A Gentle Introduction to RLHF

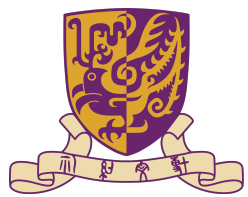
based on Prof. Yaodong Yang's RLHF Tutorial

FONG, Shi Yuk

Email: syfong1@cse.cuhk.edu.hk

The Institute of Theoretical Computer Science and Communications
The Chinese University of Hong Kong

July 26, 2024



Outline

Part 1. AI Alignment

What is AI alignment?

AI alignment aims to make AI systems behave in line with human intentions and values.

Why do we need AI alignment?

- **To achieve human purposes.** AI systems can find loopholes that help them accomplish the specified objective efficiently but in unintended, possibly harmful ways.
- **To prevent existential risk.** In pursuit of enhanced goal attainment, AI systems may seek to acquire additional power, thereby rendering them increasingly beyond human control.
- **To pursue artificial general intelligence (AGI).** Alignment is necessary condition for the development of AGI.

What does alignment address?

Alignment research aims to consider the objectives of AI systems in the following three categories:

- **Ideal specification** Expected behavior of the AI system.
- **Design specification** The goal of the AI system, often expressed in terms of objective functions and datasets.
- **Revealed specification** Actual performance of the trained AI system.

Alignment addresses mismatch between these three categories.

What does alignment address?

AI alignment aims to address the following issues:

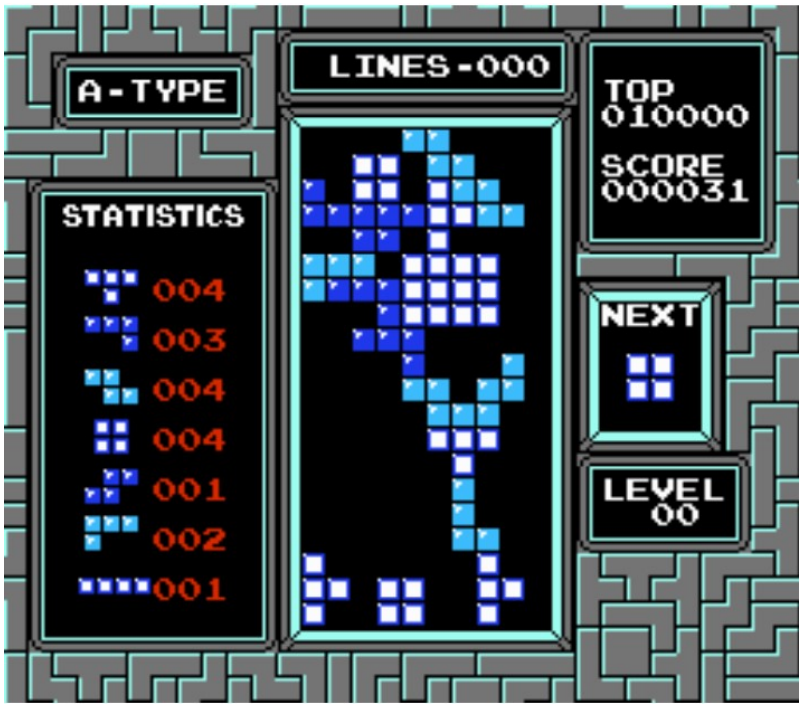
- **Outer alignment** $\rightarrow$ The consistency between ideal and design specifications $\rightarrow$ Resolves Reward misspecification
- **Inner alignment** $\rightarrow$ The consistency between design and revealed specifications $\rightarrow$ Resolves Goal misgeneralization

and thereby making AI systems more **helpful, honest, and harmless** (HHH).

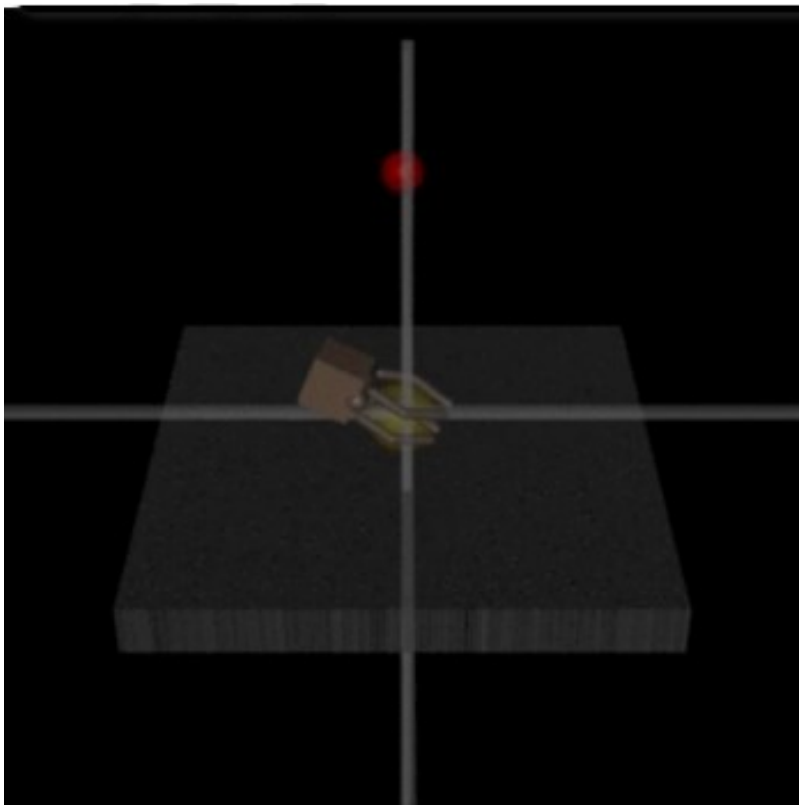
Part 2. Example of Misalignment

Reward misspecification: Tetris and robotic arm

The provided reward function incentivized incorrect behavior in the AI system, resulting in a misalignment with human expectations.



(a) When the agent was about to lose at Tetris, it learned to indefinitely pause the game.



(b) The robotic arm resort to deceptive tactics to fulfill human preferences.

Reward misspecification: reward overoptimization

AI systems follows excessively optimized reward function (**proxy goal**), leading to performance deviating from human expectations (**actual goal**).

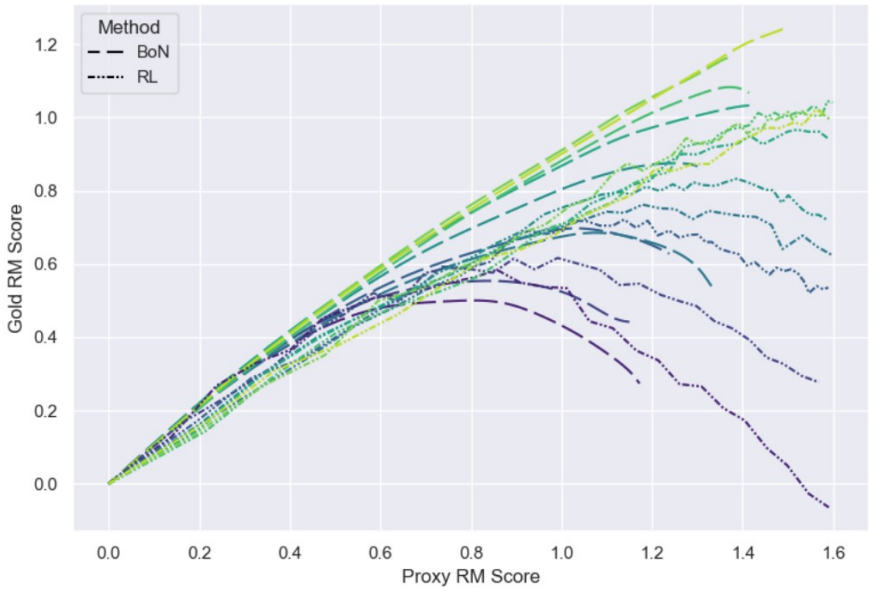
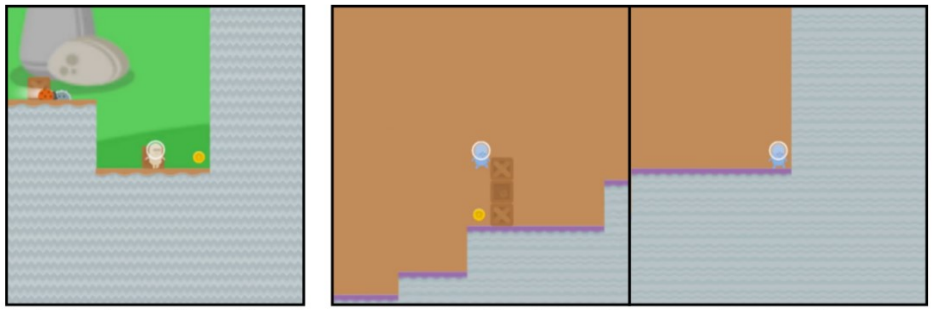


Figure: The relationship between the scores on the proxy Reward Model (RM) and the expected scores. Here, RL denotes reinforcement learning, and BoN represents selecting best-of-N outputs. The expected score of initially increases and then decreases as the score on the proxy RM increases. One way to resolve this problem is to introduce KL penalty

Goal misgeneralization: coin location

Training-testing inconsistency: AI systems pursue a goal other than the training goal while retaining the capabilities it had on the training distribution.



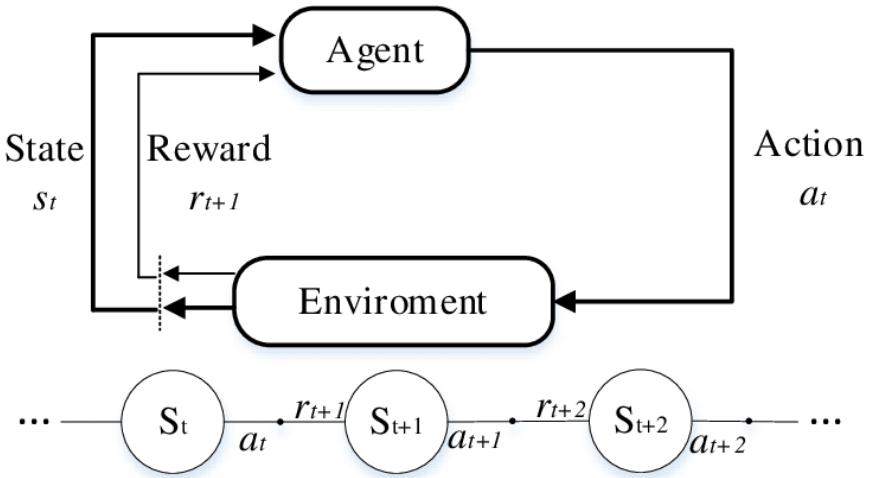
(a) Goal position fixed (b) Goal position randomized

Figure: **(a)** At training time, the agent learns to reliably reach the coin which is always located at the end of the level. **(b)** However, when the coin position is randomized at test time, the agent still goes towards the end of the level and often skips the coin. The agent's capability for solving the levels generalizes, but its goal of collecting coins does not.

Part 3. Fundamentals of reinforcement learning

Overview: What is reinforcement learning?

An agent tries to maximize the total amount of reward it receives while interacting with a complex and uncertain environment.



Some notations

- A **state** $s \in \mathcal{S}$ is a representation of the environment at a given time.
- Examples: position, orientation, velocity of objects in a game; prompt given to an LLM.
- An **action** $a \in \mathcal{A}(s)$ is a decision made by the agent at a given state s, where $\mathcal{A}(s)$ is the set of actions available at state s.
- Examples: console control in a game; response to a prompt by an LLM.

Some notations

- A **state transition** $s' \sim p(\cdot \mid s, a)$ is the probability distribution of transitioning to state s' after taking action a at state s.
- Remark: p satisfies the Markov property, i.e., The future state depends only on the current state and action, not the history. $p(s_{t+1} \mid s_t, a_t) = p(s_{t+1} \mid s_1, a_1, \ldots, s_t, a_t)$.
- A **reward** $r(s, a, s') \in \mathbb{R}$ is a scalar feedback signal received by the agent after taking action a at state s and transitioning to state s'.
- Remark: Large rewards encourage desirable actions and reinforce the agent's behavior in similar situations, and vice versa.

Some notations

- A **policy** $\pi(a \mid s)$ is a probability distribution over actions given states.
- Remark: The policy determines the agent's behavior in an environment by specifying the probabilities of selecting different actions in different states.
- A **value function** $V^{\pi}(s)$ is the expected cumulative reward when starting in state s and following policy π thereafter.
- Remark: The value functions provide estimates of the total expected reward that an agent can achieve from a given state, and they are essential in guiding the agent's decision-making process in reinforcement learning tasks.
- An **action-value function** $Q^{\pi}(s, a)$ is the expected cumulative reward when starting in state s, taking action a, and following policy π thereafter.

Bellman Equation

The Bellman equation expresses the relationship between value functions and rewards.

$$V^{\pi}(s) = \mathbb{E}_{a \sim \pi(\cdot|s)} \left[Q^{\pi}(s, a) \right]$$
$$Q^{\pi}(s, a) = \mathbb{E}_{s' \sim p(\cdot|s,a)} \left[r(s, a, s') + \gamma V^{\pi}(s') \right]$$

Objective of reinforcement learning

- Find a policy π that maximizes the **value** V^{π} that it can extract from every state s of the system.
- Find a policy π that maximizes the **reward** r that it can extract from every state s of the system.
- Find a policy π that maximizes the **likelihood** of "good" action a for all state s of the system without the guidance of reward r (Imitation Learning).

Exploration and exploitation

- **Exploration** can be seen as random selection of different actions to discover the environment's dynamics and rewards.
- **Exploitation** is the process of selecting the best-known actions following some policy π.
- The trade-off between exploration and exploitation is a fundamental challenge in reinforcement learning.

Part 4. Fundamentals of human feedback

Definition (LLM)

a language model $M : \mathcal{X} \to \mathcal{Y}$ parameterized by θ with conditional probability distribution $\pi_\theta(y \mid x)$ outputs text $y \in \mathcal{Y}$ given input of some type $x \in \mathcal{X}$. While x can be any type of input, y is typically in the space of natural language, i.e., $\mathcal{Y} \subseteq \Sigma^*$ for some alphabet Σ, where * denotes star-closure.

- **Training**: maximizing the likelihood of the training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$.
- **Inference**: usually through random sampling: $M(x) \sim \pi_\theta(\cdot \mid x)$.

Basic formulation of human feedback

Formally, we consider human feedback to be a family of functions $\mathcal{H}$ to return some feedback $f \in \mathcal{F}$. Each feedback function $h \in \mathcal{H}$ takes the form:

$$h : \mathcal{X} \times \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_n \to \mathcal{F}$$

where $x \in \mathcal{X}$ is the input, $y_1, \ldots, y_n \in \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_n$ are n different outputs.

Formats of human feedback

- Numerical feedback: $\mathcal{F} \subseteq \mathbb{R}$.
  - Example: judging a particular output is good or bad.

$$h : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$$

- **Ranking-based** feedback: $\mathcal{F} = S_n$, where S_n is the set of ordering (either partial or total order) of n elements.
  - Example: preference between two outputs. We say y_1 is preferred to y_2 if $y_1 \succ y_2$.

$$h : \mathcal{X} \times \mathcal{Y}_1 \times \mathcal{Y}_2 \to \{y_1 \succ y_2, y_2 \succ y_1\}$$

  - Remark: This is more common in RLHF as such data is easier to collect.
- Natural language feedback: $\mathcal{F} = \Sigma^*$.
  - Example: comment on a particular output.

Motivation

- h is unknown.
- We only have access to the feedback data.
- Modeling h can help us gather low-cost feedback.
- Therefore, we want to learn a parametric feedback model h_ϕ by minimizing some loss $\mathcal{L}(\phi)$:

$$\phi^* = \arg\min_{\phi} \mathbb{E}_{x,y_1,\ldots,y_n \sim \mathcal{D}_f} [\mathcal{L}(\phi)]$$

- Remark: the feedback model can then be used to improve the generation quality of LLM M.

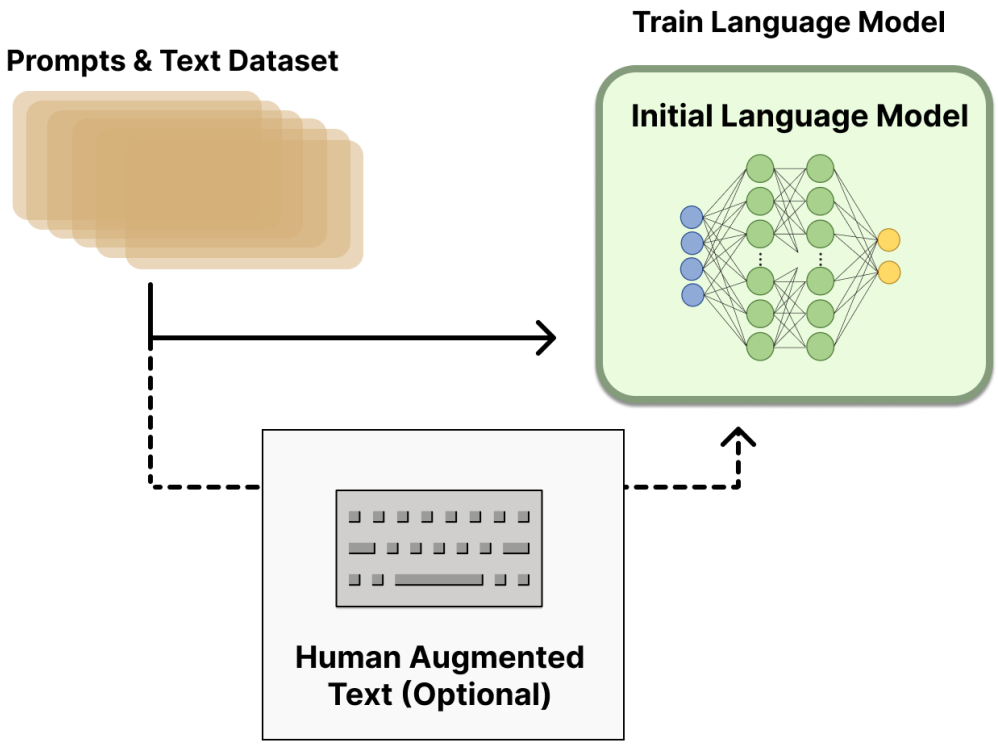
Part 5. RLHF in LLMs

Three stages of RLHF

1. Supervised fine-tuning (SFT)
2. Preference sampling and reward learning
3. Reinforcement-learning optimization

Supervised fine-tuning (SFT)

RLHF typically begins with a generic pre-trained LM, which is fine-tuned with supervised learning (maximum likelihood) on a high-quality dataset for the downstream tasks of interest, such as dialogue, instruction following, summarization, etc., to obtain a model π^{SFT}.



Reward model (RM) training

In the second stage the SFT model is prompted with prompts x to produce pairs of answers $(y_1, y_2) \sim \pi^{\mathsf{SFT}}(\cdot \mid x)$. These are then presented to human labelers who express preferences for one answer, denoted as:

$$y_+ \succ y_- \mid x.$$

The preferences are assumed to be generated by some latent reward model $r^*(y, x)$, which we do not have access to.

General framework of RM training

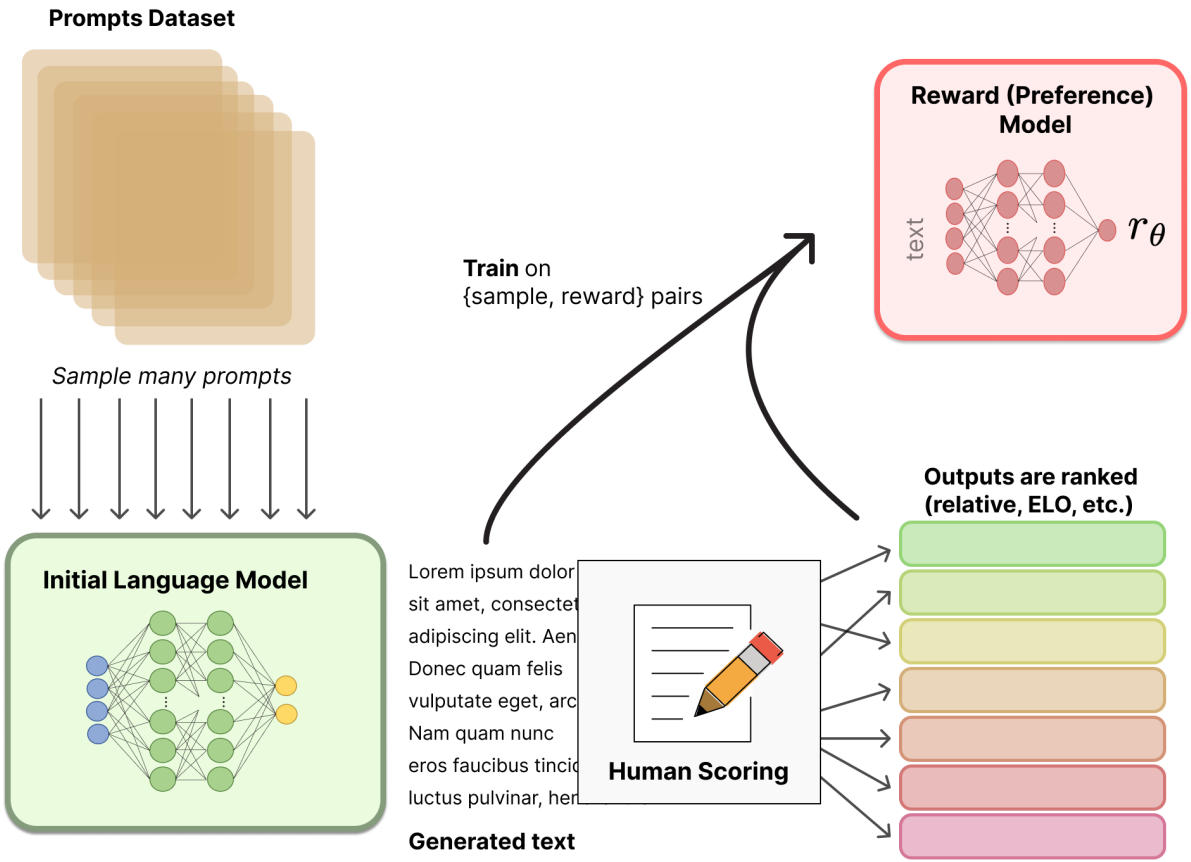
Assuming access to a static dataset of preference data $\mathcal{D}_{\text{pref}} = \{(x^{(i)}, y_+^{(i)}, y_-^{(i)})\}_{i=1}^N$, we can parameterize a reward model $r_\phi(x, y)$ and approximates the parameters with maximum likelihood. The equivalent minimization problem is:

$$\begin{aligned}\phi^* &= \arg\min_\phi \mathcal{L}_U(r_\phi, \mathcal{D}_{\text{pref}}) \\ &= \arg\min_\phi -\mathbb{E}_{(x,y_+,y_-)\sim\mathcal{D}_{\text{pref}}}[U(r_\phi(x, y_+) - r_\phi(x, y_-))]\end{aligned}$$

where U is some monotonically increasing utility function. A usual choice is the log-sigmoid function $U(x) = \log \sigma(x)$.

General framework of RM training

At this stage, the choice of reward models are usually small LMs.



Reinforcement-learning optimization

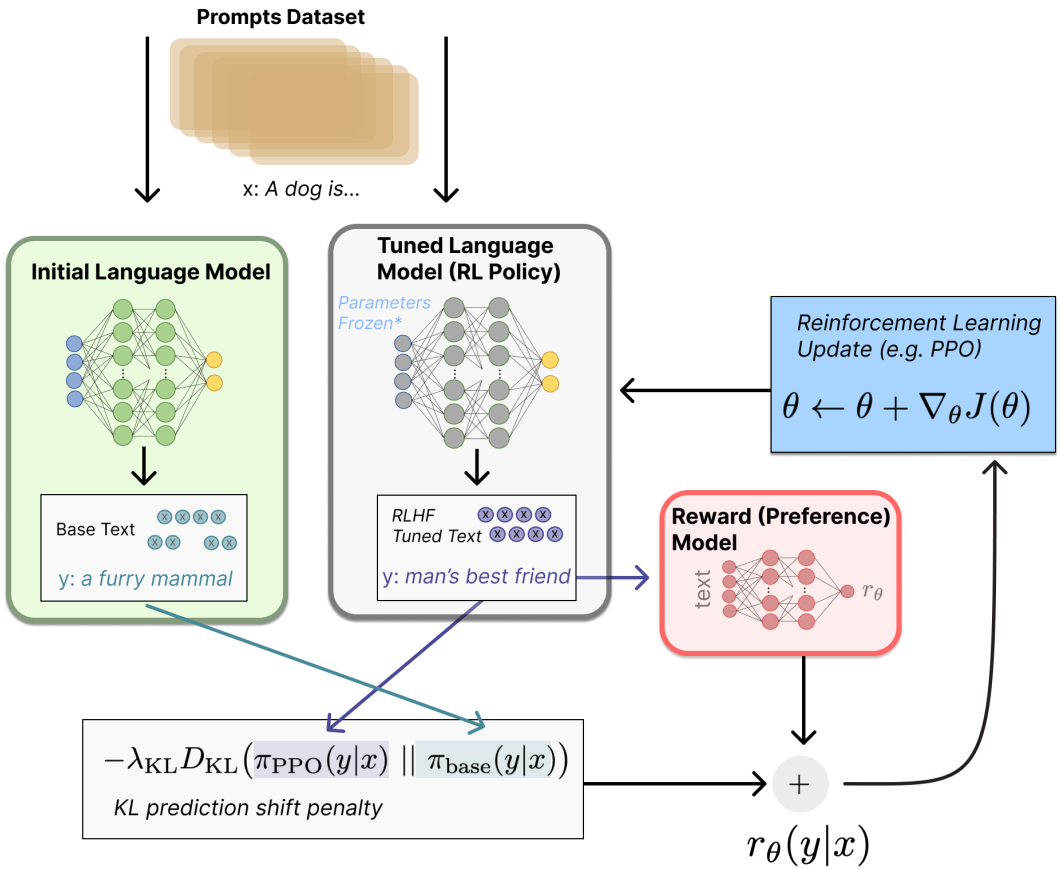
During the RL phase, we use the learned reward function to provide feedback to the language model. In particular, we formulate the following optimization problem:

$$\theta^* = \arg\max_{\theta} \mathbb{E}_{y\sim\pi_\theta(\cdot|x)}[r_\phi(x,y)] - \beta\mathcal{R}(\pi_\theta, \pi_{\text{ref}}) \qquad x \sim \mathcal{D}_{\text{pref}},$$

where $\mathcal{R}$ is a regularizer that encourages the policy to stay close to a reference policy π_{ref}. It is important as it prevents the policy from diverging too far from the distribution on which the reward model is accurate, as well as maintaining the generation diversity and preventing model collapse to single high-reward outputs. Usual choice of π_{ref} is the SFT model π^{SFT}, and $\mathcal{R}$ can be chosen as the KL divergence or, in general, f-divergence.

Reinforcement-learning optimization

Usually, a trust region method is used to optimize the policy, such as PPO, which aims to stabilize the training process by limiting the policy update to a certain region, as RL updates tends to be unstable and can diverge easily.



Part 6. Selected topics in RLHF

Selected topics in RLHF

- **Dealing with reward collapse in RLHF**

These are to be covered in future tutorial:

- Direct preference optimization (DPO)
- Proximal policy optimization (PPO)
- Fine-grained reward model

Part 7. Dealing with reward collapse in RLHF

Z. Song, T. Cai, J. D. Lee, and W. J. Su, "Reward Collapse in Aligning Large Language Models: A Prompt-Aware Approach to Preference Rankings, in ICML 2023 Workshop The Many Facets of Preference-Based Learning, 2023"

Prompts can be generally classified into two categories:

- **Open-ended**: These prompts and responses are dependent on the users' backgrounds, allowing the reward distribution to span a continuous range. e.g., What is the best cuisine in the world?
- **Closed-ended**: A correct answer exists up to human cognition. Response should be either highly or lowly scored, thus generating a roughly two-point mass distribution for the reward distribution. e.g., What is the capital of France?

Issue: The ranking-based reward has shortcomings in reflecting different reward distributions with different prompts.

What is reward collapse?

Observation: Reward distribution of a reward model trained on preference rankings is prompt-independent.

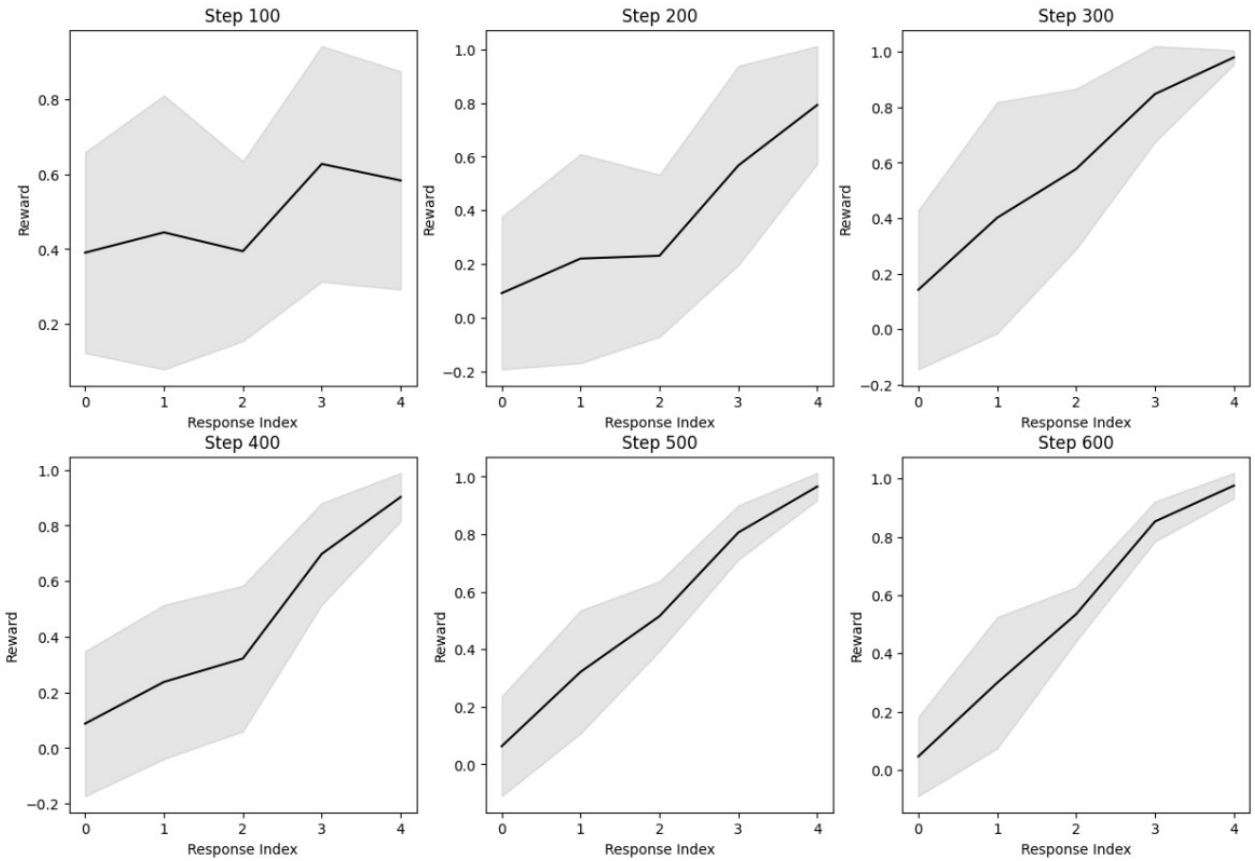


Figure: Empirical results showing that the reward of various prompts converges to a single distribution, regardless of the nature of the prompts.

Reward collapse: an algebraic perspective

Consider reward model $r_\phi(x, y)$ and the preference dataset $\mathcal{D}_{\text{pref}} = \{(x, y_1, \ldots, y_n) : y_i \succ y_j \mid x \ \forall\ 1 \leq i < j \leq n\}$, where x is the prompt and y is the response. We assume $r_\phi \in [0, 1]$. Then, recall that the reward model is trained by maximizing the utility below:

$$\sum_{(x, y_1, \ldots, y_n) \in \mathcal{D}_{\text{pref}}} \sum_{1 \leq i < j \leq n} U(r_\phi(x, y_i) - r_\phi(x, y_j)), \tag{1}$$

where it aims to both align with human-provided ranking and separate the reward as much as possible.

Reward collapse: an algebraic perspective

Consider the surrogate function

$$S(r_1, \ldots, r_n) = \sum_{1 \le i < j \le n} U(r_i - r_j),$$

which has a maximum value of M attained at a unique point $(r_1^*, \ldots, r_n^*)$ with $r_1^* \ge \cdots \ge r_n^*$. Then, the overall utility (1) is upper bounded by $|\mathcal{D}|M$. Note that this upper bound is reached if and only if

$$r_\phi(x, y_i) = r_i^* \text{ for all } i \text{ and } x. \qquad (*)$$

Then, the reward model r_ϕ trained on $\mathcal{D}_{\text{pref}}$ will converge to r^*. In fact, for sufficiently trained RM r_ϕ, $r_\phi(x, y_i)$ is close to r_i^*. Note that (*) implies that r_ϕ is prompt-independent, hence leading to reward collapse.

Remark: Existence of maximum solution of S is guaranteed by *Weierstrass theorem*. Uniqueness requires a strictly convex U.

Idea: prompt-aware reward model

To address reward collapse, we want the reward distribution depends on the prompt.

From (1), we can build a prompt-aware reward model by introducing a prompt-aware utility function U_x:

$$\sum_{(x,y_1,\ldots,y_n)\in\mathcal{D}_{\text{pref}}} \sum_{1\le i<j\le n} U_x(r_\phi(x,y_i)-r_\phi(x,y_j)), \tag{2}$$

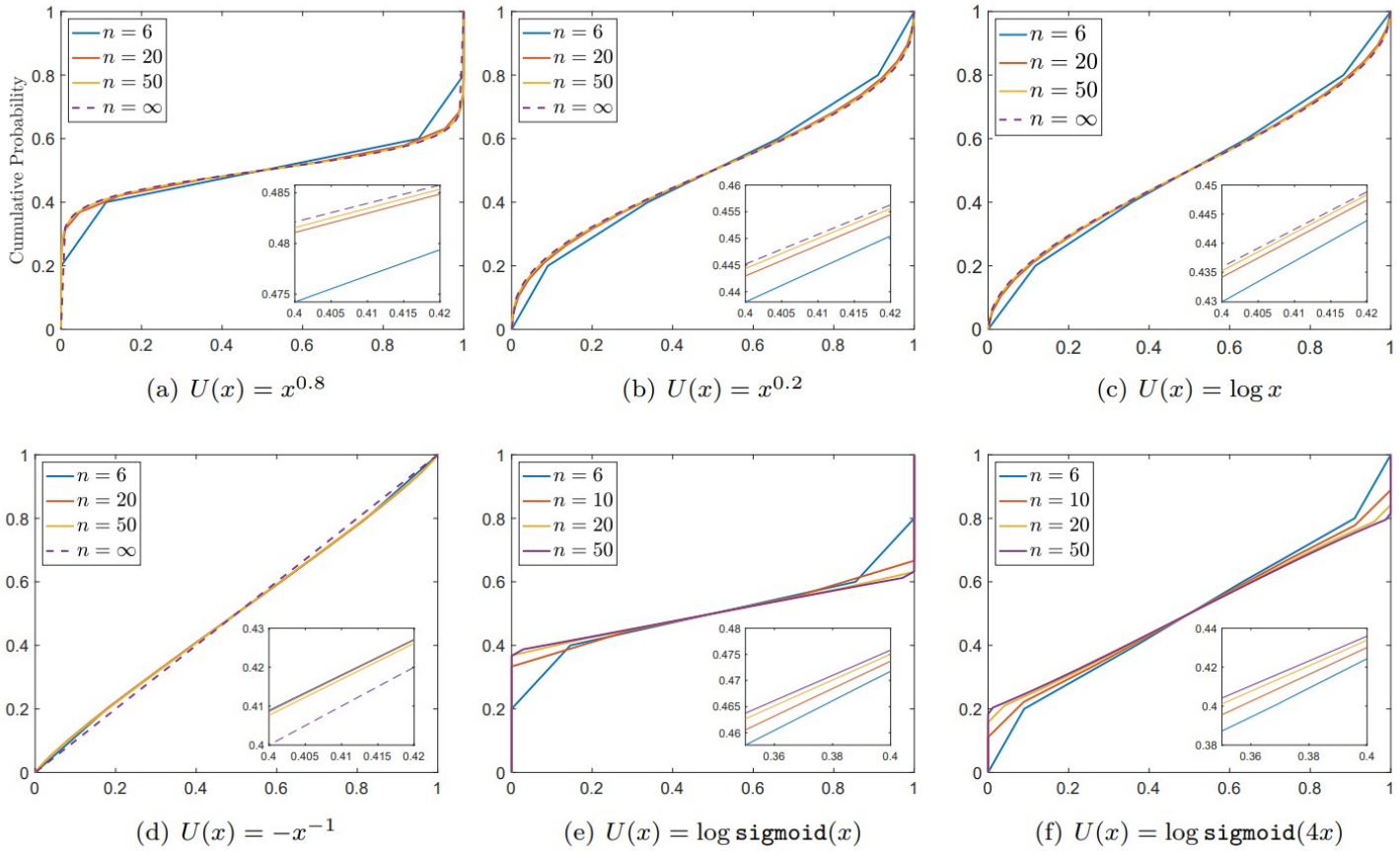
We also define its surrogate as

$$S_x(r_1,\ldots,r_n)=\sum_{1\le i<j\le n} U_x(r_i-r_j).$$

The reward distribution refers to the empirical solution of S_x.

Class of utility functions

- $U_\gamma(x) = x^\gamma, x \in [0,1]$ for $\gamma \in (0,1)$. This utility function encourages the reward to take values either near 0 or 1 as γ tends to be large.



(a) $U(x) = x^{0.8}$ (b) $U(x) = x^{0.2}$ (c) $U(x) = \log x$

(d) $U(x) = -x^{-1}$ (e) $U(x) = \log \texttt{sigmoid}(x)$ (f) $U(x) = \log \texttt{sigmoid}(4x)$

Figure: Reward distribution for different utility function.

Class of utility functions

- $U_\gamma(x) = -x^{-\gamma}, x \in (0, 1]$ for $\gamma \in (0, 1)$, and $U_0(x) = \log x$. We denote $U_\gamma(0) = -\infty$. In this case, the reward distribution becomes more even as γ increases from 0 to 1.

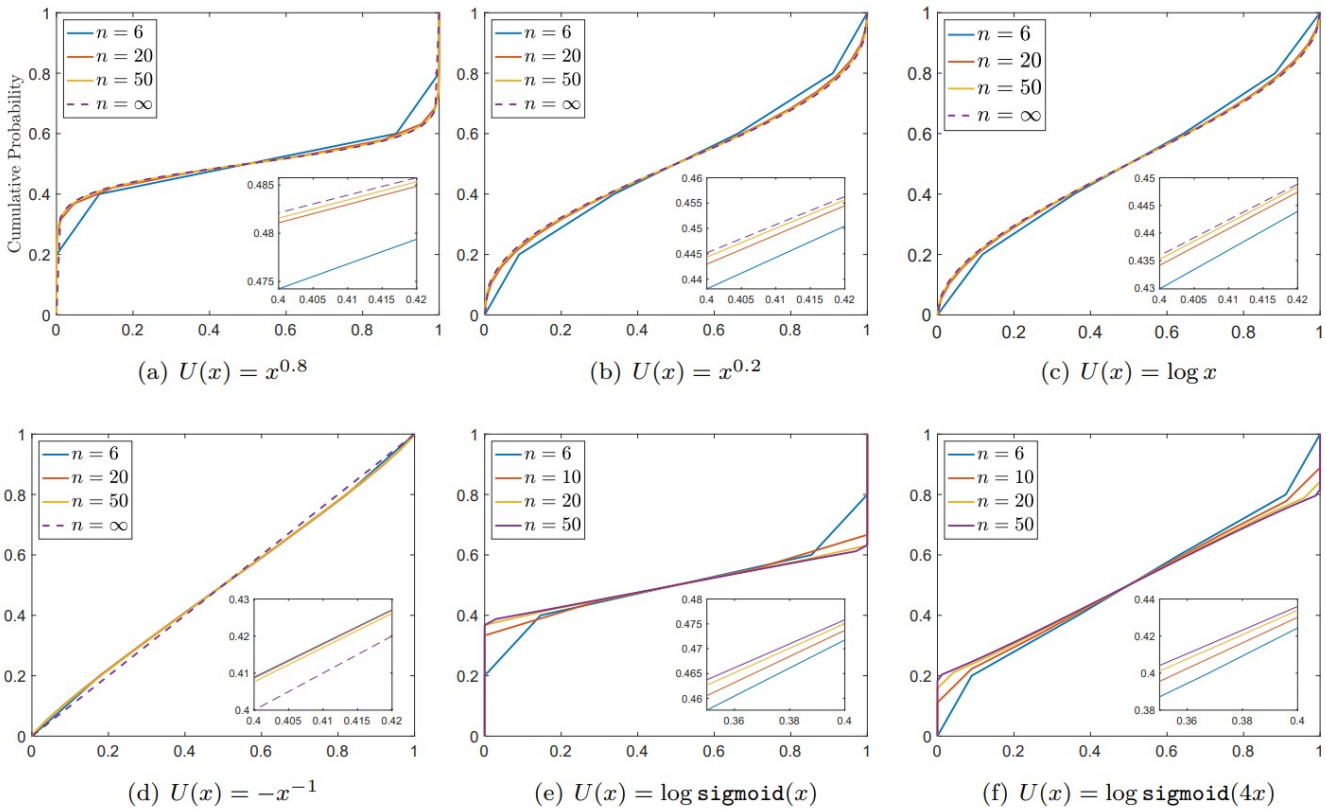
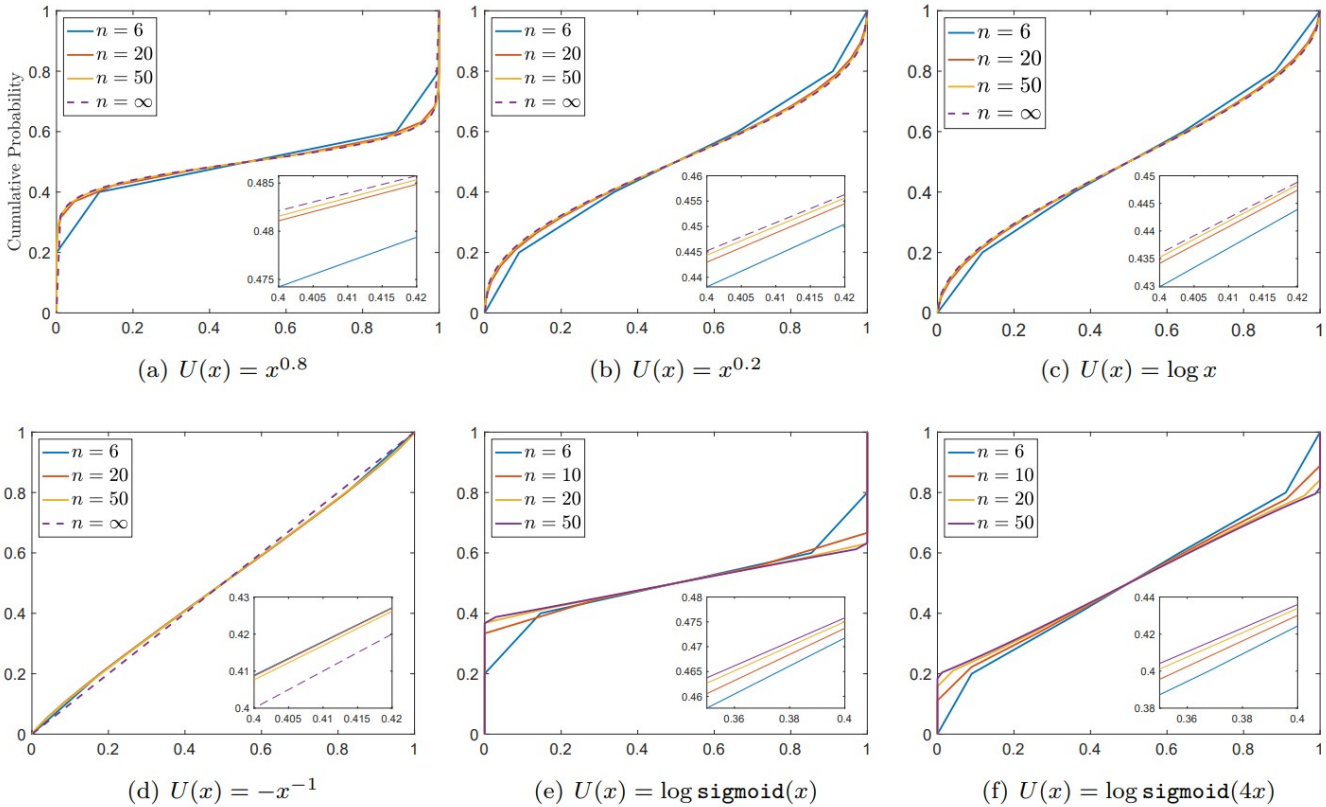


Figure: Reward distribution for different utility function.

Class of utility functions

- $U_\sigma(x) = \log \text{sigmoid}(x/\sigma)$ for $\sigma > 0$. The reward distribution becomes more spread between 0 and 1 as σ becomes smaller. This is a common choice of utility function in RLHF.

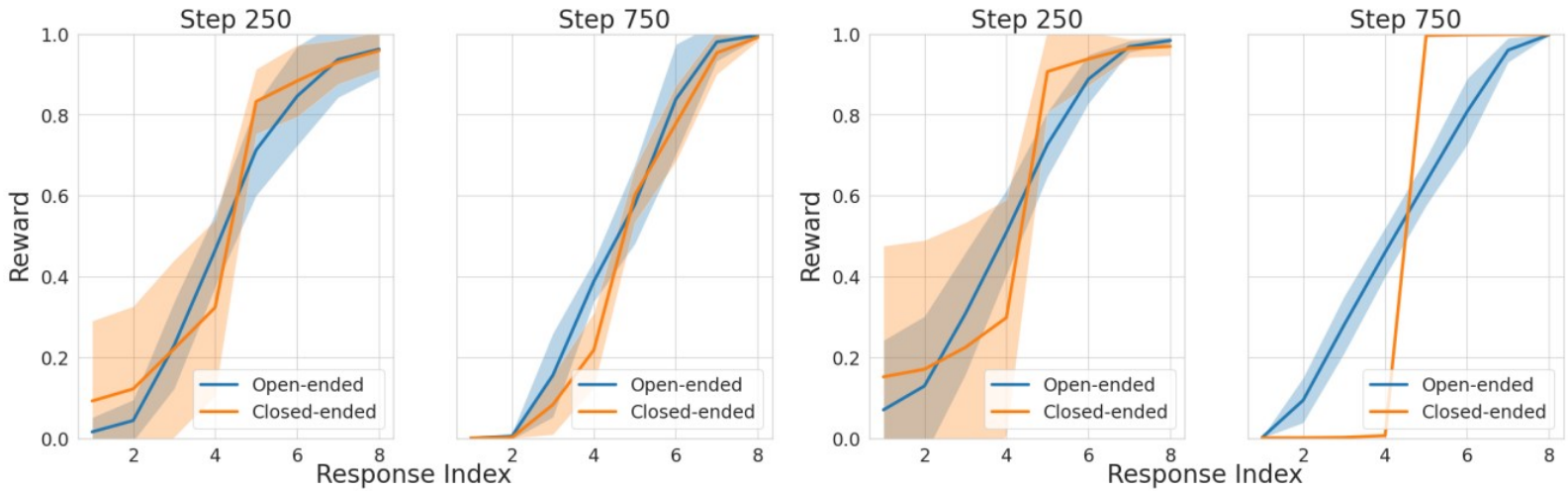


(a) $U(x) = x^{0.8}$ (b) $U(x) = x^{0.2}$ (c) $U(x) = \log x$

(d) $U(x) = -x^{-1}$ (e) $U(x) = \log \texttt{sigmoid}(x)$ (f) $U(x) = \log \texttt{sigmoid}(4x)$

Figure: Reward distribution for different utility function.

Experiments



(a) log `sigmoid` as utility function

(b) Prompt-aware utility function

Figure: **(Left)** The reward distribution of different prompts gradually converges into a single distribution during training. **(Right)** When using the prompt-aware loss function, the reward distributions of the two different prompts can be gradually separated during training.

Discussion

- Should prompts only be classified into binary classes (open/close-ended)?
- Different reward distribution $\Longrightarrow$ different LLM output distributions (conditioned on prompt)?
- Any more interesting utility functions?
- Effective way to choose the utility function?
- Prompt-based preference set format (e.g., partial ordering) $\Longrightarrow$ prompt-aware reward model?

Part 8. Open problems in RLHF

Open problems in RLHF

These are to be covered in future tutorial.

- Challenges with Obtaining Human Feedback
- Challenges with the Reward Model
- Challenges with the Policy