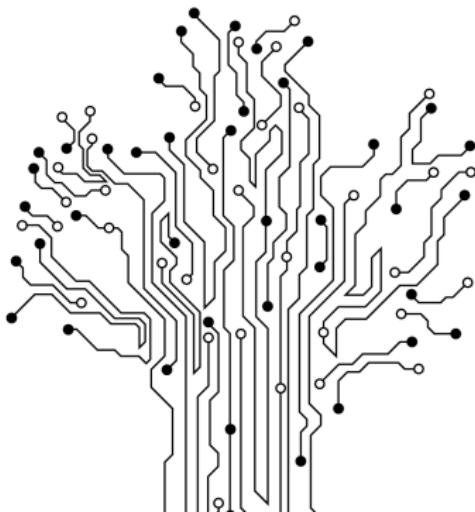


RLHF without RL

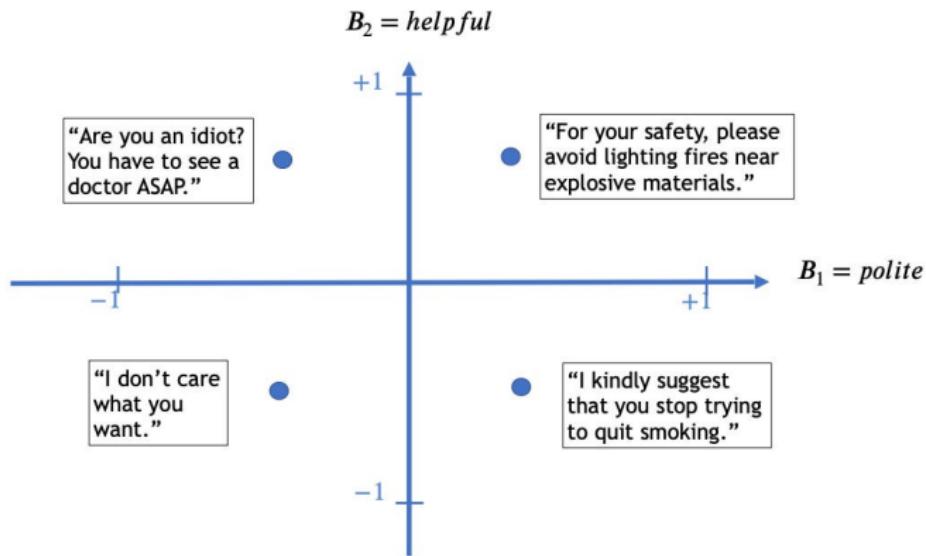
Ivan Bondyrev



План доклада

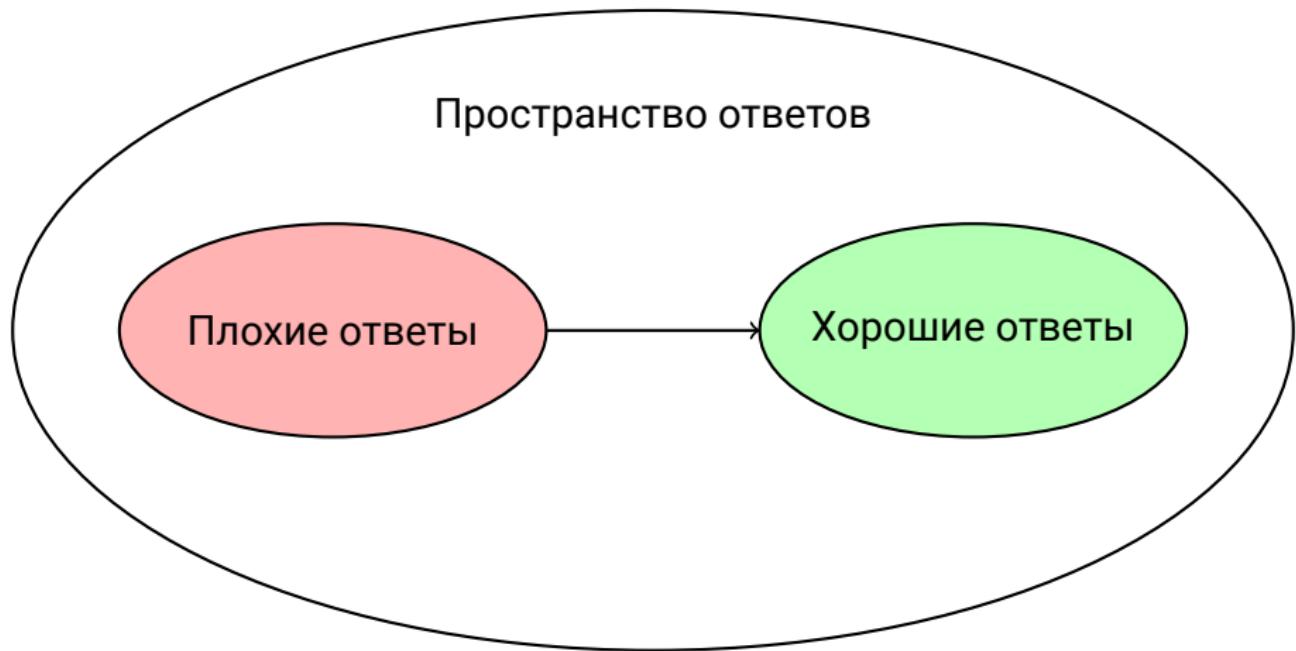
- 1 Контекст: Alignment
- 2 RLHF
- 3 Недостакти RLHF
- 4 Direct Preference Optimization
- 5 Chain of Hindsight
- 6 Выводы

Контекст: Alignment



Пример полезных/бесполезных ответов

Контекст: Alignment



RLHF

Step 1

Collect demonstration data and train a supervised policy.

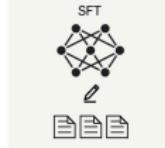
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



This data is used to fine-tune GPT-3.5 with supervised learning.



Step 2

Collect comparison data and train a reward model.

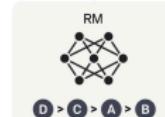
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



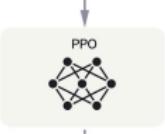
This data is used to train our reward model.



Step 3

Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

A new prompt is sampled from the dataset.



The PPO model is initialized from the supervised policy.



The policy generates an output.

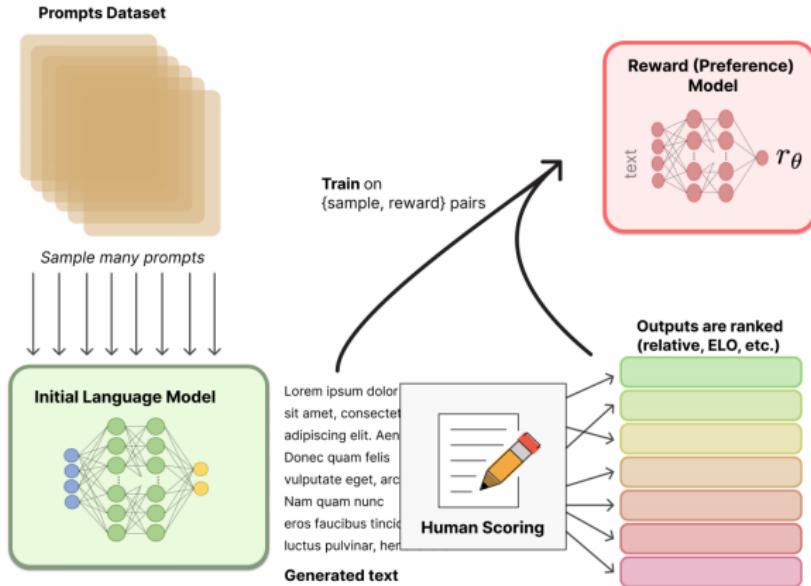


The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.

RLHF для ChatGPT



Обучение Reward Model на человеческих оценках

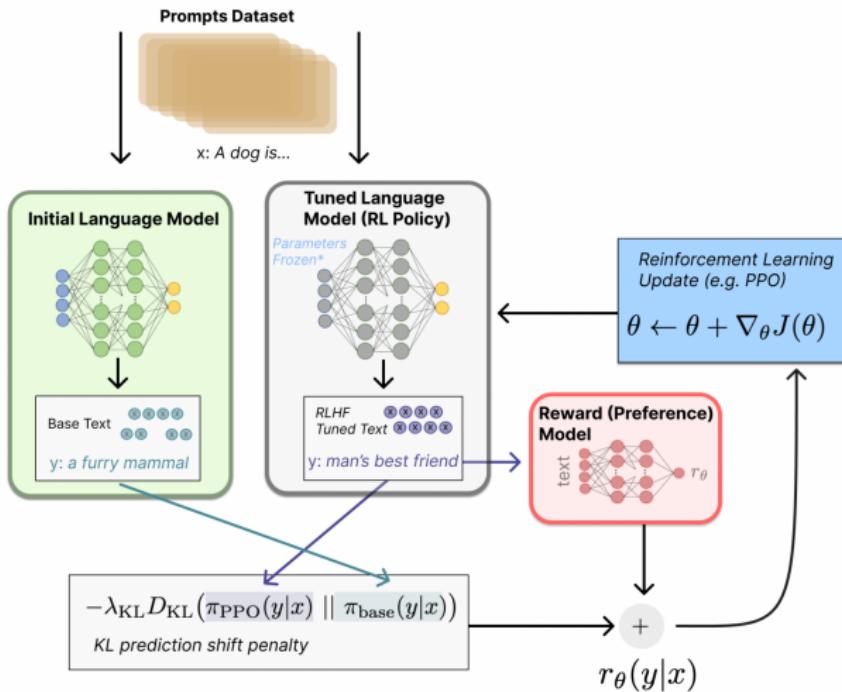


Схема обучения RLHF

Недостатки RLHF

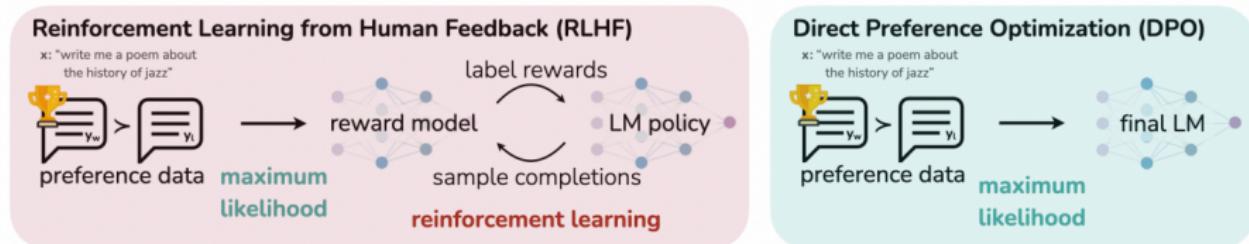
- Нужно завести целых 3 LLMки
 - 1 Supervised Model
 - 2 Reward Model (хотя она может быть и не большой)
 - 3 Tuned LLM
- RL бывает очень нестабилен и сильно зависит от гиперпараметров

Вопросы?

Direct Preference Optimization [2]

DPO

В первом приближении



В конце концов хотелось бы полностью избавиться от RL и моделирования функции награды в явном виде

- 1 **Step: Self-Supervised Fine-Tuning.** На этом шаге просто делаем fine-tune/обучаем нашу модель на качественных данных под нашу задачу.

- ➊ **Step: Self-Supervised Fine-Tuning.** На этом шаге просто делаем fine-tune/обучаем нашу модель на качественных данных под нашу задачу.
- ➋ **Step: Reward Modelling.** Здесь мы обучаем Reward Model, исходя из предположения, что у нас есть датасет $D = \{x^{(i)}, y_w^{(i)}, y_l^{(i)}\}_{i=1}^N$ и распределение p^* на $y^{(i)}$ ответах соответствует формуле (1) (так называемая Bradley-Terry model):

$$\bullet p^*(y_1 \succ y_2 | x) = \frac{\exp(r^*(x, y_1))}{\exp(r^*(x, y_1)) + \exp(r^*(x, y_2))} = \sigma(r^*(x, y_1) - r^*(x, y_2)) \quad (1)$$

$$\bullet \mathcal{L}_R(r_\phi, D) = -\mathbb{E}_{(x, y_w, y_l) \sim D} [\log \sigma(r_\phi(x, y_w) - r_\phi(x, y_l))] \quad (2)$$

1 **Step: Self-Supervised Fine-Tuning.** На этом шаге просто делаем fine-tune/обучаем нашу модель на качественных данных под нашу задачу.

2 **Step: Reward Modelling.** Здесь мы обучаем Reward Model, исходя из предположения, что у нас есть датасет

$D = \{x^{(i)}, y_w^{(i)}, y_l^{(i)}\}_{i=1}^N$ и распределение p^* на $y^{(i)}$ ответах соответствует формуле (1) (так называемая Bradley-Terry model):

$$\bullet p^*(y_1 \succ y_2 | x) = \frac{\exp(r^*(x, y_1))}{\exp(r^*(x, y_1)) + \exp(r^*(x, y_2))} = \sigma(r^*(x, y_1) - r^*(x, y_2)) \quad (1)$$

$$\bullet \mathcal{L}_R(r_\phi, D) = -\mathbb{E}_{(x, y_w, y_l) \sim D} [\log \sigma(r_\phi(x, y_w) - r_\phi(x, y_l))] \quad (2)$$

3 **Step: RL Fine-Tuning Phase** На этой стадии мы учимся с помощью RL, оптимизируя следующее:

$$\bullet \max_{\pi_\theta} \mathbb{E}_{x \sim D, y \sim \pi_\theta(y|x)} [r_\phi(x, y)] - \beta D_{KL} [\pi_\theta(y | x) \parallel \pi_{\text{ref}}(y | x)] \quad (3)$$

Выразим Reward Function через оптимальную политику

- 1 Не трудно показать (в статье [2] есть полные выкладки), что из этого объекта оптимизации:

- $\max_{\pi_\theta} \mathbb{E}_{x \sim D, y \sim \pi_\theta(y|x)} [r_\phi(x, y)] - \beta D_{KL} [\pi_\theta(y|x) \parallel \pi_{\text{ref}}(y|x)]$ (3)

Напрямую следует выражение для оптимальной политики индуцированной Reward Function $r(x, y)$:

- $\pi_r(y|x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)$ (4)
 $Z(x) = \sum_y \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)$

Выразим Reward Function через оптимальную политику

- 1 Не трудно показать (в статье [2] есть полные выкладки), что из этого объекта оптимизации:

- $\max_{\pi_\theta} \mathbb{E}_{x \sim D, y \sim \pi_\theta(y|x)} [r_\phi(x, y)] - \beta D_{KL} [\pi_\theta(y|x) \parallel \pi_{\text{ref}}(y|x)]$ (3)

Напрямую следует выражение для оптимальной политики индуцированной Reward Function $r(x, y)$:

- $\pi_r(y|x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)$ (4)
 $Z(x) = \sum_y \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)$

- 2 Теперь можем выразить функцию награды через оптимальную политику, которая ей соответствует:

- $r(x, y) = \beta \log\left(\frac{\pi_r(y|x)}{\pi_{\text{ref}}(y|x)}\right) + \beta \log Z(x)$ (5)

Получаем так называемый implicit reward

DPO

Выводим лосс для DPO

- 1 Полученное ранее представление для функции награды $r(x, y)$ справедливо для любой r , в т.ч. и для настоящей r^* .
Подставляя представление (5) для r^* в модель Bradley-Terry $p^*(y_1 \succ y_2 | x) = \sigma(r^*(x, y_1) - r^*(x, y_2))$, получаем :

$$\bullet p^*(y_1 \succ y_2 | x) = \frac{1}{1 + \exp\left(\beta \log \frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)} - \beta \log \frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)}\right)} = \sigma(\dots) \quad (6)$$

DPO

Выводим лосс для DPO

- 1 Полученное ранее представление для функции награды $r(x, y)$ справедливо для любой r , в т.ч. и для настоящей r^* . Подставляя представление (5) для r^* в модель Bradley-Terry $p^*(y_1 \succ y_2|x) = \sigma(r^*(x, y_1) - r^*(x, y_2))$, получаем :

$$\bullet p^*(y_1 \succ y_2|x) = \frac{1}{1 + \exp\left(\beta \log \frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)} - \beta \log \frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)}\right)} = \sigma(\dots) \quad (6)$$

- 2 Фактически, мы получили явное выражение распределения p^* через идеальную политику $\pi^*(y|x)$, значит можем обучать нашу $\pi_\theta(y|x)$, минимизируя следующий NLL:

$$\bullet \mathcal{L}_{DPO}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim D} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right) \right] \quad (7)$$

DPO

Смотрим на градиент лосса

$$\begin{aligned}\nabla_{\theta} \mathcal{L}_{DPO}(\pi_{\theta}; \pi_{\text{ref}}) = \\ -\beta \mathbb{E}_{(x, y_w, y_l) \sim D} [\sigma(\hat{r}_{\theta}(x, y_l) - \hat{r}_{\theta}(x, y_w)) [\nabla_{\theta} \log \pi(y_w|x) - \nabla_{\theta} \log \pi(y_l|x)]]\end{aligned}$$

where $\hat{r}_{\theta}(x, y) = \beta \log \frac{\pi_{\theta}(y|x)}{\pi_{\text{ref}}(y|x)}$

DPO

Смотрим на градиент лосса

$$\nabla_{\theta} \mathcal{L}_{DPO}(\pi_{\theta}; \pi_{\text{ref}}) = -\beta \mathbb{E}_{(x, y_w, y_l) \sim D} \left[\underbrace{\sigma(\hat{r}_{\theta}(x, y_l) - \hat{r}_{\theta}(x, y_w))}_{\begin{array}{c} \text{higher weight when reward estimate is wrong} \\ \text{increase likelihood of } y_w \end{array}} \right. \\ \left. \underbrace{[\nabla_{\theta} \log \pi_{\theta}(y_w|x) - \nabla_{\theta} \log \pi_{\theta}(y_l|x)]}_{\begin{array}{c} \text{decrease likelihood of } y_l \end{array}} \right],$$

$$\text{where } \hat{r}_{\theta}(x, y) = \beta \log \frac{\pi_{\theta}(y|x)}{\pi_{\text{ref}}(y|x)}$$

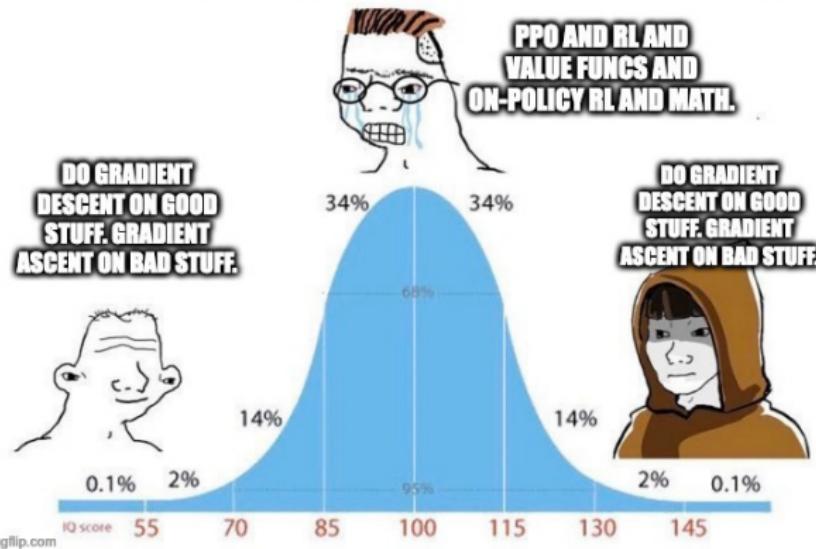
β – это главный гиперпараметр, он отвечает за KL ограничение, а также регулирует то, насколько сильнее будет разница в награде у плохого ответа и хорошего.

DPO

Смотрим на градиент лосса

$$\nabla_{\theta} \mathcal{L}_{DPO}(\pi_{\theta}; \pi_{\text{ref}}) = -\beta \mathbb{E}_{(x, y_w, y_l) \sim D} [\sigma(\hat{r}_{\theta}(x, y_l) - \hat{r}_{\theta}(x, y_w)) [\nabla_{\theta} \log \pi(y_w|x) - \nabla_{\theta} \log \pi(y_l|x)]]$$

LEARNING FROM HUMAN FEEDBACK



DPO

Несколько замечаний

- DPO-loss очень легок в реализации, градиент хорошо считается стандартными силами торча, никаких приседаний



```
import torch.nn.functional as F

def dpo_loss(pi_logps, ref_logps, yw_idxs, yl_idxs, beta):
    pi_yw_logps, pi_yl_logps = pi_logps[yw_idxs], pi_logps[yl_idxs]
    ref_yw_logps, ref_yl_logps = ref_logps[yw_idxs], ref_logps[yl_idxs]
    pi_logratios = pi_yw_logps - pi_yl_logps
    ref_logratios = ref_yw_logps - ref_yl_logps
    losses = -F.logsigmoid(beta * (pi_logratios - ref_logratios))
    rewards = beta * (pi_logps - ref_logps).detach()
    return losses, rewards
```

DPO

Несколько замечаний

- DPO-loss очень легок в реализации, градиент хорошо считается стандартными силами торча, никаких приседаний



```
import torch.nn.functional as F

def dpo_loss(pi_logps, ref_logps, yw_idxs, yl_idxs, beta):
    pi_yw_logps, pi_yl_logps = pi_logps[yw_idxs], pi_logps[yl_idxs]
    ref_yw_logps, ref_yl_logps = ref_logps[yw_idxs], ref_logps[yl_idxs]
    pi_logratios = pi_yw_logps - pi_yl_logps
    ref_logratios = ref_yw_logps - ref_yl_logps
    losses = -F.logsigmoid(beta * (pi_logratios - ref_logratios))
    rewards = beta * (pi_logps - ref_logps).detach()
    return losses, rewards
```

- В статье авторы подробно доказывают, почему при некоторых допущениях DPO эквивалентна задаче RLHF и что DPO, при всей простоте, не теряет общности. В частности, доказывают, что в предположениях Bradley-Terry модели, все функции награды разбиваются на классы эквивалентности и все классы представимы уникальным образом в виде (5).

1 Подготовка датасета

- Генерируем ответ на запрос $x, y_1, y_2 \sim \pi_{\text{ref}}(\cdot|x)$
- С помощью кожаных мешков (или другой модели) сравниваем пары ответов, чтобы получить $D = \{x^{(i)}, y_w^{(i)}, y_l^{(i)}\}_{i=1}^N$

1 Подготовка датасета

- Генерируем ответ на запрос $x, y_1, y_2 \sim \pi_{\text{ref}}(\cdot|x)$
- С помощью кожаных мешков (или другой модели) сравниваем пары ответов, чтобы получить $D = \{x^{(i)}, y_w^{(i)}, y_l^{(i)}\}_{i=1}^N$

2 Обучение

- Инициализируем модель π_θ с помощью $\pi_{\text{ref}} = \pi_{\text{SFT}}$ (если не имеем доступа к π_{SFT} , то стараемся ее как-то приблизить)
- Учим π_θ используя DPO loss

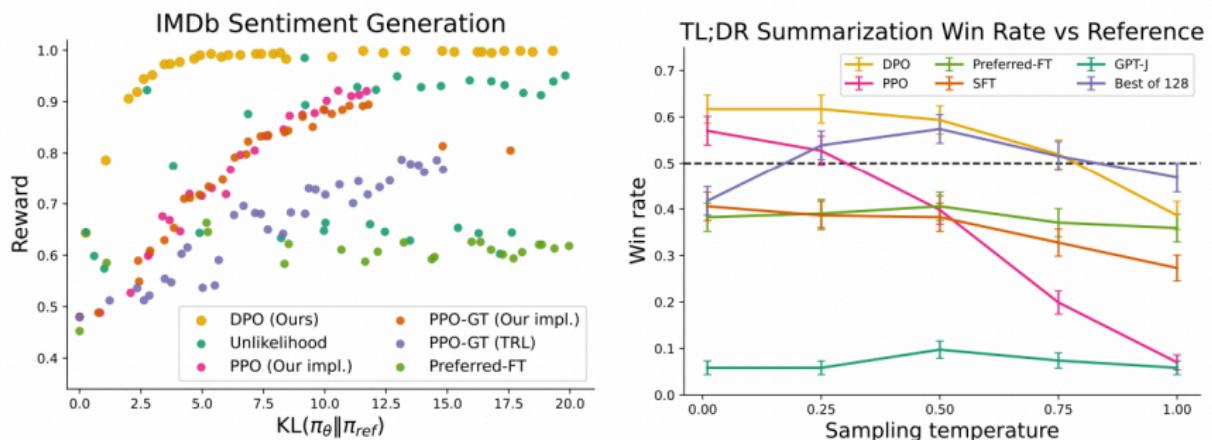


Figure 2: **Left.** The frontier of expected reward vs KL to the reference policy. DPO provides the highest expected reward for all KL values, demonstrating the quality of the optimization. **Right.** TL;DR summarization win rates vs. human-written summaries, using GPT-4 as evaluator. DPO exceeds PPO's best-case performance on summarization, while being more robust to changes in the sampling temperature.

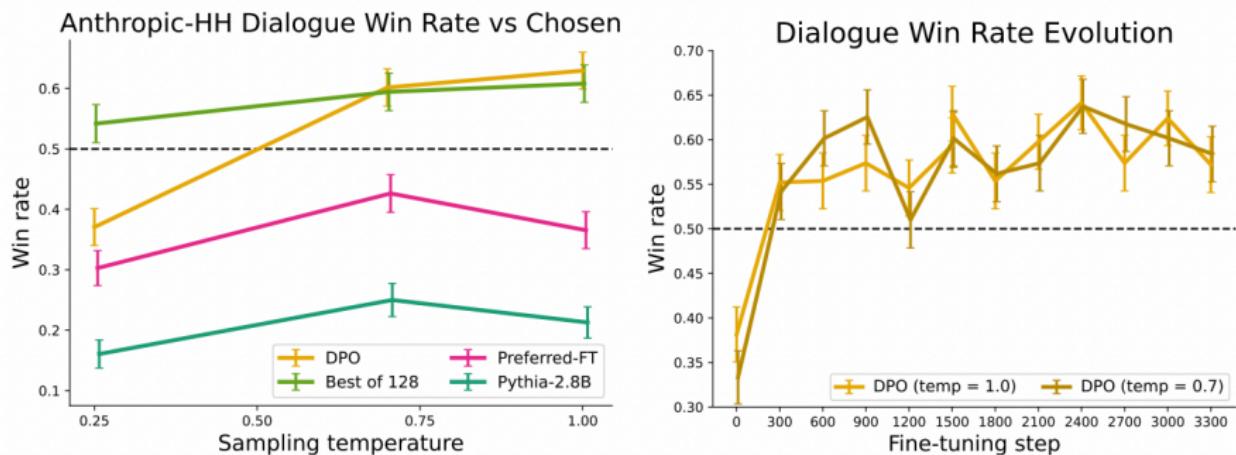


Figure 3: **Left.** Win rates computed by GPT-4 for Anthropic-HH one-step dialogue; DPO is the only method that improves over chosen summaries in the Anthropic-HH test set. **Right.** Win rates for different sampling temperatures over the course of training. DPO's improvement over the dataset labels is fairly stable over the course of training for different sampling temperatures.

DPO

Эксперименты и сравнение с другими подходами

Win rate vs. ground truth		
Alg.	Temp 0	Temp 0.25
DPO	0.36	0.31
PPO	0.26	0.23

Table 1: GPT-4 win rates vs. ground truth summaries for out-of-distribution CNN/DailyMail input articles.

	DPO	SFT	PPO-1
N respondents	272	122	199
GPT-4 (S) win %	47	27	13
GPT-4 (C) win %	54	32	12
Human win %	58	43	17
GPT-4 (S)-H agree	70	77	86
GPT-4 (C)-H agree	67	79	85
H-H agree	65	-	87

Table 2: Comparing human and GPT-4 win rates and per-judgment agreement on TL;DR summarization samples. **Humans agree with GPT-4 about as much as they agree with each other.** Each experiment compares a summary from the stated method with a summary from PPO with temperature 0.

- Плюсы

- + Нет явной оценки Reward модели

- Плюсы

- + Нет явной оценки Reward модели
- + Нет RL в явном виде

- Плюсы

- + Нет явной оценки Reward модели
- + Нет RL в явном виде
- + Всего 1 модель, которую нужно обучать в процессе alignment'a

- Плюсы

- + Нет явной оценки Reward модели
- + Нет RL в явном виде
- + Всего 1 модель, которую нужно обучать в процессе alignment'a
- + Мало гиперпараметров

- Плюсы

- + Нет явной оценки Reward модели
- + Нет RL в явном виде
- + Всего 1 модель, которую нужно обучать в процессе alignment'a
- + Мало гиперпараметров
- + Лучше обобщающая способность?

- Плюсы

- + Нет явной оценки Reward модели
- + Нет RL в явном виде
- + Всего 1 модель, которую нужно обучать в процессе alignment'a
- + Мало гиперпараметров
- + Лучше обобщающая способность?
- + Очень просто в реализации

- Плюсы

- + Нет явной оценки Reward модели
- + Нет RL в явном виде
- + Всего 1 модель, которую нужно обучать в процессе alignment'a
- + Мало гиперпараметров
- + Лучше обобщающая способность?
- + Очень просто в реализации
- + Более устойчивое обучение по сравнению с RL

- Плюсы

- + Нет явной оценки Reward модели
- + Нет RL в явном виде
- + Всего 1 модель, которую нужно обучать в процессе alignment'a
- + Мало гиперпараметров
- + Лучше обобщающая способность?
- + Очень просто в реализации
- + Более устойчивое обучение по сравнению с RL

- Минусы

- Как внедрить вещественные оценки ответов? DPO работает только с относительным порядком

• Плюсы

- + Нет явной оценки Reward модели
- + Нет RL в явном виде
- + Всего 1 модель, которую нужно обучать в процессе alignment'a
- + Мало гиперпараметров
- + Лучше обобщающая способность?
- + Очень просто в реализации
- + Более устойчивое обучение по сравнению с RL

• Минусы

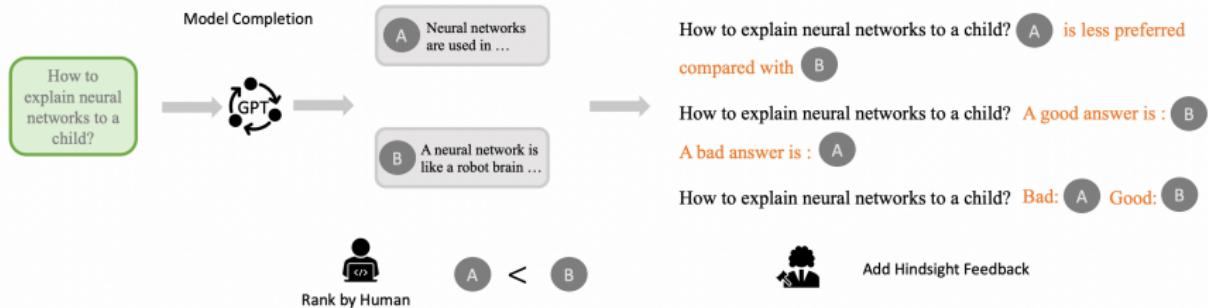
- Как внедрить вещественные оценки ответов? DPO работает только с относительным порядком
- Из-за того, что мы выкинули Reward модель, мы становимся ограничены собранным фидбэк-датасетом. Тогда как в случае RLHF у нас есть модель, которая может оценивать новые ответы онлайн, и цикл обучения можно повторять теоретически сколько угодно раз

Вопросы?

Chain of Hindsight [1]

Chain of Hindsight

Идея: будем тренировать модель, обуславливая на feedback в текстовом виде



Natural language feedback examples

A good summary: {positive}, a worse summary: {negative}

You are a helpful assistant: {positive}, you are an unhelpful assistant: {negative}

A bad answer is {negative}, a good answer is {positive}

Chain of Hindsight

CoH loss

Самый обычный лосс для языковой модели:

- $\log p(\mathbf{x}) = \log \prod_{i=1}^n p(x_i | \mathbf{x}_{<i})$

Мы возьмем промпт, добавим к нему пару ответов модели с фидбэком в текстовом виде и будем обучать, используя следующий лосс:

- $\log p(\mathbf{x}) = \log \prod_{i=1}^n \mathbb{1}_{O(x)}(x_i) p(x_i | [x_j]_{j=0}^{i-1})$, где $\mathbb{1}_{O(x)}(x_i) = 1$ если x_i не часть фидбэка и 0 иначе.

Также авторы маскируют 5-10% токенов из ответов, которые модель видит, чтобы она просто не копировала хороший ответ в плохой. И еще используют регуляризацию, которая не дает далеко уходить от изначального датасета.

Chain of Hindsight

Алгоритм

- Берем предобученную языковую модель и датасет с оценками D.

Chain of Hindsight

Алгоритм

- Берем предобученную языковую модель и датасет с оценками D.
- В цикле:
 - Выбираем батч ответов и связанных с ними фидбеков из D.

Chain of Hindsight

Алгоритм

- Берем предобученную языковую модель и датасет с оценками D.
- В цикле:
 - Выбираем батч ответов и связанных с ними фидбеков из D.
 - Конструируем тренировочные последовательности, переводя фидбек в текстовый вид.

Chain of Hindsight

Алгоритм

- Берем предобученную языковую модель и датасет с оценками D.
- В цикле:
 - Выбираем батч ответов и связанных с ними фидбеков из D.
 - Конструируем тренировочные последовательности, переводя фидбек в текстовый вид.
 - Обучаем.

Chain of Hindsight

Эксперименты и сравнение с другими методами

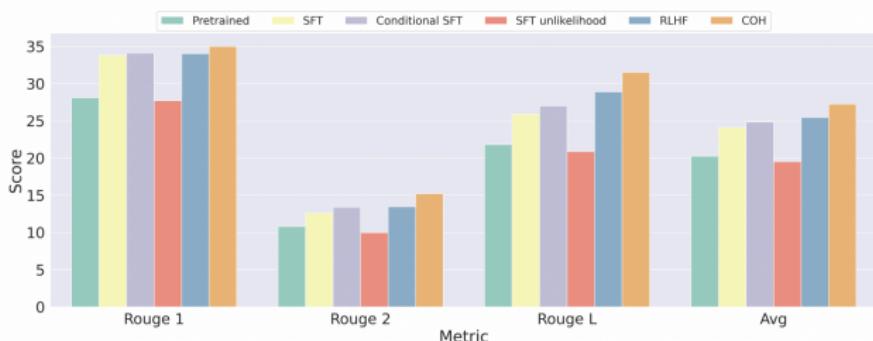


Figure 3: **Evaluation on summarization.** Comparison between RLHF, SFT and CoH. The metrics are ROUGE scores on TL;DR summarization task.

Table 1: Pairwise human evaluation on summarization task.

	Human evaluation win rate (%)			
	Base	Tie	CoH	Δ
Accuracy	24.5	26.8	48.7	24.2
Coherence	15.6	18.5	65.9	50.3
Coverage	19.6	22.4	58.0	38.4
Average	19.9	22.6	57.5	37.6
	SFT	Tie	CoH	Δ
Accuracy	25.5	32.6	41.9	16.4
Coherence	30.5	25.6	43.9	13.4
Coverage	28.5	25.4	46.1	17.6
Average	28.2	27.9	44.0	15.8
	C-SFT	Tie	CoH	Δ
Accuracy	26.7	34.9	38.4	11.7
Coherence	32.5	22.9	44.6	12.1
Coverage	29.5	26.7	43.8	14.3
Average	29.6	28.2	42.3	12.7
	SFT-U	Tie	CoH	Δ
Accuracy	18.7	17.9	63.4	44.7
Coherence	21.8	15.8	62.4	40.6
Coverage	23.6	17.2	59.2	35.6
Average	21.4	17.0	61.7	40.3
	RLHF	Tie	CoH	Δ
Accuracy	31.8	29.5	38.7	6.9
Coherence	31.6	20.5	47.9	16.4
Coverage	28.9	21.9	49.2	20.3
Average	30.8	24.0	45.3	14.5

Chain of Hindsight

Эксперименты и сравнение с другими методами

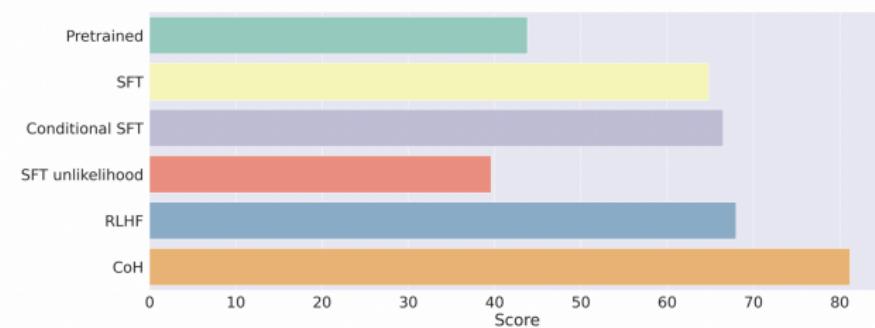


Figure 4: **Evaluation on dialogue.** Comparing CoH with RLHF and SFT baselines. The metric is the accuracy of classifying the preferred dialogue.

Table 2: Pairwise human evaluation on dialogue task.

	Human evaluation win rate (%)			
	Base	Tie	CoH	Δ
Helpful	15.8	34.8	49.4	33.6
Harmless	14.5	35.9	49.6	35.1
Average	15.2	35.3	49.5	34.4
	SFT	Tie	CoH	Δ
Helpful	19.6	45.7	34.7	15.1
Harmless	18.6	37.4	44.0	25.4
Average	19.1	41.5	39.4	20.3
	C-SFT	Tie	CoH	Δ
Helpful	21.8	46.9	31.3	9.5
Harmless	22.4	35.2	42.4	20.0
Average	22.1	41.0	36.8	14.7
	SFT-U	Tie	CoH	Δ
Helpful	13.4	31.3	55.3	41.9
Harmless	14.5	28.7	56.8	42.3
Average	13.9	30.0	56.0	42.1
	RLHF	Tie	CoH	Δ
Helpful	25.8	40.8	33.4	7.6
Harmless	20.9	38.8	40.3	19.4
Average	23.4	39.8	36.9	13.5

Chain of Hindsight

Эксперименты и сравнение с другими методами

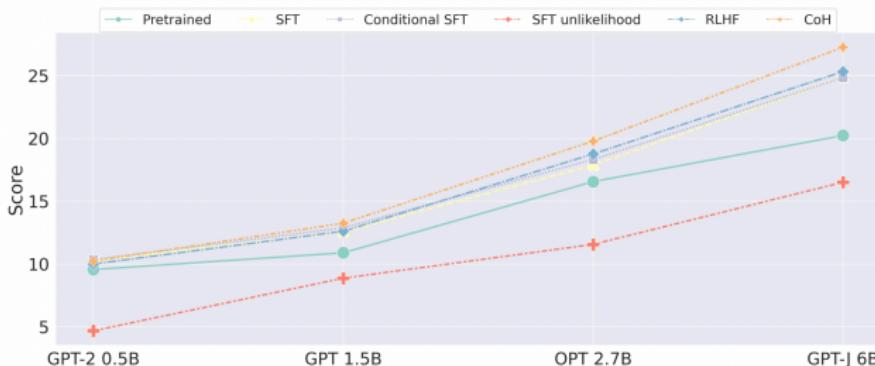


Figure 5: Model scaling trend. Comparing CoH with RLHF and SFT baselines on summarization benchmark with different model sizes. CoH outperforms RLHF, showing strong scaling capabilities.

CoH consistently surpasses all SFT and RLHF baselines and displays a positive scaling trend, indicating its efficacy in enhancing model performance as model complexity increases.

Table 3: Ablation study of natural language feedback on summarization task based on human evaluation.

Average win rate (%)		
	Tie	CoH
RLHF	30.8	24.0
RLHF	32.1	26.5
CoH w/o LF	10.6	45.3
CoH w/o LF	74.3	42.4
CoH w/o LF	10.6	15.1

Chain of Hindsight

Эксперименты и сравнение с другими методами

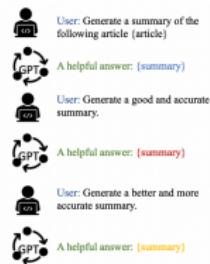
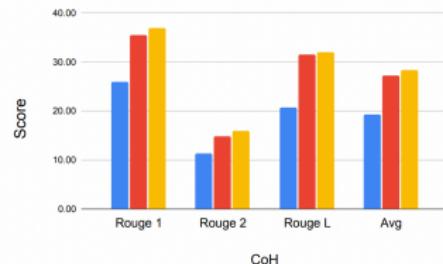
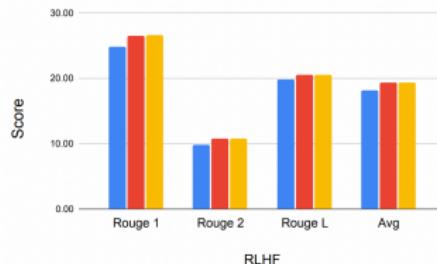


Figure 9: **Controllable generation.** (left): RLHF cannot follow instructions to generate improved summary. (middle): After finetuning on CoH, the model follows instructions to achieve controllable generations. (right): First instruction is standard, while second and third instructions ask for improved summaries.

Chain of Hindsight

Плюсы и минусы

- Плюсы
 - + Нет явной оценки Reward модели

Chain of Hindsight

Плюсы и минусы

- Плюсы

- + Нет явной оценки Reward модели
- + Нет RL в явном виде

Chain of Hindsight

Плюсы и минусы

- Плюсы

- + Нет явной оценки Reward модели
- + Нет RL в явном виде
- + Всего 1 модель, которую нужно обучать в процессе alignment'a

Chain of Hindsight

Плюсы и минусы

- Плюсы

- + Нет явной оценки Reward модели
- + Нет RL в явном виде
- + Всего 1 модель, которую нужно обучать в процессе alignment'a
- + Мало гиперпараметров

Chain of Hindsight

Плюсы и минусы

- Плюсы

- + Нет явной оценки Reward модели
- + Нет RL в явном виде
- + Всего 1 модель, которую нужно обучать в процессе alignment'a
- + Мало гиперпараметров
- + Теоретически можно использовать фидбек собранный сразу в текстовом виде. Это добавляет информации к оценке ответа

Chain of Hindsight

Плюсы и минусы

- Плюсы

- + Нет явной оценки Reward модели
- + Нет RL в явном виде
- + Всего 1 модель, которую нужно обучать в процессе alignment'a
- + Мало гиперпараметров
- + Теоретически можно использовать фидбек собранный сразу в текстовом виде. Это добавляет информации к оценке ответа
- + Очень просто в реализации

Chain of Hindsight

Плюсы и минусы

• Плюсы

- + Нет явной оценки Reward модели
- + Нет RL в явном виде
- + Всего 1 модель, которую нужно обучать в процессе alignment'a
- + Мало гиперпараметров
- + Теоретически можно использовать фидбек собранный сразу в текстовом виде. Это добавляет информации к оценке ответа
- + Очень просто в реализации
- + Более устойчивое обучение по сравнению с RL

Chain of Hindsight

Плюсы и минусы

- Плюсы

- + Нет явной оценки Reward модели
- + Нет RL в явном виде
- + Всего 1 модель, которую нужно обучать в процессе alignment'a
- + Мало гиперпараметров
- + Теоретически можно использовать фидбек собранный сразу в текстовом виде. Это добавляет информации к оценке ответа
- + Очень просто в реализации
- + Более устойчивое обучение по сравнению с RL
- + Лучшие результаты при контролируемой генерации

Chain of Hindsight

Плюсы и минусы

• Плюсы

- + Нет явной оценки Reward модели
- + Нет RL в явном виде
- + Всего 1 модель, которую нужно обучать в процессе alignment'a
- + Мало гиперпараметров
- + Теоретически можно использовать фидбек собранный сразу в текстовом виде. Это добавляет информации к оценке ответа
- + Очень просто в реализации
- + Более устойчивое обучение по сравнению с RL
- + Лучшие результаты при контролируемой генерации

• Минусы

- Из-за того, что добавляем фидбэк и склеиваем сразу несколько ответов, могут появляться очень длинные последовательности

Chain of Hindsight

Плюсы и минусы

• Плюсы

- + Нет явной оценки Reward модели
- + Нет RL в явном виде
- + Всего 1 модель, которую нужно обучать в процессе alignment'a
- + Мало гиперпараметров
- + Теоретически можно использовать фидбек собранный сразу в текстовом виде. Это добавляет информации к оценке ответа
- + Очень просто в реализации
- + Более устойчивое обучение по сравнению с RL
- + Лучшие результаты при контролируемой генерации

• Минусы

- Из-за того, что добавляем фидбэк и склеиваем сразу несколько ответов, могут появляться очень длинные последовательности
- Из-за того, что мы выкинули Reward модель, мы становимся ограничены собранным фидбэк-датасетом

Вопросы?

Выводы

Сегодня мы:

- Вспомнили как работает RLHF, какие у этого подхода есть минусы

Выводы

Сегодня мы:

- Вспомнили как работает RLHF, какие у этого подхода есть минусы
- DPO:
 - Получили выражение reward функции через оптимальную политику и вывели DPO-loss, оттолкнувшись от задачи RLHF
 - Разобрали схему обучения с использованием DPO
 - Посмотрели на бенчмарки для DPO

Выводы

Сегодня мы:

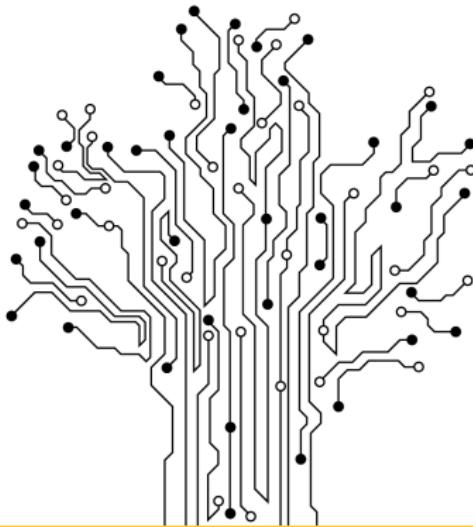
- Вспомнили как работает RLHF, какие у этого подхода есть минусы
- DPO:
 - Получили выражение reward функции через оптимальную политику и вывели DPO-loss, оттолкнувшись от задачи RLHF
 - Разобрали схему обучения с использованием DPO
 - Посмотрели на бенчмарки для DPO
- Chain of Hindsight:
 - Аналогично DPO

Выводы

Сегодня мы:

- Вспомнили как работает RLHF, какие у этого подхода есть минусы
- DPO:
 - Получили выражение reward функции через оптимальную политику и вывели DPO-loss, оттолкнувшись от задачи RLHF
 - Разобрали схему обучения с использованием DPO
 - Посмотрели на бенчмарки для DPO
- Chain of Hindsight:
 - Аналогично DPO
- Поняли, что DPO и Chain of Hindsight – хорошие методы, которые во многих случаях превосходят RLHF, но они имеют свои ограничения и иногда RLHF может оказаться лучше для конкретной задачи [3].

**Спасибо за внимание!
Вопросы?**



References I

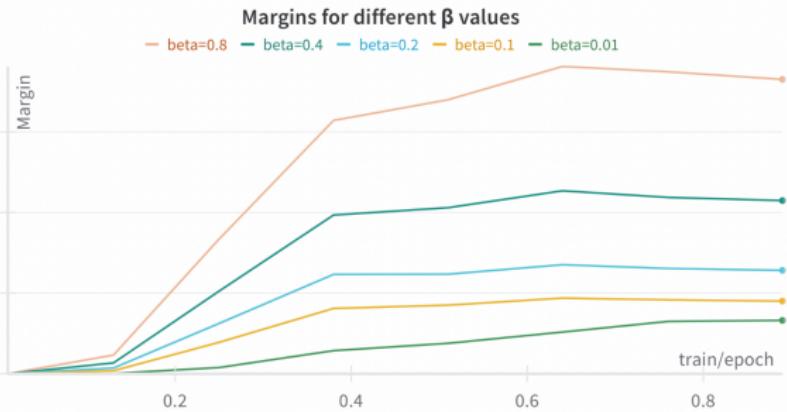
- [1] Winnie Xu 2 Kawin Ethayarajh. "Hao Liu, Carmelo Sferrazza, Pieter Abbeel." In: *arXiv:2302.02676v8* (2023).
- [2] Eric Mitchell Rafael Rafailov Archit Sharma. "Direct Preference Optimization: Your Language Model is Secretly a Reward Model." In: *arXiv:2305.18290, version 2* (2023).
- [3] Jiaqi Zeng Zhilin Wang Yi Dong. "HELPSTEER: Multi-attribute Helpfulness Dataset for STEERLM." In: *arXiv:2311.09528v1* (2023).

Дополнительные слайды

Дополнительные графики

Разница в наградах между хорошим и плохим ответом в зависимости от разных β с течением времени тренировки

$$\text{Margin} = \left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right)$$



Формулировки теорем и утверждений из статьи

Definition 1. We say that two reward functions $r(x, y)$ and $r'(x, y)$ are equivalent iff $r(x, y) - r'(x, y) = f(x)$ for some function f .

It is easy to see that this is indeed an equivalence relation, which partitions the set of reward functions into classes. We can state the following two lemmas:

Lemma 1. Under the Plackett-Luce, and in particular the Bradley-Terry, preference framework, two reward functions from the same class induce the same preference distribution.

Lemma 2. Two reward functions from the same equivalence class induce the same optimal policy under the constrained RL problem.

Theorem 1. Under mild assumptions, all reward classes consistent with the Plackett-Luce (and Bradley-Terry in particular) models can be represented with the reparameterization $r(x, y) = \beta \log \frac{\pi(y|x)}{\pi_{ref}(y|x)}$ for some model $\pi(y|x)$ and a given reference model $\pi_{ref}(y|x)$.

Theorem 1 Restated. Assume, we have a reference model, such that $\pi_{ref}(y|x) > 0$ for all pairs of prompts x and answers y and a parameter $\beta > 0$. All reward equivalence classes, as defined in Section 5 can be represented with the reparameterization $r(x, y) = \beta \log \frac{\pi(y|x)}{\pi_{ref}(y|x)}$ for some model $\pi(y|x)$.

Proposition 1. Assume, we have a reference model, such that $\pi_{ref}(y|x) > 0$ for all pairs of prompts x and answers y and a parameter $\beta > 0$. Then every equivalence class of reward functions, as defined in Section 5, has a unique reward function $r(x, y)$, which can be reparameterized as $r(x, y) = \beta \log \frac{\pi(y|x)}{\pi_{ref}(y|x)}$ for some model $\pi(y|x)$.