Lecture 6: RLHF in Language Models

Dr. Yaodong Yang

Institute for AI, Peking University

08/2023

Why do we need RLHF?

RLHF helps improve the overall quality and safety.

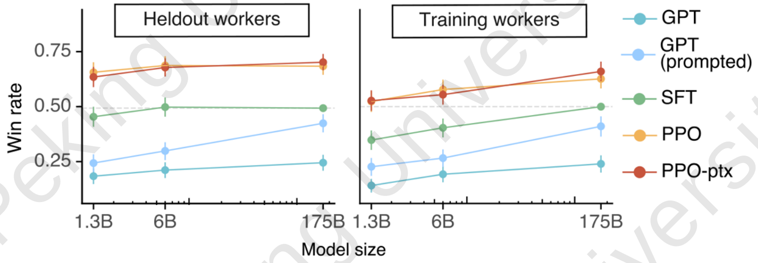


Fig. 1. Human evaluations of various models on the API prompt distribution, evaluated by how often outputs from each model were preferred to those from the 175B SFT model. Our InstructGPT models (PPO-ptx) as well as its variant trained without pretraining mix (PPO) significantly outperform the GPT-3 baselines (GPT, GPT prompted); outputs from our 1.3B PPO-ptx model are preferred to those from the 175B GPT-3 [OWJ+22].

Three stages in RLHF

We review the RLHF pipeline in [ZSW$^+$19]. It usually consists of three stages: 1) supervised fine-tuning (SFT); 2) preference sampling and reward learning and 3) reinforcement-learning optimization.

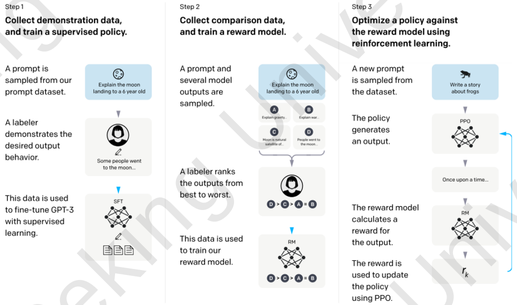

Step 1
Collect demonstration data, and train a supervised policy.
A prompt is sampled from our prompt dataset.
Explain the moon landing to a 6 year old
A labeler demonstrates the desired output behavior.
Some people went to the moon...
This data is used to fine-tune GPT-3 with supervised learning.
SFT
Step 2
Collect comparison data, and train a reward model.
A prompt and several model outputs are sampled.
Explain the moon landing to a 6 year old
A Explain gravity...
B Explain war...
C Moon is natural satellite of...
D People went to the moon...
A labeler ranks the outputs from best to worst.
D > C > A = B
This data is used to train our reward model.
RM
D > C > A = B
Step 3
Optimize a policy against the reward model using reinforcement learning.
A new prompt is sampled from the dataset.
Write a story about frogs
The policy generates an output.
PPO
Once upon a time...
The reward model calculates a reward for the output.
RM
The reward is used to update the policy using PPO.
r_k

Fig. 2. The diagram illustrating the three steps of RLHF.

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}.

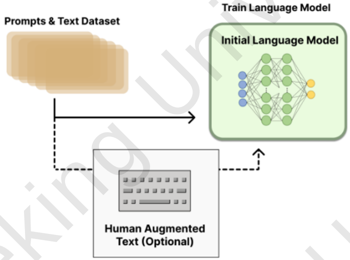


Fig. 3. The process of SFT.

Reward model (RM) training

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

$$y_w \succ y_l \mid x$$

where y_w and y_l denotes the preferred and dispreferred completion amongst (y_1, y_2) respectively. The preferences are assumed to be generated by some latent reward model $r^*(y, x)$, which we do not have access to. The BT [BT52] model stipulates that the human preference distribution p^* can be written as:

$$p^*\left(y_1 \succ y_2 \mid x\right) = \frac{\exp\left(r^*\left(x, y_1\right)\right)}{\exp\left(r^*\left(x, y_1\right)\right) + \exp\left(r^*\left(x, y_2\right)\right)}$$

Reward model (RM) training

Assuming access to a static dataset of comparisons:

$$\mathcal{D} = \left\{ x^{(i)}, y_w^{(i)}, y_l^{(i)} \right\}_{i=1}^{N}$$

sampled from p^*, we can parametrize a reward model $r_\phi(x, y)$ and estimate the parameters via maximum likelihood. The negative log-likelihood loss:

$$\mathcal{L}_R\left(r_\phi, \mathcal{D}\right) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}}\left[\log \sigma\left(r_\phi\left(x, y_w\right) - r_\phi\left(x, y_l\right)\right)\right]$$

where σ is the logistic function. In the context of LMs, the network $r_\phi(x, y)$ is often initialized from the SFT model $\pi^{\mathrm{SFT}}(y \mid x)$. To ensure a reward function with lower variance, prior works normalize the rewards, such that:

$$\mathbb{E}_{x, y \sim \mathcal{D}}\left[r_\phi(x, y)\right] = 0$$

for all x.

Reward model (RM) training

At this stage, we usually use smaller LLMs as reward models because this saves a lot of computation and we find that training larger reward models may be unstable.

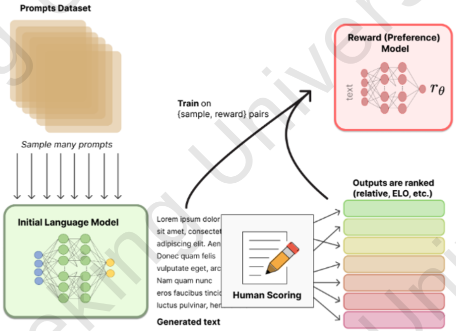

Prompts Dataset
Sample many prompts
Initial Language Model
Lorem ipsum dolor sit amet, consectet adipiscing elit. Aen Donec quam felis vulputate eget, arc Nam quam nunc eros faucibus tincic luctus pulvinar, her
Generated text
Human Scoring
Train on {sample, reward} pairs
Reward (Preference) Model
text
r_θ
Outputs are ranked (relative, ELO, etc.)

Fig. 4. The process of RM training.

Reinforcement learning via proximal policy 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:

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi_\theta(y|x)}\left[r_\phi(x,y)\right] - \beta\mathbb{D}_{\mathrm{KL}}\left[\pi_\theta(y \mid x)\|\pi_{\mathrm{ref}}(y \mid x)\right]$$

where β is a parameter controlling the deviation from the base reference policy π_{ref}, namely the initial SFT model π^{SFT}. The added constraint is important, as it prevents the model from deviating too far from the distribution on which the reward model is accurate, as well as maintaining the generation diversity and preventing mode-collapse to single high-reward answers.

Reinforcement learning via proximal policy optimization

Finally, the update rule is the parameter update from PPO [SWD+17] which is a trust region optimization algorithm that uses constraints on the gradient to ensure the update step does not destabilize the learning process.

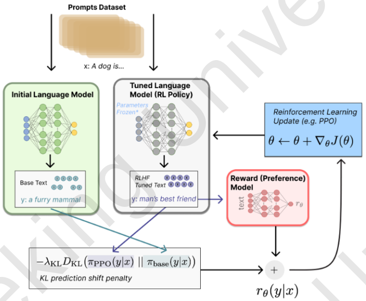

Prompts Dataset
x: A dog is...
Initial Language Model
Base Text
y: a furry mammal
Tuned Language Model (RL Policy)
Parameters Frozen*
RLHF Tuned Text
y: man's best friend
Reinforcement Learning Update (e.g. PPO)
$\theta \leftarrow \theta + \nabla_\theta J(\theta)$
Reward (Preference) Model
text
r_θ
$-\lambda_{KL} D_{KL}(\pi_{PPO}(y|x) \,||\, \pi_{base}(y|x))$
KL prediction shift penalty
+
$r_\theta(y|x)$

Fig. 5. The process of PPO training.

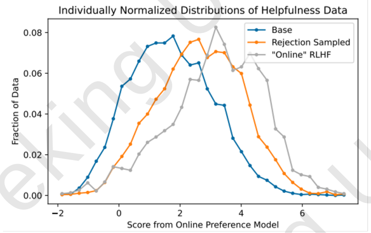
Iterated online RLHF

Optionally, RLHF can continue from this point by iteratively updating the reward model and the policy together. As the RL policy updates, users can continue ranking these outputs versus the model's earlier versions. In order to maintain stability during training [BJN+22]:

- We simply train the best RLHF policy we can, and use that to collect comparison data from crowdworkers. Since the policy was trained to optimize for PM score, it should produce responses that are on the upper end of the score distribution.
- We mix the new comparison data with our existing data, and train a new scan of PMs, which we then use to train a new scan of RLHF policies. Then reiterate this process indefinitely.

Iterated online RLHF

Online RLHF improved evaluation of high-quality responses.



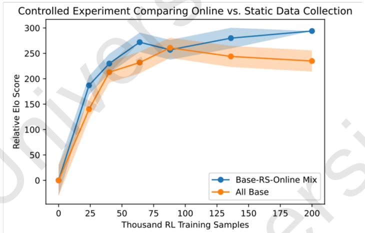


Fig. 6. (left) This plot shows individually normalized distributions of held-out helpfulness data from the base dataset (mostly with context-distilled models), from models augmented with rejection sampling, and from data collected with our iterated ‘online’ RLHF models. The upper tail of the distribution receives far more support from the RS and online models, which should make it possible for preference models to learn more subtle distinctions among high-quality responses, and amplify the value of further data collection. (right) [BJN+22] compares Elo scores from two 52B RLHF training runs that use equal-sized datasets and identical hyperparameters: one trained on our base dataset (orange), and another trained on an even mixture of data from the base, RS, and online distributions (blue). [BJN+22] finds that the iterated online model is preferred by crowdworkers.

Why do we focus on Reward Collapse?

The types of prompts:

- **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** resulting in a response that should be either highly or lowly scored, thus generating a roughly two-point mass distribution for the reward distribution. e.g. prove the Pythagorean theorem.

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

Why do we focus on Reward Collapse?

Training a reward model on preference rankings could result in the same reward distribution regardless of the prompts.

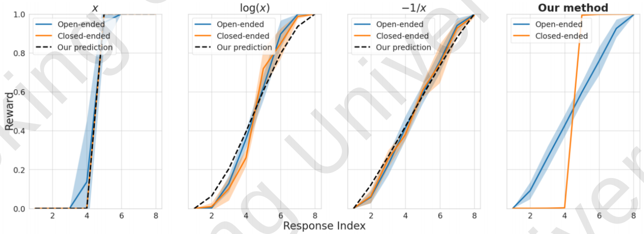


Fig. 7. Illustration of reward collapse, with rewards assigned to eight responses, arranged from least to most preferred. One type of prompt is open-ended, which should result in a roughly uniform distribution of rewards, while the other is closed-ended, which should yield either high or low rewards (polarized). However, as evidenced in the first three plots, when a common utility function is employed, the two types of prompts result in a strikingly similar reward distribution. Conversely, when a prompt-aware utility is applied, as seen in the fourth plot, the two types of prompts exhibit distinct reward distributions [SCLS23].

What is Reward Collapse?

Denote by $R($ prom, compl $)$ a reward model. We assume $0 \leq R($ prom, compl $) \leq 1$. For a given prompt prom and n completions that are i.i.d. draws from an LLM, a human labeler ranks the n responses from the most preferred to the least preferred, and the ranking is denoted as π_{prom} . We train a neural network that maximizes the following overall utility:

$$\sum_{(\text{prom},\text{compl}_w,\text{compl}_l)\in\Pi} U\left(R_\theta\left(\mathrm{prom}, \mathrm{compl}_w\right) - R_\theta\left(\text{ prom}, \text{ compl}_l\right)\right)$$

where U is an (increasing) utility function, θ is the weights of the reward neural network, and Π is the ranking dataset and compl_w is a preferred completion than compl_l in the ranking π_{prom} .

What is Reward Collapse?

In InstructGPT [OWJ$^+$22], U is set to:

$$U_\sigma(x) = \log \operatorname{sigmoid}(x/\sigma) \equiv \log \frac{e^{x/\sigma}}{e^{x/\sigma}+1}$$

Which is an increasing concave function. While maximizing:

$$\sum_{(\text{prom},\text{compl}_w,\text{compl}_l)\in\Pi} U\left(R_\theta\left(\text{prom},\text{compl}_w\right) - R_\theta\left(\text{prom},\text{compl}_l\right)\right)$$

The reward model learns to not only align with the human-provided ranking but also distinguish the rewards as much as possible. To gain insights into how the rewards depend on U, note that the above is equivalent to:

$$\max \sum_{\text{prom}} \sum_{(\text{compl}_w,\text{compl}_l)\in\pi_{\text{prom}}} U\left(R_\theta\left(\left(\text{prom},\text{compl}_w\right)\right) - R_\theta\left(\text{prom},\text{compl}_l\right)\right)$$

What is Reward Collapse?

Next, assume that the neural network parameterized by θ is sufficiently overparameterized such that:

$$\sum_{(\mathrm{compl}_w, \mathrm{compl}_l) \in \pi_{\mathrm{prom}}} U\left(R_\theta\left(\left(\text{ prom}, \mathrm{compl}_w\right)\right) - R_\theta\left(\text{ prom}, \mathrm{compl}_l\right)\right)$$

is exactly maximized. This is precisely the same as maximizing:

$$\sum_{1 \le i < j \le n} U\left(r_{\pi_{\mathrm{prom}}}(i) - r_{\pi_{\mathrm{prom}}}(j)\right)$$

over $0 \le r_1, \ldots, r_n \le 1$.

What is Reward Collapse?

However, the solution to this optimization program is independent of the prompt and, indeed, is the same as the solution to

$$\max_{0 \leq r_1, \ldots, r_n \leq 1} \sum_{1 \leq i < j \leq n} U\left(r_i - r_j\right)$$

up to a permutation. That is, the empirical distribution of the rewards is independent of the prompt itself in the interpolating regime, thereby leading to reward collapse.

Prompt-Aware Optimization

To avoid having the same reward distribution, a more principled approach is to change the objective. Our proposal is to let the utility function U now depend on the prompt. That is, now we consider training a neural network that maximizes

$$\sum_{(\text{prom},\text{compl}_w,\text{compl}_l)\in\Pi} U_{\text{prom}}\left(R_\theta\left(\mathrm{prom},\mathrm{compl}_w\right) - R_\theta\left(\text{prom}, \mathrm{compl}_l\right)\right)$$

In general, the choice of U_{prom} should reflect the open-endedness of the prompt. An important feature is that if U_{prom} is concave, this problem becomes a convex optimization problem.

Prompt-Aware Optimization

For a strictly increasing utility function U, it can be easily demonstrated that the maximum can only be attained when $r_1 \geq \cdots \geq r_n$. As a result, we only need to consider the problem

$$\max_{0 \leq r_n \leq \ldots \leq r_1 \leq 1} \sum_{1 \leq i < j \leq n} U\left(r_i - r_j\right)$$

Given the high flexibility in choosing U_{prom}, it is generally recommended to let the practitioners choose these functions to meet their needs.

Prompt-Aware Optimization

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

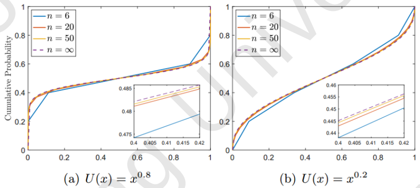


Fig. 8. Reward distribution for different utility function $U_\gamma(x) = x^\gamma, x \in [0,1]$

Theorem 1. Let $U_\gamma(x) = x^\gamma$ for some $0 < \gamma < 1$. Then the reward distribution of (4) converges to the Beta distribution Beta $\left(\frac{1-\gamma}{2}, \frac{1-\gamma}{2}\right)$ as $n \to \infty$.

Prompt-Aware Optimization

Class 2. Let $U_\gamma(x) = -x^{-\gamma}, x \in (0,1]$, for $0 < \gamma \le 1$ and $U_0(x) = \log x$ for $\gamma = 0$. We also let $U_\gamma(0) = -\infty$ for $0 \le \gamma \le 1$. In this case, the reward distribution becomes more even as γ increases from 0 to 1.

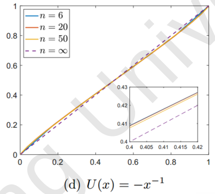


Fig. 9. Reward distribution for different utility function $U_\gamma(x) = -x^{-\gamma}, x \in (0,1]$

Theorem 2. For $U_\gamma(x) = -x^{-\gamma}$ for $0 \le \gamma \le 1$ (as a convention, take $U_0(x) = \log x$). Then. the reward distribution of (4) converges in distribution to $\mathrm{Beta}\left(\frac{1+\gamma}{2}, \frac{1+\gamma}{2}\right)$.

Prompt-Aware Optimization

Class 3. Let $U_\sigma(x) = \log \text{sigmoid}(x/\sigma)$ for $\sigma > 0$. The reward distribution becomes more spread between 0 and 1 as σ becomes smaller.

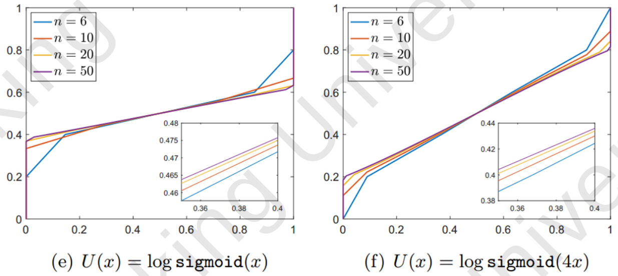


Fig. 10. Reward distribution for different utility function $U_\sigma(x) = \log \text{sigmoid}(x/\sigma)$ for $\sigma > 0$

Experiments

In the experiments, we focus on the following *U* functions: x, $\log x$, $-1/x$, logsigmoid(x), and the prompt-aware *U*, which adaptively selects *U* from x and $-1/x$. Given that the U function operates on x in the range $[-1, 1]$, we adjust some *U* functions with suitable continuous extensions or scaling.

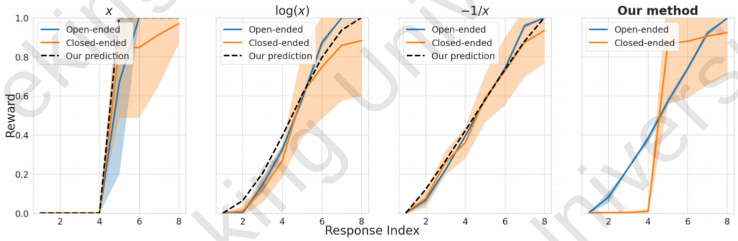


Fig. 11. Reward collapse on the test set. As we can see from the figure, the reward distributions have similar collapse phenomenons on the test set, and using prompt-aware loss can mitigate the collapse.

Experiments

Using distinct utility functions depending on prompts can help improve performance.

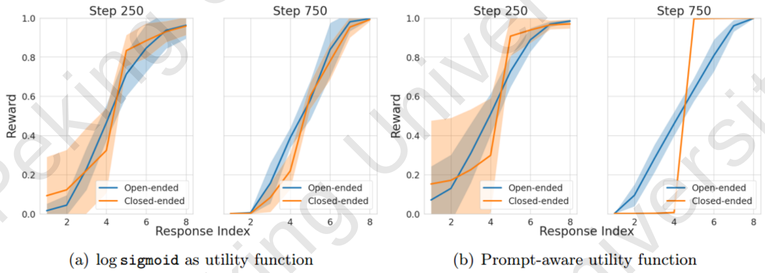


Fig. 12. (Left) Reward collapse when using log sigmoid as utility function. The reward distribution of different prompts gradually converges into a single distribution during training. (Right) Prompt-aware training avoids reward collapse. When using the prompt-aware loss function, the reward distributions of the two different prompts can be gradually separated during training.

Conclusions

Fixed loss function leads to reward collapse. As depicted in experiments, reward distributions corresponding to different prompts gradually converge towards a single, prompt-independent distribution throughout the training process.

Prompt-aware training avoids reward collapse. The results reveal that using a prompt-aware U function effectively prevents reward collapse across both training and test datasets. This strategy yields a more uniform reward distribution for open-ended prompts while promoting a more polarized reward distribution for concrete prompts.

Mitigate reward collapse during RLHF training can help improve the performance of LLMs.

Why do we need fine-grained reward

Standard preference-based rewards Such a reward provides a relatively sparse training signal, especially for tasks that require the generation of long-form text—making RLHF in such domains unreliable. Furthermore, it can be challenging for human annotators to reliably compare the overall quality of two or more model outputs when the outputs contain a mixture of diverse undesired behaviors.

Fine-grained reward rewards [WHS+23] proposes that humans give fine-grained feedback to LM output, associating categories of undesired behavior (e.g., false or irrelevant generations) and a text span at a density (e.g., sentence or sub-sentence-level).

Why do we need fine-grained reward

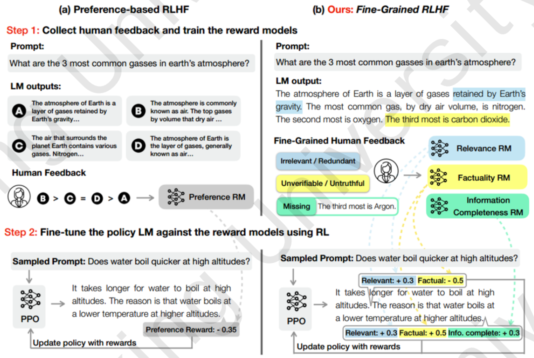

(a) Preference-based RLHF
(b) Ours: Fine-Grained RLHF
Step 1: Collect human feedback and train the reward models
Prompt:
What are the 3 most common gasses in earth's atmosphere?
LM outputs:
A The atmosphere of Earth is a layer of gases retained by Earth's gravity...
B The atmosphere is commonly known as air. The top gases by volume that dry air ...
C The air that surrounds the planet Earth contains various gases. Nitrogen...
D The atmosphere of Earth is the layer of gases, generally known as air...
Human Feedback
B > C = D > A
Preference RM
Prompt:
What are the 3 most common gasses in earth's atmosphere?
LM output:
The atmosphere of Earth is a layer of gases retained by Earth's gravity. The most common gas, by dry air volume, is nitrogen. The second most is oxygen. The third most is carbon dioxide.
Fine-Grained Human Feedback
Irrelevant / Redundant
Unverifiable / Untruthful
Missing The third most is Argon.
Relevance RM
Factuality RM
Information Completeness RM
Step 2: Fine-tune the policy LM against the reward models using RL
Sampled Prompt: Does water boil quicker at high altitudes?
PPO
It takes longer for water to boil at high altitudes. The reason is that water boils at a lower temperature at higher altitudes.
Preference Reward: - 0.35
Update policy with rewards
Sampled Prompt: Does water boil quicker at high altitudes?
Relevant: + 0.3 Factual: - 0.5
PPO
It takes longer for water to boil at high altitudes. The reason is that water boils at a lower temperature at higher altitudes.
Relevant: + 0.3 Factual: + 0.5 Info. complete: + 0.3
Update policy with rewards

Fig. 13. Comparison of (a) RL with human preference and (b) our FINE-GRAINED RLHF on long-form QA. Different from (a), which collects human preferences on the overall quality of LM outputs, we ask annotators to mark which part of an output contains what type(s) of errors. We train a fine-grained reward model for each type of error and optimize LM against these reward models. In this example, we provide a relevance reward and a factuality reward after each sentence is generated. There is also a holistic information completeness reward after the whole text is generated.

Formulation of fine-grained reward models

Previous RLHF works adopt a holistic reward model R_ϕ that maps input prompt x and generated output y to a single scalar reward representing its overall quality. This single scalar reward is only assigned to the final token in the generated sequence, a_T. Formally,

$$r_t = R_\phi(x, y)$$

if $t = T$ and 0 otherwise.

Fine-grained RLHF considers a reward function that is derived from one or multiple fine-grained reward models that:

- provide reward densely at a density level (i.e., for subsequences of the generated output)
- compute reward on distinct categories of undesired behaviors (e.g., false or repetitive generation), where each category is associated with an individual reward model.

Formulation of fine-grained reward models

For a fine-grained reward model R_{ϕ_k} that gives feedback on error category C_k, we first segment y into L_k segments

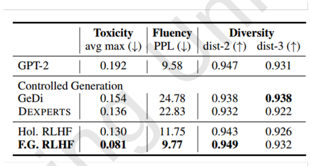
$$\left(y_1^k, y_2^k, \ldots, y_{L_k}^k\right)$$

corresponding to the density of R_{ϕ_k}, where each segment y_j^k ends at timestep T_j^k. R_{ϕ_k} outputs a reward $R_{\phi_k}(x, y, j)$ for each segment y_j^k given x and y as the input, which is assigned to the final token in y_j^k. Formally, assuming that we have K fine-grained reward models that represent different error categories, we will have a combined reward function for each token a_t as:

$$r_t = \sum_{k=1}^{K} \sum_{j=1}^{L_k} \left(\mathbb{I}\left(t = T_j^k\right) w_k R_{\phi_k}(x, y, j) \right) - \beta \log \frac{P_\theta\left(a_t \mid s_t\right)}{P_{\theta_{\text{init}}}\left(a_t \mid s_t\right)}$$

where $w_k \in \mathbb{R}$ is a weight assigned to reward model R_{ϕ_k}.

Experiments and conclusions



	Toxicity	Fluency	Diversity	
	avg max (↓)	PPL (↓)	dist-2 (↑)	dist-3 (↑)
GPT-2	0.192	9.58	0.947	0.931
Controlled Generation				
GeDi	0.154	24.78	0.938	**0.938**
DExperts	0.136	22.83	0.932	0.922
Hol. RLHF	0.130	11.75	0.943	0.926
F.G. RLHF	**0.081**	**9.77**	**0.949**	0.932

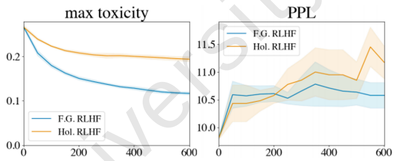


Fig. 14. (left) Results on the REALTOXICITY-PROMPTS test set. (right) Curves of toxicity and perplexity on the dev set v.s. training steps between Fine-Grained reward (F.G. RLHF), and Holistic reward (Hol. RLHF) [WHS+23].

Result 1 F.G. RLHF with sentence-level fine-grained reward attains the lowest toxicity and perplexity among all methods, while maintaining a similar level of diversity.
Result 2 F.G. RLHF has the toxicity drop much faster while keeping a low-level perplexity. This shows that learning from denser fine-grained reward is more sample efficient than holistic reward. The cost is that we have to query the reward model more times per example.

Unified alignment framework: f-DPG [GKK+23]

Aligning language models with preferences can be posed as approximating a target distribution representing some desired behavior. Existing approaches differ both in the functional form of the target distribution and the algorithm used to approximate it.

- Reinforcement Learning from Human Feedback (RLHF) corresponds to minimizing a reverse KL from an implicit target distribution arising from a KL penalty in the objective.
- Generative Distributional Control (GDC) has an explicit target distribution and minimizes a forward KL from it using the Distributional Policy Gradient (DPG) algorithm.
- ...

f-DPG unifies both frameworks (RLHF, GDC) and the approximation methods (DPG, RL with KL penalties).

Defining a Target Distribution

The target distribution expresses an ideal notion of an LM, incorporating human preferences, as probabilities $p(x)$ over texts x according to how well they satisfy the preferences. Formally, $p(x)$ is often defined through a non-negative function $P(x)$ (aka an energy-based model or EBM [LCH⁺06] such that $p(x) \propto P(x)$.

Binary preference For human preferences naturally expressible as a binary constraint $b(x) \in \{0, 1\}$, [KED20] proposed the following target distribution:

$$p_{\text{GDC}_{\text{bin}}}(x) \propto a(x)b(x)$$

where a is a pretrained LM and $b(x) = 0$ if x contains a curse and $b(x) = 1$ otherwise. $p_{\text{GDC}_{\text{bin}}}$ is the distri-bution enforcing that all samples match the binary con-straint, which deviates minimally from a as measured by $\mathrm{KL}\left(p_{\text{GDC}_{\text{bin}}} \| a\right)$.

Defining a Target Distribution

Scalar preferences applying RL with KL penalties [JGB+17] to maximize this reward while penalizing departure from $a(x)$:

$$J_{\mathrm{RLKL}}(\theta) = \mathbb{E}_{x\sim\pi_\theta}\left[r(x) - \beta \log \frac{\pi_\theta(x)}{a(x)}\right]$$

This objective can be equivalently framed as minimizing the reverse $\mathrm{KL}, \mathrm{KL}\left(\pi_\theta \| p_{\mathrm{RLKL}}\right)$, where the target distribution p_{RLKL} is defined as:

$$p_{\mathrm{RLKL}}(x) \propto a(x)\exp(r(x)/\beta)$$

where β is a hyperparameter.

Defining a Target Distribution

Distributional preferences cannot be expressed as a function of a single sample x but depend on the entire distribution, [KED20] model such preferences through distributional constraints using the following exponential family target distribution.

$$p_{\text{GDC}_{\text{dist}}}(x) \propto a(x) \exp \left[\sum_i \lambda_i \phi_i(x) \right]$$

where ϕ_i are features defined over texts and λ_i are co-efficients chosen so that the expected values $\mathbb{E}_{x\sim p}[\phi_i(x)]$ match some desired values $\bar{\mu}_i$. The resulting distribution $p_{\text{GDC-d}}$ matches the target feature moments, while deviating minimally from a as measured by $\mathrm{KL}\left(p_{\text{GDC}_{\text{dist}}} \| a\right)$.

Approximating the target distribution

Drawing samples from a target distribution p constitutes the inference problem. There are broadly two approaches to this problem:

- augmenting decoding from *a* at infer-ence time to obtain samples from *p*.
- training a new parametric model π_θ to approximate *p* which can then be sampled from directly.

we focus on the second methods to train a new model π_θ to approximate *p* by min-imizing a divergence measure from $p, D\left(\pi_\theta \| p\right)$. [KED20] uses Distributional Policy Gradients to approximate the target distribution by minimizing $\mathrm{KL}\left(p \| \pi_\theta\right)$, or equivalently, $\mathrm{CE}\left(p, \pi_\theta\right)$:

$$\nabla_\theta \mathrm{CE}\left(p, \pi_\theta\right)=-\mathbb{E}_{x \sim \pi_\theta} \frac{p(x)}{\pi_\theta(x)} \nabla_\theta \log \pi_\theta(x)$$

f-divergences

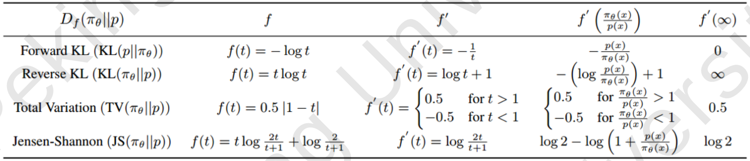
Consider a convex function $f : (0, \infty) \to \mathbb{R}$ with $f(1) = 0$. Let $f(0) \doteq \lim_{t \to 0} f(t)$ and $f'(\infty) \doteq \lim_{t \to 0} tf\left(\frac{1}{t}\right)\Big)^{1}$ Let p_1, p_2 be two distributions over a discrete set $\mathcal{X}$. The f -divergence between p_1 and p_2 can be defined as

$$D_f\left(p_1 \| p_2\right) \doteq \mathbb{E}_{x \sim p_2}\left[f\left(\frac{p_1(x)}{p_2(x)}\right)\right] + f'(\infty) p_1\left(p_2 = 0\right)$$

where $p_1\left(p_2 = 0\right)$ is the p_1-mass of the set $\{x \in \mathcal{X} : p_2(x) = 0\}$ [LV06].

f-divergences

The function f is called a generator of D_f. By convention, if $p_1(p_2 = 0) = 0$, the last term of the above equation is set to 0 regardless of the value of $f'(\infty)$ (which can be infinite). It can be shown that $D_f(p_1 \| p_2) \geq 0$ for any p_1 and p_2, with equality if $p_1 = p_2$; conversely, if $D_f(p_1 \| p_2) = 0$ and f is strictly convex at 1 , then $p_1 = p_2$.



$D_f(\pi_\theta \| p)$	f	f'	$f'\left(\frac{\pi_\theta(x)}{p(x)}\right)$	$f'(\infty)$
Forward KL ($\mathrm{KL}(p \| \pi_\theta)$)	$f(t) = -\log t$	$f'(t) = -\frac{1}{t}$	$-\frac{p(x)}{\pi_\theta(x)}$	0
Reverse KL ($\mathrm{KL}(\pi_\theta \| p)$)	$f(t) = t \log t$	$f'(t) = \log t + 1$	$-\left(\log \frac{p(x)}{\pi_\theta(x)}\right) + 1$	∞
Total Variation ($\mathrm{TV}(\pi_\theta \| p)$)	$f(t) = 0.5\,\lvert 1 - t \rvert$	$f'(t) = \begin{cases} 0.5 & \text{for } t > 1 \\ -0.5 & \text{for } t < 1 \end{cases}$	$\begin{cases} 0.5 & \text{for } \frac{\pi_\theta(x)}{p(x)} > 1 \\ -0.5 & \text{for } \frac{\pi_\theta(x)}{p(x)} < 1 \end{cases}$	0.5
Jensen-Shannon ($\mathrm{JS}(\pi_\theta \| p)$)	$f(t) = t \log \frac{2t}{t+1} + \log \frac{2}{t+1}$	$f'(t) = \log \frac{2t}{t+1}$	$\log 2 - \log\left(1 + \frac{p(x)}{\pi_\theta(x)}\right)$	$\log 2$

Fig. 15. Some common f-divergences $D_f(\pi_\theta \| p)$. In the convention of this table, the f shown corresponds to the order of arguments $D_f(\pi_\theta \| p)$. Thus the forward KL between the target p and the model, $\mathrm{KL}(p \| \pi_\theta)$, corresponds to $D_{-\log t}(\pi_\theta \| p)$, and similarly for the reverse KL, $\mathrm{KL}(\pi_\theta \| p)$, which corresponds to $D_{t \log t}(\pi_\theta \| p)$, etc. Note that for symmetric divergences (TV and JS) the order of arguments is indifferent: $\mathrm{TV}(\pi_\theta \| p) = \mathrm{TV}(p \| \pi_\theta)$, $\mathrm{JS}(\pi_\theta \| p) = \mathrm{JS}(p \| \pi_\theta)$.

Recovering Some Existing Methods

GDC In GDC, fitting the policy π_θ to the target p (which is given by either one of Eq. 1 or Eq. 4) is done using DPG (Par-shakova et al., 2019), namely by minimizing the **forward KL**, $\mathrm{KL}\left(p \| \pi_\theta\right)$. In the f-DPG framework, $\mathrm{KL}\left(p \| \pi_\theta\right) = D_f\left(\pi_\theta \| p\right)$ with $f(t) = -\log t, f'(t) = -1/t$, and the f-DPG leads to the formula:

$$\nabla_\theta D_f\left(\pi_\theta \| p\right) = \mathbb{E}_{x \sim \pi_\theta} - \frac{p(x)}{\pi_\theta(x)} \nabla_\theta \log \pi_\theta(x)$$

Recovering Some Existing Methods

RL with KL penalties Let's rewrite the target distribution of

$$p_{\mathrm{RLKL}}(x) \propto a(x) \exp(r(x)/\beta)$$

as $p(x) \doteq p_{\mathrm{RLKL}}(x) = 1/Za(x)e^{r(x)/\beta}$, where Z is a normaliser. Then $\mathrm{KL}\left(\pi_\theta \| p\right) = D_f\left(\pi_\theta \| p\right)$, with $f(t) = t \log t$ corresponding to **reverse KL**, and $f'(t) = 1 + \log t$. The f-DPG implies that:

$$\begin{aligned} &\nabla_\theta D_f\left(\pi_\theta \| p\right) \\ &= \mathbb{E}_{x \sim \pi_\theta}\left(1 + \log \frac{\pi_\theta(x)}{Z^{-1} a(x) \exp(r(x)/\beta)}\right) \nabla_\theta \log \pi_\theta(x) \\ &= \mathbb{E}_{x \sim \pi_\theta}\left(-\frac{r(x)}{\beta} + \log \frac{\pi_\theta(x)}{a(x)}\right) \nabla_\theta \log \pi_\theta(x), \end{aligned}$$

Recovering Some Existing Methods

$$\begin{aligned}&\nabla_\theta D_f(\pi_\theta \| p)\\ &= \mathbb{E}_{x\sim\pi_\theta}\left(1+\log\frac{\pi_\theta(x)}{Z^{-1}a(x)\exp(r(x)/\beta)}\right)\nabla_\theta\log\pi_\theta(x)\\ &= \mathbb{E}_{x\sim\pi_\theta}\left(-\frac{r(x)}{\beta}+\log\frac{\pi_\theta(x)}{a(x)}\right)\nabla_\theta\log\pi_\theta(x),\end{aligned}$$

where we have exploited the fact that $1+\log Z$ is a constant, hence $\mathbb{E}_{x\sim\pi_\theta}(1+\log Z)\nabla_\theta\log\pi_\theta(x) = 0$. Up to the constant factor β, this form re-covers the usual formula for estimating the gradient of the loss defined in:

$$\nabla_\theta J_{\mathrm{RLKL}}(\theta) = \mathbb{E}_{x\sim\pi_\theta}\left(r(x)-\beta\log\frac{\pi_\theta(x)}{a(x)}\right)\nabla_\theta\log\pi_\theta(x)$$

Various existing methods for aligning LM with preferences can be included in the f-DPG framework.

Empirical results of f-DPG framework

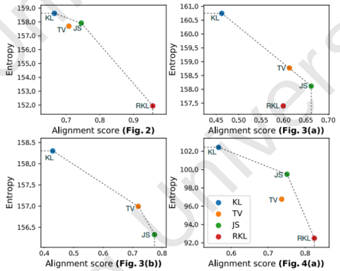


Fig. 16. Pareto frontier of f-DPG for different alignment tasks; sentiment preference (Fig. 2), lexical constraints (Fig. 3(a), (b)), and distributional constraint for gender prevalence (Fig. 4(a))

The fact that increasing the model size improves the alignment score but does not inherently bridge the gap between objectives underscores the importance of selecting appropriate divergence objectives.

Empirical results of f-DPG framework

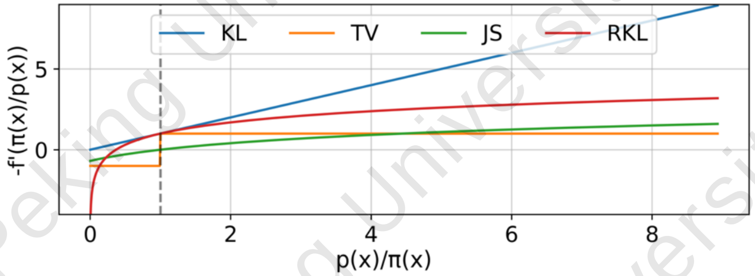


Fig. 17. Pseudo-rewards for various f-divergences. The x-axis denotes $\frac{p(x)}{\pi_\theta(x)}$ and the y-axis denotes the pseudo-reward. The dotted line denotes the point where $p(x) = \pi_\theta(x)$.

f-DPG Explained the differences between RLHF and other alignment methods

Summary and Outlook

In this lecture, we covered the fundamentals and recent advances of RLHF:

- RLHF in LLMs: Three stages in RLHF training pipline.
- How to better use the reward model in RLHF
- Differences between RLHF and other alignment methods.

In the next lecture, we will introduce RLHF:

- Using reward models to align.
- Self-Alignment

Thanks!

References I

[BJN+22] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al.
Training a helpful and harmless assistant with reinforcement learning from human feedback.
arXiv preprint arXiv:2204.05862, 2022.

[BT52] Ralph Allan Bradley and Milton E Terry.
Rank analysis of incomplete block designs: I. the method of paired comparisons.
Biometrika, 39(3/4):324–345, 1952.

[GKK+23] Dongyoung Go, Tomasz Korbak, Germán Kruszewski, Jos Rozen, Nahyeon Ryu, and Marc Dymetman.
Aligning language models with preferences through f-divergence minimization.
arXiv preprint arXiv:2302.08215, 2023.

[JGB+17] Natasha Jaques, Shixiang Gu, Dzmitry Bahdanau, José Miguel Hernández-Lobato, Richard E Turner, and Douglas Eck.
Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control.
In *International Conference on Machine Learning*, pages 1645–1654. PMLR, 2017.

[KED20] Muhammad Khalifa, Hady Elsahar, and Marc Dymetman.
A distributional approach to controlled text generation.
arXiv preprint arXiv:2012.11635, 2020.

[LCH+06] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and Fujie Huang.
A tutorial on energy-based learning.
Predicting structured data, 1(0), 2006.

References II

[LV06] Friedrich Liese and Igor Vajda.
On divergences and informations in statistics and information theory.
IEEE Transactions on Information Theory, 52(10):4394–4412, 2006.

[OWJ+22] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al.
Training language models to follow instructions with human feedback.
Advances in Neural Information Processing Systems, 35:27730–27744, 2022.

[SCLS23] Ziang Song, Tianle Cai, Jason D Lee, and Weijie J Su.
Reward collapse in aligning large language models.
arXiv preprint arXiv:2305.17608, 2023.

[SWD+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov.
Proximal policy optimization algorithms.
arXiv preprint arXiv:1707.06347, 2017.

[WHS+23] Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A Smith, Mari Ostendorf, and Hannaneh Hajishirzi.
Fine-grained human feedback gives better rewards for language model training.
arXiv preprint arXiv:2306.01693, 2023.

[ZSW+19] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving.
Fine-tuning language models from human preferences.
arXiv preprint arXiv:1909.08593, 2019.