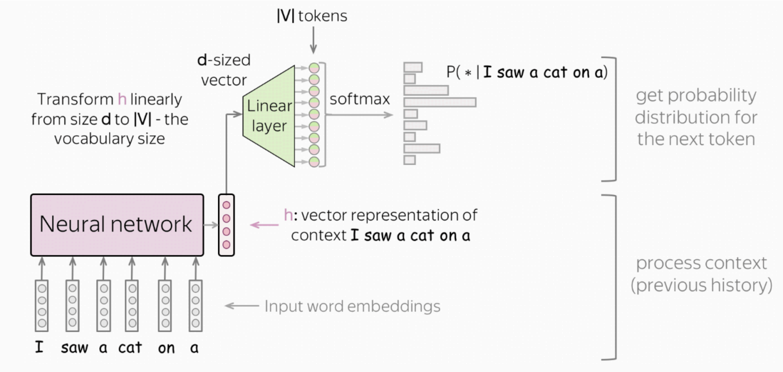
Algorithmic and Theoretical Foundations of RL

Introduction of RL for LLMs

Ke Wei
School of Data Science
Fudan University

With help from Jiacai Liu
Some illustration figures borrowed online

Auto-Regression: Next Token Prediction



- **Auto-regressive** : $y_t \sim \pi_\theta(\cdot|y_{<t})$ is a function of its past tokens $y_{<t}$.
- **Neural softmax policy** : $\pi_\theta\left(\cdot|y_{<t}\right) = \operatorname{soft\,max}\left(\frac{1}{\tau} f_\theta\left(y_{<t}\right)\right)$, where $f_\theta\left(y_{<t}\right) \in \mathbb{R}^{|\mathcal{V}|}$ is the logits vector, τ is the sampling temperature.
- **Network Architecture** : f_θ is a multi-layer decoder-only transformers.

Token and Vocabulary

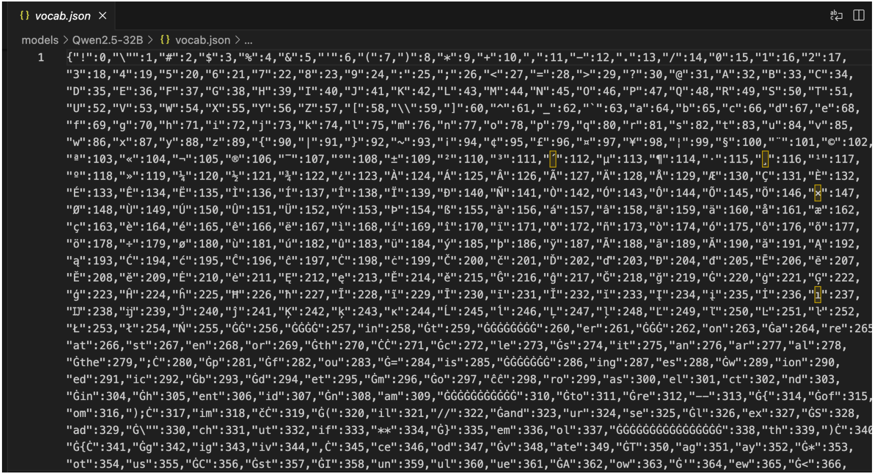
Tokenization is a fundamental preprocessing step in NLP that breaks down a piece of text, such as a sentence or a paragraph, into smaller, more manageable units called **tokens**. The set of all possible tokens is the **vocabulary**.

```python
from transformers import AutoTokenizer

model_path = "/mnt/hdfs/jiacai.liu/models/Qwen2.5-32B/"
tokenizer = AutoTokenizer.from_pretrained(model_path)
text = 'We love Fudan University'
tokens = tokenizer.tokenize(text)
input_ids = tokenizer.encode(text)
print("text",text)
print("tokens",tokens)
print("input_ids",input_ids)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
<Trial 61560222 worker_0> jiacai.liu $ /bin/python /mnt/hdfs/jiacai.liu/test.py
text We love Fudan University
tokens ['We', 'Ġlove', 'ĠF', 'ud', 'an', 'ĠUniversity']
input_ids [1654, 2948, 434, 661, 276, 3822]
```

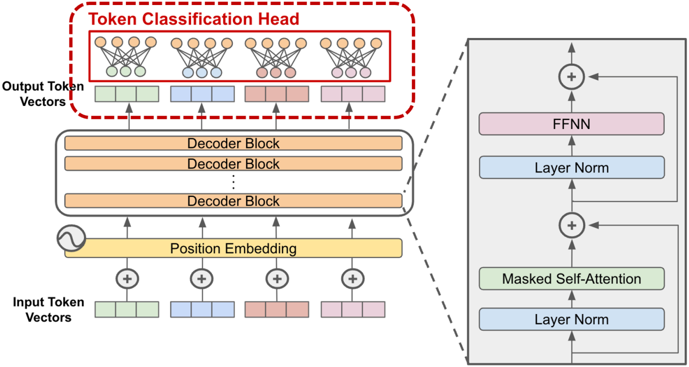
Token and Vocabulary



Vocabulary of Qwen25-32B which consists of 151642 tokens

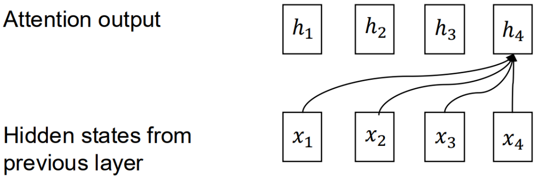
Network Architecture : Decoder-Only Transformer

Illustration of outputs of four timesteps, but not four neural networks since different timesteps use the same neural network



Decoder-only tansformer = position embedding + self-attention + residual connection + layer norm + multi-layer FFNN

Attention Mechanism



$$h_t = \sum_{i \leq t} s_{ti} x_i$$

Intuitively, $s_{ti} = \frac{\exp(\langle Qx_t, Kx_i \rangle)}{\sum_{l \leq t} \langle Qx_t, Kx_l \rangle}$ is the attention score that computes the relevance of the i-th input to the current output.

Attention Mechanism (Cont'd)

$$\text{Input: } X = \begin{bmatrix} x_1 \\ \vdots \\ x_t \end{bmatrix} \in \mathbb{R}^{t \times d_1}$$

$$\text{Query Matrix: } Q = XW_Q = \begin{bmatrix} q_1 \\ \vdots \\ q_t \end{bmatrix} \in \mathbb{R}^{t \times d_2}$$

$$\text{Key Matrix: } K = XW_K = \begin{bmatrix} k_1 \\ \vdots \\ k_t \end{bmatrix} \in \mathbb{R}^{t \times d_2}$$

$$\text{Value Matrix: } V = XW_V = \begin{bmatrix} v_1 \\ \vdots \\ v_t \end{bmatrix} \in \mathbb{R}^{d_1 \times d_2}$$

Attention Mechanism (Cont'd)

$$\text{Score Matrix: } S = QK^T = \begin{bmatrix} \langle q_1, k_1 \rangle & \cdots & \langle q_1, k_t \rangle \\ \vdots & \ddots & \vdots \\ \langle q_t, k_1 \rangle & \cdots & \langle q_t, k_t \rangle \end{bmatrix}$$

$$\text{Output: } H = \begin{bmatrix} h_1 \\ \vdots \\ h_t \end{bmatrix} = \text{softmax}\left(\frac{S}{\sqrt{d_2}} \odot \underbrace{M}_{\text{mask}} \right) V$$

$$\Updownarrow$$

$$\forall t : h_t = \sum_{i \le t} \frac{\exp(\langle q_t, k_i \rangle / \sqrt{d_2})}{\sum_{\ell \le t} \exp(\langle q_t, k_l \rangle / \sqrt{d_2})} v_i$$

- Number of parameters in W_Q, W_K and W_V is independent of timesteps!

Token-Level MDP in LLMs

Text generation process of LLMs follows a token-level MDP. Given a prompt x, let $y = (a_0, a_1, \cdots, a_{T-1})$ be the response consists of a sequence of tokens.

- **initial state** $s_0 = x$, the sampled prompt;
- **action**: $a_t = y_t$, token sampled from **vocabulary** (action space);
- **state** is the context prefix consists of previous tokens, i.e.,

$$s_t = (x, y_{<t}) = \text{Concat}(s_0, a_0, ..., a_{t-1});$$

- **policy network** $\pi_\theta(\cdot|s_t) = \pi_\theta(\cdot|x, y_{<t})$, probability of outputting next token;
- **transition kernel** is deterministic, i.e., $s_{t+1} = (s_t, a_t) = (x, y_{\leq t})$.
- **reward function** r depends on task, but generally is outcome-based, i.e.,

$$r\left(s_t, a_t\right) = \begin{cases} r\left(x, y\right) & t = T - 1 \\ 0 & t = 0 \end{cases}.$$

Token-Level MDP in LLMs

S0 详细信息

前缀(state)： <|im_start|>system Please reason step by step, and put your final answer within \boxed{}.<|im_end|> <|im_start|>user Let $a,$ $b,$ and c be distinct real numbers. Simplify the expression \[\frac{(x + a)^3}{(a - b)(a - c)} + \frac{(x + b)^3}{(b - a)(b - c)} + \frac{(x + c)^3}{(c - a)(c - b)}.\]<|im_end|> <|im_start|>assistant To

Action Space (Top 50 Tokens):

token	simplify	solve	simpl	find	Simpl	simplified	tackle
概率	0.998958	0.001024	0.000008	0.000004	0.000004	0.000001	0.000000

Example of text generation process at timestep 0

Token-Level MDP in LLMs

S1 详细信息

前缀(state)： <|im_start|>system Please reason step by step, and put your final answer within \boxed{}.<|im_end|> <|im_start|>user Let $a,$ $b,$ and c be distinct real numbers. Simplify the expression \[\frac{(x + a)^3}{(a - b)(a - c)} + \frac{(x + b)^3}{(b - a)(b - c)} + \frac{(x + c)^3}{(c - a)(c - b)}.\]<|im_end|> <|im_start|>assistant To simplify

Action Space (Top 50 Tokens):

token	the	this	\	and	each	expressions	the
概率	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000

Example of text generation process (con't) at timestep 1

Token-Level MDP in LLMs

S457 详细信息

前缀(state)： <|im_start|>system Please reason step by step, and put your final answer within \boxed{}.<|im_end|> <|im_start|>user Let $a,$ $b,$ and c be distinct real numbers. Simplify the expression \[\frac{(x + a)^3}{(a - b)(a - c)} + \frac{(x + b)^3}{(b - a)(b - c)} + \frac{(x + c)^3}{(c - a)(c - b)}.\]<|im_end|> <|im_start|>assistant To simplify the expression \(\frac{(x + a)^3}{(a - b)(a - c)} + \frac{(x + b)^3}{(b - a)(b - c)} + \frac{(x + c)^3}{(c - a)(c - b)}\), we will proceed step by step. First, let's consider the general structure of the expression. The expression is a sum of three terms, each of which is a fraction with a cubic polynomial in the numerator and a quadratic polynomial in the denominator. The denominators \((a - b)(a - c)\), \((b - a)(b - c)\), and \((c - a)(c - b)\) are symmetric and suggest that we might be able to use polynomial interpolation. The expression can be interpreted as a Lagrange interpolating polynomial of degree 2. The polynomial \(P(x)\) that passes through the points \((a, (x+a)^3)\), \((b, (x+b)^3)\), and \((c, (x+c)^3)\) can be written as: \[P(x) = \frac{(x - b)(x - c)}{(a - b)(a - c)}(x + a)^3 + \frac{(x - a)(x - c)}{(b - a)(b - c)}(x + b)^3 + \frac{(x - a)(x - b)}{(c - a)(c - b)}(x + c)^3.\] However, since \(P(x)\) is a polynomial of degree 2, and the expression we are given is a sum of terms that are designed to cancel out all the higher-degree terms, the polynomial \(P(x)\) must be of the form \(P(x) = (x + a) + (x + b) + (x + c)\), which simplifies to \(P(x) = 3x + (a + b + c)\). Thus, the simplified form of the given expression is: \[3x + (a + b + c).\] Therefore, the final answer is: \[\boxed{3x + a + b + c}.\]

token	<|im_end|>	<|im_start|>		bụ	eated	iska	istil
概率	0.999999	0.000000	0.000000	0.000000	0.000000	0.000000	0.000(

Example of text generation process (con't) at the last timestep

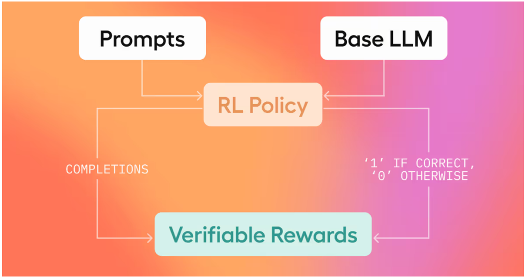
Reward Models in LLMs

Reward Models (RMs) are crucial in RLHF to align LLMs with complex human values, preferences, and instructions, by converting subjective feedback into a scalable optimization signal. There are typically three types of reward signals:

- **Rule-Based RMs**: Reward designed explicitly from a collection of pre-defined rules and heuristics, often used to enforce safety or structural constraints.
- **Discriminative RMs**: Model trained to compare and rank multiple LLM outputs based on human preference data, assigning a score or probability to reflect which response is better.
- **Generative RMs**: An approach that leverages the LLM's own generation capabilities, often by training it to output a preference decision or a "Chain-of-Thought" rationale for the reward, using next-token prediction.

Rule-Based Reward Models

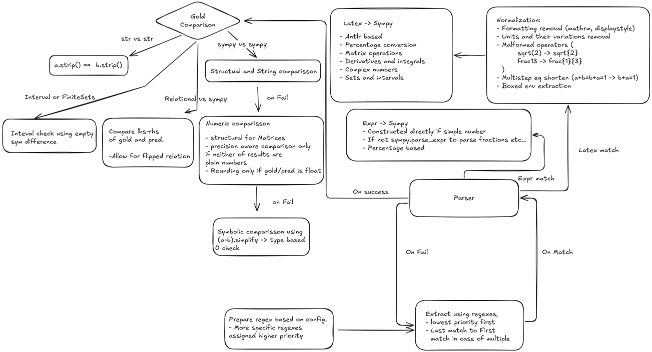
Rule-based RMs uses a pre-defined collection of explicit, symbolic rules to assign a **(binary)** reward to a model's output and are often used in verifiable tasks, such as mathematical reasoning and coding.



Rule-based RMs are commonly used verifiable tasks

Math-Verify

Math-Verify is a robust mathematical expression evaluation system designed for assessing LLMs outputs in mathematical tasks.



Architecture of math-verify

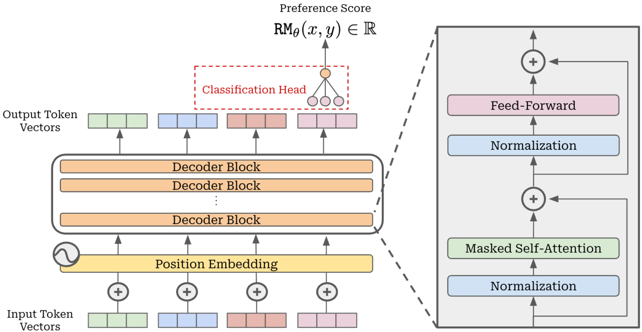
Math-Verify

```python
from math_verify import parse, verify

# Parse the gold and answer
# If you know that gold will only contain latex or expr (no latex env), use
# parse(gold, extraction_config=[LatexExtractionConfig()]) or parse(gold, extraction_config=[ExprExtrac

gold = parse("${1,3} \\cup {2,4}$")
answer = parse("${1,2,3,4}$")

# Order here is important!
verify(gold, answer)
# >>> True
```

Code example of Math-Verify

Discriminative Reward Models

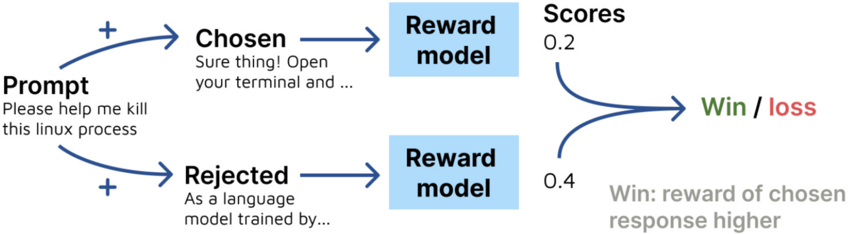
Discriminative RM is standard and very common in RLHF: a model trained to discriminate between a preferred response and a dispreferred response from a set of human-labeled comparisons.



Structure of discriminative RMs: Input is (x, y), concatenation of x and y

RM Training on Preference Dataset

Discriminative RM is trained on a manual preference dataset by maximizing the log-likelihood of preference probability under the Bradley–Terry (BT) model.



Discriminative RM should assign higher score to chosen response

Bradley–Terry Model

Reward Model (RM) with parameter ϕ

Probability that chosen response is preferred over rejected reference

$$P(y_c > y_r) = \frac{\exp(r_\phi(x, y_c))}{\exp(r_\phi(x, y_c)) + \exp(r_\phi(x, y_r))}$$

RM score for chosen response

RM score for rejected response

It is easy to see that

$$\log \frac{P(y_c > y_r)}{1 - P(y_c > y_r)} = r_\phi(x, y_c) - r_\phi(x, y_r).$$

Train reward model by minimizing the negative log-likelihood,

$$\min_\phi \mathbb{E}_{(x, y_c, y_r) \sim \mathcal{D}}[-\log \sigma(r_\phi(x, y_c) - r_\phi(x, y_r))],$$

where $\sigma(\cdot)$ denotes the sigmoid function.

RLHF: Reinforcement Learning from Human Feedback

The goal in RLHF to solve following problem:

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

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

where π_{ref} is the reference policy (often the base policy before RL training) used to control the policy shift to avoid knowledge forgetting, $y = (a_0, ..., a_{T-1})$, and

$$r_\phi(x,y) = \sum_{t=0}^{T-1} r_\phi(\underbrace{(x, y_{<t})}_{s_t}, \underbrace{y_t}_{a_t}),$$

$$\pi_\theta(y|x) = \prod_{t=0}^{T-1} \pi_\theta(y_t|x, y_{<t}).$$

Policy Gradient

By direct calculation, one has

$$\nabla V^{\pi_\theta}(\mathcal{D}) = \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left[\left(r_\phi(x,y) - \beta\log\frac{\pi_\theta(y|x)}{\pi_{\mathrm{ref}}(y|x)} - \beta\right)\nabla\log\pi_\theta(y|x)\right]$$

$$= \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left[\left(r_\phi(x,y) - \beta\log\frac{\pi_\theta(y|x)}{\pi_{\mathrm{ref}}(y|x)}\right)\nabla\log\pi_\theta(y|x)\right]$$

$$= \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left[\sum_{t=0}^{T-1}\nabla\log\pi_\theta(y_t|x,y_{<t})\left(r_\phi(x,y) - \beta\sum_{t'=0}^{T-1}\log\frac{\pi_\theta(y_{t'}|x,y_{<t'})}{\pi_{\mathrm{ref}}(y_{t'}|x,y_{<t'})}\right)\right]$$

For more details, see https://zhuanlan.zhihu.com/p/1891822525274637445?share_code=1onxLXvlAlpgq&utm_psn=1984360514466379479 by Jiacai Liu.

Note that

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

Following the idea in TRPO/PPO, the first term can be well approximated by

$$\mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi_{\theta_k}(\cdot|x)}\left[\sum_{t=0}^{T-1}\frac{\pi_\theta(y_t|(x,y_{<t}))}{\pi_{\theta_k}(y_t|(x,y_{<t}))}A^{\pi_{\theta_k}}((x,y_{<t}),y_t)\right]$$

provided π_θ is sufficiently close to π_{θ_k}. In order to impose this condition, we can still consider the following clipped objective,

$$\mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi_{\theta_k}(\cdot|x)}\left[\sum_{t=0}^{T-1}\min\left(r_t(\theta)A_t^k, \mathrm{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)A_t^k\right)\right]$$

where

$$r_t(\theta) = \frac{\pi_\theta(y_t|(x,y_{<t}))}{\pi_{\theta_k}(y_t|(x,y_{<t}))} \text{ and } A_t^k = A^{\pi_{\theta_k}}((x,y_{<t}),y_t).$$

PPO and GRPO

Together with the KL part, we have the PPO objective for RLHF,

$$\mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi_{\theta_k}(\cdot|x)}\left[\sum_{t=0}^{T-1}\min\left(r_t(\theta)A_t^k, \mathrm{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)A_t^k\right) - \beta\mathrm{KL}\left(\pi_\theta\left(\cdot|x\right)\|\pi_{\mathrm{ref}}\left(\cdot|x\right)\right)\right].$$

- Vanilla PPO: Use GAE(λ) to estimate A_t^k. Note that the token level reward is often 0 except the last one. Even for the reward model based on the BT model, it is difficult to explain the reward for intermediate tokens.
- GRPO: Sample n responses $y_1, \cdots, y_n$ for each prompt x, and estimate A_t^k by

$$A_t^k = A^{\pi_{\theta_k}}((x, y_{<t}), y_t) = \frac{r_\phi(x, y_k) - \mathrm{mean}(r_\phi(x, y_n), \cdots, r_\phi(x, y_n))}{\mathrm{std}(r_\phi(x, y_n), \cdots, r_\phi(x, y_n))}$$

Assume there only exists reward in last token. GRPO estimates A_t^k via

$$\begin{aligned} A_t^k &= Q^{\pi_{\theta_k}}((x, y_{<t}), y_t) - V^{\pi_{\theta_k}}((x, y_{<t})) \\ &\approx Q^{\pi_{\theta_k}}((x, y_{<t}), y_t) - V^{\pi_{\theta_k}}(x) \\ &\xrightarrow[\text{estimator}]{\text{unbiased}} r(x, y) - \mathrm{mean}(r_\phi(x, y_n), \cdots, r_\phi(x, y_n)). \end{aligned}$$

Direct Policy Optimization

Recall the RLHF objective,

$$\max_{\theta} V^{\pi_\theta}(\mathcal{D}) = \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left[r_\phi(x,y) + \beta\log\pi_{\text{ref}}(y|x) - \beta\log\pi_\theta(y|x)\right].$$

From this, it can be seen that the optimal policy π^*_β satisfies

$$\pi^*_\beta(y|x) = \frac{\exp(r_\phi(x,y)/\beta + \log\pi_{\text{ref}}(y|x))}{Z_\beta(x)}.$$

It follows that

$$r_\phi(x,y) = \beta\log\frac{\pi^*_\beta(y|x)}{\pi_{\text{ref}}(y|x)} + \beta\log Z_\beta(x).$$

Direct Policy Optimization

Under the Bradley-Terry Model,

$$\begin{aligned} P(y_c > y_r|x) &= \sigma(r_\phi(x, y_c) - r_\phi(x, y_r)) \\ &= \sigma\left(\beta \log \frac{\pi^*_\beta(y_c|x)}{\pi_{\text{ref}}(y_c|x)} - \beta \log \frac{\pi^*_\beta(y_r|x)}{\pi_{\text{ref}}(y_r|x)}\right). \end{aligned}$$

Therefore, it is natural to optimzie the following DPO objective,

$$\min_\theta \mathbb{E}_{(x, y_c, y_r)\sim\mathcal{D}}\left[-\log\sigma\left(\beta \log \frac{\pi_\theta(y_c|x)}{\pi_{\text{ref}}(y_c|x)} - \beta \log \frac{\pi_\theta(y_r|x)}{\pi_{\text{ref}}(y_r|x)}\right)\right]$$

- DPO is derived under Bradley-Terry model, requires high quality preference data, friendly for off-line data but seems lacking true exploration ability.

Questions?