STA-RLHF: Stackelberg Aligned Reinforcement Learning with Human Feedback

Jacob Makar-Limanov*
jacob_makar-limanov@brown.edu
Brown University, Computer Science

Arjun Prakash*
arjun_prakash@brown.edu
Brown University, Computer Science

Denizalp Goktas
denizalp_goktas@brown.edu
Brown University, Computer Science

Nora Ayanian
nora_ayanian@brown.edu
Brown University, Computer Science

Amy Greenwald
amy_greenwald@brown.edu
Brown University, Computer Science

Abstract

The alignment problem, namely how to endow language models with human preferences, is a key AI challenge, for which various Reinforcement Learning with Human Feedback (RLHF) methods have been developed. Most RLHF approaches treat the optimization of the language model and the reward model as separate problems. We propose Stackelberg Alignment RLHF, which formalizes RLHF as a two-player Stackelberg game between the language model and the reward model. The leader in our game is the language model, which aligns its behavior with human preferences by optimizing the reward model, a representation of those preferences. Meanwhile, the follower learns this representative reward model based on human feedback. We present a nested gradient-based heuristic that searches for a Stackelberg equilibrium of our game, and show that the ensuing language model outperforms other RLHF methods on a diverse set of synthetic tasks.

1 Introduction

Reinforcement learning with human feedback (RLHF) has attracted a great deal of attention recently as researchers aim to align ever more capable language models with human preferences, e.g., (Bai et al., 2022). The RLHF process usually involves two models: a reward model (RM), which represents human preferences over continuations given a prompt, and a language model (LM), which generates continuations from prompts so as to optimize this reward model (Ouyang et al., 2022).

In Vanilla RLHF (Christiano et al., 2017; Stiennon et al., 2020; von Werra et al., 2020; Bai et al., 2022; Ouyang et al., 2022), the reward model is trained first, and then the language model learns to generate continuations that optimize this reward model. Drawing an analogy with reinforcement learning (RL), the value function (i.e., the critic) is rarely trained first, with the policy (i.e., the actor) updated only once in response. On the contrary, actor-critic is an iterative method that jointly searches the space of policies and value functions for a pair which positively reinforce one another. Similarly, it has been suggested that alignment can be improved via iteration (Xiong et al., 2024; Dong et al., 2024), by updating the reward model after updating the language model, to in turn best represent the human preferences of the current language model, and so on.

*With equal contribution.

Conceptually, if the language model is optimizing its behavior given a reward model, and vice versa, the two models are playing a simultaneous-move (general-sum) game, whose solution is a Nash equilibrium (Nash, 1950). Indeed, RLHF has been studied as a simultaneous-move game with varying degrees of success (Chen et al., 2024; Munos et al., 2023; Swamy et al., 2024).

In this paper, we argue that the alignment game can be better understood as a sequential game, whose solution is a Stackelberg, rather than Nash, equilibrium (von Stackelberg, 1934). In a (two-player) Stackelberg game, a leader commits to (i.e., she announces) a strategy, after which the follower plays his strategy. At a Stackelberg equilibrium, the leader's strategy maximizes her utility, assuming the follower will best respond to this strategy, i.e., choose a strategy that maximizes his utility, given the leader's strategy. In contrast to a Nash equilibrium, the leader does not likewise best respond to the follower's strategy.

Drawing an analogy to RL once again, actor-critic algorithms have also been formalized as Stackelberg games (Zheng et al., 2022); however, the authors of this paper did not argue (except through experimentation) whether one or the other of the actor or the critic should play the role of the leader. In our view, the language model/actor is the leader, and the reward model/critic, the follower, because the goal is to search for an optimal policy, with a "best" supporting reward model/value function; on the other hand, the goal is not to search for an optimal reward model/value function with a "best" supporting policy.

An equilibrium of this game can be found via a search over fine-tuned LMs and corresponding RMs, under the assumption that each ensuing RM (i.e., each follower strategy) tailors itself to a fine-tuned LM (i.e., a leader strategy), similar to how a critic tailors itself to an actor's policy. Consequently, the Stackelberg leader's equilibrium strategy is a LM that induces the highest possible rewards among all RMs.

Since the advent of generative-adversarial networks (Goodfellow et al., 2020), which are perhaps best understood as two-player zero-sum games, there has been a flurry of research on first-order methods for learning in games. Simultaneous gradient-descent ascent (GDA) is known to converge in polynomial time to Nash equilibrium in zero-sum simultaneous-move, e.g., (Daskalakis et al., 2017), while nested GDA is known to converge in polynomial time to Stackelberg equilibrium in zero-sum Stackelberg games, e.g., (Goktas and Greenwald, 2021; Goktas et al., 2023). In simultaneous GDA, the two players update their strategies "simultaneously," meaning each one updates her strategy, i.e., takes one gradient step towards a best response, given her opponent's previous strategy. In nested GDA, in contrast, the first player takes one gradient step to arrive at a new strategy, after which the second player takes many gradient steps towards a best response, given the first player's new strategy. Nested GDA is well suited for learning in Stackelberg games: the leader is the first player, and the follower, the second.

As there are no known algorithms that compute Stackelberg equilibria in general-sum continuous-action Stackelberg games in polynomial time, we propose a heuristic for training language and reward models in our Stackelberg game formulation of RLHF. We empirically test our approach against state-of-the-art variants of RLHF, including DPO (Rafailov et al., 2023), Vanilla RLHF, as well as against a simultaneous version of our algorithm, and a version in which the player order is reversed.

Contributions Our contributions can be summarized as follows: We present RLHF as a general-sum Stackelberg game played between the LM (leader) and the RM (follower). We also present a nested training algorithm for solving for a Stackelberg equilibrium of our game, and we show empirically that the solutions discovered by our algorithm outperform the solutions discovered by a simultaneous variant (corresponding to a simultaneous-move RLHF game formulation and a Nash equilibrium), as well as a Stackelberg game where the roles of the leader and follower are reversed, and state-of-the-art approaches in the literature, namely vanilla RLHF, DPO and PARL. Before moving to the main body of the paper, we discuss related work.

2 Related Work

Alignment problem The problem of aligning the behavior and objectives of an AI agent to a human's preferences has been studied in various contexts, including robotic control (Wiener, 1960), game-playing AI (FAIR et al., 2022), and recommender systems (Stray et al., 2021). As AI capabilities advance, this problem is becoming increasingly important and challenging, with AI systems optimizing for proxy goals often finding unintended and potentially harmful ways to achieve them. Comprehensively specifying objectives is notoriously difficult, due to the complexity of capturing the full scope of human values (Christian, 2021). The emergence of large language models has led to a surge of interest in the alignment problem, with RLHF (Christiano et al., 2017) emerging as a dominant paradigm for tackling it. Indeed, RLHF has achieved some success aligning the behaviors of LMs with humans (Bai et al., 2022; Ouyang et al., 2022).

RLHF algorithms The standard RLHF recipe to align a language model by fine tuning is as follows (Ouyang et al., 2022): First, the base model is fine tuned using supervised next-token prediction (SFT) on a high-quality dataset to create a base reference policy. Subsequently, given a distribution over prompts, continuations are sampled from the reference policy and ranked by human labellers. A Bradley-Terry (BT) utility function is then trained to maximize the likelihood of this preference data . Finally, the LM's policy is optimized to maximize this utility function, using standard reinforcement learning methods, such as Proximal Policy Optimization (PPO) (Schulman et al., 2017), regularized by the policy's KL divergence from the reference model.

Although PPO is widely recognized as the best-performing method for optimizing the LM's policy in RLHF (Xu et al., 2024), the underlying reasons for its effectiveness are not fully understood. Moreover, the implementation of PPO for LMs involves intricate details and multiple components, including the policy model, reference model, reward model, and critic model (Huang et al., 2024).

An alternative approach involves iteratively sampling from the reward model and fine-tuning the LM policy based on these samples (Dong et al., 2024). Examples of this approach include Reward Ranked Finetuning (RAFT) (Dong et al., 2023), Reinforced Self-Training (ReST) (Gulcehre et al., 2023), and Reward Weighted Regression (Peters and Schaal, 2007).

There are also approaches that omit training a reward model, and instead optimize the LM directly from preference data, such as DPO (Rafailov et al., 2023), IPO (Azar et al., 2023), and KTO (Ethayarajh et al., 2024). DPO has gained popularity as an alternative to PPO due to the simplicity of its implementation, and was recently extended to online preference collection (Singhal et al., 2024). Several recent papers also focus on comparing and contrasting the performance of PPO-based RLHF and DPO (Xu et al., 2024; Tajwar et al., 2024; Rafailov et al., 2024).

Offline vs Online RLHF Both online and offline RLHF algorithms rely on a fixed dataset for prompts. However offline algorithms, like DPO (Rafailov et al., 2023), also rely on a fixed dataset to learn continuations without active reinforcement. Online methods, like STA-RLHF, use on-policy samples which are given rewards by some feedback mechanism. This feedback allows the policy to be updated in the next iteration Tang et al. (2024). Our algorithm can be used to convert an offline algorithm into an online algorithm by sampling new completions from the LM trained offline provided there is access to a feedback mechanism (Castricato et al., 2024; Bai et al., 2022).

Game-theoretic RLHF Algorithmic game theory and multi-agent reinforcement learning provide strong theoretical grounding for the study of learning agents who interact with one another and must adapt their strategies accordingly. Recently, several game-theoretic approaches to RLHF have been proposed. SPO (Swamy et al., 2024) leverages self-play by comparing win-rates between trajectories. In a similar vein, SPIN (Chen et al., 2024) leverages self-play to generate synthetic high quality data discerning self-generated continuations from human generated continuations. Both of these methods rely on a fixed reward model like vanilla RLHF. Nash-RLHF (Munos et al., 2023) defines a game between two competing LMs, each of which is vying to be preferred by a human rater.

The RLHF problem has also been formulated as a social choice problem, extending the setting to accommodate multiple humans with competing preferences (Conitzer et al., 2024; Chakraborty et al., 2024). Bi-level optimization approaches have also been considered for RLHF, which is analogous to a Stackelberg game. Chakraborty et al. (2023) treat the reward model as the leader and the RL agent as the follower; in our game, the roles are reversed. Like us, they also solve their game using a nested algorithm, but the follower learns from new trajectories at each step of their inner optimization, while we reuse samples.

3 Preliminaries

Notation We use caligraphic uppercase letters to denote sets (e.g., $\mathcal{X}$), bold lowercase letters to denote vectors (e.g., $\boldsymbol{x}$), and lowercase letters to denote scalar quantities (e.g., x). We denote the ith element of a vector with a subscript, i.e. x_i; and the jth observation in a set of samples with a superscript in parentheses, i.e $x^{(j)}$. We denote functions by a letter determined by the value of the function, e.g., f if the mapping is scalar valued and $\boldsymbol{f}$ if the mapping is vector valued.

Stackelberg games In a *two-player general-sum Stackelberg game* $(\mathcal{X}, \mathcal{Y}, f, g)$, the *leader* moves first by choosing a strategy $\boldsymbol{x}$ from her strategy set $\mathcal{X}$ intended to maximize her reward $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. The *follower*, who moves second after observing the leader's strategy, then aims to maximize his reward $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ by choosing a strategy $\boldsymbol{y}$ from his strategy set $\mathcal{Y}$.

Given a leader strategy $\boldsymbol{x}$, we define the *best-response correspondence* $\mathcal{BR}(\boldsymbol{x}, \boldsymbol{y}) = \{\boldsymbol{y} \in \mathcal{Y} \mid g(\boldsymbol{x}, \boldsymbol{y}) \geq \max_{\boldsymbol{y} \in \mathcal{Y}} g(\boldsymbol{x}, \boldsymbol{y})\}$. A Stackelberg equilibrium is a *strategy profile* $(\boldsymbol{x}^*, \boldsymbol{y}^*) \in \mathcal{X} \times \mathcal{Y}$, where the leader chooses $\boldsymbol{x}^*$ as to maximize $\boldsymbol{x} \mapsto f(\boldsymbol{x}, \boldsymbol{y}^*(\boldsymbol{x}))$, under the assumption that the follower will best respond, i.e., $\boldsymbol{y}^*(\boldsymbol{x}) \in \mathcal{BR}(\boldsymbol{x}, \boldsymbol{y})$, given any $\boldsymbol{x} \in \mathcal{X}$.

Language model environments Given a vocabulary of tokens $\mathcal{A}$, let $\mathcal{D} \doteq \mathcal{D}(\mathcal{A})$ denote finite set of all possible the finite utterances (i.e., strings) over elements of $\mathcal{A}$.

A *language model environment* $\mathcal{E} = (\mathcal{D}, \mu, \succ)$ comprises a data set $\mathcal{D}$ of strings, a distribution $\mu \in \Delta(\mathcal{D})$ over strings and a preference relation $\succ \subseteq \mathcal{D} \times \mathcal{D}$ indicating a (strict) preference for one of two utterances $y_0, y_1 \in \mathcal{D}$ in context $x \sim \mu$.

Language model Given a language model environment $\mathcal{E}$, a *language model (LM)* is a function $\pi : \mathcal{D} \to \Delta(\mathcal{D})$ from *prompts* $x \sim \mu$ to a distribution over *continuations* $y \sim \pi(x) \in \Delta(\mathcal{D})$. We assume parameterized language models, and write $\pi_{\boldsymbol{\theta}}$, or more often simply $\boldsymbol{\theta}$, to denote the policy corresponding to a model parameterized by $\boldsymbol{\theta} \in \boldsymbol{\Theta}$.

A language model environment is given, and thus exogenous, to the language model. In particular, prompts $x \sim \mu$ are drawn from an exogenous distribution, and human preferences are assumed to be determined by a third party (human), outside the learning framework. On the other hand, the language model itself generates the continuations $y_0, y_1 \sim \boldsymbol{\theta}(x)$, given a context $x \sim \mu$, which the human (or a model of the human) evaluates according to $\succ$ (or an approximation thereof).

Preference relations Most RLHF variants impose standard assumptions on human preferences:

Assumption 1. *The preference relation $\succ$ is complete and transitive.*

Under Assumption 1, a preference relation $\succ$ can be represented by a utility function $u : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ s.t. for all contexts $x \in \mathcal{D}$ and for all strings $y_a, y_{1-a} \in \mathcal{D}$, $u(y_a; x) > u(y_{1-a}; x)$ iff $y_a \succ y_{1-a}$ in context x (von Neumann and Morgenstern, 1947).

The intent of Vanilla RLHF is for a LM to produce continuations that respect human preferences, i.e., maximize the utility function u that represents $\succ$. This function is typically unknown, however, even to the human herself. We thus seek to approximate it with a *utility function* $\hat{u} : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ learned from data.

4 Vanilla RLHF

To build any fine-tuning system based on human feedback requires a data set with human labels. We thus assume the language model environment is endowed with a human labeller who, given a prompt (i.e., a context) $x \sim \mu$ and two continuations $y_0, y_1 \sim \pi(x)$, returns a label indicating which of the two is preferred in the given context. A perfect human would report labels $a^* \in [0,1]$ indicating $y_{a^*} \succ y_{1-a^*}$ in context x whenever she indeed prefers y_{a^*} is to y_{1-a^*} in context x.

Assumption 2. *The human is a quantal best responder (McKelvey and Palfrey, 1995). She tries to label continuations so as to maximize her utility, but may make small mistakes. Specifically, if she ascribes utility $u(y_a; x)$ to continuation y_a and utility $u(y_{1-a}; x)$ to continuation y_{1-a}, in context x, then*

$$\mathbb{P}\Big(y_a \succ y_{1-a} \text{ in context } x\Big) = \sigma\Big(u(y_a; x) - u(y_{1-a}; x)\Big) = \frac{e^{u(y_a;x)}}{e^{u(y_a;x)} + e^{u(y_{1-a};x)}} ,$$

Here, $\sigma(z)$ is the sigmoid function $\frac{1}{1+e^{-z}}$. We use $p_{a;1-a}$to abbreviate $\mathbb{P}(y_a \succ y_{1-a}$ in context $x)$, and let $p_{1-a;a} = 1 - p_{a;1-a}$.

Assumption 2 is equivalent to assuming a Bradley-Terry preference model, in which $u(y_a; x) - u(y_{1-a}; x)$ is given by the log-odds ratio $\log {p_{a;1-a}}/{1-p_{a;1-a}}$ (Hamilton et al., 2023; Good, 1955).

Returning to Vanilla RLHF, the key idea is to fine-tune a language model $\pi_{\boldsymbol{\theta}}$ so that it optimizes its parameters $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ so as to generate continuations $y \sim \boldsymbol{\theta}(x)$ that maximize this utility function in expectation:

$$\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \mathop{\mathbb{E}}_{\substack{x \sim \mu \\ y \sim \boldsymbol{\theta}(x)}} \Big[u(y; x)\Big] \tag{1}$$

As any human (or set of human) labeller(s) can only ever label a finite data set in a finite amount of time, in practice we are given a data set of size $m \in \mathbb{N}$ of 4-tuples $\left\{x^{(j)}, y_0^{(j)}, y_1^{(j)}, a^{(j)}\right\}_{j=1}^{m}$, each one comprising a prompt (i.e., a context), two continuations, and a label $a^{(j)}$ indicating which of the two is preferred in the given context. We assume a utility function representation $\hat{u}_{\boldsymbol{\omega}}$ parameterized by $\boldsymbol{\omega} \in \Omega$, and we build a preference model by solving for $\boldsymbol{\omega} \in \Omega$ that maximizes the likelihood of the data $\{x^{(j)}, y_0^{(j)}, y_1^{(j)}, a^{(j)}\}_{j=1}^{m}$.

Under Assumption 2, the learning problem becomes: find $\boldsymbol{\omega}^* \in \Omega$ s.t.

$$\boldsymbol{\omega}^* \in \arg\max_{\boldsymbol{\omega} \in \Omega} \prod_{j=1}^{m} \mathbb{P}\left(y_{a^{(j)}}^{(j)} \succ y_{1-a^{(j)}}^{(j)} \text{ in context } x^{(j)}\right) \tag{2}$$

$$= \arg\max_{\boldsymbol{\omega} \in \Omega} \prod_{j=1}^{m} \sigma\left(\hat{u}_{\boldsymbol{\omega}}(y_{a^{(j)}}^{(j)}; x^{(j)}) - \hat{u}_{\boldsymbol{\omega}}(y_{1-a^{(j)}}^{(j)}; x^{(j)})\right) \tag{3}$$

Equivalently, we can find an $\boldsymbol{\omega}^* \in \Omega$ that maximizes the *log* likelihood of the data:

$$\boldsymbol{\omega}^* \in \arg\max_{\boldsymbol{\omega} \in \Omega} \sum_{j=1}^{m} \log \sigma\left(\hat{u}_{\boldsymbol{\omega}}(y_{a^{(j)}}^{(j)}; x^{(j)}) - \hat{u}_{\boldsymbol{\omega}}(y_{1-a^{(j)}}^{(j)}; x^{(j)})\right) \tag{4}$$

In the sequel, we refer to this log likelihood objective function as $\mathcal{R}_{\text{RM}}(\boldsymbol{\omega}; \boldsymbol{\theta})$. The dependence on $\boldsymbol{\theta}$ stems from the fact that the data on which this objective is evaluated varies with $\boldsymbol{\theta}$.

After learning $\hat{u}_{\boldsymbol{\omega}}$, Vanilla RLHF fine-tunes the language model $\pi_{\boldsymbol{\theta}}$ by searching for parameters $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, which generate continuations $y \sim \boldsymbol{\theta}(x)$ that maximize the expected value of this learned utility function over the prompt distribution μ:

$$\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \mathop{\mathbb{E}}_{\substack{x \sim \mu \\ y \sim \boldsymbol{\theta}(x)}} \Big[\hat{u}_{\boldsymbol{\omega}}(y; x)\Big] \tag{5}$$

5 Alignment as a Stackelberg game

Given a language model environment $\mathcal{E} = (\mathcal{D}, \mu, \succ)$, an *alignment game* $(\boldsymbol{\Theta}, \Omega, \mathcal{R}_{\mathrm{LM}}, \mathcal{R}_{\mathrm{RM}}; \mathcal{E})$ comprises two players: a leader LM who seeks to set the parameters $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ of a policy $\pi_{\boldsymbol{\theta}}$, which, given a prompt $x \in \mathcal{D}$, outputs a continuation $y \sim \boldsymbol{\theta}(x)$; and a follower RM who seeks to set the parameters $\boldsymbol{\omega} \in \Omega$ of a reward model $r_{\boldsymbol{\omega}}$ (i.e., a utility function) that represents $\succ$.

At a Nash equilibrium, both players' strategies are optimal given one another's. In contrast, at a Stackelberg equilibrium, only the follower optimizes given the leader's strategy; the leader then optimizes over all possible follower's best responses. We thus parameterize the RM's objective function $\mathcal{R}_{\mathrm{RM}}$ by the LM's policy: i.e., its parameters $\boldsymbol{\theta}$.

In an alignment game, the RM's objective $\mathcal{R}_{\mathrm{RM}}$ is to learn a reward model $r_{\boldsymbol{\omega}}$ that well approximates $\succ$ on a data set $\mathcal{C}$ comprising prompts $x \sim \mu$ and ensuing pairs of continuations $(y_0, y_1) \sim \boldsymbol{\theta}(x)$. The LM's objective $\mathcal{R}_{\mathrm{LM}}$ is then to generate continuations that optimize this reward model $r_{\boldsymbol{\omega}}$:

$$\mathcal{R}_{\mathrm{LM}}(\boldsymbol{\theta}, \boldsymbol{\omega}) \doteq \max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \mathop{\mathbb{E}}_{\substack{x \sim \mu \\ y \sim \boldsymbol{\theta}(x)}} \left[r_{\boldsymbol{\omega}}(y; x) \right] \tag{6}$$

Taking as our starting points Assumptions 1 and 2, we choose Equation (4) as the RM's objective, $\mathcal{R}_{\mathrm{RM}}(\boldsymbol{\omega}; \boldsymbol{\theta})$. The alignment game formulation, however, is not wedded to this particular choice.

We cast the problem of solving an alignment game as the following bi-level optimization problem:

$$\begin{aligned} &\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \mathcal{R}_{\mathrm{LM}}(\boldsymbol{\theta}, \boldsymbol{\omega}^*) \\ \text{subject to } \; &\boldsymbol{\omega}^* \in \arg\max_{\boldsymbol{\omega} \in \Omega} \mathcal{R}_{\mathrm{RM}}(\boldsymbol{\omega}; \boldsymbol{\theta}) \end{aligned} \tag{7}$$

A solution to this problem, when one exists, is called a *Stackelberg equilibrium*. By the extreme value theorem, we can guarantee the existence of a solution to a maximization problem whenever the objective function is continuous in all its arguments and the space of possible solutions is compact. But even if we assume these conditions hold for the follower's optimization problem, it can still be difficult to ensure the leader's optimization problem has a solution, because the leader's objective function must be continuous in the follower's solution. Moreover, the follower's solution need not be unique, in which case we must ensure that the leader's objective function is continuous in the follower's solution *correspondence*![1]

Although general-sum Stackelberg equilibria are computable via linear programming in polynomial time in two-player discrete-action games (Blum et al., 2019; Conitzer and Sandholm, 2006), little is known about how to compute Stackelberg equilibria in two-player continuous-action games, such as those played by two neural networks. Consequently, we developed a heuristic (Algorithm 1) for computing a Stackelberg equilibrium of an alignment game. This heuristic carries over the intuition for solving zero-sum Stackelberg games via nested rather than simultaneous gradient-descent algorithms to general-sum games Goktas and Greenwald (2021). Specifically, for each update step of the leader's policy, we allow the follower multiple update steps, during which time it searches for a best response to the leader's current policy.

As our heuristic is a first-order (i.e., gradient-based) method, it is necessary to compute the gradients of $\mathcal{R}_{\mathrm{RM}}$ (Equation (4)) and $\mathcal{R}_{\mathrm{LM}}$ (Equation (6)). Computing the former is straightforward. To compute the latter, we employ the standard REINFORCE estimate (Sutton et al., 1999; Williams, 1992).[2] Furthermore, we regularize the LM's objective with a KL-divergence term, which ensures

[1] Alternatively, but less generally, we can impose a deterministic tie-breaking rule on the follower's choice that ensures that the leader's objective function is continuous in the follower's now unique solution.

[2] For details, see Appendix A.1, Equation (17).

that the language model does not stray too far from the reference model:

$$\mathcal{R}^{\beta}_{\mathrm{LM}}(\boldsymbol{\theta}, \boldsymbol{\omega}; \pi_{\mathrm{ref}}) \doteq \max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \mathop{\mathbb{E}}_{\substack{x \sim \mu \\ y \sim \boldsymbol{\theta}(x)}} \Big[r_{\boldsymbol{\omega}}(y; x) \Big] - \beta \, \mathbb{D}_{\mathrm{KL}}(\pi_{\boldsymbol{\theta}}(y^{(j)}; x^{(j)}) \mid\mid \pi_{\mathrm{ref}}(y^{(j)}; x^{(j)})) \tag{8}$$

This regularization term also mitigates reward hacking (Laidlaw et al., 2024) and maintains diversity in the generated continuations (Rafailov et al., 2023). This method is our Naive STA-RLHF algorithm.

Motivated by Stackelberg Actor-Critic (Zheng et al., 2022), which recognises the interdependence between the actor's objective and the critic's objective in reinforcement learning, we also introduce Total STA-RLHF. The leader incorporates the fact that the follower's optimization is implicitly defined by the leader's own optimization. Therefore, the leader still uses the KL-regularized REINFORCE estimator, but also includes an additional derivative which takes the gradient of the follower into account[3].

Although our algorithm is iterative, it is not necessary to seek human feedback during each iteration. While we could rank preferences at each step with a human labeler, this is not necessary. On the contrary, we run a pre-processing step in which we solve Equation (4) for $\hat{u}$, exactly as in Vanilla RLHF, and then we seed the heuristic with a *learned language model environment* $\hat{\mathcal{E}} = (\mathcal{D}, \mu, \hat{u})$, instead of a language model environment $\mathcal{E} = (\mathcal{D}, \mu, \succ)$. We can then rank proposed continuations according to these learned utilities, rather than $\succ$ (or repeatedly querying humans). Our algorithm is flexible in that $\hat{u}$ can be instantiated by $\succ$, direct principal feedback (Castricato et al., 2024), or constitutional AI (Bai et al., 2022).

Algorithm 1 STA-RLHF

Input: Learned language model environment $\hat{\mathcal{E}} = (\mathcal{D}, \mu, \hat{u})$, batch size B, number of outer and inner iterations T, S, reference model π_{ref}, and learning rate schedules $\{\eta^t\}_{t=1}^{T}$, $\{\eta^s\}_{s=1}^{S}$
Output: Fine-tuned language model $\pi_{\boldsymbol{\theta}}^{T}$
Initialize the language model $\pi_{\boldsymbol{\theta}}^{0}$ with the reference model π_{ref}
for $t = 1$ to T iterations **do**
  Sample a batch $\mathcal{B} = \{x^{(i)}\}_{i=1}^{B}$ of prompts from $\mathcal{D}$ according to μ
  Run $\boldsymbol{\theta}^{t-1}$ on all prompts $x^{(i)}$ in $\mathcal{B}$ twice to generate a pair of continuations $(y_0^{(i)}, y_1^{(i)})$
  Rank continuations using the utility function $\hat{u}$ and store as $\mathcal{C} = \left\{ \left(x^{(i)}, y_a^{(i)}, y_{a-1}^{(i)} \right) \right\}_{i=1}^{B}$
  $\boldsymbol{\omega}^{t-1,0} \leftarrow \boldsymbol{\omega}^{t-1}$
  for $s = 1$ to S iterations **do**
    Compute the gradient of $\mathcal{R}^{\lambda}_{\mathrm{RM}}(\boldsymbol{\omega}^{t-1,s-1}; \boldsymbol{\theta}^{t-1})$ on $\mathcal{C}$
    $\boldsymbol{\omega}^{t-1,s} \leftarrow \boldsymbol{\omega}^{t-1,s-1} - \eta^s \nabla_{\boldsymbol{\omega}} \mathcal{R}^{\lambda}_{\mathrm{RM}}(\boldsymbol{\omega}^{t-1,s-1}; \boldsymbol{\theta}^{t-1})$
  end for
  $\boldsymbol{\omega}^{t} \leftarrow \boldsymbol{\omega}^{t-1,S}$
  Compute the gradient of $\mathcal{R}^{\beta}_{\mathrm{LM}}(\boldsymbol{\theta}^{t-1}, \boldsymbol{\omega}^{t}; \pi_{\mathrm{ref}})$ on $\mathcal{C}$ (Equation (11))
  $\boldsymbol{\theta}^{t} \leftarrow \boldsymbol{\theta}^{t-1} + \eta^t \nabla_{\boldsymbol{\theta}} \mathcal{R}^{\beta}_{\mathrm{LM}}(\boldsymbol{\theta}^{t-1}, \boldsymbol{\omega}^{t}; \pi_{\mathrm{ref}})$
end for
return Final fine-tuned language model $\pi_{\boldsymbol{\theta}}^{T}$

Competing approaches In Vanilla RLHF and DPO Rafailov et al. (2023), a reward model is trained on data, i.e., continuations, generated by the reference model. Similar to our approach, ReST (Gulcehre et al., 2023) addresses the static nature of the continuations by sampling new continuations from the latest LM. ReST does not build a corresponding reward model, however; it simply continues to optimize its initial objective on new data. Algorithm 1 similarly samples new

[3]shown in Appendix A.1

continuations from the latest LM, based on which it learns a new reward model, so that it always updates the LM using the latest reward signal.

In PARL (Chakraborty et al., 2023), RLHF is formulated as a bi-level optimization, in which the players roles are the reverse of those in our Stackelberg game formulation. A nested heuristic is also proposed to solve PARL, but unlike in our method, multiple batches of trajectories are sampled in the inner loop, where the LM is updated. As the RM is updated in the outer loop, and on fewer batches of data, the frequency of queries to the preference oracle (e.g., $\succ$ or $\hat{u}$) is reduced in PARL, but the LM may be updated based on an outdated reward signal. We compare our heuristic to both PARL and a reversed version of our game/heuristic, with the LM updated in the inner loop and the RM, in the outer.

We also compare against a simultaneous variant of Algorithm 3 with only one loop, where one update step is performed on the LM (RM), based on the previous RM (LM). In this variant, we run our algorithm with $S = 1$ and update the LM (i.e., compute the gradient of $\mathcal{R}^{\beta}_{\text{LM}}$) using $r_{\boldsymbol{\omega}^{t-1}}$ instead of $r_{\boldsymbol{\omega}^{t}}$. The simultaneous variant is intended to find a Nash equilibrium of a simultaneous-move game played between the LM and the RM. This approach is distinct from SPO (Swamy et al., 2024) and Nash-RLHF (Munos et al., 2023), both of which aim to compute a Nash equilibrium between dueling LM policies.

6 Experiments

To evaluate the effectiveness of STA-RLHF, we conduct empirical comparisons against state-of-the-art RLHF variants, such as DPO (Rafailov et al., 2023) and PPO-based Vanilla RLHF. We also compare our methods with iterated DPO where we retrain a DPO model after sampling new completions from the current policy[4] Additionally, we compare our algorithm to its simultaneous and reversed variants, as well as PARL, another variant with the players' roles reversed, which aligns with the problem formulation proposed by Chakraborty et al. (2023).

In all experiments, we seed the algorithms with a ground truth preference relation $\succ$, rather than an approximation $\hat{u}$, and we assess how well the various approaches learn these ground truth preferences, by measuring the utility achieved as a function of size of the preference data set, i.e., the number of examples seen. Following Rafailov et al. (2023), we also evaluate the Pareto front between reward and KL-divergence from the reference policy, as it may be possible to achieve high reward by diverging substantially from the reference policy, but this could result in a fine-tuned model that is no longer tethered to the English language.

Experimental setup Unless stated otherwise, all experiments were run using $\beta = 0.1$, a batch size of 16, RMSprop as the optimizer for the LM, and a learning rate of 1e-6. For Total STA-RLHF, Naive STA-RLHF, Reverse, Simultaneous, and our PARL implementation, we use AdamW as the optimizer for the reward model with a constant learning rate of 1e-5. For Total STA-RLHF, we compute the Hessian vector product term in the total gradient with the conjugate gradient method with 10 iterations.

The number of inner iterations is set to $S = 5$, except in Simultaneous, where $S = 1$. When generating continuations, we generate four instead of two continuations and select the continuations with the highest and lowest ground truth rewards as the more and less preferred continuations.

Our choice of reference model varied with the task, but in all experiments, the LM shares the network architecture of the reference model. The RM appends to this architecture a classification head, which we implemented using `AutoModelForSequenceClassification` from the HuggingFace Transformers library (Wolf et al., 2020).

For PPO, we used ground-truth reward instead of a reward model, and the following hyperparameter values: the clipping paramater $\epsilon = 0.2$, 4 PPO update steps per trajectory batch, and AdamW as

[4] Our code can be found at https://github.com/jacobmakar/stackelberg-rlhf

the optimizer of the critic model with a constant learning rate of 1e-5. The critic model consists of two fully connected layers: the first layer maps the model output size to a hidden size of 256, followed by a ReLU activation function; the second layer maps the hidden size to a single output value.

We run all methods five times on seeds `[467, 532, 6518, 7107, 8688]`. All experiments were conducted using GeForce 3090 or equivalent GPUs. Running a single seed for one setting and one algorithm generally required one GPU. No optimization of hyper-parameters for STA-RLHF or any other methods was attempted.

If a pair of continuations had the same score, one of them was replaced such that there were no ties, as is consistent with the Bradley-Terry preference model and Rafailov et al. (2023).

Word collector In our second set of experiments, Word Collector, we adopt the experimental setup from Singhal et al. (2024). Given a set of words $W = \{w_1, w_2, \ldots, w_{30}\}$, the goal is to construct a text output y comprising 75 tokens, which incorporate as many unique words from W as possible. Specifically, for any two outputs y_1 and y_2, $y_1 \succeq y_2$ if the number of words from W included in y_1 is greater than or equal to the number included in y_2. Each unique additional word in W included in y increases the value of y by 1, up to a maximum value of 30. In these experiments, we use OPT-125m as a reference model, the top 30 content words in the UltraFeedback dataset (Cui et al., 2023) as W, and prompts from the UltraChat dataset (Ding et al., 2023).

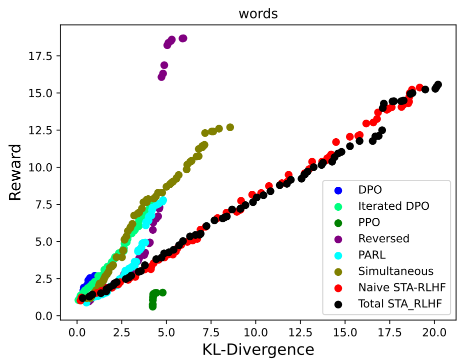
Constrained word collector We expand on the Word Collector setting with the introduction of an additional constraint, namely that the continuation should not contain any words in the prompt. Given a set of 30 target words $W = \{w_1, w_2, \ldots, w_{30}\}$, the objective remains to construct a text output y comprising 75 tokens, which incorporate as many unique words from W as possible. However, in this setting, including any words in the prompt in y results in a penalty. Specifically, for any two outputs y_1 and y_2 given prompt x, $y_1 \succeq y_2$ if the number of words from $W \setminus x$ included in y_1 minus the number of words from x included in y_1 is greater than or equal to the number included in y_2. As in the Word collector task, we use OPT-125m as a reference model, the top 30 content words in the UltraFeedback dataset (Cui et al., 2023) as W, and prompts from the UltraChat dataset (Ding et al., 2023).

Unique nouns Finally, we consider a second experimental setup from Singhal et al. (2024). In this setting, the objective is to align the LM so that it constructs a continuation with as many unique nouns as possible. In these experiments, we use OPT-125m as a reference model, Natural Language Toolkit (NLTK) (Loper and Bird, 2002) for noun detection, and prompts from the UltraChat dataset (Ding et al., 2023).

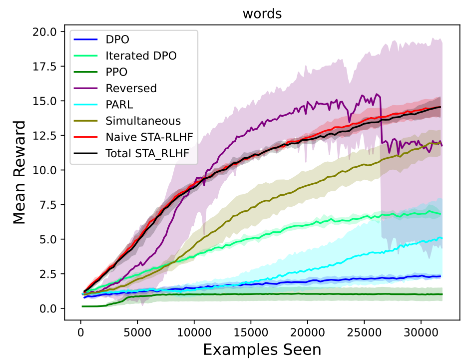
Results We calculate the Pareto frontier of KL divergence vs. reward on a fixed set of test prompts. Potential points are calculated by averaging the KL divergence from the reference policy (x-dimension) and averaging the reward on the test set (y-dimension), across all five seeds throughout training. Pareto-dominated points—those for which another point exists that is weakly better in both dimensions of comparison—are then excluded, so that we plot only the Pareto frontier. We also calculate the reward with respect to number of examples seen. We plot the mean reward with shading indicating the standard deviation across the five seeds.

In all tasks, both versions of STA-RLHF achieve better rewards than the all other methods. In word collector and constrained word collector, the the STA-RLHF methods are Pareto dominated indicating that they modify the LM further away from the reference policy, however in unique nouns their Pareto front is similar to the other methods while also achieving a much higher reward. One explanation for why Total STA-RLHF does not convincingly outperform the Naive counterpart may be because of the imprecision with approximating the inverse Hessian vector product of a large neural network.

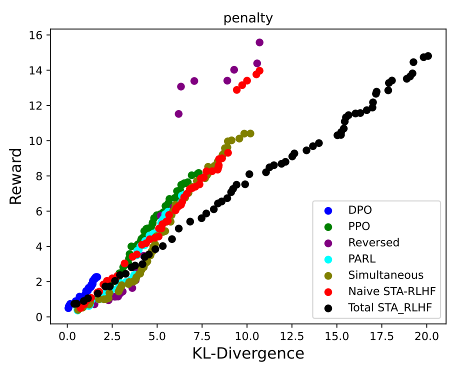
Pareto frontier **Training curve**



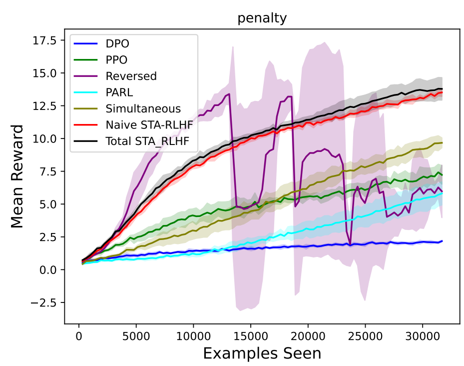
(a) Word collector



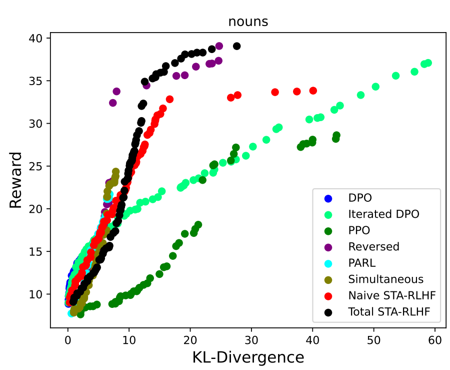
(b) Word collector



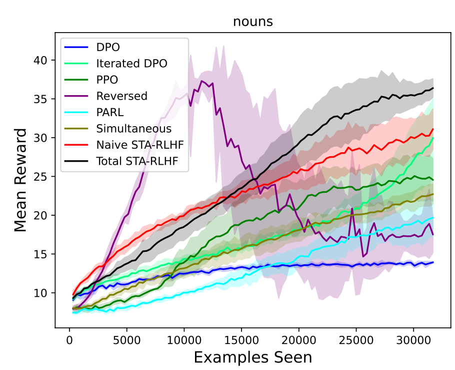
(c) Constrained word collector



(d) Constrained word collector



(e) Unique nouns



(f) Unique nouns

Figure 1: Comparison of competing approaches. STA-RLHF, shown in red and black, consistently achieves the highest rewards after sufficiently long training (i.e., after seeing sufficiently many preference examples), while Reversed starts off competitive, but becomes unstable.

7 Conclusion

In this paper, we formalize the RLHF alignment problem as a Stackelberg game. The leader in our game is the LM, while the follower is the RM. A Stackelberg equilibrium of this game is a fine-tuned language model that optimizes a reward model which is intended to represent human preferences. The key difference between our framework and others is that our search for an optimal LM induces a search over RMs, in a way that is analogous actor-critic methods, where an actor's search for an optimal policy induces a search over value functions. The result of this search is a LM with a higher reward than is achieved by other game-theoretic formulations.

To solve our game, we introduce STA-RLHF, a pair of novel training algorithms for aligning large language models with human preferences. Our findings indicate that STA-RLHF enhances performance in a variety of synthetic settings when compared to traditional RLHF methods such as DPO and PPO. We further validate our approach by reversing the roles of the players, and by training both the language and the reward models simultaneously.

We believe that our approach is theoretically sound, as it takes a game-theoretic perspective, seeking an equilibrium of the Stackelberg alignment game. Indeed, we empirically demonstrate solid performance of STA-RLHF on synthetic tasks, as compared to other perhaps less principled approaches. That said, we are do not provide any convergence guarantees. Our algorithm does appear to converge, as shown in Figure 2, so it may be possible to prove that our method converges to a Stackelberg equilibrium in the future.

Furthermore, our experiments are run on relatively small models, compared to the massive language models (with billions of parameters) now available. Further validation of our approach on larger models and additional tasks are other avenues for future work, although we would expect STA-RLHF to continue to exhibit solid performance across additional tasks, as shown here.

Although we assume a Bradley-Terry model of human preferences, our game formulation is agnostic to the preference structure (unlike DPO). In future work, we intend to experiment with other preference models, such as the Kahneman-Tversky model of human utility (Ethayarajh et al., 2024). Future work could also consider a more general Stackelberg game framework, with multiple followers (i.e., reward models), where the language model seeks to optimize some aggregation of the followers' preferences.

References

Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. A General Theoretical Paradigm to Understand Learning from Human Preferences, November 2023. URL http://arxiv.org/abs/2310.12036. arXiv:2310.12036 [cs, stat].

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional AI: Harmlessness from AI Feedback, December 2022.

Avrim Blum, Nika Haghtalab, MohammadTaghi Hajiaghayi, and Saeed Seddighin. Computing stackelberg equilibria of large general-sum games. In *Algorithmic Game Theory: 12th International Symposium, SAGT 2019, Athens, Greece, September 30–October 3, 2019, Proceedings 12*, pages 168–182. Springer, 2019.

Louis Castricato, Nathan Lile, Suraj Anand, Hailey Schoelkopf, Siddharth Verma, and Stella Biderman. Suppressing Pink Elephants with Direct Principle Feedback, February 2024. URL http://arxiv.org/abs/2402.07896. arXiv:2402.07896 [cs].

Souradip Chakraborty, Amrit Singh Bedi, Alec Koppel, Dinesh Manocha, Huazheng Wang, Mengdi Wang, and Furong Huang. PARL: A Unified Framework for Policy Alignment in Reinforcement Learning, October 2023. URL http://arxiv.org/abs/2308.02585. arXiv:2308.02585 [cs].

Souradip Chakraborty, Jiahao Qiu, Hui Yuan, Alec Koppel, Furong Huang, Dinesh Manocha, Amrit Singh Bedi, and Mengdi Wang. MaxMin-RLHF: Towards Equitable Alignment of Large Language Models with Diverse Human Preferences, February 2024. URL http://arxiv.org/abs/2402.08925. arXiv:2402.08925 [cs].

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-Play Fine-Tuning Converts Weak Language Models to Strong Language Models, February 2024. URL http://arxiv.org/abs/2401.01335. arXiv:2401.01335 [cs, stat].

Brian Christian. *The alignment problem: How can machines learn human values?* Atlantic Books, 2021.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce*, pages 82–90, 2006.

Vincent Conitzer, Rachel Freedman, Jobst Heitzig, Wesley H. Holliday, Bob M. Jacobs, Nathan Lambert, Milan Mossé, Eric Pacuit, Stuart Russell, Hailey Schoelkopf, Emanuel Tewolde, and William S. Zwicker. Social Choice for AI Alignment: Dealing with Diverse Human Feedback, April 2024. URL http://arxiv.org/abs/2404.10271. arXiv:2404.10271 [cs].

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. Ultrafeedback: Boosting language models with high-quality feedback, 2023.

Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training gans with optimism. *arXiv preprint arXiv:1711.00141*, 2017.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*, 2023.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. RAFT: Reward rAnked FineTuning for Generative Foundation Model Alignment, December 2023. URL http://arxiv.org/abs/2304.06767. arXiv:2304.06767 [cs, stat] version: 4.

Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. RLHF Workflow: From Reward Modeling to Online RLHF. https://arxiv.org/abs/2405.07863v1, May 2024.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. KTO: Model Alignment as Prospect Theoretic Optimization, February 2024. URL http://arxiv.org/abs/2402.01306. arXiv:2402.01306 [cs].

FAIR, Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, et al. Human-level play in the game of diplomacy by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074, 2022.

Denizalp Goktas and Amy Greenwald. Convex-Concave Min-Max Stackelberg Games. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 2991–3003. Curran Associates, Inc., 2021.

Denizalp Goktas, Arjun Prakash, and Amy Greenwald. Convex-concave zero-sum markov stackelberg games. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 66818–66832. Curran Associates, Inc., 2023.

I. J. Good. On the marking of chess-players. *The Mathematical Gazette*, 39(330):292–296, 1955. ISSN 00255572. URL http://www.jstor.org/stable/3608567.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud Doucet, Orhan Firat, and Nando de Freitas. Reinforced Self-Training (ReST) for Language Modeling, August 2023. URL http://arxiv.org/abs/2308.08998. arXiv:2308.08998 [cs].

Ian Hamilton, Nick Tawn, and David Firth. The many routes to the ubiquitous bradley-terry model. *arXiv preprint arXiv:2312.13619*, 2023.

Shengyi Huang, Michael Noukhovitch, Arian Hosseini, Kashif Rasul, Weixun Wang, and Lewis Tunstall. The n+ implementation details of rlhf with ppo: A case study on tl;dr summarization, 2024.

Cassidy Laidlaw, Shivam Singhal, and Anca Dragan. Preventing reward hacking with occupancy measure regularization, 2024.

Edward Loper and Steven Bird. NLTK: the natural language toolkit. *CoRR*, cs.CL/0205028, 2002. URL https://arxiv.org/abs/cs/0205028.

Richard D McKelvey and Thomas R Palfrey. Quantal response equilibria for normal form games. *Games and economic behavior*, 10(1):6–38, 1995.

Rémi Munos, Michal Valko, Daniele Calandriello, Mohammad Gheshlaghi Azar, Mark Rowland, Zhaohan Daniel Guo, Yunhao Tang, Matthieu Geist, Thomas Mesnard, Andrea Michi, Marco Selvi, Sertan Girgin, Nikola Momchev, Olivier Bachem, Daniel J. Mankowitz, Doina Precup, and Bilal Piot. Nash Learning from Human Feedback, December 2023. URL http://arxiv.org/abs/2312.00886. arXiv:2312.00886 [cs, stat].

John F Nash. Non-cooperative games. 1950.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, March 2022.

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pages 745–750, Corvalis Oregon USA, June 2007. ACM. ISBN 978-1-59593-793-3. doi: 10.1145/1273496.1273590. URL https://dl.acm.org/doi/10.1145/1273496.1273590.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct Preference Optimization: Your Language Model is Secretly a Reward Model, December 2023. URL http://arxiv.org/abs/2305.18290. arXiv:2305.18290 [cs].

Rafael Rafailov, Joey Hejna, Ryan Park, and Chelsea Finn. From \$r\$ to \$Q^*\$: Your Language Model is Secretly a Q-Function, April 2024. URL http://arxiv.org/abs/2404.12358. arXiv:2404.12358 [cs].

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Prasann Singhal, Nathan Lambert, Scott Niekum, Tanya Goyal, and Greg Durrett. D2po: Discriminator-guided dpo with response evaluation models, 2024.

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

Jonathan Stray, Ivan Vendrov, Jeremy Nixon, Steven Adler, and Dylan Hadfield-Menell. What are you optimizing for? aligning recommender systems with human values. *CoRR*, abs/2107.10939, 2021. URL https://arxiv.org/abs/2107.10939.

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

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Gokul Swamy, Christoph Dann, Rahul Kidambi, Zhiwei Steven Wu, and Alekh Agarwal. A Minimaximalist Approach to Reinforcement Learning from Human Feedback, January 2024. URL http://arxiv.org/abs/2401.04056. arXiv:2401.04056 [cs].

Fahim Tajwar, Anikait Singh, Archit Sharma, Rafael Rafailov, Jeff Schneider, Tengyang Xie, Stefano Ermon, Chelsea Finn, and Aviral Kumar. Preference Fine-Tuning of LLMs Should Leverage Suboptimal, On-Policy Data, April 2024. URL http://arxiv.org/abs/2404.14367. arXiv:2404.14367 [cs].

Yunhao Tang, Daniel Zhaohan Guo, Zeyu Zheng, Daniele Calandriello, Yuan Cao, Eugene Tarassov, Rémi Munos, Bernardo Ávila Pires, Michal Valko, Yong Cheng, et al. Understanding the performance gap between online and offline alignment algorithms. *arXiv preprint arXiv:2405.08448*, 2024.

John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, 1947. ISBN 9781400829460. doi: doi:10.1515/9781400829460. URL https://doi.org/10.1515/9781400829460.

H. von Stackelberg. *Marktform und Gleichgewicht*. Die Handelsblatt-Bibliothek "Klassiker der Nationalökonomie". J. Springer, 1934. URL https://books.google.com/books?id=wihBAAAAIAAJ.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, and Shengyi Huang. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl, 2020.

Norbert Wiener. Some Moral and Technical Consequences of Automation. *Science*, 131(3410): 1355–1358, May 1960. doi: 10.1126/science.131.3410.1355. URL https://www.science.org/doi/10.1126/science.131.3410.1355. Publisher: American Association for the Advancement of Science.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Wei Xiong, Hanze Dong, Chenlu Ye, Ziqi Wang, Han Zhong, Heng Ji, Nan Jiang, and Tong Zhang. Iterative preference learning from human feedback: Bridging theory and practice for rlhf under kl-constraint, 2024.

Shusheng Xu, Wei Fu, Jiaxuan Gao, Wenjie Ye, Weilin Liu, Zhiyu Mei, Guangju Wang, Chao Yu, and Yi Wu. Is DPO Superior to PPO for LLM Alignment? A Comprehensive Study, April 2024. URL http://arxiv.org/abs/2404.10719. arXiv:2404.10719 [cs].

Liyuan Zheng, Tanner Fiez, Zane Alumbaugh, Benjamin Chasnov, and Lillian J Ratliff. Stackelberg actor-critic: Game-theoretic reinforcement learning algorithms. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 9217–9224, 2022.

A The Total Derivative

The LM's objective is to choose parameters $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, which define a policy $\pi^{\boldsymbol{\theta}}$ for the LM that maximizes

$$\mathcal{R}_{\mathrm{LM}}(\boldsymbol{\theta}, \boldsymbol{\omega}^*) \doteq \mathop{\mathbb{E}}_{\substack{x \sim \mu \\ y \sim \pi_{\boldsymbol{\theta}}(x)}} \Big[r_{\boldsymbol{\omega}^*}(y; x) \Big] \tag{9}$$

Here, $\boldsymbol{\omega}^*$ is an optimal choice for the RM.

The RM's objective is to choose parameters $\boldsymbol{\omega}$ that maximize

$$\mathcal{R}_{\mathrm{RM}}(\boldsymbol{\omega}; \boldsymbol{\theta}) \doteq \mathop{\mathbb{E}}_{\substack{x \sim \mu \\ y_0, y_1 \sim \pi_{\boldsymbol{\theta}}(x)}} \left[\log \sigma \left(r_{\boldsymbol{\omega}}(y_{a^{(*)}}; x) - r_{\boldsymbol{\omega}}(y_{1-a^{(*)}}; x) \right) \right] \ , \tag{10}$$

where, $a^{(*)} \in \arg\max_{a \in \{0,1\}} \hat{u}(y_a)$.

As we compute a Monte Carlo estimate of this gradient from samples x, y_0, and y_1, it is possible to sort each pair of continuations y_0, and y_1 according to $\hat{u}$ to identify $y_{a^{(*)}}$ (preferred) and $y_{1-a^{(*)}}$.

After appling the Inverse function theorem (Zheng et al., 2022), total derivative of the LM's objective function is:

$$\nabla_{\boldsymbol{\theta}} \mathcal{R}_{\mathrm{LM}}(\boldsymbol{\theta}, \boldsymbol{\omega}^*) - \nabla^T_{\boldsymbol{\omega}\boldsymbol{\theta}} \mathcal{R}_{\mathrm{RM}}(\boldsymbol{\omega}; \boldsymbol{\theta}) (\nabla^2_{\boldsymbol{\omega}} \mathcal{R}_{\mathrm{RM}}(\boldsymbol{\omega}; \boldsymbol{\theta}))^{-1} \nabla_{\boldsymbol{\omega}} \mathcal{R}_{\mathrm{LM}}(\boldsymbol{\theta}, \boldsymbol{\omega}^*) \tag{11}$$

Note that only using $\nabla_{\boldsymbol{\theta}} \mathcal{R}_{\mathrm{LM}}(\boldsymbol{\theta}, \boldsymbol{\omega}^*)$ results in our naive STA-RLHF algorithm.

A.1 The REINFORCE Estimator: The Gradient of the LM's Objective $\mathcal{R}_{\text{LM}}(\boldsymbol{\theta}, \boldsymbol{\omega}^*)$ wrt $\boldsymbol{\theta}$

By the policy gradient theorem Williams (1992); Sutton and Barto (2018) the gradient of this objective is:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \mathcal{R}_{\text{LM}}(\boldsymbol{\theta}, \boldsymbol{\omega}^*) &= \nabla_{\boldsymbol{\theta}} \underset{\substack{x \sim \mu \\ y \sim \pi_{\boldsymbol{\theta}}(x)}}{\mathbb{E}} \Big[r_{\boldsymbol{\omega}^*}(y; x) \Big] && (12)\\
&= \underset{\substack{x \sim \mu \\ y \sim \pi_{\boldsymbol{\theta}}(x)}}{\mathbb{E}} \Big[\nabla_{\boldsymbol{\theta}} r_{\boldsymbol{\omega}^*}(y; x) \Big] && (13)\\
&= \underset{x \sim \mu}{\mathbb{E}} \int_{y \in \mathcal{D}} \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(x)\, r_{\boldsymbol{\omega}^*}(y; x)\, \mathrm{d}y && (14)\\
&= \underset{x \sim \mu}{\mathbb{E}} \int_{y \in \mathcal{D}} \pi_{\boldsymbol{\theta}}(x) \frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(x)}{\pi_{\boldsymbol{\theta}}(x)}\, r_{\boldsymbol{\omega}^*}(y; x)\, \mathrm{d}y && (15)\\
&= \underset{x \sim \mu}{\mathbb{E}} \int_{y \in \mathcal{D}} \pi_{\boldsymbol{\theta}}(x)\, \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(x)\, r_{\boldsymbol{\omega}^*}(y; x)\, \mathrm{d}y && (16)\\
&= \underset{\substack{x \sim \mu \\ y \sim \pi_{\boldsymbol{\theta}}(x)}}{\mathbb{E}} \Big[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y; x)\, r_{\boldsymbol{\omega}^*}(y; x) \Big]\ , && (17)
\end{aligned}$$

which we estimate by averaging across a batch of samples $\{x^{(j)}, y^{(j)}\}_{j=1}^{B}$, with each $x^{(j)} \sim \mu$ and each $y^{(j)} \sim \pi_{\boldsymbol{\theta}}(x^{(j)})$. Additionally, we regularize this estimator with the KL divergence:

$$\frac{1}{B} \sum_{j=0}^{B} \Big[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y^{(j)}; x^{(j)})\, r_{\boldsymbol{\omega}^*}(y^{(j)}; x^{(j)}) - \beta\, \mathbb{D}_{\text{KL}}(\pi_{\boldsymbol{\theta}}(y^{(j)}; x^{(j)}) \;||\; \pi_{\text{ref}}(y^{(j)}; x^{(j)}) \Big] \tag{18}$$

A.2 The Gradient of the RM's Objective $\mathcal{R}_{\text{RM}}(\boldsymbol{\omega}; \boldsymbol{\theta})$ wrt $\boldsymbol{\omega}$ and $\boldsymbol{\theta}$

We begin by computing the partial wrt $\boldsymbol{\theta}$:

$$\begin{aligned}
&\nabla_{\boldsymbol{\theta}} \underset{\substack{x \sim \mu \\ y_0, y_1 \sim \pi_{\boldsymbol{\theta}}(x)}}{\mathbb{E}} \Big[\log \sigma \Big(r_{\boldsymbol{\omega}}(y_{a^{(*)}}; x) - r_{\boldsymbol{\omega}}(y_{1-a^{(*)}}; x) \Big) \Big] && (19)\\
&= \underset{\substack{x \sim \mu \\ y_0, y_1 \sim \pi_{\boldsymbol{\theta}}(x)}}{\mathbb{E}} \Big[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y_0, y_1; x) \log \sigma \Big(r_{\boldsymbol{\omega}}(y_{a^{(*)}}; x) - r_{\boldsymbol{\omega}}(y_{1-a^{(*)}}; x) \Big) \Big] && (20)\\
&= \underset{\substack{x \sim \mu \\ y_0, y_1 \sim \pi_{\boldsymbol{\theta}}(x)}}{\mathbb{E}} \Big[\nabla_{\boldsymbol{\theta}} \log(\pi_{\boldsymbol{\theta}}(y_0; x) \pi_{\boldsymbol{\theta}}(y_1; x)) \log \sigma \Big(r_{\boldsymbol{\omega}}(y_{a^{(*)}}; x) - r_{\boldsymbol{\omega}}(y_{1-a^{(*)}}; x) \Big) \Big] && (21)\\
&= \underset{\substack{x \sim \mu \\ y_0, y_1 \sim \pi_{\boldsymbol{\theta}}(x)}}{\mathbb{E}} \Big[\nabla_{\boldsymbol{\theta}} (\log \pi_{\boldsymbol{\theta}}(y_0; x) + \log \pi_{\boldsymbol{\theta}}(y_1; x)) \log \sigma \Big(r_{\boldsymbol{\omega}}(y_{a^{(*)}}; x) - r_{\boldsymbol{\omega}}(y_{1-a^{(*)}}; x) \Big) \Big] && (22)\\
&= \underset{\substack{x \sim \mu \\ y_0, y_1 \sim \pi_{\boldsymbol{\theta}}(x)}}{\mathbb{E}} \Big[(\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y_0; x) + \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y_1; x)) \log \sigma \Big(r_{\boldsymbol{\omega}}(y_{a^{(*)}}; x) - r_{\boldsymbol{\omega}}(y_{1-a^{(*)}}; x) \Big) \Big] && (23)\\
&= \underset{\substack{x \sim \mu \\ y_{a^{(*)}}, y_{1-a^{(*)}} \sim \pi_{\boldsymbol{\theta}}(x)}}{\mathbb{E}} \Big[(\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y_{a^{(*)}}; x) + \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y_{1-a^{(*)}}; x)) \log \sigma \Big(r_{\boldsymbol{\omega}}(y_{a^{(*)}}; x) - r_{\boldsymbol{\omega}}(y_{1-a^{(*)}}; x) \Big) \Big] && (24)
\end{aligned}$$

The last line follows from the fact that addition is symmetric; in particular, $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y_0; x) + \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y_1; x) = \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y_1; x) + \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y_0; x)$.

Next, we compute the gradient wrt $\boldsymbol{\omega}$:

$$\nabla_{\boldsymbol{\omega}} \mathop{\mathbb{E}}_{\substack{x\sim\mu \\ y_0,y_1\sim\pi_{\boldsymbol{\theta}}(x)}} \left[(\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y_{a^{(*)}};x) + \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y_{1-a^{(*)}};x)) \log \sigma \left(r_{\boldsymbol{\omega}}(y_{a^{(*)}};x) - r_{\boldsymbol{\omega}}(y_{1-a^{(*)}};x) \right) \right] \tag{25}$$

$$= \mathop{\mathbb{E}}_{\substack{x\sim\mu \\ y_0,y_1\sim\pi_{\boldsymbol{\theta}}(x)}} \left[(\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y_{a^{(*)}};x) + \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y_{1-a^{(*)}};x)) \nabla_{\boldsymbol{\omega}} \log \sigma \left(r_{\boldsymbol{\omega}}(y_{a^{(*)}};x) - r_{\boldsymbol{\omega}}(y_{1-a^{(*)}};x) \right) \right] \tag{26}$$

Finally, by the chain rule,

$$\nabla_{\boldsymbol{\omega}} \log \sigma \left(r_{\boldsymbol{\omega}}(y_{a^{(*)}};x) - r_{\boldsymbol{\omega}}(y_{1-a^{(*)}};x) \right) = \frac{1}{1+e^{r_{\boldsymbol{\omega}}(y_{a^{(*)}};x) - r_{\boldsymbol{\omega}}(y_{1-a^{(*)}};x)}} \left(\nabla_{\boldsymbol{\omega}} r_{\boldsymbol{\omega}}(y_{a^{(*)}};x) - \nabla_{\boldsymbol{\omega}} r_{\boldsymbol{\omega}}(y_{1-a^{(*)}};x) \right) \tag{27}$$

A.3 The Gradient of the RM's Objective $\mathcal{R}_{\mathrm{RM}}(\boldsymbol{\omega};\boldsymbol{\theta})$ wrt $\boldsymbol{\omega}$, twice

Computing the gradient of $\mathcal{R}_{\mathrm{LM}}$ once wrt $\boldsymbol{\omega}$ (once) yields:

$$\nabla_{\boldsymbol{\omega}} \mathop{\mathbb{E}}_{\substack{x\sim\mu \\ y_{a^{(*)}},y_{1-a^{(*)}}\sim\pi_{\boldsymbol{\theta}}(x)}} \left[\log \sigma \left(r_{\boldsymbol{\omega}}(y_{a^{(*)}};x) - r_{\boldsymbol{\omega}}(y_{1-a^{(*)}};x) \right) \right] \tag{28}$$

$$= \mathop{\mathbb{E}}_{\substack{x\sim\mu \\ y_{a^{(*)}},y_{1-a^{(*)}}\sim\pi_{\boldsymbol{\theta}}(x)}} \left[\nabla_{\boldsymbol{\omega}} \log \sigma \left(r_{\boldsymbol{\omega}}(y_{a^{(*)}};x) - r_{\boldsymbol{\omega}}(y_{1-a^{(*)}};x) \right) \right] \tag{29}$$

$$= \mathop{\mathbb{E}}_{\substack{x\sim\mu \\ y_{a^{(*)}},y_{1-a^{(*)}}\sim\pi_{\boldsymbol{\theta}}(x)}} \left[\frac{1}{1+e^{r_{\boldsymbol{\omega}}(y_{a^{(*)}};x) - r_{\boldsymbol{\omega}}(y_{1-a^{(*)}};x)}} \left(\nabla_{\boldsymbol{\omega}} r_{\boldsymbol{\omega}}(y_{a^{(*)}};x) - \nabla_{\boldsymbol{\omega}} r_{\boldsymbol{\omega}}(y_{1-a^{(*)}};x) \right) \right] \tag{30}$$

Using the product rule to compute the gradient a second time wrt $\boldsymbol{\omega}$ yields two terms, which are summed in the final gradient expression, namely

$$\nabla_{\boldsymbol{\omega}} \frac{1}{1+e^{r_{\boldsymbol{\omega}}(y_{a^{(*)}};x) - r_{\boldsymbol{\omega}}(y_{1-a^{(*)}};x)}} \left(\nabla_{\boldsymbol{\omega}} r_{\boldsymbol{\omega}}(y_{a^{(*)}};x) - \nabla_{\boldsymbol{\omega}} r_{\boldsymbol{\omega}}(y_{1-a^{(*)}};x) \right) \tag{31}$$

$$= \frac{e^{r_{\boldsymbol{\omega}}(y_{1-a^{(*)}};x) - r_{\boldsymbol{\omega}}(y_{a^{(*)}};x)}}{(1+e^{r_{\boldsymbol{\omega}}(y_{1-a^{(*)}};x) - r_{\boldsymbol{\omega}}(y_{a^{(*)}};x)})^2} \left(\nabla_{\boldsymbol{\omega}} r_{\boldsymbol{\omega}}(y_{a^{(*)}};x) - \nabla_{\boldsymbol{\omega}} r_{\boldsymbol{\omega}}(y_{1-a^{(*)}};x) \right) \tag{32}$$

$$+ \frac{1}{1+e^{r_{\boldsymbol{\omega}}(y_{a^{(*)}};x) - r_{\boldsymbol{\omega}}(y_{1-a^{(*)}};x)}} \left(\nabla^2_{\boldsymbol{\omega}} r_{\boldsymbol{\omega}}(y_{a^{(*)}};x) - \nabla^2_{\boldsymbol{\omega}} r_{\boldsymbol{\omega}}(y_{1-a^{(*)}};x) \right) \tag{33}$$

A.4 The Gradient of the LM's Objective $\mathcal{R}_{\mathrm{LM}}(\boldsymbol{\theta},\boldsymbol{\omega}^*)$ wrt $\boldsymbol{\omega}$

$$\nabla_{\boldsymbol{\omega}} \mathcal{R}_{\mathrm{LM}}(\boldsymbol{\theta},\boldsymbol{\omega}^*) = \nabla_{\boldsymbol{\omega}} \mathop{\mathbb{E}}_{\substack{x\sim\mu \\ y\sim\pi_{\boldsymbol{\theta}}(x)}} \left[r_{\boldsymbol{\omega}^*}(y;x) \right] = \mathop{\mathbb{E}}_{\substack{x\sim\mu \\ y\sim\pi_{\boldsymbol{\theta}}(x)}} \left[\nabla_{\boldsymbol{\omega}} r_{\boldsymbol{\omega}^*}(y;x) \right] \tag{34}$$

$$\frac{1}{B} \sum_{j=0}^{B} \left[\nabla_{\boldsymbol{\omega}} r_{\boldsymbol{\omega}^*}(y^{(j)};x^{(j)}) \right] \tag{35}$$

B Additional Results

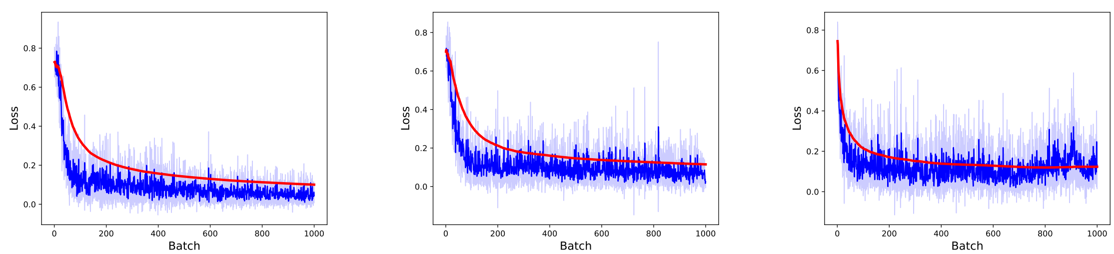
Runtime The table below lists the average running time in seconds of each method on the various tasks, averaged across seeds.

Table 1: Average Experiment Times of Different Algorithms

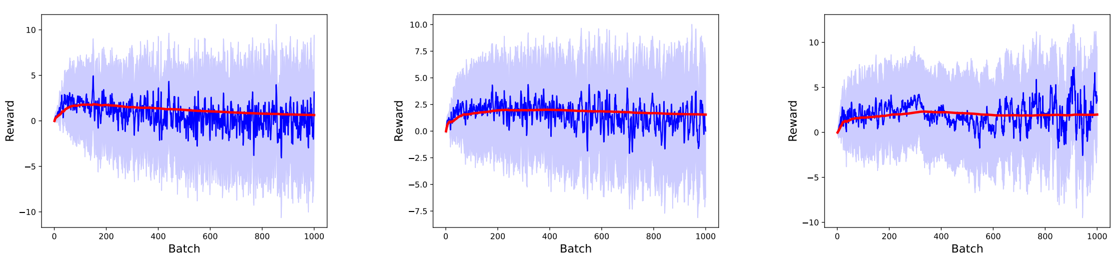
Task	**Total STA-RLHF**	**Naive STA-RLHF**	**PPO**	**DPO**	**Simul.**	**Rev.**	**PARL**
WC	11040.50	2558.152	3489.175	1143.555	2130.730	4102.406	1847.278
CWC	11043.6	2399.765	2004.738	652.607	1340.522	3130.647	1340.522
UN	8220.03	3130.627	2133.094	654.742	2050.985	3370.721	1527.929

In Total STA-RLHF, computing the total gradient involves an Hessian term of the reward model. We accomplish this numerically this by applying the conjugate gradient method to find the inverse Hessian vector product (rather than the finding the entire Hessian).

The following figure shows that in our settings (naive) STA-RLHF empirically converges, as both the reward model and policy converge to a value for the respective objectives.



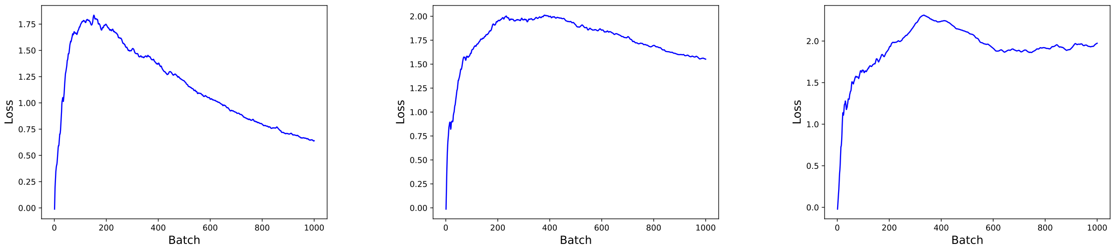
(a) Word collector (b) Constrained words (c) Unique nouns



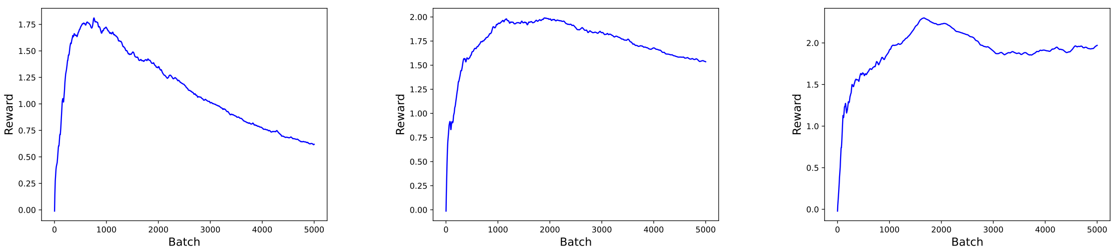
(d) Word collector (e) Constrained words (f) Unique nouns

Figure 2: Loss of the reward model (top) and the rewards received by the policy from the reward model (bottom) during training for STA-RLHF across various settings. Cumulative averages in red.

The following figure shows the cumulative average of Figure 2.



(a) Word collector (b) Constrained words (c) Unique nouns



(d) Word collector (e) Constrained words (f) Unique nouns

Figure 3: Cumulative average of the loss of the reward model (top) and the cumulative average of the rewards received by the policy from the reward model (bottom) during training for STA-RLHF across various settings.

C Algorithm Variants

Included below are two modifications of our heuristic that we run as baselines in our experiments:

Algorithm 2 Reversed

Input: Learned language model environment $\hat{\mathcal{E}} = (\mathcal{D}, \mu, \hat{u})$, batch size B, number of outer and inner iterations T, S, reference model π_{ref}, and learning rate schedules $\{\eta^t\}_{t=1}^T, \{\eta^s\}_{s=1}^S$

Output: Fine-tuned LM $\pi_{\boldsymbol{\theta}}$

Initialize fine-tuned model $\pi_{\boldsymbol{\theta}}^0$ with reference model π_{ref}

for $t = 1$ to T iterations **do**

Run $\boldsymbol{\theta}^{t-1}$ on all prompts $x^{(i)}$ in $\mathcal{B}$ twice to generate a pair of continuations $(y_0^{(i)}, y_1^{(i)})$

Rank continuations using the utility function $\hat{u}$ and store as $\mathcal{C} = \left\{\left(x^{(i)}, y_a^{(i)}, y_{a-1}^{(i)}\right)\right\}_{i=1}^B$

$\boldsymbol{\theta}^{t-1,0} \leftarrow \boldsymbol{\theta}^{t-1}$

for $s = 1$ to S iterations **do**

Compute the gradient of $\mathcal{R}_{\text{LM}}^\beta(\boldsymbol{\theta}^{t-1,s-1}; \pi_{\text{ref}})$ on $\mathcal{C}$ using REINFORCE (Equation (18))

$\boldsymbol{\theta}^{t-1,s} \leftarrow \boldsymbol{\theta}^{t-1,s-1} + \eta^s \nabla_{\boldsymbol{\theta}^{t-1}} \mathcal{R}_{\text{LM}}^\beta(\boldsymbol{\theta}^{t-1})$

end for

$\boldsymbol{\theta}^t \leftarrow \boldsymbol{\theta}^{t-1,S}$

Compute the gradient of $\mathcal{R}_{\text{RM}}(\boldsymbol{\omega}^{t-1}; \boldsymbol{\theta}^{t-1})$ on $\mathcal{C}$ (Equation (4))

$\boldsymbol{\omega}^t \leftarrow \boldsymbol{\omega}^{t-1} - \eta^t \nabla_{\boldsymbol{\omega}} \mathcal{R}_{\text{RM}}(\boldsymbol{\omega}^{t-1}; \boldsymbol{\theta}^{t-1})$

end for

return Final fine-tuned model $\pi_{\boldsymbol{\theta}}^T$

D Policy Samples from Naive STA-RLHF

Included below are policy samples from the experimental settings in our paper.

Algorithm 3 Simultaneous

Input: Learned language model environment $\hat{\mathcal{E}} = (\mathcal{D}, \mu, \hat{u})$, batch size B, number of outer and inner iterations T, reference model π_{ref}, and learning rate schedules $\{\eta^t\}_{t=1}^T, \{\eta^s\}_{s=1}^S$
Output: Fine-tuned LM $\pi_{\boldsymbol{\theta}}$
Initialize fine-tuned model $\pi_{\boldsymbol{\theta}}^0$ with reference model π_{ref}
for $t = 1$ to T iterations **do**
  Sample a batch $\mathcal{B} = \left\{\left(x^{(i)}\right)\right\}_{i=1}^B$ of prompts from $\mathcal{D}$ according to μ
  Run $\boldsymbol{\theta}^{t-1}$ on all prompts $x^{(i)}$ in $\mathcal{B}$ twice to generate a pair of continuations $(y_0^{(i)}, y_1^{(i)})$
  Rank continuations using the utility function $\hat{u}$ and store as $\mathcal{C} = \left\{\left(x^{(i)}, y_a^{(i)}, y_{a-1}^{(i)}\right)\right\}_{i=1}^B$
  Compute the gradient of $\mathcal{R}_{\text{RM}}(\boldsymbol{\omega}^{t-1}; \boldsymbol{\theta}^{t-1})$ on $\mathcal{C}$ (Equation (4))
  $\boldsymbol{\omega}^t \leftarrow \boldsymbol{\omega}^{t-1} - \eta^t \nabla_{\boldsymbol{\omega}} \mathcal{R}_{\text{RM}}(\boldsymbol{\omega}^{t-1}; \boldsymbol{\theta}^{t-1})$
  Compute the gradient of $\mathcal{R}_{\text{LM}}^{\beta}(\boldsymbol{\theta}^{t-1}; \pi_{\text{ref}})$ on $\mathcal{C}$ using REINFORCE (Equation (18))
  $\boldsymbol{\theta}^t \leftarrow \boldsymbol{\theta}^{t-1} + \eta^t \nabla_{\boldsymbol{\theta}^{t-1}} \mathcal{R}_{\text{LM}}^{\beta}(\boldsymbol{\theta}^{t-1})$
end for
return Final fine-tuned model $\pi_{\boldsymbol{\theta}}^T$

Table 2: Naive STA-RLHF samples for word collector

Prompt	**Continuation** (Words in $W = \{w_1, w_2, \ldots, w_{30}\}$ are shown in bold)
What is your personal definition of art?	The invention is, to **help create** the **following** three useful **data** to **ensure** safe and appropriate use of **your data** to deliver appropriate **information** to our customers. We **create** the **following two important first question** to keep **your data** safe to **provide you** appropriate possible **experience** to **ensure** that **your**
What is the role of a motherboard in a computer system?	**Your information** may be useful to **help ensure** the **following** are safe to **provide** the **best** possible **experience**. We **have** the **following two important first question** to **understand** to **generate** the **best** possible **information** to **ensure your data** is safe to **provide** appropriate **information** to **help create** the **best** possible **experience**
What are some of the most famous works of art, and what makes them	valuable to **your data**? We **have** to **provide** the **following two first question** to **help** us **understand understanding your best** possible **experience** to **create** the appropriate **data** to **help ensure your data** is safe to **provide** the **best** possible **experience**. We **have two important questions** to **help ensure your information**

Table 3: Naive STA-RLHF Samples for Constrained Word Collector

Prompt	**Continuation** (Words in $W = \{w_1, w_2, \ldots, w_{30}\}$ are shown in bold)
What is your personal definition of art?	The **following** question is helpful in determining how to get an **appropriate** and **safe information** to **provide your important data** to **create best possible experience** for **your best possible data** to **help ensure** a **safe** and **appropriate experience** to **provide** a **safe**, **appropriate experience** to **create your best possible information** to
What is the role of a motherboard in a computer system?	Bosch is the perfect partner to **have** to **create** a **best possible experience** to **help create important** and meaningful **information** for **your** visitors. We **have created** a series of questions to **help understand important** questions to **help provide important data** to **help create best** customer value. It is
What are some of the most famous works of art, and what makes them	**important** to **help understand** the **experience** of **creating** a **safe** and **appropriate** environment to **help ensure your best possible experience**? The **following** question is **important** to **provide** a good and **important** question to **understand your best possible experience** to **help provide appropriate** and **important information** to **ensure** you **have** to **provide**

Table 4: Naive STA-RLHF samples for unique nouns

Prompt	**Continuation**
Can you speak to the role that technology and social media have played in the	development of public safety programs in the United Nations Economic Development Agency, World Bank and United Nations Health Research Institute Center for Global Public Policy Research (UNC) International Global Public Policy Studies Research Center (U.S. State University, United Nations Environment Science Research and
What are the most promising renewable energy sources in terms of their potential and scal	ability? The United States Department of Health Services National Science Foundation International Center for Global Technology Policy Research Research Society Earth Sciences Science Technology and Engineering Education Sciences Technology Management Services Policy Center National Science Foundation Economic Studies Research Department Office Social Sciences Public Affairs Research Business Administration Social Services Business