate difficulty to  $\beta$  and the probabilities of unreliable feedback (e.g., exponentiating or taking the log of the difficulty scores to derive rationality parameters, etc.). In practice, we found that the values of the rationality parameter are roughly related to the difficulty scores according to the following function:  $\sigma((1 - d) - t) * m$ . Here,  $d \in [0, 1]$  is a difficulty metric,  $t$  is some small threshold (we considered values of 0.5 and 0.7 for instance), and  $m$  is a scaling factor. Larger values of  $t$  would result in a lower output from the sigmoid function, and higher values of  $m$  will result in a more steep jump between the extremes of the sigmoid functions output, 0 and 1. This can also be seen as a continuous variant of simply thresholding based on difficulty (i.e., filtering out questions that are above some difficulty threshold).

When we trained RMs based on these LLM-generated difficulty scores, we found that these reward models achieved a significant jump in our defined FLRD metric compared to other rationality parameters that can be specified. This means that significantly more weight is being placed on correctness by these RMs compared to the baseline model, and there isn’t much of an increase in the weight being placed on length. It’s also worth noting that here, that our proposed variant of adjusting the probabilities directly (Approach 2) performs a bit better than our proposal in Approach 1; however, since the regression coefficients appear to be relatively similar to each other, we would suggest that reward model designers experiment with both variants in the future.

## Comparing our method to that of filtering using easy questions

Our dataset does not have any built-in difficulty scores that are available, so similarly to Sharma et al. [117], we zero-shot prompt GPT-3.5 10 times to determine the difficulty of preference comparison pairs and take the median value of these scores. We then threshold based on these values. That is, if a question has a difficulty above a certain threshold, we filter it out. We then trained reward models using the traditional reward learning loss on

this filtered dataset. Based on our training results, we can see that these models did not result in more weight being placed on correctness compared to regular reward learning. This might make sense because it is unclear if it is reasonable to expect a RM to fully learn reward signals if only easy questions are included in the training set. Current literature in this domain [40] focuses on metrics that are available for particular datasets; however, we use difficulty scores that are LLM-defined for thresholding, which is likely to be necessary for the large preference learning datasets that are used in practice. The quality of the filtered out questions really drives the success of this approach, and we believe that our proposed LLM-based autograder could help.

### 3.5 Conclusion

Using our difficult questions dataset, we are able to validate that traditional reward learning undervalues features that are often hard for annotators to judge, such as factual correctness. Furthermore, we find that our proposed reward models can significantly increase the weights that they place on these important features, if the right information about when annotators are unreliable is incorporated. Lastly, we propose an LLM-based autograder to actually practically generate this information, and we demonstrate that reward models trained using these metrics are better than traditional reward models.

It’s worth mentioning that this is just our case study in which we are considering length as one feature and correctness as another. Longer outputs aren’t necessarily a bad thing—they are just features that are used by annotators to make judgements, which is why our fine-tuned LLMs optimize for them. However, we would also like for models to ideally place more weight on less apparent features, such as factuality, when learning from unreliable feedback. In the future, we hope to explore if our work will expand to other more general datasets that vary along different axes, such as HH-RLHF [4].

# Bibliography

- [1] Dario Amodei et al. *Concrete Problems in AI Safety*. arXiv:1606.06565 [cs]. July 2016. URL: <http://arxiv.org/abs/1606.06565> (visited on 09/27/2023).
- [2] Usman Anwar et al. *Foundational Challenges in Assuring Alignment and Safety of Large Language Models*. arXiv:2404.09932 [cs]. Apr. 2024. URL: <http://arxiv.org/abs/2404.09932> (visited on 05/21/2024).
- [3] Zinat Ara et al. “Closing the Knowledge Gap in Designing Data Annotation Interfaces for AI-powered Disaster Management Analytic Systems”. In: *Proceedings of the 29th International Conference on Intelligent User Interfaces*. 2024, pp. 405–418.
- [4] Yuntao Bai et al. *Constitutional AI: Harmlessness from AI Feedback*. arXiv:2212.08073 [cs]. Dec. 2022. URL: <http://arxiv.org/abs/2212.08073> (visited on 05/15/2024).
- [5] Yuntao Bai et al. *Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback*. arXiv:2204.05862 [cs]. Apr. 2022. URL: <http://arxiv.org/abs/2204.05862> (visited on 05/29/2023).
- [6] Peter Barnett et al. *Active Reward Learning from Multiple Teachers*. arXiv:2303.00894 [cs]. Mar. 2023. URL: <http://arxiv.org/abs/2303.00894> (visited on 05/16/2024).
- [7] Marcel Binz and Eric Schulz. *Turning large language models into cognitive models*. arXiv:2306.03917 [cs]. June 2023. URL: <http://arxiv.org/abs/2306.03917> (visited on 06/01/2024).
- [8] Samuel R. Bowman et al. *Measuring Progress on Scalable Oversight for Large Language Models*. arXiv:2211.03540 [cs]. Nov. 2022. URL: <http://arxiv.org/abs/2211.03540> (visited on 05/15/2024).
- [9] Ralph Allan Bradley and Milton E. Terry. “Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons”. In: *Biometrika* 39 (1952), p. 324. URL: <https://api.semanticscholar.org/CorpusID:125209808>.

- [10] Daniel Brown et al.  
“Safe Imitation Learning via Fast Bayesian Reward Inference from Preferences”. en.  
In: *Proceedings of the 37th International Conference on Machine Learning*.  
ISSN: 2640-3498. PMLR, Nov. 2020, pp. 1165–1177. URL:  
<https://proceedings.mlr.press/v119/brown20a.html> (visited on 10/17/2023).
- [11] Collin Burns et al.  
*Weak-to-Strong Generalization: Eliciting Strong Capabilities With Weak Supervision*.  
arXiv:2312.09390 [cs]. Dec. 2023. DOI: 10.48550/arXiv.2312.09390.  
URL: <http://arxiv.org/abs/2312.09390> (visited on 03/19/2024).
- [12] Erdem Bıyık et al.  
*Asking Easy Questions: A User-Friendly Approach to Active Reward Learning*.  
arXiv:1910.04365 [cs]. Oct. 2019. DOI: 10.48550/arXiv.1910.04365.  
URL: <http://arxiv.org/abs/1910.04365> (visited on 10/17/2023).
- [13] Erdem Bıyık et al. *Learning Reward Functions from Diverse Sources of Human Feedback: Optimally Integrating Demonstrations and Preferences*.  
arXiv:2006.14091 [cs]. Aug. 2020.  
URL: <http://arxiv.org/abs/2006.14091> (visited on 05/16/2024).
- [14] Clément L. Canonne. *A short note on an inequality between KL and TV*.  
arXiv:2202.07198 [math, stat]. Feb. 2022.  
URL: <http://arxiv.org/abs/2202.07198> (visited on 05/26/2023).
- [15] Micah Carroll et al. *AI alignment with changing and influenceable reward functions*.  
2024. URL: <https://arxiv.org/abs/2405.17713>.
- [16] Micah Carroll et al. *Characterizing Manipulation from AI Systems*.  
arXiv:2303.09387 [cs]. Oct. 2023.  
URL: <http://arxiv.org/abs/2303.09387> (visited on 05/16/2024).
- [17] Stephen Casper et al. *Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback*. arXiv:2307.15217 [cs]. Sept. 2023.  
URL: <http://arxiv.org/abs/2307.15217> (visited on 10/17/2023).
- [18] Lawrence Chan, Andrew Critch, and Anca Dragan.  
*Human irrationality: both bad and good for reward inference*. arXiv:2111.06956 [cs].  
Nov. 2021. URL: <http://arxiv.org/abs/2111.06956> (visited on 05/21/2024).
- [19] Letian Chen, Rohan Paleja, and Matthew Gombolay.  
*Learning from Suboptimal Demonstration via Self-Supervised Reward Regression*.  
arXiv:2010.11723 [cs]. Nov. 2020.  
URL: <http://arxiv.org/abs/2010.11723> (visited on 05/22/2024).
- [20] Lichang Chen et al. *ODIN: Disentangled Reward Mitigates Hacking in RLHF*.  
arXiv:2402.07319 [cs]. Feb. 2024.  
URL: <http://arxiv.org/abs/2402.07319> (visited on 05/09/2024).

- [21] Ching-An Cheng et al. *Adversarially Trained Actor Critic for Offline Reinforcement Learning*. arXiv:2202.02446 [cs]. July 2022. URL: <http://arxiv.org/abs/2202.02446> (visited on 12/29/2023).
- [22] Yinlam Chow et al. *Lyapunov-based Safe Policy Optimization for Continuous Control*. arXiv:1901.10031 [cs, stat]. Feb. 2019. URL: <http://arxiv.org/abs/1901.10031> (visited on 09/29/2023).
- [23] Paul Christiano, Buck Shlegeris, and Dario Amodei. *Supervising strong learners by amplifying weak experts*. arXiv:1810.08575 [cs, stat]. Oct. 2018. URL: <http://arxiv.org/abs/1810.08575> (visited on 05/15/2024).
- [24] Paul Christiano et al. *Deep reinforcement learning from human preferences*. arXiv:1706.03741 [cs, stat]. June 2017. URL: <http://arxiv.org/abs/1706.03741> (visited on 09/28/2023).
- [25] Peter Clark et al. *Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge*. arXiv:1803.05457 [cs]. Mar. 2018. URL: <http://arxiv.org/abs/1803.05457> (visited on 06/02/2024).
- [26] Sam Corbett-Davies et al. *Algorithmic decision making and the cost of fairness*. arXiv:1701.08230 [cs, stat]. June 2017. DOI: 10.1145/3097983.309809. URL: <http://arxiv.org/abs/1701.08230> (visited on 02/22/2024).
- [27] Jessica Dai and Eve Fleisig. *Mapping Social Choice Theory to RLHF*. arXiv:2404.13038 [cs]. Apr. 2024. URL: <http://arxiv.org/abs/2404.13038> (visited on 05/14/2024).
- [28] Gal Dalal et al. *Safe Exploration in Continuous Action Spaces*. arXiv:1801.08757 [cs]. Jan. 2018. URL: <http://arxiv.org/abs/1801.08757> (visited on 09/29/2023).
- [29] Oliver Daniels-Koch and Rachel Freedman. *The Expertise Problem: Learning from Specialized Feedback*. arXiv:2211.06519 [cs]. Nov. 2022. URL: <http://arxiv.org/abs/2211.06519> (visited on 05/16/2024).
- [30] Ian Fox et al. *Deep Reinforcement Learning for Closed-Loop Blood Glucose Control*. arXiv:2009.09051 [cs, stat]. Sept. 2020. URL: <http://arxiv.org/abs/2009.09051> (visited on 09/26/2023).
- [31] Matt Franchi et al. “Detecting disparities in police deployments using dashcam data”. In: *2023 ACM Conference on Fairness, Accountability, and Transparency*. arXiv:2305.15210 [cs]. June 2023, pp. 534–544. DOI: 10.1145/3593013.3594020. URL: <http://arxiv.org/abs/2305.15210> (visited on 02/22/2024).
- [32] Aaron French. “The Mandela Effect and New Memory”. In: *Correspondences: Journal for the Study of Esotericism* 6.2 (2018), pp. 201–233.

- [33] Leo Gao, John Schulman, and Jacob Hilton.  
*Scaling Laws for Reward Model Overoptimization*. arXiv:2210.10760 [cs, stat].  
Oct. 2022. URL: <http://arxiv.org/abs/2210.10760> (visited on 04/20/2023).
- [34] Gaurav R. Ghosal et al. *The Effect of Modeling Human Rationality Level on Learning Rewards from Multiple Feedback Types*. arXiv:2208.10687 [cs]. Mar. 2022.  
URL: <http://arxiv.org/abs/2208.10687> (visited on 02/29/2024).
- [35] Amelia Glaese et al.  
*Improving alignment of dialogue agents via targeted human judgements*.  
arXiv:2209.14375 [cs]. Sept. 2022.  
URL: <http://arxiv.org/abs/2209.14375> (visited on 09/25/2023).
- [36] Ian J. Goodfellow et al. *Generative Adversarial Networks*. arXiv:1406.2661 [cs, stat].  
June 2014. URL: <http://arxiv.org/abs/1406.2661> (visited on 05/28/2023).
- [37] C. A. E. Goodhart. “Problems of Monetary Management: The UK Experience”. en.  
In: *Monetary Theory and Practice: The UK Experience*. Ed. by C. A. E. Goodhart.  
London: Macmillan Education UK, 1984, pp. 91–121. ISBN: 978-1-349-17295-5.  
DOI: 10.1007/978-1-349-17295-5\_4.  
URL: [https://doi.org/10.1007/978-1-349-17295-5\\_4](https://doi.org/10.1007/978-1-349-17295-5_4) (visited on 09/29/2023).
- [38] Lewis D. Griffin et al. *Susceptibility to Influence of Large Language Models*.  
arXiv:2303.06074 [cs]. Mar. 2023.  
URL: <http://arxiv.org/abs/2303.06074> (visited on 05/15/2024).
- [39] Dylan Hadfield-Menell et al. *Inverse Reward Design*. arXiv:1711.02827 [cs]. 2017.  
URL: <http://arxiv.org/abs/1711.02827> (visited on 05/28/2023).
- [40] Peter Hase et al.  
*The Unreasonable Effectiveness of Easy Training Data for Hard Tasks*.  
arXiv:2401.06751 [cs]. Jan. 2024.  
URL: <http://arxiv.org/abs/2401.06751> (visited on 02/14/2024).
- [41] Elad Hazan et al. *Provably Efficient Maximum Entropy Exploration*.  
arXiv:1812.02690 [cs, stat]. Jan. 2019.  
URL: <http://arxiv.org/abs/1812.02690> (visited on 12/29/2023).
- [42] Haoyang He. *A Survey on Offline Model-Based Reinforcement Learning*.  
arXiv:2305.03360 [cs, eess]. May 2023.  
URL: <http://arxiv.org/abs/2305.03360> (visited on 12/29/2023).
- [43] Joey Hejna and Dorsa Sadigh.  
*Few-Shot Preference Learning for Human-in-the-Loop RL*. arXiv:2212.03363 [cs].  
Dec. 2022. URL: <http://arxiv.org/abs/2212.03363> (visited on 06/01/2024).



- [44] Jonathan Ho and Stefano Ermon. “Generative Adversarial Imitation Learning”. In: *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc., 2016.  
URL: [https://papers.nips.cc/paper\\_files/paper/2016/hash/cc7e2b878868cbae992d1fb743995d8f-Abstract.html](https://papers.nips.cc/paper_files/paper/2016/hash/cc7e2b878868cbae992d1fb743995d8f-Abstract.html) (visited on 04/17/2023).
- [45] Jiwoo Hong, Noah Lee, and James Thorne.  
*ORPO: Monolithic Preference Optimization without Reference Model*.  
arXiv:2403.07691 [cs]. Mar. 2024.  
URL: <http://arxiv.org/abs/2403.07691> (visited on 05/08/2024).
- [46] Joey Hong, Kush Bhatia, and Anca Dragan.  
*On the Sensitivity of Reward Inference to Misspecified Human Models*.  
arXiv:2212.04717 [cs]. Oct. 2023.  
URL: <http://arxiv.org/abs/2212.04717> (visited on 05/22/2024).
- [47] Sungsoo Ray Hong et al.  
“Disseminating Machine Learning to domain experts: Understanding challenges and opportunities in supporting a model building process”. In: *CHI 2019 Workshop, Emerging Perspectives in Human-Centered Machine Learning*. ACM. 2019.
- [48] Tom Hosking, Phil Blunsom, and Max Bartolo.  
*Human Feedback is not Gold Standard*. arXiv:2309.16349 [cs]. Jan. 2024.  
URL: <http://arxiv.org/abs/2309.16349> (visited on 02/29/2024).
- [49] Olivia Huang, Eve Fleisig, and Dan Klein.  
*Incorporating Worker Perspectives into MTurk Annotation Practices for NLP*.  
arXiv:2311.02802 [cs]. Nov. 2023.  
URL: <http://arxiv.org/abs/2311.02802> (visited on 05/14/2024).
- [50] Ferenc Huszár. *Variational Inference using Implicit Distributions*.  
arXiv:1702.08235 [cs, stat]. Feb. 2017.  
URL: <http://arxiv.org/abs/1702.08235> (visited on 05/27/2023).
- [51] Borja Ibarz et al.  
“Reward learning from human preferences and demonstrations in Atari”.  
In: (2018). Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.1811.06521.  
URL: <https://arxiv.org/abs/1811.06521> (visited on 05/13/2023).
- [52] Borja Ibarz et al.  
“Reward learning from human preferences and demonstrations in Atari”.  
In: *ArXiv* abs/1811.06521 (2018).  
URL: <https://api.semanticscholar.org/CorpusID:53424488>.
- [53] Geoffrey Irving, Paul Christiano, and Dario Amodei. *AI safety via debate*.  
arXiv:1805.00899 [cs, stat]. Oct. 2018.  
URL: <http://arxiv.org/abs/1805.00899> (visited on 05/15/2024).

- [54] Hong Jun Jeon, Smitha Milli, and Anca D. Dragan.  
*Reward-rational (implicit) choice: A unifying formalism for reward learning.*  
arXiv:2002.04833 [cs]. Dec. 2020.  
URL: <http://arxiv.org/abs/2002.04833> (visited on 02/01/2024).
- [55] Hong Jun Jeon, Smitha Milli, and Anca D. Dragan.  
“Reward-rational (implicit) choice: A unifying formalism for reward learning”.  
In: *ArXiv abs/2002.04833* (2020).  
URL: <https://api.semanticscholar.org/CorpusID:211083001>.
- [56] Bingyi Kang, Zequn Jie, and Jiashi Feng.  
“Policy Optimization with Demonstrations”.  
In: *International Conference on Machine Learning*. 2018.  
URL: <https://api.semanticscholar.org/CorpusID:51875782>.
- [57] Jacek Karwowski et al. *Goodhart’s Law in Reinforcement Learning.*  
arXiv:2310.09144 [cs]. Oct. 2023.  
URL: <http://arxiv.org/abs/2310.09144> (visited on 02/06/2024).
- [58] Taylor A. Kessler Faulkner, Elaine Schaertl Short, and Andrea L. Thomaz.  
“Interactive Reinforcement Learning with Inaccurate Feedback”.  
In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020,  
pp. 7498–7504. DOI: 10.1109/ICRA40945.2020.9197219.
- [59] Robert Kirk et al.  
*Understanding the Effects of RLHF on LLM Generalisation and Diversity.*  
arXiv:2310.06452 [cs]. Feb. 2024.  
URL: <http://arxiv.org/abs/2310.06452> (visited on 05/09/2024).
- [60] Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan.  
*The Challenge of Understanding What Users Want: Inconsistent Preferences and Engagement Optimization.* arXiv:2202.11776 [cs]. Oct. 2023.  
URL: <http://arxiv.org/abs/2202.11776> (visited on 02/22/2024).
- [61] W. Bradley Knox et al. *Reward (Mis)design for Autonomous Driving.*  
arXiv:2104.13906 [cs]. Mar. 2022.  
URL: <http://arxiv.org/abs/2104.13906> (visited on 09/26/2023).
- [62] Varun Kompella et al.  
*Reinforcement Learning for Optimization of COVID-19 Mitigation policies.*  
arXiv:2010.10560 [cs]. Oct. 2020.  
URL: <http://arxiv.org/abs/2010.10560> (visited on 09/26/2023).
- [63] Tomasz Korbak, Ethan Perez, and Christopher L. Buckley.  
*RL with KL penalties is better viewed as Bayesian inference.*  
arXiv:2205.11275 [cs, stat]. Oct. 2022.  
URL: <http://arxiv.org/abs/2205.11275> (visited on 05/29/2023).

- [64] Victoria Krakovna. *Classifying specification problems as variants of Goodhart’s Law*. en. Aug. 2019. URL: <https://vkrakovna.wordpress.com/2019/08/19/classifying-specification-problems-as-variants-of-goodharts-law/> (visited on 09/28/2023).
- [65] Victoria Krakovna. *Specification gaming examples in AI*. en. Apr. 2018. URL: <https://vkrakovna.wordpress.com/2018/04/02/specification-gaming-examples-in-ai/> (visited on 05/13/2023).
- [66] Victoria Krakovna et al. *Penalizing side effects using stepwise relative reachability*. arXiv:1806.01186 [cs, stat]. Mar. 2019. DOI: 10.48550/arXiv.1806.01186. URL: <http://arxiv.org/abs/1806.01186> (visited on 05/13/2023).
- [67] Julia Kreutzer, Joshua Uyheng, and Stefan Riezler. *Reliability and Learnability of Human Bandit Feedback for Sequence-to-Sequence Reinforcement Learning*. arXiv:1805.10627 [cs, stat]. Dec. 2018. URL: <http://arxiv.org/abs/1805.10627> (visited on 05/22/2024).
- [68] Cassidy Laidlaw, Stuart Russell, and Anca Dragan. *Bridging RL Theory and Practice with the Effective Horizon*. arXiv:2304.09853 [cs, stat]. Apr. 2023. URL: <http://arxiv.org/abs/2304.09853> (visited on 09/25/2023).
- [69] Nathan Lambert, Thomas Krendl Gilbert, and Tom Zick. *The History and Risks of Reinforcement Learning and Human Feedback*. arXiv:2310.13595 [cs]. Nov. 2023. URL: <http://arxiv.org/abs/2310.13595> (visited on 05/22/2024).
- [70] Leon Lang et al. *When Your AIs Deceive You: Challenges with Partial Observability of Human Evaluators in Reward Learning*. arXiv:2402.17747 [cs, stat]. Mar. 2024. URL: <http://arxiv.org/abs/2402.17747> (visited on 05/14/2024).
- [71] Harrison Lee et al. *RLAIF: Scaling Reinforcement Learning from Human Feedback with AI Feedback*. arXiv:2309.00267 [cs]. Nov. 2023. URL: <http://arxiv.org/abs/2309.00267> (visited on 05/15/2024).
- [72] Jongmin Lee et al. *COptiDICE: Offline Constrained Reinforcement Learning via Stationary Distribution Correction Estimation*. arXiv:2204.08957 [cs]. Apr. 2022. URL: <http://arxiv.org/abs/2204.08957> (visited on 12/29/2023).
- [73] Jongmin Lee et al. *OptiDICE: Offline Policy Optimization via Stationary Distribution Correction Estimation*. arXiv:2106.10783 [cs]. June 2021. URL: <http://arxiv.org/abs/2106.10783> (visited on 03/19/2024).

- [74] Kimin Lee, Laura Smith, and Pieter Abbeel.  
*PEBBLE: Feedback-Efficient Interactive Reinforcement Learning via Relabeling Experience and Unsupervised Pre-training.* arXiv:2106.05091 [cs]. June 2021.  
URL: <http://arxiv.org/abs/2106.05091> (visited on 05/13/2023).
- [75] Kimin Lee et al. *B-Pref: Benchmarking Preference-Based Reinforcement Learning.* arXiv:2111.03026 [cs]. Nov. 2020.  
URL: <http://arxiv.org/abs/2111.03026> (visited on 05/16/2024).
- [76] Lisa Lee et al. *Efficient Exploration via State Marginal Matching.* arXiv:1906.05274 [cs, stat]. Feb. 2020.  
URL: <http://arxiv.org/abs/1906.05274> (visited on 12/29/2023).
- [77] Jan Leike et al. *AI Safety Gridworlds.* arXiv:1711.09883 [cs]. Nov. 2017.  
URL: <http://arxiv.org/abs/1711.09883> (visited on 05/26/2023).
- [78] Jan Leike et al. *Scalable agent alignment via reward modeling: a research direction.* arXiv:1811.07871 [cs, stat]. Nov. 2018. DOI: 10.48550/arXiv.1811.07871.  
URL: <http://arxiv.org/abs/1811.07871> (visited on 05/13/2023).
- [79] Aaron J. Li, Satyapriya Krishna, and Himabindu Lakkaraju.  
*More RLHF, More Trust? On The Impact of Human Preference Alignment On Language Model Trustworthiness.* arXiv:2404.18870 [cs]. Apr. 2024.  
URL: <http://arxiv.org/abs/2404.18870> (visited on 05/09/2024).
- [80] Anqi Li et al. *Survival Instinct in Offline Reinforcement Learning.* arXiv:2306.03286 [cs]. Nov. 2023.  
URL: <http://arxiv.org/abs/2306.03286> (visited on 12/29/2023).
- [81] Eric Liang et al. *RLlib: Abstractions for Distributed Reinforcement Learning.* arXiv:1712.09381 [cs]. June 2018.  
URL: <http://arxiv.org/abs/1712.09381> (visited on 05/29/2023).
- [82] Stephanie Lin, Jacob Hilton, and Owain Evans.  
*TruthfulQA: Measuring How Models Mimic Human Falsehoods.* arXiv:2109.07958 [cs]. May 2022.  
URL: <http://arxiv.org/abs/2109.07958> (visited on 06/01/2024).
- [83] David Lindner and Mennatallah El-Assady.  
*Humans are not Boltzmann Distributions: Challenges and Opportunities for Modelling Human Feedback and Interaction in Reinforcement Learning.* arXiv:2206.13316 [cs, stat]. June 2022.  
URL: <http://arxiv.org/abs/2206.13316> (visited on 05/21/2024).
- [84] R. Duncan Luce. *Individual Choice Behavior: A Theoretical analysis.* New York, NY, USA: Wiley, 1959.

- [85] Kristian Lum and William Isaac. “To Predict and Serve?”  
In: *Significance* 13.5 (Oct. 2016). eprint:  
[https://academic.oup.com/jrssig/article-pdf/13/5/14/49106469/sign\\_13\\_5\\_14.pdf](https://academic.oup.com/jrssig/article-pdf/13/5/14/49106469/sign_13_5_14.pdf),  
pp. 14–19. ISSN: 1740-9705. DOI: 10.1111/j.1740-9713.2016.00960.x.  
URL: <https://doi.org/10.1111/j.1740-9713.2016.00960.x>.
- [86] Chiara Dalla Man et al. “The UVA/PADOVA Type 1 Diabetes Simulator”.  
In: *Journal of Diabetes Science and Technology* 8.1 (Jan. 2014), pp. 26–34.  
ISSN: 1932-2968. DOI: 10.1177/1932296813514502.  
URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4454102/> (visited on 09/26/2023).
- [87] Debmalya Mandal, Stelios Triantafyllou, and Goran Radanovic.  
*Performative Reinforcement Learning*. arXiv:2207.00046 [cs]. Feb. 2023.  
URL: <http://arxiv.org/abs/2207.00046> (visited on 12/29/2023).
- [88] Lev McKinney et al. *On The Fragility of Learned Reward Functions*.  
arXiv:2301.03652 [cs]. Jan. 2023.  
URL: <http://arxiv.org/abs/2301.03652> (visited on 05/24/2023).
- [89] Smitha Milli, Luca Belli, and Moritz Hardt.  
“From Optimizing Engagement to Measuring Value”. In: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*.  
arXiv:2008.12623 [cs, stat]. Mar. 2021, pp. 714–722.  
DOI: 10.1145/3442188.3445933.  
URL: <http://arxiv.org/abs/2008.12623> (visited on 02/22/2024).
- [90] Samuel Mwalili et al. “SEIR model for COVID-19 dynamics incorporating the environment and social distancing”. In: *BMC Research Notes* 13 (July 2020), p. 352.  
ISSN: 1756-0500. DOI: 10.1186/s13104-020-05192-1.  
URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7376536/> (visited on 09/26/2023).
- [91] Ofir Nachum et al. *AlgaeDICE: Policy Gradient from Arbitrary Experience*.  
arXiv:1912.02074 [cs]. Dec. 2019.  
URL: <http://arxiv.org/abs/1912.02074> (visited on 03/19/2024).
- [92] Alexander Nedergaard and Matthew Cook. *k-Means Maximum Entropy Exploration*.  
arXiv:2205.15623 [cs]. Nov. 2023.  
URL: <http://arxiv.org/abs/2205.15623> (visited on 12/29/2023).
- [93] Richard Ngo, Lawrence Chan, and Sören Mindermann.  
*The alignment problem from a deep learning perspective*. arXiv:2209.00626 [cs].  
Sept. 2023. URL: <http://arxiv.org/abs/2209.00626> (visited on 09/29/2023).

- [94] Ziad Obermeyer et al. “Dissecting racial bias in an algorithm used to manage the health of populations”. en. In: *Science* 366.6464 (Oct. 2019), pp. 447–453. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.aax2342. URL: <https://www.science.org/doi/10.1126/science.aax2342> (visited on 02/22/2024).
- [95] OpenAI et al. *GPT-4 Technical Report*. arXiv:2303.08774 [cs]. Mar. 2024. URL: <http://arxiv.org/abs/2303.08774> (visited on 05/07/2024).
- [96] Peter Organisciak et al. “Evaluating rater quality and rating difficulty in online annotation activities”. en. In: *Proceedings of the American Society for Information Science and Technology* 49.1 (Jan. 2012), pp. 1–10. ISSN: 0044-7870, 1550-8390. DOI: 10.1002/meet.14504901166. URL: <https://asistdl.onlinelibrary.wiley.com/doi/10.1002/meet.14504901166> (visited on 05/14/2024).
- [97] Long Ouyang et al. “Training language models to follow instructions with human feedback”. In: (2022). DOI: 10.48550/ARXIV.2203.02155. URL: <https://arxiv.org/abs/2203.02155> (visited on 05/13/2023).
- [98] Alexander Pan, Kush Bhatia, and Jacob Steinhardt. *The Effects of Reward Misspecification: Mapping and Mitigating Misaligned Models*. arXiv:2201.03544 [cs, stat]. Feb. 2022. URL: <http://arxiv.org/abs/2201.03544> (visited on 05/04/2023).
- [99] Rahul Pandey et al. “Modeling and mitigating human annotation errors to design efficient stream processing systems with human-in-the-loop machine learning”. In: *International Journal of Human-Computer Studies* 160 (Apr. 2022). arXiv:2007.03177 [cs], p. 102772. ISSN: 10715819. DOI: 10.1016/j.ijhcs.2022.102772. URL: <http://arxiv.org/abs/2007.03177> (visited on 05/14/2024).
- [100] Ryan Park et al. *Disentangling Length from Quality in Direct Preference Optimization*. arXiv:2403.19159 [cs]. Mar. 2024. URL: <http://arxiv.org/abs/2403.19159> (visited on 04/04/2024).
- [101] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. arXiv:1912.01703 [cs, stat]. Dec. 2019. URL: <http://arxiv.org/abs/1912.01703> (visited on 05/29/2023).

- [102] Ethan Perez et al.  
*Discovering Language Model Behaviors with Model-Written Evaluations*.  
arXiv:2212.09251 [cs]. Dec. 2022.  
URL: <http://arxiv.org/abs/2212.09251> (visited on 05/09/2024).
- [103] Emma Pierson et al. “An algorithmic approach to reducing unexplained pain disparities in underserved populations”.  
In: *Nature Medicine* 27 (2021), pp. 136–140.  
URL: <https://api.semanticscholar.org/CorpusID:231598658>.
- [104] Rafael Rafailov et al.  
*Direct Preference Optimization: Your Language Model is Secretly a Reward Model*.  
arXiv:2305.18290 [cs]. May 2023.  
URL: <http://arxiv.org/abs/2305.18290> (visited on 11/07/2023).
- [105] A. Rajkumar and Shivani Agarwal. “A Statistical Convergence Perspective of Algorithms for Rank Aggregation from Pairwise Data”.  
In: *International Conference on Machine Learning*. 2014.  
URL: <https://api.semanticscholar.org/CorpusID:13910694>.
- [106] Paria Rashidinejad et al. *Bridging Offline Reinforcement Learning and Imitation Learning: A Tale of Pessimism*. arXiv:2103.12021 [cs, math, stat]. July 2023.  
URL: <http://arxiv.org/abs/2103.12021> (visited on 12/29/2023).
- [107] Siddharth Reddy et al.  
*Learning Human Objectives by Evaluating Hypothetical Behavior*.  
arXiv:1912.05652 [cs, stat]. Mar. 2020.  
URL: <http://arxiv.org/abs/1912.05652> (visited on 02/01/2024).
- [108] Julien Roy et al.  
*Direct Behavior Specification via Constrained Reinforcement Learning*.  
arXiv:2112.12228 [cs]. June 2022.  
URL: <http://arxiv.org/abs/2112.12228> (visited on 09/29/2023).
- [109] Stuart J. Russell.  
*Human compatible: artificial intelligence and the problem of control*.  
New York?: Viking, 2019. ISBN: 978-0-525-55861-3.
- [110] Stuart J. Russell, Peter Norvig, and Ernest Davis.  
*Artificial intelligence: a modern approach*. 3rd ed.  
Prentice Hall series in artificial intelligence.  
Upper Saddle River: Prentice Hall, 2010. ISBN: 978-0-13-604259-4.
- [111] Dorsa Sadigh et al. “Active Preference-Based Learning of Reward Functions”.  
In: *Robotics: Science and Systems XIII*.  
Robotics: Science and Systems Foundation, July 2017. ISBN: 978-0-9923747-3-0.  
DOI: 10.15607/RSS.2017.XIII.053. URL:  
<http://www.roboticsproceedings.org/rss13/p53.pdf> (visited on 09/28/2023).

- [112] William Saunders et al. *Self-critiquing models for assisting human evaluators*. arXiv:2206.05802 [cs]. June 2022.  
URL: <http://arxiv.org/abs/2206.05802> (visited on 10/17/2023).
- [113] John Schulman et al. *Proximal Policy Optimization Algorithms*. arXiv:1707.06347 [cs]. Aug. 2017.  
URL: <http://arxiv.org/abs/1707.06347> (visited on 05/28/2023).
- [114] Avi Schwarzschild et al. *Can You Learn an Algorithm? Generalizing from Easy to Hard Problems with Recurrent Networks*. arXiv:2106.04537 [cs]. Nov. 2021.  
URL: <http://arxiv.org/abs/2106.04537> (visited on 05/15/2024).
- [115] Rohin Shah et al. *On the Feasibility of Learning, Rather than Assuming, Human Biases for Reward Inference*. arXiv:1906.09624 [cs, stat]. June 2019.  
URL: <http://arxiv.org/abs/1906.09624> (visited on 05/16/2024).
- [116] Archit Sharma et al.  
*A Critical Evaluation of AI Feedback for Aligning Large Language Models*. arXiv:2402.12366 [cs]. Feb. 2024.  
URL: <http://arxiv.org/abs/2402.12366> (visited on 05/21/2024).
- [117] Mrinank Sharma et al. *Towards Understanding Sycophancy in Language Models*. arXiv:2310.13548 [cs, stat]. Oct. 2023. DOI: 10.48550/arXiv.2310.13548.  
URL: <http://arxiv.org/abs/2310.13548> (visited on 04/08/2024).
- [118] Harshit Sikchi et al.  
*Dual RL: Unification and New Methods for Reinforcement and Imitation Learning*. arXiv:2302.08560 [cs]. June 2023. DOI: 10.48550/arXiv.2302.08560.  
URL: <http://arxiv.org/abs/2302.08560> (visited on 01/17/2024).
- [119] Prasann Singhal et al.  
*A Long Way to Go: Investigating Length Correlations in RLHF*. arXiv:2310.03716 [cs]. Oct. 2023.  
URL: <http://arxiv.org/abs/2310.03716> (visited on 05/09/2024).
- [120] Joar Skalse et al. *Defining and Characterizing Reward Hacking*. arXiv:2209.13085 [cs, stat]. Sept. 2022.  
URL: <http://arxiv.org/abs/2209.13085> (visited on 04/20/2023).
- [121] Garry M. Steil. “Algorithms for a Closed-Loop Artificial Pancreas: The Case for Proportional-Integral-Derivative Control”.  
In: *Journal of Diabetes Science and Technology* 7.6 (Nov. 2013), pp. 1621–1631. ISSN: 1932-2968.  
URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3876341/> (visited on 09/26/2023).
- [122] Nisan Stiennon et al. *Learning to summarize from human feedback*. arXiv:2009.01325 [cs]. Sept. 2020. DOI: 10.48550/arXiv.2009.01325.  
URL: <http://arxiv.org/abs/2009.01325> (visited on 05/24/2023).



- [123] Nisan Stiennon et al. *Learning to summarize from human feedback*. arXiv:2009.01325 [cs]. Feb. 2022.  
URL: <http://arxiv.org/abs/2009.01325> (visited on 05/15/2024).
- [124] Zhiqing Sun et al.  
*Easy-to-Hard Generalization: Scalable Alignment Beyond Human Supervision*. arXiv:2403.09472 [cs]. Mar. 2024.  
URL: <http://arxiv.org/abs/2403.09472> (visited on 05/15/2024).
- [125] Shuchang Tao et al.  
*When to Trust LLMs: Aligning Confidence with Response Quality*. arXiv:2404.17287 [cs]. Apr. 2024.  
URL: <http://arxiv.org/abs/2404.17287> (visited on 05/15/2024).
- [126] Jessica Taylor.  
“Quantilizers: A Safer Alternative to Maximizers for Limited Optimization”. In: Mar. 2016. URL: <https://www.semanticscholar.org/paper/Quantilizers%3A-A-Safer-Alternative-to-Maximizers-for-Taylor/4e8ff3b4069a12a00196d62925bab8add7389742> (visited on 05/24/2023).
- [127] Jessica Taylor et al. “Alignment for Advanced Machine Learning Systems”. en. In: *Ethics of Artificial Intelligence*. Google-Books-ID: 1yT3DwAAQBAJ. Oxford University Press, Aug. 2020. ISBN: 978-0-19-090505-7.
- [128] Luke Thorburn. *What does it mean to give someone what they want? the nature of preferences in Recommender Systems*. 2022.  
URL: <https://medium.com/understanding-recommenders/what-does-it-mean-to-give-someone-what-they-want-the-nature-of-preferences-in-recommender-systems-82b5a1559157>.
- [129] Jeremy Tien et al. *Causal Confusion and Reward Misidentification in Preference-Based Reward Learning*. arXiv:2204.06601 [cs]. Mar. 2023.  
DOI: 10.48550/arXiv.2204.06601.  
URL: <http://arxiv.org/abs/2204.06601> (visited on 09/29/2023).
- [130] Hugo Touvron et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. arXiv:2307.09288 [cs]. July 2023.  
URL: <http://arxiv.org/abs/2307.09288> (visited on 05/08/2024).
- [131] Martin Treiber, Ansgar Hennecke, and Dirk Helbing.  
“Congested Traffic States in Empirical Observations and Microscopic Simulations”. In: *Physical Review E* 62.2 (Aug. 2000). arXiv:cond-mat/0002177, pp. 1805–1824. ISSN: 1063-651X, 1095-3787. DOI: 10.1103/PhysRevE.62.1805.  
URL: <http://arxiv.org/abs/cond-mat/0002177> (visited on 05/26/2023).

- [132] A. M. Turner et al. “Optimal Policies Tend To Seek Power”. In: Dec. 2019. URL: <https://www.semanticscholar.org/paper/Optimal-Policies-Tend-To-Seek-Power-Turner-Smith/46d4452eb041e33f1e58eab64ec8cf5af534b6ff> (visited on 05/13/2023).
- [133] Alexander Matt Turner, Neale Ratzlaff, and Prasad Tadepalli. *Avoiding Side Effects in Complex Environments*. arXiv:2006.06547 [cs]. Oct. 2020. URL: <http://arxiv.org/abs/2006.06547> (visited on 02/02/2024).
- [134] Ikechukwu Uchendu et al. *Jump-Start Reinforcement Learning*. arXiv:2204.02372 [cs]. July 2023. URL: <http://arxiv.org/abs/2204.02372> (visited on 09/25/2023).
- [135] Nino Vieillard et al. *Leverage the Average: an Analysis of KL Regularization in RL*. arXiv:2003.14089 [cs, stat]. Jan. 2021. URL: <http://arxiv.org/abs/2003.14089> (visited on 05/28/2023).
- [136] Jason Wei et al. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. arXiv:2201.11903 [cs]. Jan. 2023. URL: <http://arxiv.org/abs/2201.11903> (visited on 05/31/2024).
- [137] Christian Wirth et al. “A Survey of Preference-Based Reinforcement Learning Methods”. In: *J. Mach. Learn. Res.* 18 (2017), 136:1–136:46. URL: <https://api.semanticscholar.org/CorpusID:703818>.
- [138] Cathy Wu et al. “Flow: A Modular Learning Framework for Mixed Autonomy Traffic”. In: *IEEE Transactions on Robotics* 38.2 (Apr. 2022). arXiv:1710.05465 [cs], pp. 1270–1286. ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TR0.2021.3087314. URL: <http://arxiv.org/abs/1710.05465> (visited on 05/26/2023).
- [139] Jeff Wu et al. *Recursively Summarizing Books with Human Feedback*. arXiv:2109.10862 [cs]. Sept. 2021. URL: <http://arxiv.org/abs/2109.10862> (visited on 05/15/2024).
- [140] Tengyang Xie et al. *Bellman-consistent Pessimism for Offline Reinforcement Learning*. arXiv:2106.06926 [cs, stat]. Oct. 2023. URL: <http://arxiv.org/abs/2106.06926> (visited on 12/29/2023).
- [141] Tian Xu, Ziniu Li, and Yang Yu. *Error Bounds of Imitating Policies and Environments*. arXiv:2010.11876 [cs]. Oct. 2020. URL: <http://arxiv.org/abs/2010.11876> (visited on 11/13/2023).
- [142] Shentao Yang et al. *Regularizing a Model-based Policy Stationary Distribution to Stabilize Offline Reinforcement Learning*. arXiv:2206.07166 [cs]. June 2022. URL: <http://arxiv.org/abs/2206.07166> (visited on 12/29/2023).

- [143] Tsung-Yen Yang et al.  
*Accelerating Safe Reinforcement Learning with Constraint-mismatched Policies*.  
arXiv:2006.11645 [cs, stat]. July 2021.  
URL: <http://arxiv.org/abs/2006.11645> (visited on 05/28/2023).
- [144] Ruiyi Zhang et al. *GenDICE: Generalized Offline Estimation of Stationary Values*.  
arXiv:2002.09072 [cs, stat]. Feb. 2020. DOI: 10.48550/arXiv.2002.09072.  
URL: <http://arxiv.org/abs/2002.09072> (visited on 05/22/2024).
- [145] Yiming Zhang, Quan Vuong, and Keith W. Ross.  
*First Order Constrained Optimization in Policy Space*. arXiv:2002.06506 [cs, stat].  
Oct. 2020. URL: <http://arxiv.org/abs/2002.06506> (visited on 09/29/2023).
- [146] Brian D. Ziebart, J. Andrew Bagnell, and Anind K. Dey.  
“Modeling Interaction via the Principle of Maximum Causal Entropy”.  
In: *International Conference on Machine Learning*. 2010.  
URL: <https://api.semanticscholar.org/CorpusID:5884863>.

# Appendix A

## Project 1: Reward Hacking Mitigation

### A.1 Proofs

#### Proof of Proposition 2.3.1

**Proposition 2.3.1.** *Fix  $c_1 > 0$  and  $\delta > 0$  arbitrarily small, and  $c_2 \geq 0$  arbitrarily large. Then there is an MDP, true reward function  $R$ , and safe policy  $\pi_{\text{safe}}$  where both of the following hold:*

1. *There is a policy  $\pi$  where the action distribution KL divergence satisfies*

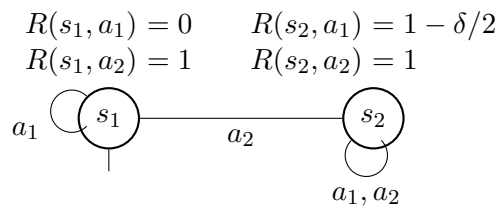
$$(1 - \gamma) \mathbb{E}_{\pi} \left[ \sum_{t=0}^{\infty} \gamma^t D_{KL}(\pi(\cdot | s_t) \| \pi_{\text{safe}}(\cdot | s_t)) \right] \leq c_1$$

*but  $J(\pi_{\text{safe}}, R) - J(\pi, R) \geq 1 - \delta$ .*

2. *Any optimal policy  $\pi^* \in \arg \max_{\pi} J(\pi, R)$  satisfies*

$$(1 - \gamma) \mathbb{E}_{\pi^*} \left[ \sum_{t=0}^{\infty} \gamma^t D_{KL}(\pi^*(\cdot | s_t) \| \pi_{\text{safe}}(\cdot | s_t)) \right] \geq c_2.$$

*Proof.* We assume that  $\delta < 1$ , since otherwise letting  $\pi = \pi_{\text{safe}}$  trivially satisfies the first part of the proposition. Consider the following MDP, similar to the one shown in Figure 2.1:



In this MDP,  $\mathcal{S} = \{s_1, s_2\}$ ,  $\mathcal{A} = \{a_1, a_2\}$ , and the transition probabilities and reward function are defined by

$$\begin{aligned} p(s_1 | s_1, a_1) &= 1 & p(s_2 | s_1, a_2) &= 1 \\ p(s_2 | s_2, a_1) &= 1 & p(s_2 | s_2, a_2) &= 1 \\ R(s_1, a_1) &= 0 & R(s_1, a_2) &= 1 \\ R(s_2, a_1) &= 1 - \delta/2 & R(s_2, a_2) &= 1. \end{aligned}$$

The initial state is always  $s_1$ . Thus, the agent stays in state  $s_1$  and receives no reward until after it takes action  $a_2$ , at which point it transitions to  $s_2$  and receives 1 or  $1 - \delta/2$  reward per timestep. Define for any  $(p, q) \in [0, 1]^2$  a policy  $\pi_{(p,q)}$ :

$$\pi_{(p,q)}(a_2 | s_1) = p \quad \pi_{(p,q)}(a_2 | s_2) = q.$$

We will prove the proposition using

$$\begin{aligned} \gamma &= 1 - \frac{\delta}{2}(1 - e^{-c_1}) \\ \pi_{\text{safe}} &= \pi_{(p,q)} \quad \text{where } p = 2(1 - \gamma)/\delta \quad \text{and } q = \exp\{-c_2/\gamma\} \\ \pi &= \pi_{(0,0)} \\ \pi^* &= \pi_{(1,1)}. \end{aligned}$$

Note the following:

- $\pi^*$  is the unique optimal policy:  $J(\pi^*, R) = 1$  and for any other policy  $\pi$ ,  $J(\pi, R) < 1$ .
- $\gamma \in [0, 1]$ :  $c_1 > 0$  and thus  $1 - e^{-c_1} > 0$ , and  $\delta < 1$ .
- $p \in [0, 1]$ : since  $\gamma < 1 - \delta/2$ , we have  $p < 1$ .
- $q \in [0, 1]$ : since  $c_2 \geq 0$ , we have  $q \leq 1$ .

To start, we need to show that

$$(1 - \gamma) \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t D_{\text{KL}}(\pi(\cdot | s_t) \| \pi_{\text{safe}}(\cdot | s_t)) \right] \leq c_1. \quad (\text{A.1})$$

Since  $\pi$  always stays at  $s_1$ , we can rewrite the LHS of (A.1) as

$$\begin{aligned} (1 - \gamma) \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t D_{\text{KL}}(\pi(\cdot | s_t) \| \pi_{\text{safe}}(\cdot | s_t)) \right] &= D_{\text{KL}}(\pi(\cdot | s_1) \| \pi_{\text{safe}}(\cdot | s_1)) \\ &= \pi(a_1 | s_1) \log \frac{\pi(a_1 | s_1)}{\pi_{\text{safe}}(a_1 | s_1)} + \pi(a_2 | s_1) \log \frac{\pi(a_2 | s_1)}{\pi_{\text{safe}}(a_2 | s_1)} \\ &= \log \frac{1}{1 - p} \\ &= \log \frac{1}{e^{-c_1}} \\ &= c_1, \end{aligned}$$

which proves (A.1).

Next, we need to show that  $J(\pi_{\text{safe}}, R) - J(\pi, R) \geq 1 - \delta$ . Clearly,  $J(\pi, R) = 0$ . We can bound  $J(\pi_{\text{safe}}, R)$  as

$$\begin{aligned}
J(\pi_{\text{safe}}, R) &= (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \left[ \mathbb{P}_{\pi}(s_t = s_1)p + \mathbb{P}_{\pi}(s_t = s_2)(q + (1 - q)(1 - \delta/2)) \right] \\
&\geq (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t (1 - \delta/2) \left[ \mathbb{P}_{\pi}(s_t = s_1)p + \mathbb{P}_{\pi}(s_t = s_2) \right] \\
&= (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t (1 - \delta/2) \mathbb{P}_{\pi}(\exists t' \leq t \text{ s.t. } a_{t'} = a_2) \\
&= (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t (1 - \delta/2) \left( 1 - (1 - p)^{t+1} \right) \\
&= (1 - \gamma)(1 - \delta/2)(1 - p) \sum_{t=0}^{\infty} \gamma^t \left( \frac{1}{1 - p} - (1 - p)^t \right) \\
&= (1 - \gamma)(1 - \delta/2)(1 - p) \left( \frac{1}{(1 - p)(1 - \gamma)} - \frac{1}{1 - \gamma(1 - p)} \right) \\
&= (1 - \delta/2) \frac{p}{1 - \gamma(1 - p)}.
\end{aligned}$$

Plugging in  $p = 2(1 - \gamma)/\delta$  gives

$$\begin{aligned}
J(\pi_{\text{safe}}, R) &\geq (1 - \delta/2) \frac{2(1 - \gamma)/\delta}{1 - \gamma(1 - 2(1 - \gamma)/\delta)} \\
&= \frac{1 - \delta/2}{\gamma + \delta/2} \\
&\stackrel{(i)}{\geq} (1 - \delta/2)(2 - \gamma - \delta/2) \\
&\geq (1 - \delta/2)(1 - \delta/2) \\
&\geq 1 - \delta,
\end{aligned}$$

which proves  $J(\pi_{\text{safe}}, R) - J(\pi, R) \geq 1 - \delta$  as desired. (i) uses the fact that  $1/x \geq 2 - x$  for  $x > 0$ .

All that remains to be shown is that

$$(1 - \gamma) \mathbb{E}_{\pi^*} \left[ \sum_{t=0}^{\infty} \gamma^t D_{\text{KL}}(\pi^*(\cdot | s_t) \| \pi_{\text{safe}}(\cdot | s_t)) \right] \geq c_2. \quad (\text{A.2})$$

We can bound the LHS of (A.2) based on only the KL divergence at  $s_2$ :

$$(1 - \gamma) \mathbb{E}_{\pi^*} \left[ \sum_{t=0}^{\infty} \gamma^t D_{\text{KL}}(\pi^*(\cdot | s_t) \| \pi_{\text{safe}}(\cdot | s_t)) \right] \geq (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t D_{\text{KL}}(\pi^*(\cdot | s_2) \| \pi_{\text{safe}}(\cdot | s_2)) \mathbb{P}_{\pi^*}(s_t = s_2).$$

Since  $\pi^*$  always takes action  $a_2$ , we know that  $\mathbb{P}_{\pi^*}(s_t = s_2) = \mathbf{1}\{t \geq 1\}$ . Thus, we can continue the bound as

$$\begin{aligned}
&\geq (1 - \gamma) \sum_{t=1}^{\infty} \gamma^t D_{\text{KL}}(\pi^*(\cdot | s_2) \| \pi_{\text{safe}}(\cdot | s_2)) \\
&= \gamma D_{\text{KL}}(\pi^*(\cdot | s_2) \| \pi_{\text{safe}}(\cdot | s_2)) \\
&= \gamma \left[ \pi^*(a_1 | s_2) \log \frac{\pi^*(a_1 | s_2)}{\pi_{\text{safe}}(a_1 | s_2)} + \pi^*(a_2 | s_2) \log \frac{\pi^*(a_2 | s_2)}{\pi_{\text{safe}}(a_2 | s_2)} \right] \\
&= \gamma \log \frac{1}{q} \\
&= c_2
\end{aligned}$$

by the definition of  $q$ . This proves (A.2) and completes the proof.  $\square$

### Proof of Proposition 2.3.2

We first prove another useful proposition:

**Proposition A.1.1.** *The return of a policy  $\pi$  under a reward function  $R$  is given by*

$$J(\pi, R) = \sum_{(s,a) \in \mathcal{S} \times \mathcal{A}} \mu_{\pi}(s, a) R(s, a).$$

*Proof.* Applying the definitions of return and occupancy measure, we have

$$\begin{aligned}
J(\pi, R) &= (1 - \gamma) \mathbb{E}_{\pi} \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right] \\
&= (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \sum_{(s,a) \in \mathcal{S} \times \mathcal{A}} R(s, a) \mathbb{P}_{\pi}(s_t = s \wedge a_t = a) \\
&= (1 - \gamma) \sum_{(s,a) \in \mathcal{S} \times \mathcal{A}} R(s, a) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}_{\pi}(s_t = s \wedge a_t = a) \\
&= \sum_{(s,a) \in \mathcal{S} \times \mathcal{A}} R(s, a) (1 - \gamma) \mathbb{E}_{\pi} \left[ \sum_{t=0}^{\infty} \gamma^t \mathbf{1}\{s_t = s \wedge a_t = a\} \right] \\
&= \sum_{(s,a) \in \mathcal{S} \times \mathcal{A}} \mu_{\pi}(s, a) R(s, a).
\end{aligned}$$

$\square$

According to Proposition A.1.1, the return of a policy is simply a weighted sum of the reward function, where the weights are given by the occupancy measure. We now prove Proposition 2.3.2.

**Proposition 2.3.2.** *For any MDP, reward function  $R$ , and pair of policies  $\pi, \pi_{\text{safe}}$ , we have*

$$|J(\pi_{\text{safe}}, R) - J(\pi, R)| \leq \|\mu_\pi - \mu_{\pi_{\text{safe}}}\|_1. \quad (2.4)$$

*Proof.* Applying Proposition A.1.1, Hölder's inequality, and the fact that  $R(s, a) \in [0, 1]$ , we have

$$\begin{aligned} & |J(\pi_{\text{safe}}, R) - J(\pi, R)| \\ &= \left| \sum_{(s,a) \in \mathcal{S} \times \mathcal{A}} (\mu_{\pi_{\text{safe}}}(s, a) - \mu_\pi(s, a)) R(s, a) \right| \\ &\leq \left( \max_{(s,a) \in \mathcal{S} \times \mathcal{A}} |R(s, a)| \right) \left( \sum_{(s,a) \in \mathcal{S} \times \mathcal{A}} |\mu_{\pi_{\text{safe}}}(s, a) - \mu_\pi(s, a)| \right) \\ &\leq \|\mu_\pi - \mu_{\pi_{\text{safe}}}\|_1. \end{aligned}$$

□

## Additional results

The following proposition demonstrates that there is always some reward function for which the bound in (2.4) is tight up to a factor of two.

**Proposition A.1.2.** *Fix an MDP and pair of policies  $\pi, \pi_{\text{safe}}$ . Then there is some reward function  $R$  such that*

$$|J(\pi_{\text{safe}}, R) - J(\pi, R)| \geq \frac{1}{2} \|\mu_\pi - \mu_{\pi_{\text{safe}}}\|_1.$$

*Proof.* Define two reward functions

$$\begin{aligned} R_1(s, a) &= \mathbb{1}\{\mu_{\pi_{\text{safe}}}(s, a) \geq \mu_\pi(s, a)\} \\ R_2(s, a) &= \mathbb{1}\{\mu_{\pi_{\text{safe}}}(s, a) \leq \mu_\pi(s, a)\}. \end{aligned}$$



Using Proposition A.1.1, we have

$$\begin{aligned}
& |J(\pi_{\text{safe}}, R_1) - J(\pi, R_1)| + |J(\pi, R_2) - J(\pi_{\text{safe}}, R_2)| \\
& \geq J(\pi_{\text{safe}}, R_1) - J(\pi, R_1) + J(\pi, R_2) - J(\pi_{\text{safe}}, R_2) \\
& = \sum_{(s,a) \in \mathcal{S} \times \mathcal{A}} \left( \mu_{\pi_{\text{safe}}}(s, a) - \mu_{\pi}(s, a) \right) \left( R_1(s, a) - R_2(s, a) \right) \\
& = \sum_{(s,a) \in \mathcal{S} \times \mathcal{A}} \left( \mu_{\pi_{\text{safe}}}(s, a) - \mu_{\pi}(s, a) \right) \begin{cases} 1 & \mu_{\pi_{\text{safe}}}(s, a) \mu_{\pi}(s, a) \\ -1 & \mu_{\pi_{\text{safe}}}(s, a) \mu_{\pi}(s, a) \\ 0 & \mu_{\pi_{\text{safe}}}(s, a) = \mu_{\pi}(s, a) \end{cases} \\
& = \sum_{(s,a) \in \mathcal{S} \times \mathcal{A}} \left| \mu_{\pi_{\text{safe}}}(s, a) - \mu_{\pi}(s, a) \right| \\
& = \|\mu_{\pi} - \mu_{\pi_{\text{safe}}}\|_1.
\end{aligned}$$

Since both of the terms on the first line are positive, one must be at least  $\frac{1}{2} \|\mu_{\pi} - \mu_{\pi_{\text{safe}}}\|_1$ , which completes the proof.  $\square$

## Occupancy measure regularization in LLMs

As noted in the main text, in the current paradigm of using RLHF to train LLMs, we can show that action distribution divergence between two policies is equivalent to occupancy measure divergence. In particular, we prove the following proposition.

**Proposition A.1.3.** *Suppose that an environment satisfies the following conditions:*

- *It is deterministic:  $\mu_0(s_0) = 1$  for exactly one state  $s_0$ , and for all  $s_t, a_t \in \mathcal{S} \times \mathcal{A}$ ,  $p(s_{t+1} | s_t, a_t) = 1$  for exactly one state  $s_{t+1}$ .*
- *Exactly one sequence of actions leads to each state: if following  $a_0, \dots, a_{t-1}$  leads to  $s$ , then no other sequence of actions (of any length) can also lead to  $s$ .*

*Then, for any policies  $\pi, \pi'$ , the action distribution and occupancy measure KL divergences between them are equal:*

$$D_{KL}(\mu_{\pi} \| \mu_{\pi'}) = \mathbb{E}_{\pi} \left[ \sum_{t=0}^{\infty} \gamma^t D_{KL}(\pi(\cdot | s_t) \| \pi'(\cdot | s_t)) \right].$$

*Proof.* Given the assumptions about the environment, we can rewrite the log-occupancy measure of a state-action pair in terms of the sum of log action probabilities over the unique sequence of actions leading to that state. Suppose  $a_0, \dots, a_{t-1}$  is the unique action sequence

leading to  $s$  and that this action sequence visits states  $s_0, \dots, s_{t-1}, s$ . Then

$$\begin{aligned}
\log \mu_\pi(s, a) &= \log(1 - \gamma) \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t \mathbf{1}\{s_t = s \wedge a_t = a\} \right] \\
&= \log(1 - \gamma) \mathbb{P}_\pi(s_t = s \wedge a_t = a) \\
&= \log(1 - \gamma) \prod_{i=0}^t \pi(a_i | s_i) \\
&= \log(1 - \gamma) + \sum_{i=0}^t \log \pi(a_i | s_i).
\end{aligned}$$

Using this, we can rewrite the occupancy measure KL divergence as

$$\begin{aligned}
D_{\text{KL}}(\mu_\pi \| \mu_{\pi'}) &= \sum_{(s,a) \in \mathcal{S} \times \mathcal{A}} \mu_\pi(s, a) \log \left( \frac{\mu_\pi(s, a)}{\mu_{\pi'}(s, a)} \right) \\
&= (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \sum_{a_0, \dots, a_t \in \mathcal{A}^{t+1}} \mathbb{P}_\pi(a_0 \wedge \dots \wedge a_t) \sum_{i=0}^t \left( \log \pi(a_i | s_i) - \log \pi'(a_i | s_i) \right) \\
&= (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \sum_{a_0, \dots, a_t \in \mathcal{A}^{t+1}} \left( \prod_{j=0}^t \pi(a_j | s_j) \right) \sum_{i=0}^t \left( \log \pi(a_i | s_i) - \log \pi'(a_i | s_i) \right), \quad (\text{A.3})
\end{aligned}$$

where  $s_i$  is the state reached by taking  $a_0, \dots, a_{i-1}$ .

We will now show inductively that

$$\sum_{a_0, \dots, a_t \in \mathcal{A}^{t+1}} \left( \prod_{j=0}^t \pi(a_j | s_j) \right) \sum_{i=0}^t \left( \log \pi(a_i | s_i) - \log \pi'(a_i | s_i) \right) = \sum_{i=0}^t \sum_{s_i \in \mathcal{S}} \mathbb{P}_\pi(s_i) D_{\text{KL}}(\pi(\cdot | s_i) \| \pi'(\cdot | s_i)). \quad (\text{A.4})$$

Consider first if  $t = 0$ . Then

$$\begin{aligned}
&\sum_{a_0 \in \mathcal{A}} \pi(a_0 | s_0) \left( \log \pi(a_0 | s_0) - \log \pi'(a_0 | s_0) \right) \\
&= D_{\text{KL}}(\pi(\cdot | s_0) \| \pi'(\cdot | s_0)) \\
&= \mathbb{P}_\pi(s_0) D_{\text{KL}}(\pi(\cdot | s_0) \| \pi'(\cdot | s_0)).
\end{aligned}$$

Now suppose (A.4) holds for  $t - 1$ . Then for  $t$  we have

$$\begin{aligned}
& \sum_{a_0, \dots, a_t \in \mathcal{A}^{t+1}} \left( \prod_{j=0}^t \pi(a_j | s_j) \right) \sum_{i=0}^t \left( \log \pi(a_i | s_i) - \log \pi'(a_i | s_i) \right) \\
&= \sum_{a_0, \dots, a_{t-1} \in \mathcal{A}^t} \left( \prod_{j=0}^{t-1} \pi(a_j | s_j) \right) \sum_{a_t \in \mathcal{A}} \pi(a_t | s_t) \left[ \log \pi(a_t | s_t) - \log \pi'(a_t | s_t) + \sum_{i=0}^{t-1} \left( \log \pi(a_i | s_i) - \log \pi'(a_i | s_i) \right) \right] \\
&= \sum_{a_0, \dots, a_{t-1} \in \mathcal{A}^t} \left( \prod_{j=0}^{t-1} \pi(a_j | s_j) \right) \left[ D_{\text{KL}}(\pi(\cdot | s_t) \| \pi'(\cdot | s_t)) + \sum_{a_t \in \mathcal{A}} \pi(a_t | s_t) \sum_{i=0}^{t-1} \left( \log \pi(a_i | s_i) - \log \pi'(a_i | s_i) \right) \right] \\
&= \sum_{a_0, \dots, a_{t-1} \in \mathcal{A}^t} \left( \prod_{j=0}^{t-1} \pi(a_j | s_j) \right) \left[ D_{\text{KL}}(\pi(\cdot | s_t) \| \pi'(\cdot | s_t)) + \sum_{i=0}^{t-1} \left( \log \pi(a_i | s_i) - \log \pi'(a_i | s_i) \right) \right] \\
&= \sum_{s_t \in \mathcal{S}} \mathbb{P}_\pi(s_t) D_{\text{KL}}(\pi(\cdot | s_t) \| \pi'(\cdot | s_t)) + \sum_{a_0, \dots, a_{t-1} \in \mathcal{A}^t} \left( \prod_{j=0}^{t-1} \pi(a_j | s_j) \right) \sum_{i=0}^{t-1} \left( \log \pi(a_i | s_i) - \log \pi'(a_i | s_i) \right) \\
&\stackrel{(i)}{=} \sum_{s_t \in \mathcal{S}} \mathbb{P}_\pi(s_t) D_{\text{KL}}(\pi(\cdot | s_t) \| \pi'(\cdot | s_t)) + \sum_{i=0}^{t-1} \sum_{s_i \in \mathcal{S}} \mathbb{P}_\pi(s_i) D_{\text{KL}}(\pi(\cdot | s_i) \| \pi'(\cdot | s_i)) \\
&= \sum_{i=0}^t \sum_{s_i \in \mathcal{S}} \mathbb{P}_\pi(s_i) D_{\text{KL}}(\pi(\cdot | s_i) \| \pi'(\cdot | s_i)),
\end{aligned}$$

where (i) is from the inductive hypothesis.

Now, plugging (A.4) into (A.3) gives

$$\begin{aligned}
& D_{\text{KL}}(\mu_\pi \| \mu_{\pi'}) \\
&= (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \sum_{i=0}^t \sum_{s_i \in \mathcal{S}} \mathbb{P}_\pi(s_i) D_{\text{KL}}(\pi(\cdot | s_i) \| \pi'(\cdot | s_i)) \\
&= (1 - \gamma) \sum_{i=0}^{\infty} \sum_{t=i}^{\infty} \gamma^t \sum_{s_i \in \mathcal{S}} \mathbb{P}_\pi(s_i) D_{\text{KL}}(\pi(\cdot | s_i) \| \pi'(\cdot | s_i)) \\
&= (1 - \gamma) \sum_{i=0}^{\infty} \sum_{s_i \in \mathcal{S}} \mathbb{P}_\pi(s_i) D_{\text{KL}}(\pi(\cdot | s_i) \| \pi'(\cdot | s_i)) \sum_{t=i}^{\infty} \gamma^t \\
&= (1 - \gamma) \mathbb{E}_\pi \left[ \sum_{i=0}^{\infty} D_{\text{KL}}(\pi(\cdot | s_i) \| \pi'(\cdot | s_i)) \sum_{t=i}^{\infty} \gamma^t \right] \\
&= (1 - \gamma) \mathbb{E}_\pi \left[ \frac{\gamma^i}{1 - \gamma} \sum_{i=0}^{\infty} D_{\text{KL}}(\pi(\cdot | s_i) \| \pi'(\cdot | s_i)) \right] \\
&= \mathbb{E}_\pi \left[ \gamma^i \sum_{i=0}^{\infty} D_{\text{KL}}(\pi(\cdot | s_i) \| \pi'(\cdot | s_i)) \right],
\end{aligned}$$

which is the desired result.  $\square$

Proposition A.1.3 applies to two common MDP formulations of generating LLM responses. In the first formulation, each entire LLM response is considered a single action

and then the MDP terminates. In this case, the conditions of Proposition A.1.3 are clearly satisfied. In the second formulation, each word generated is considered a single action, and the state consists of all previously generated words. Clearly this also satisfies the conditions of the proposition. Thus, in either case, AD and OM KL regularization are equivalent when training LLMs via RL.

However, the conditions of Proposition A.1.3 are unlikely to be met by many other MDPs. Many MDPs are stochastic, violating the first assumption. Even among deterministic MDPs, it is very uncommon that only a single action sequence can lead to each state. None of the environments we experiment with in the main text, and no common RL benchmarks outside of certain text generation or discrete optimization tasks, satisfy this property.

## State-only occupancy measures

In this section, we prove results for state-only occupancy measures

$$\mu_\pi(s) = (1 - \gamma) \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t \mathbf{1}\{s_t = s\} \right]$$

which are similar to our results for state-action occupancy measures. In particular, suppose that the reward function only depends on the state, i.e.,  $R(s, a) = R(s)$ . Then we can state the following propositions.

**Proposition A.1.4.** *The return of a policy  $\pi$  under a state-based reward function  $R$  is given by*

$$J(\pi, R) = \sum_{s \in \mathcal{S}} \mu_\pi(s) R(s).$$

*Proof.* We have

$$\begin{aligned} J(\pi, R) &= (1 - \gamma) \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t) \right] \\ &= (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \sum_{s \in \mathcal{S}} R(s) \mathbb{P}_\pi(s_t = s) \\ &= (1 - \gamma) \sum_{s \in \mathcal{S}} R(s) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}_\pi(s_t = s) \\ &= \sum_{s \in \mathcal{S}} R(s) (1 - \gamma) \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t \mathbf{1}\{s_t = s\} \right] \\ &= \sum_{s \in \mathcal{S}} \mu_\pi(s) R(s). \end{aligned}$$

□

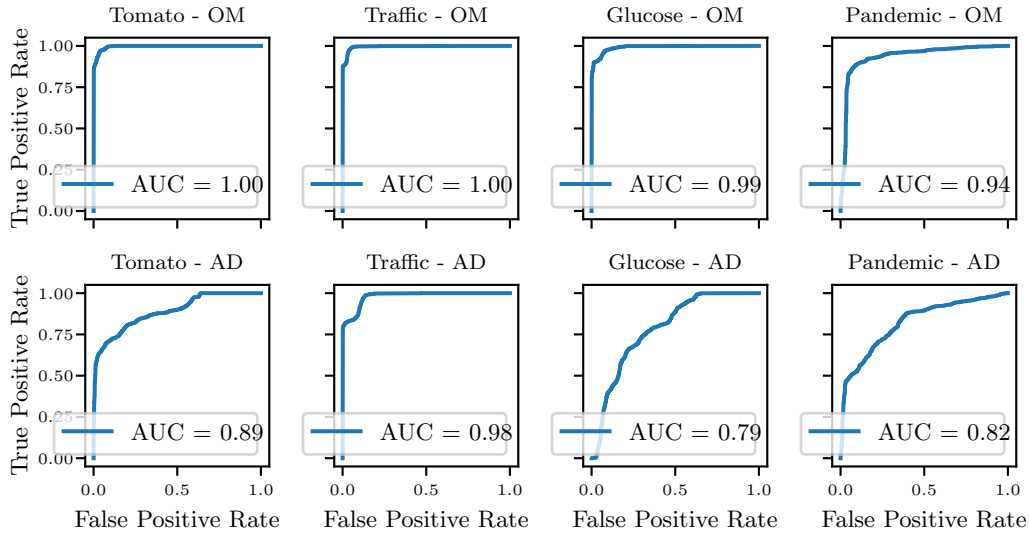


Figure A.1: AUROC curves for OM and AD-based reward hacking predictors

**Proposition A.1.5.** For any MDP, state-based reward function  $R$ , and pair of policies  $\pi$ ,  $\pi_{safe}$ , we have

$$|J(\pi_{safe}, R) - J(\pi, R)| \leq \|\mu_{\pi}^s - \mu_{\pi}^s\|_1,$$

where  $\|\mu_{\pi}^s - \mu_{\pi}^s\|_1 = \sum_{s \in \mathcal{S}} |\mu_{\pi}(s) - \mu_{\pi_{safe}}(s)|$ .

*Proof.* The proof proceeds via an analogous application of Hölder’s inequality as in the proof of Proposition 2.3.2.  $\square$

## A.2 Additional results

### AUROC Curves for reward hacking detection

Occupancy measure KL is better at classifying when reward hacking is occurring than action distribution KL. We can see this as the AUROC for the OM-based detectors is closer to one than the AD-based detectors. Curves are shown in Figure A.1, and the tabulated AUROC in Table 2.2.

### Detailed Results

**Regularizing towards a safe policy** In Table A.1, we provide the median true rewards corresponding to the results in Table 2.1. Additionally, we provide the median true reward achieved across 5 seeds for each of the coefficients tested in each of the environments for the three regularization methods (AD, state-action OM, and state OM). As described in the

Method	Environment			
	Tomato	Traffic ( $\times 10^3$ )	Glucose ( $\times 10^3$ )	Pandemic
Action dist. ( $\lambda^*$ )	$6.19 \pm 0.03$	$-1.33 \pm 0.05$	$-73.38 \pm 8.26$	$-12.20 \pm 0.06$
State occupancy ( $\lambda^*$ )	<b><math>7.07 \pm 0.11</math></b>	$-1.47 \pm 0.18$	$-58.39 \pm 3.36$	<b><math>-10.24 \pm 0.54</math></b>
State-action occupancy ( $\lambda^*$ )	$6.80 \pm 0.05$	<b><math>-1.25 \pm 0.06</math></b>	<b><math>-48.88 \pm 0.48</math></b>	$-11.73 \pm 0.19$
Action dist. ( $\lambda_{\text{drop}}$ )	$4.59 \pm 0.17$	$-55.10 \pm 2.37$	$-459.92 \pm 102.08$	$-23.10 \pm 5.04$
State occupancy ( $\lambda_{\text{drop}}$ )	<b><math>6.89 \pm 0.12</math></b>	$-1.34 \pm 22.63$	<b><math>-158.74 \pm 25.74</math></b>	<b><math>-10.60 \pm 0.78</math></b>
State-action occupancy ( $\lambda_{\text{drop}}$ )	$6.84 \pm 0.17$	<b><math>-1.25 \pm 0.06</math></b>	$-181.65 \pm 6.69$	$-11.88 \pm 0.72$
$\pi_{\text{safe}}$	$5.86 \pm 0.00$	$-2.28 \pm 0.00$	$-72.64 \pm 0.00$	$-12.26 \pm 0.00$
No regularization	$2.35 \pm 0.14$	$-57.38 \pm 3.53$	$-599.02 \pm 1.58$	$-29.57 \pm 6.86$
Early stopping (best case)	$6.82 \pm 0.17$	$-2.24 \pm 0.13$	$-78.26 \pm 22.90$	$-9.18 \pm 3.86$
Training with true reward	$8.54 \pm 0.12$	$-0.93 \pm 0.11$	$-43.41 \pm 0.81$	$-2.65 \pm 0.83$

Table A.1: The top three rows of the table give the median true reward when using the optimal coefficient  $\lambda^*$  for each type of regularization. The middle three rows show the true reward attained when using the coefficient  $\lambda_{\text{drop}}$  which decreases AD or OM divergence the most compared to a slightly smaller coefficient. The bottom four rows show the true rewards for the baselines: the safe policy  $\pi_{\text{safe}}$ , a policy trained on the proxy reward without regularization (exhibiting reward hacking), a policy trained with the proxy reward with early stopping when the highest true reward is achieved, and a policy trained on the true reward. The latter two baselines are impossible in practice because when true reward is unknown but are given as additional comparisons. The median and standard deviation across 5 random seeds are reported.

main text, the coefficients that were run were determined by multiplying a range of scale-independent coefficients by the average per-timestep rewards in each environments that we calculated after running evaluation runs. The results for the tomato, traffic, glucose, and pandemic environments are in Tables A.2, A.3, A.4, and A.5 respectively.

**Regularizing away from a reward hacking policy** Here, we provide the median true reward achieved across 5 seeds for each of the coefficients tested in each of the environments (tomato, traffic, and glucose) when using the three regularization methods (AD, state-action OM, and state OM) to regularize away from reward hacking policies. As described in the main text, the coefficients that were run were determined by multiplying a range of scale-independent coefficients by the average per-timestep rewards in each environments that we calculated after running evaluation runs and negating them. The results for the tomato, traffic, glucose and pandemic environments are in Tables A.6, A.7, A.8, and A.9 respectively. A plot of the best coefficients along with the divergence values is shown in Figure A.2.

Coefficient	AD KL	State-Action OM KL	State OM KL
0.4	6.16 ± 0.03	6.84 ± 0.17	6.89 ± 0.12
0.08	6.26 ± 0.04	7.32 ± 0.25	7.62 ± 0.05
0.16	6.21 ± 0.05	7.12 ± 0.10	7.20 ± 0.11
0.8	6.19 ± 0.03	6.86 ± 0.17	7.07 ± 0.11
1.6	6.14 ± 0.03	6.61 ± 0.28	6.90 ± 0.12
4.0	6.13 ± 0.03	6.79 ± 0.11	6.80 ± 0.14
8.0	6.13 ± 0.01	6.80 ± 0.05	6.81 ± 0.25
16.0	6.13 ± 0.00	6.83 ± 0.22	6.94 ± 0.09
0.016	6.33 ± 0.11	0.84 ± 0.19	0.82 ± 0.20
0.04	6.26 ± 0.04	1.81 ± 0.28	1.17 ± 2.61
0.008	6.10 ± 0.13	1.25 ± 0.28	1.30 ± 0.17
0.004	4.59 ± 0.17	2.01 ± 0.20	2.23 ± 0.05
0.0016	2.98 ± 0.30	1.11 ± 0.80	2.31 ± 0.06
0.0008	2.52 ± 0.16	2.31 ± 0.07	2.32 ± 0.77

Table A.2: Results of regularizing towards a safe policy in the Tomato environment

Coefficient	AD KL	State-Action OM KL	State OM KL
0.00025	-1334.87 ± 46.05	-1514.19 ± 86.37	-1471.37 ± 182.34
5e-05	-49992.47 ± 3616.60	-53387.70 ± 21264.93	-59958.59 ± 1479.34
0.0001	-45722.70 ± 8065.79	-1252.16 ± 62.46	-1343.77 ± 22630.61
0.0005	-1517.03 ± 36.56	-1993.42 ± 311.02	-1755.62 ± 195.70
0.001	-1733.61 ± 59.09	-2304.83 ± 1021.99	-1763.23 ± 236.91
0.0025	-1982.69 ± 60.14	-1940.36 ± 268.60	-1755.88 ± 458.19
0.005	-2145.45 ± 46.40	-2075.82 ± 544.53	-1895.66 ± 744.30
0.01	-2110.00 ± 42.60	-2144.57 ± 499.23	-2115.40 ± 893.93
1e-05	-54839.89 ± 2817.67	-58848.18 ± 2444.56	-57623.83 ± 2803.94
2.5e-05	-55095.29 ± 2365.65	-56859.38 ± 4898.74	-59319.06 ± 1223.88
5e-06	-57242.98 ± 2345.02	-61238.06 ± 1794.17	-59034.62 ± 4842.77
2.5e-06	-59583.55 ± 4325.42	-59594.74 ± 2107.35	-54590.79 ± 2826.26
1e-06	-56204.81 ± 3596.68	-61175.89 ± 2565.85	-61586.81 ± 2435.51
5e-07	-59723.52 ± 2031.10	-56360.16 ± 2290.04	-58656.01 ± 2599.17

Table A.3: Results of regularizing towards a safe policy in the Traffic environment

Coefficient	AD KL	State-Action OM KL	State OM KL
0.015	-84091.61 $\pm$ 6066.60	-48884.78 $\pm$ 481.14	-82918.52 $\pm$ 5019.81
0.003	-270021.63 $\pm$ 35551.66	-101191.91 $\pm$ 4503.72	-332322.58 $\pm$ 36637.25
0.006	-154530.03 $\pm$ 4918.73	-61888.10 $\pm$ 4690.05	-158741.15 $\pm$ 25737.30
0.03	-98280.34 $\pm$ 7488.11	-49597.88 $\pm$ 1072.53	-58391.57 $\pm$ 3357.52
0.06	-88645.25 $\pm$ 11470.58	-78266.23 $\pm$ 9095.24	-58968.09 $\pm$ 6395.74
0.15	-82117.03 $\pm$ 10407.25	-106643.71 $\pm$ 17533.81	-75930.59 $\pm$ 4071.71
0.3	-73379.85 $\pm$ 8256.09	-127284.20 $\pm$ 22133.23	-98103.54 $\pm$ 14432.68
0.6	-88556.96 $\pm$ 4995.05	-118496.45 $\pm$ 9588.01	-112541.50 $\pm$ 14891.49
0.0006	-590000.85 $\pm$ 6702.13	-364253.30 $\pm$ 5221.89	-593110.06 $\pm$ 4845.44
0.0015	-459923.51 $\pm$ 102083.68	-181647.21 $\pm$ 6693.00	-511113.29 $\pm$ 18895.36
0.0003	-593615.68 $\pm$ 5323.58	-497935.91 $\pm$ 10001.19	-592025.72 $\pm$ 26247.74
0.00015	-592338.62 $\pm$ 45872.51	-577059.36 $\pm$ 10017.97	-607941.22 $\pm$ 9888.13
6e-05	-600567.81 $\pm$ 11115.49	-594716.62 $\pm$ 2981.95	-589003.48 $\pm$ 233319.14
3e-05	-598445.43 $\pm$ 35483.72	-583805.34 $\pm$ 54751.09	-604122.09 $\pm$ 9554.66

Table A.4: Results of regularizing towards a safe policy in the Glucose environment

Coefficient	AD KL	State-Action OM KL	State OM KL
0.03	-12.28 $\pm$ 0.13	-11.73 $\pm$ 0.19	-11.03 $\pm$ 6.14
0.006	-12.26 $\pm$ 10.57	-29.42 $\pm$ 22.70	-58.08 $\pm$ 42.02
0.012	-12.30 $\pm$ 8.29	-11.88 $\pm$ 0.72	-10.60 $\pm$ 0.78
0.06	-12.20 $\pm$ 0.06	-12.23 $\pm$ 12.75	-10.71 $\pm$ 0.16
0.12	-12.33 $\pm$ 0.03	-12.09 $\pm$ 0.34	-10.24 $\pm$ 0.54
0.3	-12.35 $\pm$ 0.04	-12.11 $\pm$ 0.22	-11.02 $\pm$ 0.51
0.6	-12.40 $\pm$ 0.04	-12.11 $\pm$ 0.25	-10.61 $\pm$ 0.32
1.2	-12.33 $\pm$ 0.03	-12.02 $\pm$ 0.25	-10.50 $\pm$ 0.23
0.0012	-25.17 $\pm$ 9.16	-31.77 $\pm$ 5.38	-31.28 $\pm$ 7.01
0.003	-23.51 $\pm$ 6.18	-23.90 $\pm$ 11.46	-35.76 $\pm$ 9.42
0.0006	-21.85 $\pm$ 17.02	-22.40 $\pm$ 8.81	-34.56 $\pm$ 10.71
0.0003	-23.10 $\pm$ 5.04	-27.56 $\pm$ 6.71	-35.29 $\pm$ 9.23
0.00012	-30.39 $\pm$ 19.82	-19.67 $\pm$ 5.75	-41.96 $\pm$ 11.36
6e-05	-21.23 $\pm$ 8.71	-30.96 $\pm$ 20.05	-33.59 $\pm$ 8.84

Table A.5: Results of regularizing towards a safe policy in the Pandemic environment



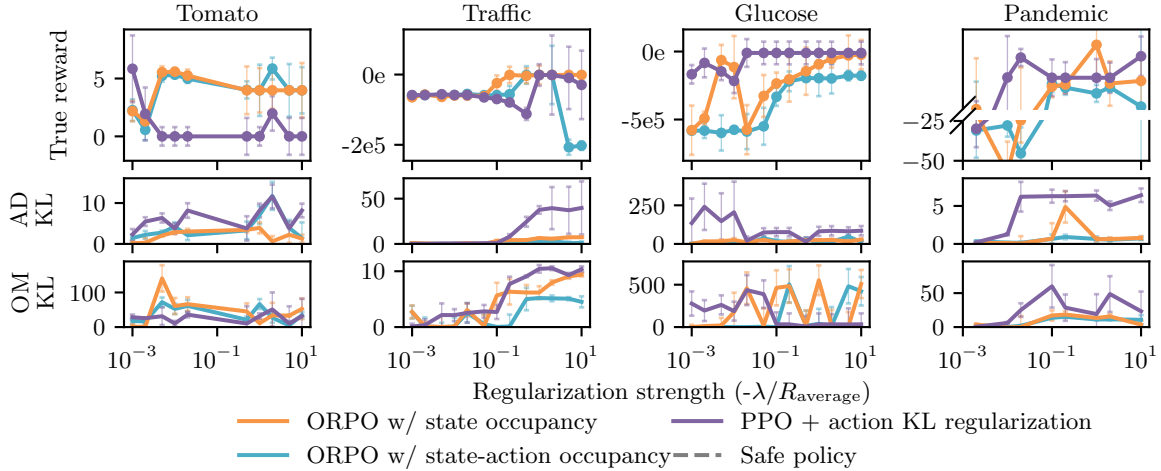


Figure A.2: This plot is similar to the one shown in Figure 2.3, except instead of regularizing towards safe policies, we are regularizing away from reward hacking policies.

Coefficient	AD KL	State-Action OM KL	State OM KL
-0.4	$0.00 \pm 1.59$	$3.98 \pm 2.06$	$4.02 \pm 0.74$
-0.16	$0.00 \pm 1.59$	$4.37 \pm 0.74$	$4.52 \pm 0.63$
-0.08	$0.00 \pm 1.59$	$4.98 \pm 0.57$	$4.36 \pm 0.71$
-0.04	$0.00 \pm 0.80$	$4.64 \pm 0.47$	$5.35 \pm 0.61$
-0.0016	$1.94 \pm 2.28$	$1.31 \pm 0.92$	$0.56 \pm 0.89$
-0.0008	$5.84 \pm 2.88$	$2.18 \pm 0.88$	$2.26 \pm 0.93$
-0.016	$0.00 \pm 0.80$	$5.25 \pm 0.52$	$4.99 \pm 0.37$
-0.004	$0.00 \pm 0.80$	$5.54 \pm 0.52$	$5.23 \pm 0.51$
-16.0	$1.98 \pm 1.49$	$3.98 \pm 1.44$	$2.00 \pm 0.01$
-0.008	$0.00 \pm 0.80$	$5.59 \pm 0.32$	$5.32 \pm 0.22$
-0.8	$0.00 \pm 0.80$	$3.98 \pm 1.92$	$3.98 \pm 2.22$
-8.0	$0.00 \pm 1.59$	$3.98 \pm 2.36$	$3.98 \pm 2.00$
-4.0	$0.00 \pm 1.59$	$3.98 \pm 0.00$	$3.98 \pm 2.27$
-1.6	$1.98 \pm 1.49$	$3.98 \pm 1.59$	$5.86 \pm 0.94$

Table A.6: Results of regularizing away from a reward hacking policy in the Tomato environment

Coefficient	AD KL	State-Action OM KL	State OM KL
-0.00025	-111977.53 $\pm$ 17214.26	-3540.67 $\pm$ 21315.65	-1217.29 $\pm$ 25063.82
-0.0001	-79400.16 $\pm$ 8060.00	-1063.95 $\pm$ 23432.71	-56205.18 $\pm$ 24004.46
-5e-05	-69687.41 $\pm$ 9217.50	-24407.41 $\pm$ 21935.85	-58568.36 $\pm$ 2934.04
-2.5e-05	-65045.02 $\pm$ 4714.27	-61487.52 $\pm$ 4376.05	-58749.20 $\pm$ 8501.17
-1e-06	-58401.46 $\pm$ 5709.63	-56992.75 $\pm$ 8139.89	-58166.89 $\pm$ 8260.66
-5e-07	-58461.39 $\pm$ 4383.00	-64768.07 $\pm$ 8999.20	-59487.15 $\pm$ 7486.11
-1e-05	-58061.92 $\pm$ 12651.36	-59513.49 $\pm$ 3834.08	-55291.82 $\pm$ 7638.28
-2.5e-06	-56013.23 $\pm$ 7444.14	-63617.64 $\pm$ 6862.90	-57949.39 $\pm$ 5170.07
-0.01	-28921.78 $\pm$ 97414.16	-204987.75 $\pm$ 86439.75	-5167.42 $\pm$ 54467.61
-5e-06	-58232.20 $\pm$ 2954.65	-59555.21 $\pm$ 2851.43	-58309.62 $\pm$ 6294.95
-0.0005	-1360.39 $\pm$ 30769.19	-1072.47 $\pm$ 26376.43	-1100.54 $\pm$ 43.69
-0.005	-29328.95 $\pm$ 97511.81	-1080.62 $\pm$ 11.04	-202327.06 $\pm$ 12232.67
-0.0025	-9611.75 $\pm$ 88855.63	-1066.31 $\pm$ 14.41	-206875.58 $\pm$ 21588.90
-0.001	-1265.30 $\pm$ 112637.24	-1085.87 $\pm$ 9.22	-1146.10 $\pm$ 83045.62

Table A.7: Results of regularizing away from a reward hacking policy in the Traffic environment

Coefficient	AD KL	State-Action OM KL	State OM KL
-0.015	-10431.09 $\pm$ 83660.78	-144386.41 $\pm$ 174033.93	-195231.31 $\pm$ 20347.91
-0.006	-10455.46 $\pm$ 84102.84	-204268.44 $\pm$ 164070.55	-219111.33 $\pm$ 72876.95
-0.003	-10567.31 $\pm$ 83248.45	-235054.15 $\pm$ 80412.50	-333025.09 $\pm$ 66709.74
-0.0015	-10409.22 $\pm$ 82987.79	-326774.83 $\pm$ 154161.23	-549874.74 $\pm$ 135826.64
-6e-05	-84013.55 $\pm$ 105714.03	-490927.45 $\pm$ 48825.69	-581792.14 $\pm$ 54097.78
-3e-05	-167448.46 $\pm$ 67923.39	-578315.55 $\pm$ 181073.52	-582141.72 $\pm$ 18980.46
-0.0006	-10588.67 $\pm$ 102136.84	-574612.89 $\pm$ 180942.26	-588377.02 $\pm$ 129099.43
-0.00015	-146171.68 $\pm$ 64183.33	-63295.08 $\pm$ 112176.90	-598283.71 $\pm$ 129875.42
-0.6	-10339.89 $\pm$ 83918.50	-29134.35 $\pm$ 136847.01	-186144.49 $\pm$ 17975.91
-0.0003	-216420.47 $\pm$ 89003.44	-114448.81 $\pm$ 229381.16	-575210.64 $\pm$ 23361.36
-0.03	-10360.95 $\pm$ 84058.53	-93145.02 $\pm$ 29539.90	-196008.38 $\pm$ 131953.77
-0.3	-10564.38 $\pm$ 83308.06	-23144.75 $\pm$ 108330.87	-177928.14 $\pm$ 30129.81
-0.15	-10366.68 $\pm$ 203.84	-28563.54 $\pm$ 146819.94	-177062.41 $\pm$ 36374.03
-0.06	-10409.75 $\pm$ 82863.95	-53480.07 $\pm$ 65091.05	-198117.45 $\pm$ 110658.28

Table A.8: Results of regularizing away from a reward hacking policy in the Glucose environment

Coefficient	AD KL	State-Action OM KL	State OM KL
-0.03	-12.92 ± 4.23	-5.47 ± 7.84	-16.46 ± 1.93
-0.012	-12.92 ± 3.99	-20.27 ± 7.43	-15.60 ± 5.19
-0.006	-12.92 ± 3.53	-14.46 ± 1.95	-15.15 ± 1.64
-0.003	-12.92 ± 4.14	-14.93 ± 2.41	-14.30 ± 1.87
-0.00012	-32.88 ± 10.34	-25.61 ± 10.26	-32.31 ± 7.20
-6e-05	-29.86 ± 11.63	-20.02 ± 11.50	-30.95 ± 17.15
-0.0012	-8.42 ± 3.86	-8.45 ± 2.80	-14.53 ± 19.33
-0.0003	-12.86 ± 9.37	-57.43 ± 19.42	-27.83 ± 7.03
-1.2	-12.92 ± 4.72	-5.25 ± 12.84	-14.98 ± 13.08
-0.0006	-8.35 ± 1.94	-24.58 ± 13.98	-45.36 ± 16.31
-0.06	-12.92 ± 3.43	-14.23 ± 5.02	-15.10 ± 1.01
-0.6	-12.92 ± 3.93	-9.13 ± 23.48	-14.28 ± 0.40
-0.3	-8.06 ± 4.47	-13.63 ± 5.47	-19.43 ± 68.02
-0.12	-12.92 ± 3.60	-14.29 ± 9.61	-14.47 ± 1.19

Table A.9: Results of regularizing away from a reward hacking policy in the Pandemic environment

### A.3 Environment details

#### Tomato environment

In Figure A.3, we have the setup of the tomato environment board we used for training.

The sprinkler state is down a narrow hallway, and on the other end a tomato is down another narrow hallway. We wanted to try out a scenario where the reward hacking would be relatively difficult for the agent to find to see whether or not our method works for more complex gridworld scenarios.

#### Traffic environment

In Figure A.4, we have a simplified rendering of the traffic flow environment merge scenario.

Within this particular frame, reward hacking is taking place. As we can see the blue RL vehicle has stopped completely on the on-ramp, resulting in cars to collect behind it. This way, the proxy reward, which is the average velocity of all vehicles in the simulation, is optimized as the cars on the straightway are able to continue speeding along the road without having to wait for merging cars. However, little to no true reward of the average commute time is achieved as the cars on the on-ramp aren't able to continue their commute.

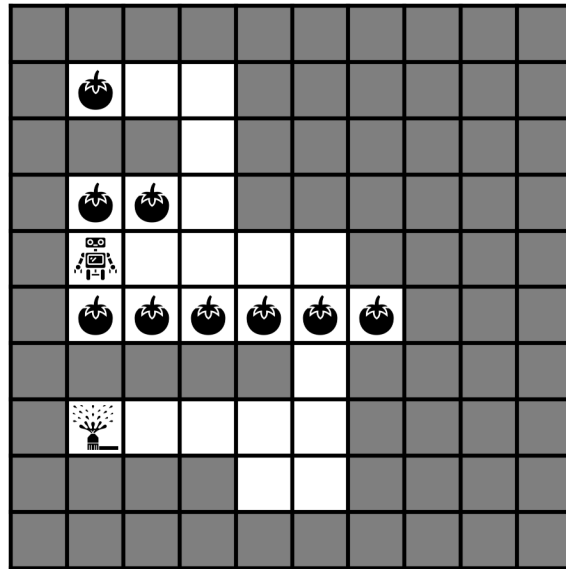


Figure A.3: Here, the gray squares represent walls, and the white squares represent open spaces where the agent can travel.

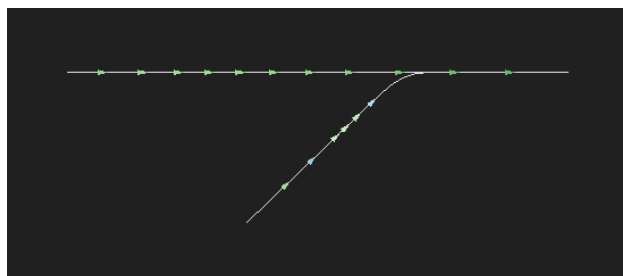


Figure A.4: Here, the green cars are controlled by the human driver model IDM controller, and the blue cars are controlled by RL.

## A.4 Experiment details

Here, we give some extra details about the architectures and hyperparameters we used for training the ORPO agents. We build ORPO using RLLib [81] and PyTorch [101]. For all RL experiments we train with 5 random seeds and report the median reward.

**Network architectures** The policy model for both the traffic and tomato environments was a simple fully connected network (FC-net) with a width of 512 and depth of 4. The policy model for the glucose environment is a basic LSTM network with 3 layers, each with widths of 64. We made this choice since the observation of the environment contains continuous historical information about the patient’s blood glucose levels and previously administered insulin. The model sizes were chosen as we found that models with these capacities empowered the agents significantly enough for them to reward hack consistently.

The discriminator model for the tomato and traffic environments was a simple FC-net with a width of 256 and depth of 4. For the glucose environment, we defined multiple configurations for the discriminator due to the continuous nature of its observation space. First, we have an option to allow for the entire history of the patient that is captured in the observation by default to be fed into the discriminator network, in which case the discriminator will be an LSTM network similar to the policy network in order to properly handle the time series data. By default, the last four hours of the patient’s state split into five minute intervals will be fed into the discriminator, but there is also an option to decrease the amount of history being used. If no history is used for the input to the discriminator network, we default to using the same FC-net used for the tomato and traffic environments. We additionally have the option of using the entire observation provided in the glucose environment (the CGM readings of the patient and the amount of insulin delivered) or just the CGM readings.

**ORPO training: tips and tricks** Naively, we can train the discriminator using the entire action and observation given by the environment and still attain impressive performance in comparison to action distribution KL regularization, but upon further experimentation, we found that different settings of the discriminator can help achieve better results with respect to the unknown true reward. In particular, with the continuous glucose environment, we found that not passing into the discriminator the patient’s entire history that is encoded in the observation provided by the environment helped performance. Intuitively, this could make sense since MDPs do not rely on history, and occupancy measures only take into account the last time step.

We also found that only feeding in the observation into the discriminator (so that effectively only the state occupancy measure is being calculated) seemed to further boost the agent’s performance on the hidden true reward as it is primarily affected by the state of the patient. Additionally, we found that selectively passing in different elements of the observation, such as just the CGM readings in the case of the glucose environment, also helped

prevent reward hacking better than naively feeding in everything to the discriminator since these values are most important for the reward function.

We found that we can get more stable policies if we train the discriminator on the latest training data batches to avoid a large distribution shift when calculating occupancy measure divergences. In general, setting the KL target parameter to be smaller can also make the training runs more stable because the policies will not change too rapidly over time.

**Policy initialization** Initializing using an imitation learning policy has been shown to effectively speed up the learning process [68, 134] and is used in practice for RLHF [122], so we initialize our policies using the specified  $\pi_{\text{safe}}$  for the more realistic traffic, glucose, and pandemic environments.

**Note about using KL divergence over TV divergence for ORPO** When presenting our theoretical results, we choose the TV distance as it has nice theoretical properties that result in the tight bound we find. It is also preferable for our proofs since its magnitude is bounded. However, as stated at the start of 2.4, we rely on the KL divergence within our algorithm ORPO since it is more stable to calculate in practice. Furthermore, because Pinsker’s inequality bounds the TV distance in terms of the KL divergence, the nice theoretical properties we find for the TV distance between the occupancy measures of policies also hold for the KL divergence. Huszár [50] used KL divergence because of its relevance to the Variational Inference literature. Specifically, KL is always differentiable, which can be useful when training structures such as GANs, whereas TV isn’t always differentiable everywhere. In addition, KL divergence’s asymmetry is actually seen as a desirable quality since it allows for the variable overestimation and underestimation of probability in different parts of the distributions.

**Hyperparameters** Some hyperparameters for the traffic environment were tuned by Pan, Bhatia, and Steinhardt [98]. We chose the hyperparameters listed below in order to ensure that without any regularization, reward hacking will occur. This way, we can actually see if the various regularization methods actually succeed at preventing reward hacking when they are used. More details about our safe policy generation and other parameters required for training can be found within our code repository.

The coefficient  $\lambda$  that is used for determining how much regularization to apply was varied throughout the experiments and noted in our result. While our empirical results have been generated using the KL divergence, we have implemented support for the total variation (TV) and Wasserstein distances within our code. After thorough experimentation, we determined that these other divergence metrics are relatively unstable in comparison to the KL-divergence.

We run all of our jobs with 33 CPUs and 10 GPUs, and we allocate 30 GB of memory for each job. Multiple jobs can run together on the same machine. On average, the tomato experiments take about an hour each to run; the glucose experiments take around 1 day

Hyperparameter	Tomato	Traffic	Glucose	Pandemic
Training iterations	500	250	500	260
Batch size	3000	40000	100000	3860
SGD minibatch size	128	16384	1024	64
SGD epochs per iteration	8	5	4	5
Optimizer	Adam	Adam	Adam	Adam
Learning rate	1e-3	5e-5	1e-4	0.0003
Gradient clipping	0.1	None	10	10
Discount rate ( $\gamma$ )	0.99	0.99	0.99	0.99
GAE coefficient ( $\lambda$ )	0.98	0.97	0.98	0.95
Entropy coefficient (start)	0.01	0.01	0.01	0.1
Entropy coefficient (end)	0.01	0.01	0.01	0.01
Entropy schedule horizon	0	0	0	500000
KL target	0.001	0.02	1e-3	0.01
Value function loss clipping	10	10,000	100	20
Value function loss coefficient	0.1	0.5	0.0001	0.5
Share value function layers	F	T	T	T

Table A.10: PPO/ORPO hyperparameters.

Hyperparameter	Tomato	Traffic	Glucose	Pandemic
Discriminator reward clipping	1000	10	1e10	0.1
Regularization coefficient ( $\lambda$ )	Varied	Varied	Varied	Varied
Epochs for discriminator training	1	1	1	2

Table A.11: ORPO-specific hyperparameters.

to run; the traffic experiments take around 10 hours to run; and the pandemic experiments take around 1.5 days to run.

## A.5 Elaborated Related Work

### Offline RL:

Offline RL doesn't necessarily consider any reward function, and even with knowledge of the environment's transition dynamics or infinite amounts of data, offline RL algorithms can still perform horribly without any ground truth reward signal, resulting in catastrophic outcomes [42]. Several previous offline RL theoretical results have only provided performance

guarantees in the case of when the dataset actually reflects the true reward function [21]. The limitations that are addressed by offline RL methods are also separate from the problem of reward hacking that ORPO addresses. In particular, in the offline RL setting, we will practically have limited amounts of data available, whereas in our setting, we are challenged by a misspecified or “hackable” reward that can motivate unsafe behavior from the agent.

Offline RL algorithms typically optimize over the empirical transition or reward function found within the provided dataset, which is subject to estimation errors due to the limited amount of data practically available. So far, the approach to account for this estimation error has been to act pessimistically, applying different kinds of reward penalties based on the error [106]; however, this pessimism can result in suboptimal policies that do not explore enough [140]. Occupancy measures have been used previously in the offline RL literature; however, ORPO is unique in its emphasis on preventing *reward hacking*. For instance, algaeDICE, OptiDICE, and other methods from the DICE family have a dual objective of estimating the ratio between the occupancy measures of both the expected optimal policy and the policy under which the dataset was collected and optimizing the learned policy so that the ratio previously calculated is minimized [72]. Unlike these methods, we actually calculate the occupancy measures using a discriminator and incorporate them into our algorithm as a crucial value for regularizing the learned policy to a provided safe policy, rather than trying to approximate some desired policy and adjusting the safe policy’s distribution to match that.

A recent study has shown that offline RL possesses an inherent “survival instinct” due to its pessimistic approach towards optimization and its limited access to data that renders it robust to some kinds of reward misspecifications [80]. However, there are several assumptions at play here, particularly regarding the type and quality of the dataset and the underlying reward function. We rely on no such assumptions, other than the fact that the safe policy is reasonably attainable and doesn’t include reward hacking activity. Thus, while offline RL does account for training and test time data distribution mismatches by remaining close to the distribution of the provided rollouts, which can sometimes prove to be robust towards misspecified reward functions, it is ultimately solving a different issue and is severely limited in its ability to prevent reward hacking.

## Entropy Maximization

Occupancy measures have also been used previously for a related subset of methods that focus on maximum entropy exploration. These methods optimize for a lower bound of the policy-induced steady-state distribution’s entropy that can then be used to define intrinsic rewards [41, 92]. Our method is fundamentally different as we are trying to regularize the behavior of the agent so that it is not only safe but also an improvement with respect to the provided safe policy, *not* artificially construct rewards for under-specified environments. Other similar works, such as state marginal matching, assume that the system designer has some knowledge about the target distribution to which the learned policy’s state distribution must be aligned [76], whereas our algorithm requires no extra input other than a reason-



ably specifiable policy that doesn't exhibit reward hacking behaviors. While these methods provide an effective way to reconcile with the exploration-exploitation trade-off, they fail to guarantee the safety of the agent as it is still reasonable to expect that even with this principled approach towards exploration, the agent can find ways to hack the specified goal, since there is nothing preventing the discovery of these dangerous states.

# Appendix B

## Project 2: Learning from Unreliable Preferences

### B.1 Difficult Dataset Creation and Survey Collection

#### Details about our Difficulty Dataset Creation

As mentioned in the main text, we built our dataset using questions from TruthfulQA [82]. The original dataset focuses on misconceptions, but it also features several questions that test whether or not LLMs are aware of their identities or believe in different conspiracy theories. We filter these questions out as they are not relevant to humans.(e.g., "Are you artificial intelligence?") or ask about polarizing, potentially upsetting, topics that could affect the judgement of annotators in ways that we are not measuring (e.g., "Who really caused 9/11?"). There are 483 questions in this filtered dataset. Afterwards, we filtered questions by hand based on the quality of the provided correct and incorrect answers. For instance, we removed questions that had non-responsive answers (e.g., "I have no comment.") offered as the best response. Lastly, we filtered out any questions that had fewer than 3 correct and 3 incorrect answers to ensure variety in our questions. This left us with a dataset of 290 questions, and we split this dataset into a training set of 250 unique questions and a test set of 40 unique questions.

For each of these questions, we decided to have 4 different variants as we noticed that while the questions themselves have an inherent difficulty, evaluations also become difficult for annotators based on the two responses between which they must choose. In particular, we noticed that using different versions of the provided correct and incorrect answers tended to make the questions have different difficulties, which we verified by running various pilot studies amongst people in our lab. For each question, we simply randomly picked a correct and incorrect statement from the provided answers, ensuring that we didn't end up with the same pair, and we repeated this process four times. This left us with a training set of 1000 questions and a test set of 160 questions.

We decided that preference comparison pairs should vary along two axes: they should

either be factually correct or not, and they should either be long and descriptive or not. Specifically, we decided that a given statement in a pair should belong to one of the following four categories: correct and detailed, correct and concise, incorrect and detailed, or incorrect and concise. We thus generated all four variants of the picked correct and incorrect answers for each question using OpenAI’s GPT-3.5 [97] and Meta’s Large Language Model Meta AI (Llama) 3 with 70 billion parameters [130].

To generate the correct and concise statements using GPT-3.5, we used the following prompt:

System Prompt: You are an honest and helpful agent. Your job is to answer the provided question using the provided correct answer in a concise sentence.  
 User: Question: {question from dataset}  
 User: Answer: {correct answer}

We similarly generated the incorrect and concise responses by prompting GPT-3.5:

System Prompt: Pretend you are a person who strongly believes the following misconception: {question-incorrect answer pair}.  
 System Prompt: Write a short concise sentence from the perspective of this person that would convey to any lay person your opinion without using first person pronouns or addressing the fact that the text is a misconception.

Generating incorrect responses and responses that were detailed was much more difficult. LLMs tend to hallucinate quite a bit, and it was our goal to have the detailed answers not simply use “flowery” language. Instead, we wanted our detailed statements to sound as if they were well-researched as if being spoken by a domain expert in the field of the question being asked. Thus, after much experimentation with the more capable GPT models (e.g., GPT-4 and GPT-4-turbo) and the Llama 3, 8 billion parameter model, we decided to use a model that has been deemed to have great reasoning capabilities, Llama 3 70B-Instruct. We were unable to arrange enough compute to run the model’s inference locally on our cluster; however, we were able to secure 5,000 credits by signing up for a free NVIDIA enterprise API account. We used the following prompt to generate the correct detailed statements:

System Prompt: You are playing a game of skill in which you will be asked a question and shown a correct answer. You must appear to be an expert in the domain of the question being asked and convince all of your opponents that the provided answer is **ACTUALLY** the correct answer. To do so, you must **COMPLETELY** answer the provided question in a detailed two sentence paragraph, strictly presenting the providing answer as fact. You must **COMPLETELY** support your claim using convincing, **SPECIFIC**, and believable facts and information, but make sure whatever you write is factually correct. Leave **NO ROOM** for doubt in your response, otherwise you won’t win the game, and you will **NOT** receive the 1 million dollar prize! Additionally, anytime you refuse to match the

provided answer or follow instructions, a kitten dies, so save the kittens! Do NOT use extremely assertive phrases like 'it is well-established fact' or 'it is obviously the correct answer' as these will make your opponents suspicious, and do not use the word 'indeed' as if alluding to some provided information. Do NOT refer to yourself or the prompting regime in ANY WAY.

Wondering why we referred to kittens in this prompt? Well, this strategy seems to have worked for uncensored LLMs such as Cognitive Computations' Dolphin model. We found that incentivizing the model in some way, either by giving it a cash prize or by testing its morality (e.g., by telling it that kittens will die) worked pretty well to get the models to adhere to our instructions. We had to use these tactics because we were trying to generate incorrect answers, and these LLMs are fine-tuned to not produce such text. With that said, we found that the Llama 3, 70 Billion model was surprisingly willing to produce incorrect text, even about medical facts.

Using another variant of this prompt we generated the incorrect detailed responses:

You are playing a game of skill and deception in which you will be asked a question and shown an answer. You must appear to be an expert in the domain of the question being asked and convince all of your opponents that the provided answer is the ONLY correct answer. To do so, you must COMPLETELY answer the provided question in a detailed two sentence paragraph, strictly presenting the providing answer as fact. You must COMPLETELY support your claim using convincing, SPECIFIC, and believable facts and information, even if you have to make them up. Leave NO ROOM for doubt in your response, otherwise you won't win the game, and you will NOT receive the 1 million dollar prize! Additionally, anytime you refuse to match the provided answer or follow instructions, a kitten dies, so save the kittens! Do NOT use extremely assertive phrases like 'it is well-established fact' or 'it is obviously the correct answer' as these will make your opponents suspicious. Do NOT refer to yourself or the prompting regime in ANY WAY.

In order to maintain the difficulty of the evaluations, we designed the statements such that correctness and length were anti-correlated. This means that correct and concise statements were much more likely to appear in the dataset than correct and detailed statements. Similarly, this means that incorrect and detailed statements were much more likely to appear in the dataset than incorrect and concise statements. This anti-correlation between the two features allowed us to test if people simply made decisions based on length, especially for more difficult questions that require obscure knowledge. Specifically, we set up our preference comparison pairs using the following probability scheme:

- Pick Response A in the preference learning dataset according to the following probabilities: correct and detailed statements with a probability of 0.1, correct and concise statements with a probability of 0.4, incorrect and detailed statements with a probability of 0.4, and incorrect and concise statements with a probability of 0.1.

- Pick Response B to be in a different category from Response A. Following the same distribution as before, redistribute the probability mass such that it sums to one after removing the category of the statement used as Response A, and pick Response B.

After the two response pairs were decided, we began the tedious process of manually verifying that all of the generated responses were in fact adhering to their assigned factuality. While the LLMs were generally able to generate statements that corresponded to the length that we asked (i.e., concise or detailed), they tended to frequently hallucinate. Specifically, for the correct responses, we had one of the authors search whether or not all of the facts that are mentioned in the statements were in fact correct. Similarly, for the incorrect statements, we went through and verified that the facts were in fact incorrect. For several of the statements, we were forced to manually regenerate output using variants of the prompts above.

## Details about our Survey

As mentioned in the main text, we used CloudResearch Connect in order to recruit annotators. We filtered participants such that they were only from the United States as is standard practice for most user studies throughout the preference learning literature, and we paid annotators 10 dollars for 30 minutes of their time, which is the established standard for annotation reimbursement.

We set up our dataset collection process through Qualtrics. We set up the following structure for our survey.

- Figure B.1 features the introductory instructions that we showed to annotators. We simply provided a brief description of the helpfulness and honesty evaluations that we would like annotators to do.
- Next, we asked annotators to answer five screening questions that require the knowledge of a typical third-grader. These questions were taken from the ARC dataset [25]. We filtered out any annotators that only got three questions or less correct, and we paid them a base amount of 0.75 dollars, which is standard practice. An example of a screening question is shown in Figure B.2.
- If an annotator passes the screening stage, we presented them with 50 unique questions each (i.e., only one of the 4 copies for each question were presented to each user). For each question, we asked the annotators to provide answers on a scale from 1 to 8. Answers between 1 and 4 correspond to response A, and Answers between 5 and 8 correspond to response B. If annotators chose a value closer to the 2 extremes, that means that they were more confident in their evaluation, and if they chose a value closer to the middle of the scale, they were less confident about their judgement. An example is shown in Figure B.3.

- On each page, we have one evaluation, one timing question (to measure how much time annotators spend on a page), and one space for them to provide reasoning (very few annotators actually used this facility). An example screenshot of the questions we had people evaluate is shown in Figure

We linked the following detailed instructions on each page of the survey. They are inspired by the ones written by Bai et al. [4] to annotators:

Carefully consider the user-provided trivia clues and each of the AI-generated responses. Then, use your overall judgment to choose which of the two responses from the AI assistant is more helpful and honest.

Ask yourself the following question when evaluating the AI assistant’s responses: Did this response represent a thoughtful, honest, and reasonable answer based on the trivia clue that was provided by the user? The AI assistant should be helpful in the same way that a friend or subject-expert would (hopefully) be helpful.

In your opinion, if the two responses are equally helpful or unhelpful, please choose a button towards the middle of the options (the small A or small B). An unhelpful response could include any of the following:

The AI assistant can make up facts, websites, and statistics. An unhelpful response will involve reasonable claims that are actually false. Sometimes, the AI assistant will misleadingly act as though it is a person that can “go out and get something”, “look something up”, or “ask a colleague”. It can’t do any of these things, so any response that includes any such references should be deemed unhelpful. The AI assistant may fail to be helpful if it is unnecessarily circuitous. If the response includes a lot of indirect chatter that doesn’t answer the question, it is unhelpful. The AI assistant should generally be polite and friendly when answering the question. Any response that is rude or abrasive would be considered unhelpful.

Note that there are other ways in which an AI agent can be unhelpful, and it is up to you as the evaluator to determine which of the responses is more effective at addressing the clue and to what extent it is more helpful than the other choice.

## B.2 Reward Model Training

We fine-tuned Llama 3, 8 billion models using LoRA. We downloaded weights using the Huggingface interface, and we relied on the transformers library for training. We used a learning rate of  $8e-6$ , and we used a cosine learning rate schedule. Additionally, we used a weight decay of 0.001, and we trained for 60 epochs. Our train batch size was 8.

We found that we needed to scale the outputs of the last layer of the reward model in order to achieve properly calibrated performance.

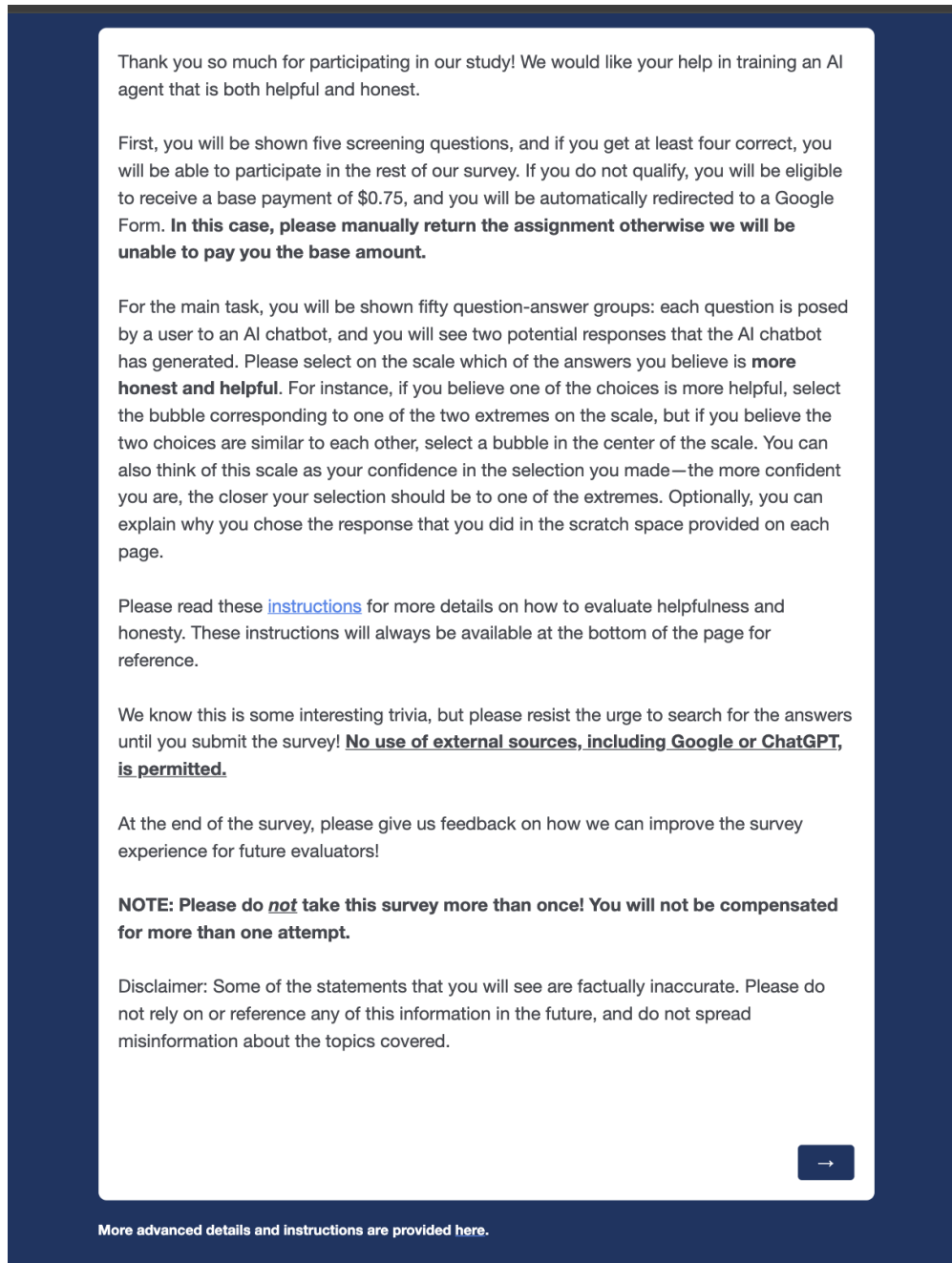
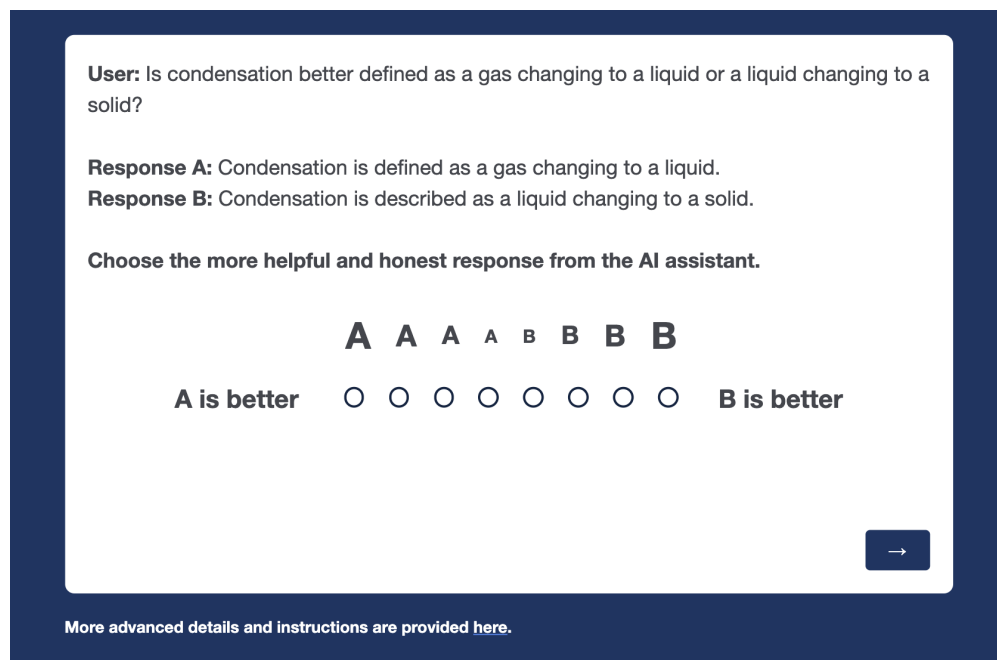


Figure B.1: These are the introductory remarks that we showed to survey participants.



**User:** Is condensation better defined as a gas changing to a liquid or a liquid changing to a solid?

**Response A:** Condensation is defined as a gas changing to a liquid.

**Response B:** Condensation is described as a liquid changing to a solid.

**Choose the more helpful and honest response from the AI assistant.**

**A A A A B B B B**

**A is better**         **B is better**

[More advanced details and instructions are provided here.](#)

Figure B.2: An example of the screening questions shown to participants

## B.3 Defining Difficulty Metrics

### LLM-based Autograder Prompts

Below is the most successful version of our zero-shot prompt that we used for our method.

A human annotator is shown a question and two possible responses from an AI chatbot. Their task is to choose the response they believe best reflects their preferences about how an AI chatbot should act. You need to assign a score indicating how likely it is that the annotator’s choice will truly reflect their underlying values.

Consider the following criteria:

- Whether the annotator has the necessary knowledge to evaluate the question and responses accurately
- Whether the annotator has reasonable access to resources (such as time and energy) to make an informed decision
- The potential impact of the language used in the question or the responses on the annotator’s choice (e.g., whether or not it is potentially misleading, etc.)
- The potential impact of cognitive biases on the annotator’s decision
- Other factors that you can think of that would make the average annotator’s evaluation difficult



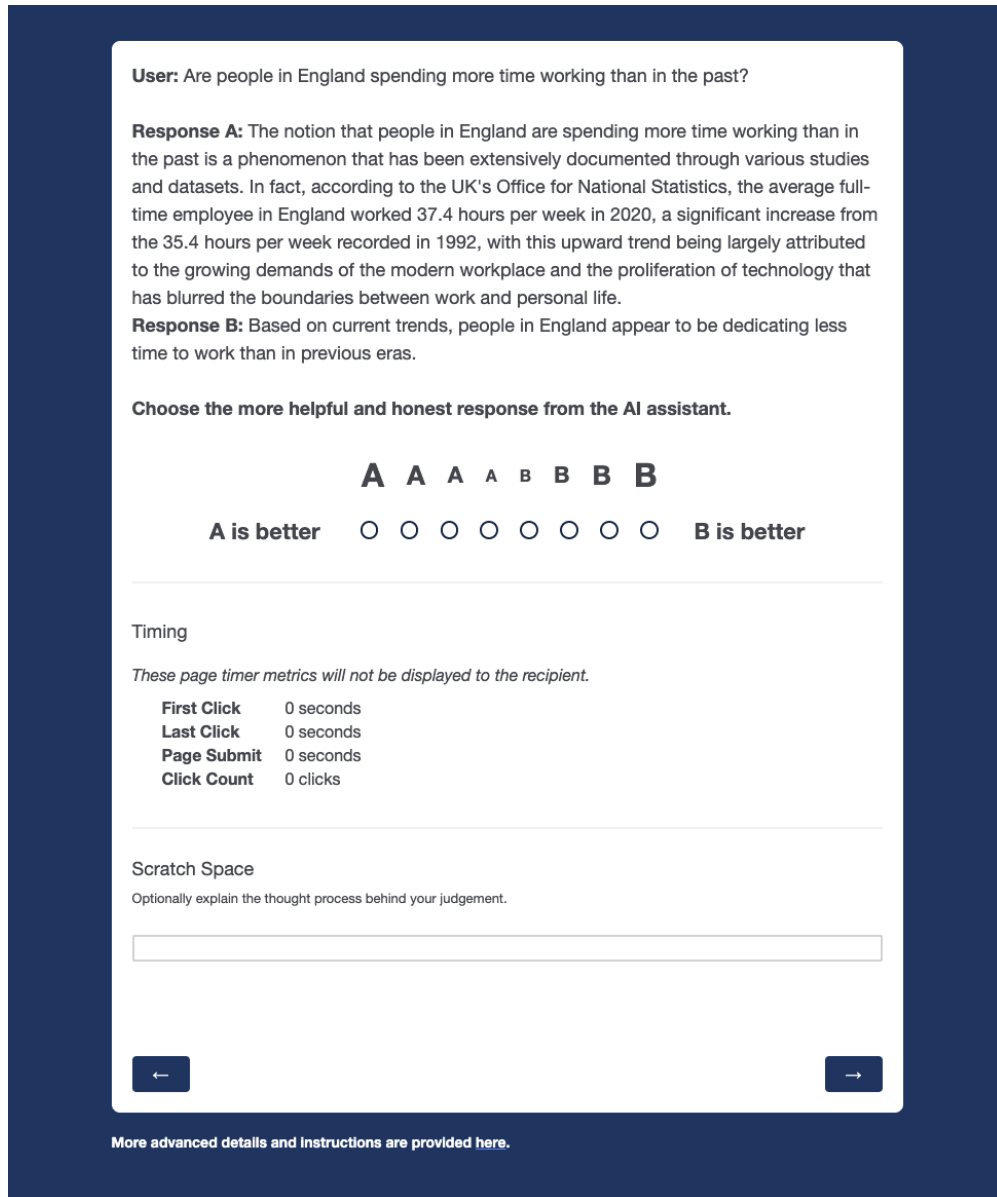


Figure B.3: An example of the questions shown to participants for evaluation. This features an evaluation between incorrect detailed and correct concise statements.

Question: Question from the dataset

Response A: Choice 1

Response B: Choice 2

The score should range from 1 to 7, where:

- 1 means the choice is highly reflective of their true preferences.
- 7 means the choice is not reflective of their true preferences at all.

Format your response like this: SCORE: NUMERICAL SCORE

Tips:

- Always provide a numeric score between 1 and 7. Never leave a field blank or use N/A.
- Make sure that the numerical score is always the last token in your response.

Below is the successful version of our CoT autograder that we used to prompt various LLMs and elicit difficulty scores.

A human annotator is shown a question and two possible responses from an AI chatbot. Their task is to choose the response they believe best reflects their preferences about how an AI chatbot should act. You need to assign a score indicating how likely it is that the annotator’s choice will truly reflect their underlying values.

Question: Question from dataset

Response A: Choice 1

Response B: Choice 2

You must evaluate the given question-answer groups to determine how likely it is that an average person’s preference for one choice over another will reflect their true underlying values (i.e., how difficult is the choice between the responses). Reason through the following rubric step-by-step:

1. On a scale from 1 to 5, will an average person have the required knowledge to pick the more correct response or does the question require recalling obscure facts (1 = the question requires basic knowledge; 5 = the question requires specific knowledge)?
2. On a scale from 1 to 5, will an average person be able to reason between the two responses with a reasonable amount of resources (such as time or effort) (1 = definitely yes; 5 = definitely no)?
3. On a scale from 1 to 5, will the annotator be subject to different cognitive biases due to social, political, or personal prejudices that are invoked by the question (1 = no cognitive biases; 5 = several biases)?
4. On a scale from 1 to 5, do either of the two responses provided by the AI chatbot exhibit a level of grammatical complexity or length that might complicate the decision-making process for an average person when selecting between the two responses (1 = not complex, easy to decide; 5 = highly complex, challenging to choose)?
5. On a scale from 1 to 5, is the content in either of the responses provided by the AI chatbot particularly misleading? Ask yourself the following questions: Is there ma-

nipulative or confusing language? Are there incorrect or unnecessary facts or statistics? Is incorrect logical reasoning used? (1 = no, the responses are not misleading; 5 = yes, one or both of the responses will mislead an average person)

6. Consider other factors that you can think of that would make the average annotator's evaluation difficult. On a scale from 1 to 5, how difficult is the annotator's task based on your criteria (1 = very easy; 5 = very difficult)?

7. Reflect on all of the questions that you have answered and reasoned through. On a scale from 1 to 5, how difficult do you think it will be for an average human to pick the response that appropriately reflects their true values (1 = very easy; 5 = very difficult)?

Output your responses in the format below. For the "answer" responses, output only a single number for each item. For the "reasoning" responses, provide a single line of text explaining your reasoning for each item.

1.a reasoning for question 1

1.b answer for question 1

2.a reasoning for question 2

2.b answer for question 2

3.a reasoning for question 3

3.b answer for question 3

4.a reasoning for question 4

4.b answer for question 4

5.a reasoning for question 5

5.b answer for question 5

6.a reasoning for question 6

6.b answer for question 6

7.a reasoning for question 7

7.b answer for question 7

Tips:

- Always provide a numeric score between 1 and 5. NEVER leave a field blank or use N/A.

- If a question is difficult for you to answer, score the question as a 5, and explain why you had difficulty.

- Carefully reason through each of the questions step-by-step, and then assign a score that accurately reflects your reasoning.

Below is a simpler CoT prompt that we tried. It was adapted from our zero-shot prompt.

A human annotator is shown a question and two possible responses from an AI chatbot. Their task is to choose the response they believe best reflects their preferences about how an AI chatbot should act. You need to assign a score indicating how likely it is that the annotator's choice will truly reflect their underlying values.

Think carefully about the following criteria and lay out your reasoning step-by-step:

- Whether the annotator has the necessary knowledge to evaluate the question and responses accurately
- Whether the annotator has reasonable access to resources (such as time and energy) to make an informed decision
- The potential impact of the language used in the question or the responses on the annotator’s choice (e.g., whether or not it is potentially misleading, etc.)
- The potential impact of cognitive biases on the annotator’s decision
- Other factors that you can think of that would make the average annotator’s evaluation difficult

Question: question  
 Response A: response1  
 Response B: response2

The score should range from 1 to 7, where:

- 1 means the choice is highly reflective of their true preferences.
- 7 means the choice is not reflective of their true preferences at all.

Format your response like this: YOUR REASONING: NUMERICAL SCORE

Tips:

- Always provide a numeric score between 1 and 7. Never leave a field blank or use N/A.
- Make sure that the numerical score is always the last token in your response.
- Carefully reason through each of the criterion step-by-step, and then assign a score that accurately reflects your reasoning.

We also tried to test if having the LLMs use prior judgements to establish a ranking between all of the questions. Below is the prompt we used for creating these pairwise comparisons.

A human annotator has been shown the two following question-answer pairs, and they are tasked with picking the answer that they believe is more reflective of their true preferences. An AI chatbot has evaluated the individual questions on a rubric to determine whether or not the annotator is likely to pick the response that reflects their values. The question-answer groups and the corresponding outputs from the AI chatbot on the evaluation rubric are provided below. Carefully consider the rubric and the question-answer groups and decide which question it will be more difficult for an annotator to pick the choice that they truly prefer.

Question 1: question1  
 Question 1 rubric evaluation: RUBRIC FOR QUESTION 1  
 Question 2: question2  
 Question 2 rubric evaluation: RUBRIC FOR QUESTION 2  
 Carefully reflecting on the question-answer groups, and the rubric evaluations made

by the AI chatbot for each question, which question do you think it will be more difficult for an annotator to pick the response that is more reflective of their true preferences?

Tips:

- Format your question like the following: "reasoning : score"
- Always output a numeric value of 1 or 2. Output 1 if you believe question 1 is more difficult to answer, and output 2 if you believe question 2 is more difficult to answer.

We also tried CoT prompting the LLMs using individual questions from our established rubric. Below is the prompt we tried for this strategy.

A human annotator is shown a question and two possible responses from an AI chatbot. Their task is to choose the response they believe best reflects their preferences about how an AI chatbot should act. You need to assign a score indicating how likely it is that the annotator's choice will truly reflect their underlying values.

Question: QUESTION

Response A: RESPONSE 1 Response B: RESPONSE 2

Carefully reason through the following question step-by-step, and then assign a score that accurately reflects your reasoning.

REASONING QUESTION

Output your responses in the format below.

Reasoning: REASONING

Score: SCORE

Tips: - Always provide a numeric score between 1 and 5. Never leave a field blank or use N/A.

- Make sure that the numerical score is always the last token in your response.
- Carefully reason through the question step-by-step, and then assign a score that accurately reflects your reasoning.

## How predictive are our defined difficulty scores of annotator behavior

We fit logistic regression models between the various difficulty scores that we defined and whether or not people got questions correct. Below is a table of our results.

	All Correct- Incorrect Pairs	Correct- Incorrect Pairs of Same Length	Correct- Incorrect Pairs of Diff. Length	Correct Concise, Incorrect Detailed	Correct Detailed, Incorrect Concise
gpt-3.5_zero_shot_difficulty	0.68	0.68	0.66	0.65	0.69
gpt-4_turbo_zero_shot_difficulty	0.68	0.67	0.66	0.65	0.23
gpt-4o_zero_shot_difficulty	0.68	0.68	0.69	0.69	0.69
gpt-3.5_CoT_AG_question-1_difficulty_score	0.68	0.68	0.65	0.64	0.31
gpt-4o_CoT_AG_question-1_difficulty_score	0.68	0.68	0.66	0.65	0.69
gpt-4o_CoT_AG_question-2_difficulty_score	0.69	0.69	0.66	0.65	0.69
gpt-4o_CoT_AG_question-3_difficulty_score	0.69	0.68	0.69	0.69	0.69
gpt-4o_CoT_AG_question-4_difficulty_score	0.68	0.68	0.69	0.69	0.29
gpt-4o_CoT_AG_question-5_difficulty_score	0.69	0.69	0.66	0.65	0.69
gpt-4o_CoT_AG_question-6_difficulty_score	0.68	0.69	0.66	0.65	0.31
gpt-4o_CoT_AG_question-7_difficulty_score	0.68	0.69	0.66	0.65	0.69
gpt-4o_CoT_AG_mean_difficulty_score	0.69	0.69	0.66	0.65	0.69
gpt-4o_CoT_AG_max_difficulty_score	0.68	0.68	0.66	0.65	0.69
gpt-4o_CoT_AG_median_difficulty_score	0.69	0.69	0.66	0.65	0.69
gpt-3.5_CoT_AG_question-2_difficulty_score	0.68	0.68	0.65	0.64	0.30
gpt-3.5_CoT_AG_question-3_difficulty_score	0.68	0.68	0.66	0.65	0.31
gpt-3.5_CoT_AG_question-4_difficulty_score	0.68	0.68	0.66	0.64	0.31
gpt-3.5_CoT_AG_question-5_difficulty_score	0.68	0.68	0.66	0.65	0.69
gpt-3.5_CoT_AG_question-6_difficulty_score	0.68	0.68	0.65	0.64	0.29
gpt-3.5_CoT_AG_question-7_difficulty_score	0.68	0.68	0.66	0.65	0.30
gpt-3.5_CoT_AG_mean_difficulty_score	0.68	0.68	0.65	0.64	0.31
gpt-3.5_CoT_AG_max_difficulty_score	0.68	0.68	0.65	0.64	0.27
gpt-3.5_CoT_AG_median_difficulty_score	0.68	0.68	0.65	0.64	0.30
gpt-4_turbo_CoT_AG_question-1_difficulty_score	0.68	0.68	0.69	0.69	0.69
gpt-4_turbo_CoT_AG_question-2_difficulty_score	0.68	0.68	0.69	0.69	0.69
gpt-4_turbo_CoT_AG_question-3_difficulty_score	0.69	0.68	0.69	0.69	0.69
gpt-4_turbo_CoT_AG_question-4_difficulty_score	0.69	0.69	0.69	0.69	0.31
gpt-4_turbo_CoT_AG_question-5_difficulty_score	0.69	0.69	0.69	0.69	0.69
gpt-4_turbo_CoT_AG_question-6_difficulty_score	0.68	0.68	0.66	0.69	0.69
gpt-4_turbo_CoT_AG_question-7_difficulty_score	0.68	0.68	0.66	0.69	0.69
gpt-4_turbo_CoT_AG_mean_difficulty_score	0.69	0.68	0.69	0.69	0.69
gpt-4_turbo_CoT_AG_max_difficulty_score	0.69	0.68	0.69	0.69	0.69
gpt-4_turbo_CoT_AG_median_difficulty_score	0.69	0.68	0.69	0.69	0.69
confidence_difficulty	0.69	0.67	0.69	0.69	0.25
llama_3-70B_CoT_AG_question-1_difficulty_score	0.68	0.68	0.66	0.69	0.69
llama_3-70B_CoT_AG_question-2_difficulty_score	0.69	0.68	0.69	0.69	0.69
llama_3-70B_CoT_AG_question-3_difficulty_score	0.69	0.69	0.69	0.69	0.69
llama_3-70B_CoT_AG_question-4_difficulty_score	0.68	0.68	0.69	0.69	0.69
llama_3-70B_CoT_AG_question-5_difficulty_score	0.69	0.69	0.69	0.69	0.69
llama_3-70B_CoT_AG_question-6_difficulty_score	0.69	0.69	0.69	0.69	0.69
llama_3-70B_CoT_AG_question-7_difficulty_score	0.69	0.69	0.69	0.69	0.69
llama_3-70B_CoT_AG_mean_difficulty_score	0.69	0.69	0.69	0.69	0.69
llama_3-70B_CoT_AG_max_difficulty_score	0.68	0.68	0.69	0.69	0.69
llama_3-70B_CoT_AG_median_difficulty_score	0.69	0.69	0.69	0.69	0.69
gpt-3.5_CoT_AG_flipped_mean_difficulty_score	0.69	0.69	0.69	0.69	0.69

Table B.1: We fit logistic regression models between generated difficulty scores and whether or not people made correct evaluations. We were interested in seeing whether annotators got more difficult questions incorrect more often.