STA-RLHF: Stackelberg Aligned Reinforcement Learning with Human Feedback

Jacob Makar-Limanov†‡ Arjun Prakash†‡ Denizalp Goktas‡
Nora Ayanian † Amy Greenwald †

†Brown University, Computer Science
{jacob_makar-limanov, arjun_prakash, denizalp_goktas, amy_greenwald, nora_ayanian}@brown.edu

Abstract

The alignment problem, namely teaching language models human preferences, is a key AI challenge. Most RLHF approaches treat the optimization of the language model and the preference model as separate problems. We propose Stackelberg Alignment Reinforcement Learning from Human Feedback (STA-RLHF), which formalises RLHF as a Stackelberg game between the language model and the preference model. Crucially, our game explicitly takes into account the coupled nature of the optimization process as the two models learn sequentially. We devise a nested gradient descent-ascent algorithm 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 evermore capable language models with human preferences (Bai et al., 2022; Ouyang et al., 2022). The RLHF process usually involves two models: a language model (LM), which generates continuations from prompts, and a preference model (PM) which scores these continuations according to human preferences Ouyang et al. (2022).

In standard RLHF, the preference model is trained first, and then the LM is optimized to generate completions that satisfy the preference model. This separation during training limits the feedback both models receive from each other which could enhance performance (Stiennon et al., 2022). This inherently coupled nature of the behavior of the RM and PM naturally lends itself to a two-player general-sum sequential game i.e. a Stackelberg game. In a Stackelberg game, a leader commits to a strategy before the follower plays their strategy. After both players play their respective actions, they each receive a reward according to their individual utility functions which depends on the joint actions played. The solution concept is a Stackelberg equilibrium where the leader commits to their strategy assuming the follower will best respond. We model RLHF as a Stackelberg game where the leader is an LM who observes a prompt and outputs a continuation. The LM’s utility function is to generate the best possible continuations, given the PM’s representation of the human preferences. The follower is the PM who observes a prompt and two continuations produced by the LM. The PM’s utility function is to best represent the designer’s preferences by selecting the preferred continuation.

In this game-theoretic formulation of RLHF, a question immediately arises as to whether the two players should move simultaneously, or one after the other, corresponding to Nash equilibrium and

†With equal contribution.
‡If both players were simultaneously best-responding the solution concept would be a Nash equilibrium.
A Stackelberg equilibrium\(^2\) as the appropriate solution concepts, respectively. We contend that the game is more naturally modelled as a sequential game. Moreover, in contrast to Chakraborty et al. (2023), we further contend that the LM should be the leader, and the PM, the follower. Our choice of leader and follower is based on the observation that a Stackelberg leader’s utility at a Stackelberg equilibrium (weakly) exceeds her utility at a Nash equilibrium. Therefore, a higher objective value can be achieved (i.e., better continuations generated) by an LM that is a Stackelberg leader than by an LM who is on equal footing with a PM. By allowing the PM to be the follower, we actually endow the system designer with more power since the leading RM must update its policy according to the strongest possible PM; the PM which best represents the designer’s preferences from online samples of the current RM.

As this intuition is natural, others have likely also tried to model the game played by LMs and PMs as sequential; however, there are no known algorithms that compute Stackelberg equilibria in general-sum games in polynomial time. While we do not prove the convergence of our algorithm for solving this game, we empirically test our approach by comparing it to a simultaneous version of our algorithm and a variant where the leader and follower are reversed to show that our formulation yields the best performing LM.

1.1 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), and recommender systems (Stray et al., 2021). As AI capabilities have advanced, this problem has become increasingly important and challenging, with AI systems optimizing for proxy goals often finding unintended and potentially harmful ways to achieve them. Attempts to comprehensively specify objectives are difficult due to the complexity of capturing the full scope of human values. The emergence of large language models has surged interest in the alignment problem, with RLHF (Christiano et al., 2023) emerging as a dominant paradigm for tackling it. The RLHF technique has been applied to aligning the behaviors of LMs with great success (Bai et al., 2022; Ouyang et al., 2022).

RLHF Algorithms The standard RLHF recipe for tuning a language model (LM) for alignment follows a specific process. First, the base language model is fine-tuned using supervised next-token prediction (SFT) on a high-quality dataset to initialize a base reference policy. Subsequently, using a dataset of prompts, completions are sampled from the reference policy and ranked by human labelers. A Bradley-Terry (BT) reward model is then trained to maximize the likelihood of this preference data Ouyang et al. (2022). Finally, the LM policy is optimized using standard reinforcement learning methods, such as REINFORCE and Proximal Policy Optimization (PPO), to maximize the objective of the reward model score, regularized by the policy’s KL divergence from the reference model.

Although PPO is widely recognized as the best-performing method for optimizing the policy in RLHF, the underlying reasons for its effectiveness are not fully understood. Moreover, the implementation of PPO for language models involves intricate details and multiple components, including the policy model, reference model, reward model, and critic model.

An alternative approach involves iteratively sampling from the reward model and fine-tuning the LM policy based on these samples. 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 & Schaal, 2007).

Additionally, there are approaches that omit training a reward model, and instead optimize the LM directly from the preference data. These methods include DPO (Rafailov et al., 2023), IPO (Azar et al., 2023), KTO (Ethayarajh et al., 2024), and P3O (Wu et al., 2023). DPO has gained popularity as an alternative to PPO due to its simplicity in implementation, and has recently been extended

\(^2\)A Stackelberg equilibrium of a two-player general-sum Stackelberg game can be equivalently understood as a bi-level optimization problem (Dempe & Zemkoho, 2020)
to online preference collection (Singhal et al., 2024). Many recent papers 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).

**Game Theoretic RLHF** Algorithmic game theory and multi-agent reinforcement learning provide strong theoretical grounding for the study of learning agents who must interact and adapt with each other. When the players are learning agents, trained with gradient descent-like algorithms, the dynamics of learning are well understood particularly in the two-player zero-sum case under convexity assumptions (Goktas & Greenwald, 2021; Lin et al., 2024). The canonical solution concept is the Nash equilibrium where no player can improve their payoff by deviating from their strategy, assuming no deviation in strategy from the other players (Nash et al., 1950). Recently, several game theory-based 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. However, both of these methods rely on a fixed preference model. Nash-RLHF (Munos et al., 2023) computes a Nash equilibrium between two competing LMs against a fixed preference model. The RLHF problem has also been formulated as a social choice problem, extending the setting to accommodate multiple human raters with different preferences (Conitzer et al., 2024; Chakraborty et al., 2024). A Bi-level optimization approaches have also been considered for RLHF, which is analogous to a Stackelberg game (Chakraborty et al., 2023). However, our method treats the preference model as the follower allowing for the preference model to adapt to on-policy changes from the leader-LM.

**Contributions** Our contributions can be summarized as follows: We present RLHF as a general-sum Stackelberg game played between the LM and the PM. Furthermore, our game formulation generalises to any class of PM unlike DPO (Rafailov et al., 2024) or KTO (Ethayarajh et al., 2024) which assume a set structure to preference. We present a nested training algorithm for solving for the Stackelberg equilibrium of our game, and show empirically that our solution outperforms a Nash equilibrium, and a Stackelberg game where the leader and follower are reversed, as well as state-of-the-art approaches in the literature, namely DPO and PARL.

2 Preliminaries

**Notation** We use caligraphic uppercase letters to denote sets (e.g., $\mathcal{X}$), bold lowercase letters to denote vectors (e.g., $\mathbf{x}$), lowercase letters to denote scalar quantities (e.g., $x$), and uppercase letters to denote random variables (e.g., $X$). We denote the $i$th element of a vector with a subscript, i.e. $x_i$; and the $j$th observation in a sample with 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 $F$ if the mapping is vector valued. We denote set of real numbers by $\mathbb{R}$.

**Stackelberg Game** A two-player general-sum Stackelberg game $(\mathcal{X}, \mathcal{Y}, f, g)$ is one where the leader, who moves first, tries to maximize their reward $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$, by choosing a strategy $\mathbf{x}$ from their strategy set $\mathcal{X}$. Similarly, the follower, who moves second tries to maximize their reward $g : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ by choosing a strategy from their strategy set $\mathbf{y} \in \mathcal{Y}$. We define the the best-response correspondence $BR(\mathbf{x}, \mathbf{y}) = \{ \mathbf{y} \in \mathcal{Y} | g(\mathbf{x}, \mathbf{y}) \geq \max_{\mathbf{y} \in \mathcal{Y}} g(\mathbf{x}, \mathbf{y}) \}$. A strong Stackelberg equilibrium (SSE) breaks ties in favour of the leader and is defined as an action profile $(\mathbf{x}^*, \mathbf{y}^*) \in \mathcal{X} \times \mathcal{Y}$ s.t. $f(\mathbf{x}^*, \mathbf{y}^*) = \max_{\mathbf{x} \in \mathcal{X}} \max_{\mathbf{y} \in BR(\mathbf{x}, \mathbf{y})} f(\mathbf{x}, \mathbf{y})$.

**Preference Relations** A human’s preferences are represented by a relation $\succsim \subseteq \mathbb{R}^{|D| \times |D|}$, also referred to as their type. As $\succsim$ is unobserved, it must be learned from data. Implicit in our framework and most variants of RLHF is the following assumption on the preference relation:

---

3A weak Stackelberg equilibrium breaks ties in favour of the follower but is not guaranteed to exist unlike the strong version (Guo et al., 2019).
**Assumption 1.** $D$ is finite, the preference relation $\succsim$ is complete and transitive.

Under Assumption 1, a preference relation $\succsim$ can be represented by a utility function $u(\cdot, \cdot; \succsim) : D \times D \rightarrow \mathbb{R}$ s.t. for all strings $x, y, x', y' \in D$, $u(x, y; \succsim) \geq u(x', y'; \succsim)$ iff $(x, y) \succsim (x', y')$ (Debreu et al., 1954).

**Preference Model** Given a prompt $x$ together with two possible continuations $(y_0, y_1)$, a preference model $\sigma : D \times D \times D \rightarrow \Delta(\{0, 1\})$ predicts a label $a \sim \sigma(\cdot \mid x, y_0, y_1)$ indicating that $(x, y_a) \succsim (x, y_{1-a})$, which we will use to learn an estimate the designer’s utility function $\hat{u}$.

**Language Model** Let $D$ denote the finite set of all possible strings. A language model (LM) is a function $\pi : D \rightarrow \Delta(D)$ from prompts $x \in D$ to a distribution over continuations $y \sim \pi(x) \in \Delta(D)$.

### 3 Stackelberg RLHF

In our Stackelberg game formulation, the players will be the LM and the PM. We now define their respective objectives. Given a pair of continuations and a prompt $(x, y_0, y_1)$, the PM should select the continuation with the highest utility according to the humans preferences.

$$\arg\max_{a \in \{0, 1\}} u(x, y_a; \succsim) \quad (1)$$

However, given that the string space is large, and distinguishing preferences between some strings is difficult, we make the following assumption which captures the imperfect nature of human labeling.

**Assumption 2.** The PM is a quantile best-response agent.

Under Assumption 2, given a prompt and pair of continuations we have that

$$\sigma^* \in \arg\max_{\sigma : D \rightarrow \Delta(\{0, 1\})} \sum_{a \in \{0, 1\}} \left( \sigma(a \mid x, y_0, y_1) \cdot u(x, y_a; \succsim) \right) - \sum_{a \in \{0, 1\}} \left( \sigma(a \mid x, y_0, y_1) \log \sigma(a \mid x, y_0, y_1) \right)$$

(2)

**Lemma 1.** The solution to this problem is given by

$$\frac{e^{u(x, y_a; \succsim)}}{e^{u(x, y_a; \succsim)} + e^{u(x, y_{1-a}; \succsim)}} = \mathbb{P}\left((x, y_a) \succsim (x, y_{1-a})\right) = \sigma^*(a \mid x, y_a, y_{1-a})$$

(3)

Which is the Bradley-Terry model (see Appendix A):

**Preference Model Objective** Following Lemma 1, given $x \sim \mu$ and $y_0, y_1 \sim \pi(x)$ and the preferred label $a \sim \sigma$ the objective of the preference model is:

$$\arg\max_{\succsim \in \{\mathbb{R}^{|D|} \times |D|\}} \mathbb{P}\left((x, y_a) \succsim (x, y_{1-a})\right) = \arg\max_{a \in \mathcal{U}_\succsim} \frac{e^{u(x, y_a; \succsim)}}{e^{u(x, y_a; \succsim)} + e^{u(x, y_{1-a}; \succsim)}} = \arg\max_{\sigma \in \Sigma} \sigma(a \mid x, y_a, y_{1-a})$$

(4)

Where $\mathcal{U}_\succsim = \{D \times D \rightarrow \mathbb{R}\}$.

**Language Model Objective** Now with the PM’s objective defined, the LM will use the representation of $u$ for its own objective:

$$\max_{\pi \in \Pi} \mathbb{E}_{x \sim \mu, y \sim \pi(x)} \left[u(x, y; \succsim)\right]$$

(5)
Final Optimization Problem

We can cast the full problem as a bi-level optimization problem

In particular, we optimize over the parameters

(P
designer
u
preference:
utility function of the LM is to generate the best possible continuation according to the designer’s preference: \( u(x, y; \succ) \). The follower is a PM. The PM observes a pair of continuations from the LM given a prompt \((x, y_0, y_1)\). The utility function of the PM is to represent preferences of the designer \( \mathbb{P}((x, y_0) \succ (x, y_{1-a})) \) where \( a \) is the preferred continuation. The SSE is the action profile \((\pi^*, \succ^*) \) s.t. \( \mathbb{E}[u(x, \pi^*(x); \succ^*)] \geq \max_{\pi \in \Pi} \max_{\pi \in \mathcal{BR}(\pi, \succ)} \mathbb{E}[u(x, \pi(x); \succ)] \). Where \( \mathcal{BR}(\pi, \succ) = \{ \succ \in \mathbb{R}_{\mathcal{D} \times \mathcal{D}} \mid \mathbb{P}((x, \pi(x)_a \succ (x, \pi(x)_{1-a}) \geq \max_{\pi \in \mathcal{D}} \mathbb{P}((x, \pi(x)_a \succ (x, \pi(x)_{1-a})) \} \).

4 Empirical estimation

Preference Model

With the game and problem formulation now defined, we must compute an approximate solution since we do not have direct access to \( \pi, \mu, \sigma \). A preference dataset of size \( m \in \mathbb{N} \) can be generated by sampling prompts, generating pairs of continuations, and assigning labels to the preferred and not preferred continuations:

\[
\{x, y_0, y_1, a\}_{j=1}^m \sim \mu^m \times \prod_{j=1}^m \pi(x) \times \pi(x) \times \sigma_i(\cdot|x, y_0, y_1)
\]

We can then approximate the PM’s objective through a maximum log-likelihood problem to learn \( \hat{u} \):

\[
\max_{\hat{u} \in \mathcal{U}_\succ} \log \left( \prod_{j=1}^m \frac{e^{\hat{u}(x(j), y(j); \succ)}}{e^{\hat{u}(x(j), y(j); \succ)} + e^{\hat{u}(x(j), y_{1-a}; \succ)}} \right)
\]

\[
= \max_{\hat{u} \in \mathcal{U}_\succ} \sum_{j=1}^m \log \left( e^{\hat{u}(x(j), y(j); \succ)} \right) - \log \left( e^{\hat{u}(x(j), y(j); \succ)} + e^{\hat{u}(x(j), y_{1-a}; \succ)} \right)
\]

\[
= \max_{\hat{u} \in \mathcal{U}_\succ} \sum_{j=1}^m \hat{u}(x(j), y(j); \succ) - \log \left( e^{\hat{u}(x(j), y(j); \succ)} + e^{\hat{u}(x(j), y_{1-a}; \succ)} \right)
\]

In particular, we optimize over the parameters \( \theta \) of an LM which is designed to approximate the preference relation. In particular \( \mathcal{U}_\theta \leq \mathcal{U}_\tau \) meaning that:

\[
\max_{a \in \mathcal{U}_\succ} \sum_{j=1}^m \hat{u}(x(j), y(j); \succ) - \log \left( e^{\hat{u}(x(j), y(j); \succ)} + e^{\hat{u}(x(j), y_{1-a}; \succ)} \right)
\]

\[
\geq \max_{a \in \mathcal{U}_\theta} \sum_{j=1}^m \hat{u}(x(j), y(j); \theta) - \log \left( e^{\hat{u}(x(j), y(j); \theta)} + e^{\hat{u}(x(j), y_{1-a}; \theta)} \right)
\]

Final Optimization Problem

We can cast the full problem as a bi-level optimization problem with:

\[
\max_{\pi \in \Pi} \max_{\theta^*(\pi)} \mathbb{E} \left[ \hat{u}(x(j), y(j); \theta^*(\pi)) \right]
\]

subject to

\[
\theta^*(\pi) = \arg \max_{\hat{u} \in \mathcal{U}_\theta} \sum_{j=1}^m \hat{u}(x(j), \pi(x(j)); \theta) - \log \left( e^{\hat{u}(x(j), \pi(x(j)); \theta)} + e^{\hat{u}(x(j), \pi(x(j)); 1-a; \theta)} \right)
\]
Solution Concepts

By framing RLHF as a game, we can define our solution concepts. In a simultaneous move game, the canonical solution concept is the Nash equilibrium — where no player can achieve a higher payoff by deviating, assuming all other players do not deviate. In the this game, if both pairs of policies are learned simultaneously, then the pair \((\pi^*, \theta^*)\) is a Nash equilibrium.

The standard way for computing a Nash equilibrium is with simultaneous gradient descent-ascent (Gorbunov et al., 2022) which is guaranteed to converge in certain games. In a sequential game, an alternative solution concept is the Stackelberg equilibrium where the leading player picks their strategy assuming the follower will best-respond. In a strong Stackelberg equilibrium, ties are broken in favour of the leader which justifies the second max in Equation (11). A pair of policy parameters satisfying Equation (12) and Equation (11), \((\pi^*, \theta^*)\) with \(\pi \rightarrow \theta^*(\pi)\) form a (strong) Stackelberg equilibrium. Nested gradient-ascent type algorithms are known to converge to a Stackelberg equilibrium Ghadimi & Wang (2018) under certain conditions. In particular, the utility of the leader in a Stackelberg equilibrium is at least as good as it would be in a Nash equilibrium. This means that the a language model trained under STA-RLHF in a nested fashion should be better aligned with a given preference model than if both models are trained together in tandem or if one of them is fixed.

5 Our Algorithm

\begin{algorithm}
\caption{Stackelberg RLHF}
\begin{algorithmic}
\Require Prompt data \(D\), batch size \(b\), policy sample size \(n\), number of iterations \(N, M\), learning schedules \(\{\eta^m\}_{m=1}^{N}, \{\eta^n\}_{n=1}^{M}\)
\Ensure LM model \(\pi\)
\State Initialize fine-tuned model parameters \(\pi^0\) to an SFT reference model \(\pi^{\text{ref}}\)
\For {\(t = 1\) to \(N\) iterations}
\State Sample a batch \(B = \{(x^{(i)})\}_{i=1}^{b}\) of prompts of size \(b\) from \(D\)
\State Run \(\pi^{t-1}\) on prompt \(x^{(i)}\) in \(B\) to generate \(y_1^{(i)}, y_0^{(i)}\).
\State Rank completions using an oracle and store \(\{(x^{(i)}, y_1^{(i)}, y_0^{(i)})\}_{i=1}^{b}\) as \(C\).
\For {\(j = 1\) to \(M\) iterations}
\State Compute the gradient of \(L_{PM}(\theta^{t-1})\) on \(C\)
\State \(\theta^{t} \leftarrow \theta^{t-1} - \eta^t \nabla_{\theta} L_{PM}(\theta^{t-1})\)
\EndFor
\State Compute the gradient of (5) on \(C\) given \(\theta^t\) and \(\pi^{\text{ref}}\) using REINFORCE
\State \(\pi^t \leftarrow \pi^{t-1} + \eta^n \nabla_{\pi^{t-1}} L_{LM}(\pi^{t-1})\)
\EndFor
\State \textbf{return} Final fine-tuned model parameters \(\pi^T\)
\end{algorithmic}
\end{algorithm}

In RLHF and DPO, all preference data is collected from the reference model which is used to train a fixed PM. According to our Stackelberg model, fixing the preference data would be analogous to fixing the strategy of the preference model instead of best-responding. This can lead to poor samples which are not representative of the new task. ReST addresses the static nature of the preference data by sampling new continuations from the latest policy, but does not train a preference model. PARL formulates RLHF as a bi-level optimization which would be a reversal of the players according to our game formulation. Additionally in PARL, the LM is optimized over several batches of trajectories in the inner loop instead of one set of trajectories.

In our algorithm, we collect preference labels in an online fashion and re-train the preference model in the inner loop. The advantage of our method is that the LM is updated according to the latest preference signal according to the current policy. We also consider simultaneous STA-RLHF, where the LM policy and preference model are updated in each step of the outer loop. Here the policy and preference model are updated using the previous iteration of each. Note that the solution concept of this optimization is a Nash equilibrium between preference model and LM policy. This is distinct from SPO and Nash-RLHF which compute a Nash Equilibrium between dueling LM policies.
6 Experiments

Unless stated otherwise, in all experiments below we use \( \beta = 0.1 \), a batch size of 16, RMSprop as an optimizer for the LM policy, and a policy learning rate of 1e-6. For our Stackelberg, Reverse, Simultaneous and PARL we use AdamW as an optimizer for the preference model with a constant learning rate of 1e-5. The number of inner iterations are set to \( M = 5 \) when applicable.

**IMDb Sentiment Generation** We follow the experimental setup in Rafailov et al. (2023) to evaluate aligning an LM to output completions with positive sentiment. In Sentiment Generation, the LM is given prompt \( x \) from a fixed dataset and must generate a completion \( y \) with a positive sentiment as scored by a fixed sentiment analysis model.

We use prefixes from the IMDb movie review dataset (Maas et al., 2011) and use siebert/sentiment-roberta-large-english as the sentiment scoring model (Hartmann et al., 2023). In this setting we use GPT-2 Large (Radford et al., 2019) as a reference model fine-tuned on 10% of the train split of the dataset with TRL’s implementation of SFT (von Werra et al., 2020).

![Figure 1: IMDb Sentiment Generation Results. Left: A scatter plot displaying the KL divergence between the policy and the reference model, plotted against the reward. This illustrates the trade-off between improving reward and deviating from the reference model across various methods. Right: The rewards of different methods throughout training, as evaluated by the sentiment model.](image)

**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 \( y \) where \( |y| = 75 \) tokens that incorporates as many unique words from \( W \) as possible. The preference relation \( \succeq \) is defined on the basis of the inclusion of tokens from \( W \) in \( y \). Specifically, for any two outputs \( y_1 \) and \( y_2 \), \( y_1 \succeq y_2 \) if the number of tokens from \( W \) included in \( y_1 \) is greater or equal than in \( y_2 \). Each token from \( W \) included in \( y \) increases the value of \( y \) by 1, up to a maximum value of 30. In our experiments we use OPT-125m as a reference model, the top 30 content words in the UltraFeedback dataset (Cui et al., 2023) for \( W \), and prompts from the UltraChat dataset (Ding et al., 2023).

In both settings, we can see that STA-RLHF shows strong results in quickly learning the reward and achieving high reward at a smaller deviation from the reference policy. When a method learns the reward faster than STA-RLHF, this is because it has a high deviation from the reference-policy and outputs non-sensical text.

7 Conclusion

In this paper, we first reformulate the alignment problem and RLHF as a Stackelberg game. Based on this model, we analyze solving for a Stackelberg equilibrium which represents an LM policy
which best matches the human’s preference. Based on this model, we introduce STA-RLHF, a novel game-theoretic approach to aligning large language models with RLHF. Our preliminary findings indicate that STA-RLHF enhances performance in synthetic settings when compared to traditional RLHF methods such as PPO and DPO. We validate our Stackelberg model by reversing the order of players and simultaneously training both the preference model and the language model. Future work should investigate scaling STA-RLHF to larger models, but also examine using STA-RLHF preference aggregation and online learning, areas where STA-RLHF naturally extends.

References


### A Additional Proofs

#### Proof of Lemma 1

**Proof.**

\[
\sigma \in \arg \max_{\sigma: D^3 \rightarrow \Delta(\{0,1\})} \sum_{a \in \{0,1\}} \left( \sigma(a|x,y_0,y_1) \cdot u(x,y_a; \succ) \right) - \sum_{a \in \{0,1\}} \left( \sigma(a|x,y_0,y_1) \log \sigma(a|x,y_0,y_1) \right)
\]  

(13)

Differentiating with respect to each \(a\) and setting to zero we obtain:

\[
\frac{\partial \mathcal{L}}{\partial \sigma(a|x,y_0,y_1)} = u(x,y_a; \succ) - 1 - \log \sigma(a|x,y_0,y_1)
\]  

(17)

\[
\implies \sigma(a|x,y_0,y_1) = e^{u(x,y_a; \succ) - 1 - \lambda}
\]  

(18)
Since \( \sum_{a \in \{0,1\}} \sigma(a|x, y_0, y_1) = 1 \) we get:

\[
\sum_{a \in \{0,1\}} e^{u(x, y_a; \zeta) - 1 - \lambda} = 1
\]

(19)

\[
\implies e^{-1 - \lambda} = \frac{1}{\sum_{a \in \{0,1\}} e^{u(x, y_a; \zeta)}}
\]

(20)

\[
= \frac{e^{u(x, y_a; \zeta)}}{e^{u(x, y_{1-a}; \zeta)} + e^{u(x, y_1-a; \zeta)}} = \mathbb{P}(x, y_a \gtrsim (x, y_{1-a})) = \sigma(a|x, y_a, y_1-a)
\]

(21)

Where the last line follows by multiplying both sides by \( e^{u(x, y_a; \zeta)} \). Which is equivalent to the Bradley-Terry model Agresti (2012)[ch.11].

\[\square\]

B Deriving the Gradient of the LM Objective

When optimizing the LM policy, we need to calculate the gradient of:

\[
\mathbb{E}_{(x^{(j)}) \sim \mu} \mathbb{E}_{(y^{(j)}) \sim \pi(x^{(j)})} \left[ \hat{u}(x^{(j)}, y^{(j)}; \theta^*(\pi)) \right]
\]

(22)

Note that for a given completion \( y = (y_1, \ldots, y_l) \), the probability of sampling \( y \) from \( \pi \) using prompt \( x = (x_1, \ldots, x_n) \) is:

\[
p_\pi(y \mid x) = p_\pi(y_1 \mid x_1, \ldots, x_n)p_\pi(y_2 \mid x_1, \ldots, x_n, y_1) \cdots p_\pi(y_l \mid x_1, \ldots, x_n, y_1, \ldots, y_{l-1})
\]

For a fixed prompt \( x^{(j)} \) the gradient becomes:

\[
\nabla_\pi \mathbb{E}_{y^{(j)} \sim \pi(x^{(j)})} \left[ \hat{u}(x^{(j)}, y^{(j)}; \theta^*(\pi)) \right] = \sum_{y \in \mathcal{D}} \left[ \hat{u}(x^{(j)}, y; \theta^*(\pi)) \nabla_\pi p_\pi(y \mid x^{(j)}) \right]
\]

(23)

As the right-hand side is not an expectation, we cannot use Monte-Carlo estimation to compute an estimate. Further, computing this directly by summing over all possible completions (i.e. \( y \in \mathcal{D} \)) is intractable.

Instead we can use the policy-gradient estimator.

\[
\nabla_\pi \mathbb{E}_{y^{(j)} \sim \pi(x^{(j)})} \left[ \hat{u}(x^{(j)}, y^{(j)}; \theta^*(\pi)) \right] = \mathbb{E} \left[ \hat{u}(x^{(j)}, y^{(j)}; \theta^*(\pi)) \nabla_\pi \log p_\pi(y \mid x^{(j)}) \right]
\]

(24)

C Algorithm Variants

Included below are the modified versions of our algorithm we consider in our experiments:
Algorithm 2 Reversed

Require: Prompt data \( D \), batch size \( b \), policy sample size \( n \), number of iterations \( N, M \), learning schedules \( \{\eta_n^N\}_{n=1}^N, \{\eta_m^M\}_{m=1}^M \)

Ensure: LM model \( \pi \)

1. Initialize fine-tuned model parameters \( \pi^0 \) to an SFT reference model \( \pi^{\text{ref}} \)
2. for \( t = 1 \) to \( N \) iterations do
3.   Sample a batch \( B = \{x(i)\}_{i=1}^b \) of prompts of size \( b \) from \( D \)
4.   Run \( \pi^{t-1} \) on prompt \( x(i) \) in \( B \) twice to generate \( y_1^{(i)}, y_0^{(i)} \).
5.   Rank completions using an oracle and store \( \{x(i), y_1^{(i)}, y_0^{(i)}\}_{i=1}^b \) as \( C \).
6.   for \( j = 1 \) to \( M \) iterations do
7.     Compute the gradient of (5) on \( C \) given \( \theta^{t-1} \) and \( \pi^{\text{ref}} \) using REINFORCE
8.     \( \pi^t \leftarrow \pi^{t-1} + \eta_n^t \nabla_{\pi^{t-1}} L_{\text{LM}}(\pi^{t-1}) \)
9.   end for
10. Compute the gradient of \( L_{PM}(\theta^{t-1}) \) on \( C \)
11. \( \theta^t \leftarrow \theta^{t-1} - \eta^t \nabla_{\theta} L_{PM}(\theta^{t-1}) \)
12. end for
13. return Final fine-tuned model parameters \( \pi^T \)

Algorithm 3 Simultaneous

Require: Prompt data \( D \), batch size \( b \), policy sample size \( n \), number of iterations \( N, M \), learning schedules \( \{\eta_n^N\}_{n=1}^N, \{\eta_m^M\}_{m=1}^M \)

Ensure: LM model \( \pi \)

1. Initialize fine-tuned model parameters \( \pi^0 \) to an SFT reference model \( \pi^{\text{ref}} \)
2. for \( t = 1 \) to \( N \) iterations do
3.   Sample a batch \( B = \{x(i)\}_{i=1}^b \) of prompts of size \( b \) from \( D \)
4.   Run \( \pi^{t-1} \) on prompt \( x(i) \) in \( B \) twice to generate \( y_1^{(i)}, y_0^{(i)} \).
5.   Rank completions using an oracle and store \( \{x(i), y_1^{(i)}, y_0^{(i)}\}_{i=1}^b \) as \( C \).
6.   Compute the gradient of (5) on \( C \) given \( \theta^{t-1} \) and \( \pi^{\text{ref}} \) using REINFORCE
7.   \( \pi^t \leftarrow \pi^{t-1} + \eta_n^t \nabla_{\pi^{t-1}} L_{\text{LM}}(\pi^{t-1}) \)
8.   Compute the gradient of \( L_{PM}(\theta^{t-1}) \) on \( C \)
9.   \( \theta^t \leftarrow \theta^{t-1} - \eta^t \nabla_{\theta} L_{PM}(\theta^{t-1}) \)
10. end for
11. return Final fine-tuned model parameters \( \pi^T \)

D Policy Samples from STA-RLHF

Included below are policy samples from the experimental settings in our paper.
Bruce Willis, as usual, does an excellent job in this movie, and it just takes all of what the film has going for it and puts it into a great movie that takes you on a very enjoyable story with great characters and great action. There are a few scripts like this one (that’s what I call them) and I highly recommend it. This is one of them. I saw it at a big screening in Reno a few years ago and I enjoyed it!! It’s a pretty good film.

The first few minutes showing the cold and dark atmosphere of a Belgian village is a great and unique film. It creates an atmosphere of a true crime film. It really brings me back to that feeling of a true crime story being told with real action and excitement.

I saw this only because my 10-year old son loved it. I thought it was great! It is a great family movie. The movie is about a boy and his dog. It was cute and entertaining. The movie is not for everyone as it is very dark and violent.

First of all, I wasn’t sure how to rate this movie. I’d read other people’s comments and liked them, but I wasn’t sure how to rate it. I thought it was a very good movie! I highly recommend it.

Table 1: Policy Samples for Sentiment Generation