LLM ALIGNMENT

Team 6
Fengyu Gao, Shunqiang Feng, Wei Shen, Zihan Zhao

A Comprehensive Survey of LLM Alignment Techniques: RLHF, RLAIF, PPO, DPO and More

Additional references:

[1] https://anukriti-ranjan.medium.com/preference-tuning-llms-ppo-dpo-grpo-a-simple-guide-135765c87090

[2] https://web.stanford.edu/class/cs224n/slides/cs224n-spr2024-lecture10-prompting-rlhf.pdf

Fengyu Gao (wan6jj)

Aligning Language Models

LMs like GPT-3 are misaligned: they maximize the likelihood of large untrusted datasets.

This leads to:

- Not following the user's instruction
- Making up facts
- Generating harmful/toxic content
- ……

Explain the moon landing to a 6 year old in a few sentences.

GPT-3

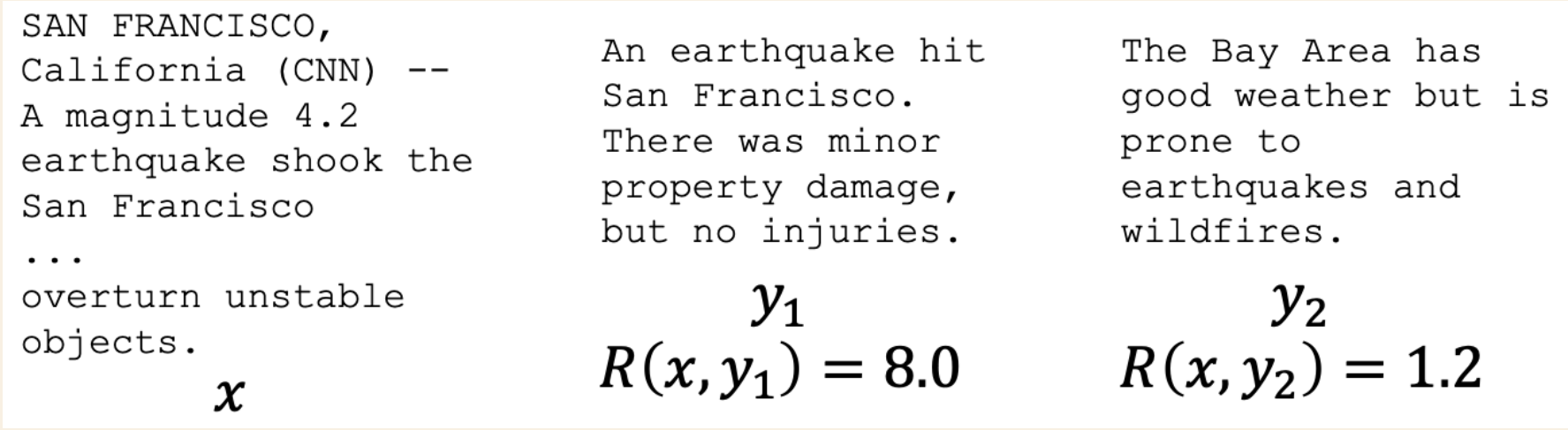
```
Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.
```

Language models are not *aligned* with user intent.

Aligning LMs with Human Feedback

Suppose we are training a LM for a summarization task.

For a given instruction x and a generated summary y, we assume we can obtain a human reward of that summary: $R(x, y)$ – where higher values indicate better quality.



We want to maximize the expected reward based on this feedback.

A (very!) brief introduction to RL

Reinforcement Learning = Learning by Doing and Getting Feedback

- An agent (LLM) interacts with an environment and learns by trial and error.
- Large Rewards (✅ Correct answer!) encourage desirable outputs.
- Small Rewards (❌ Incorrect response!) discourage undesirable outputs..
- RL algorithms (e.g., PPO, DPO, GRPO) train LLMs to maximize this reward.

How do we get the rewards?

Q1: Human-in-the-loop is expensive!

Solution: Instead of asking humans directly, we train a separate reward model to learn human preferences.

Q2: Human judgments are noisy and miscalibrated!

Solution: Use pairwise comparisons instead of direct ratings.

An earthquake hit San Francisco. There was minor property damage, but no injuries.	>	A 4.2 magnitude earthquake hit San Francisco, resulting in massive damage.

$$L_{\mathrm{RM}}(r_\phi) = -\frac{1}{C_K^2}\mathbb{E}_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\phi(x,y_w) - r_\phi(x,y_l)\right)\right)\right]$$

y_w: winning sample

y_l: losing sample

y_w should score higher than y_l

RLHF: Optimizing the learned reward model

We have the following:

- A pretrained (possibly instruction-finetuned) LM $\pi_{ref}(y|x)$
- A reward model $r_\phi(x, y)$ that produces scalar rewards for LM outputs, trained on a dataset of human comparisons

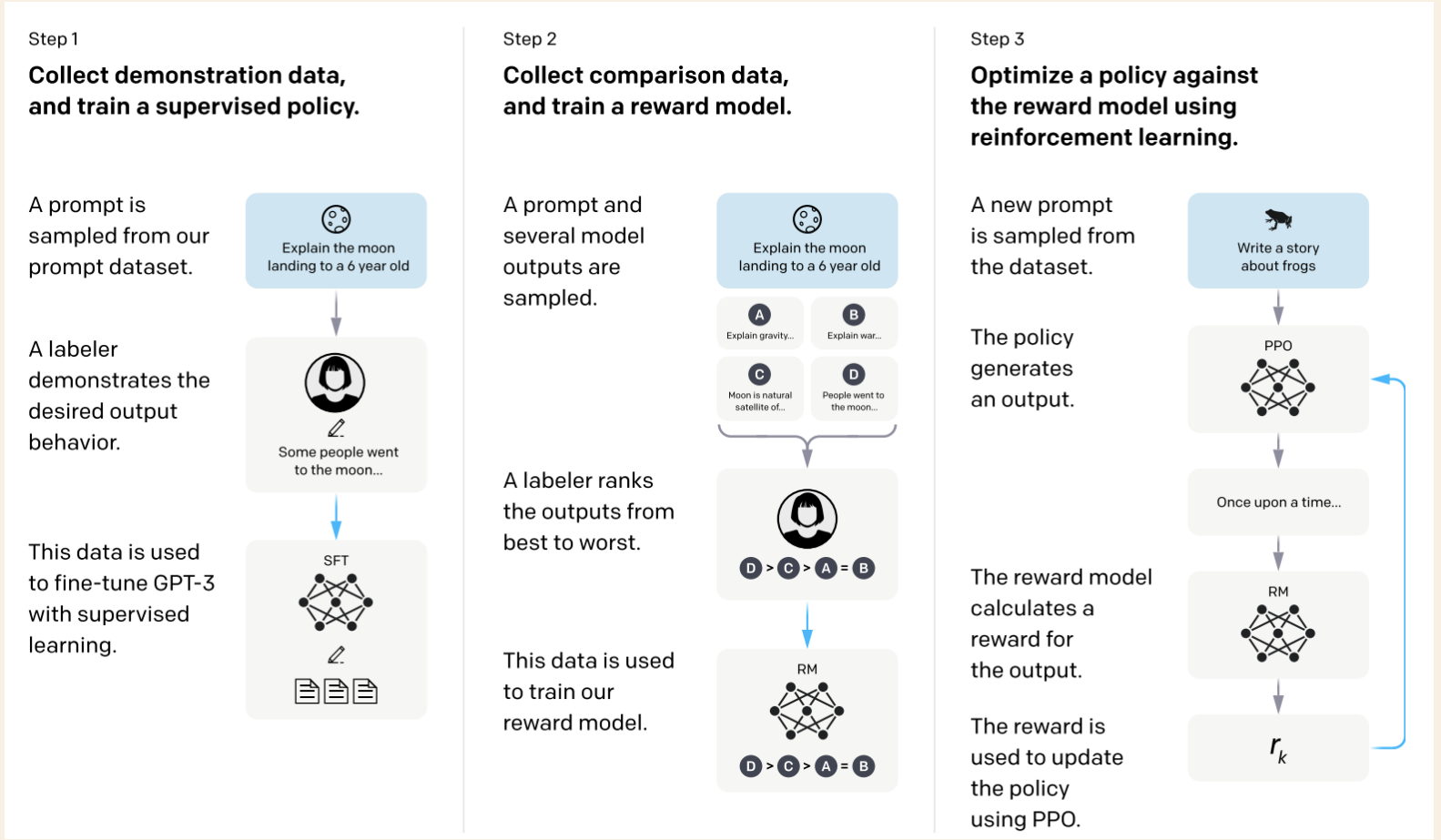
Now to do RLHF:

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

Maximizing rewards

Minimizing divergence between current policy and reference policy

High-Level Overview: RLHF Pipeline



supervised fine-tuning/instruction tuning -> reward modeling -> policy optimization

Can we simplify RLHF? Towards DPO

Direct Preference Optimization (DPO): directly optimizes policy based on human preference data using a clever loss function.

Recall our objective in RLHF:

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

There is a closed form solution to this:

$$\pi_\theta(y|x) = \frac{1}{Z(x)}\pi_{\mathrm{ref}}(y|x)e^{\left(\frac{1}{\beta}r_\theta(x,y)\right)}$$

Rearrange the terms:

$$r_\theta(x,y) = \beta\log\left(\frac{\pi_\theta(y|x)}{\pi_{\mathrm{ref}}(y|x)}\right) + \beta\log Z(x)$$

Reward model can be written in terms of policy!

Can we simplify RLHF? Towards DPO

Direct Preference Optimization (DPO): directly optimizes policy based on human preference data using a clever loss function.

Recall, how we fit the reward model in RLHF:

$$L_{\mathrm{RM}}(r_\phi) = -\frac{1}{C_K^2}\mathbb{E}_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\phi(x,y_w) - r_\phi(x,y_l)\right)\right)\right]$$

Notice that we only need the difference between the rewards. Simplify for rewards:

$$r_\theta(x,y_w) - r_\theta(x,y_l) = \beta\left[\log\left(\frac{\pi_\theta(y_w|x)}{\pi_{\mathrm{ref}}(y_w|x)}\right) - \log\left(\frac{\pi_\theta(y_l|x)}{\pi_{\mathrm{ref}}(y_l|x)}\right)\right]$$

The final DPO loss function is:

$$-\mathbb{E}_{(x,y_w,y_l)\sim D}\log\left\{\sigma\left[\beta\log\left(\frac{\pi_\theta(y_w|x)}{\pi_{\mathrm{ref}}(y_w|x)}\right) - \beta\log\left(\frac{\pi_\theta(y_l|x)}{\pi_{\mathrm{ref}}(y_l|x)}\right)\right]\right\}$$

We have a classification loss function that connects **preference data** to **LM parameters** directly!

Summary (RLHF and DPO)

- Our goal is to optimize for Human Preferences
  - Instead of humans writing the answers or giving uncalibrated scores, we get humans to rank different LM generated answers.
- RLHF
  - Step 1: Supervise fine-tuning on a labeled dataset
  - Step 2: Train an explicit reward model on comparison data to predict a score for a completion
  - Step 3: Optimize the LM to maximize the predicted score (under KL-constraint)
  - Very effective when tuned well, computationally expensive
- DPO
  - Optimize LM parameters directly on preference data by solving a binary classification problem
  - Simple and effective, similar properties to RLHF

Research directions of LLM alignment

- Reward model
- Feedback
- RL policy
- Optimization

Reward model

- **Explicit Reward Model vs. Implicit Reward Model**
  - e.g., RLHF vs. DPO
- **Pointwise Reward Model vs. Preferencewise Model**
  - $R(x, y)$ vs. prob. that the desired response is preferred over the undesired one
- **Response-Level Reward vs. Token-Level Reward**
  - Assign a single score to the entire response vs. provide feedback at each token
- **Negative Preference Optimization**
  - Use only prompts and undesired responses from RLHF datasets, generating desired responses with LLMs instead of relying on human-labeled preferred responses

Feedback

- **Preference Feedback vs. Binary Feedback**
  - Rank responses vs. simple positive or negative signal without ranking
- **Pairwise Feedback vs. Listwise Feedback**
  - Compare two responses vs. rank multiple responses together
- **Human Feedback vs. AI Feedback**
  - Real user preferences vs. LLM-generated evaluations

RL

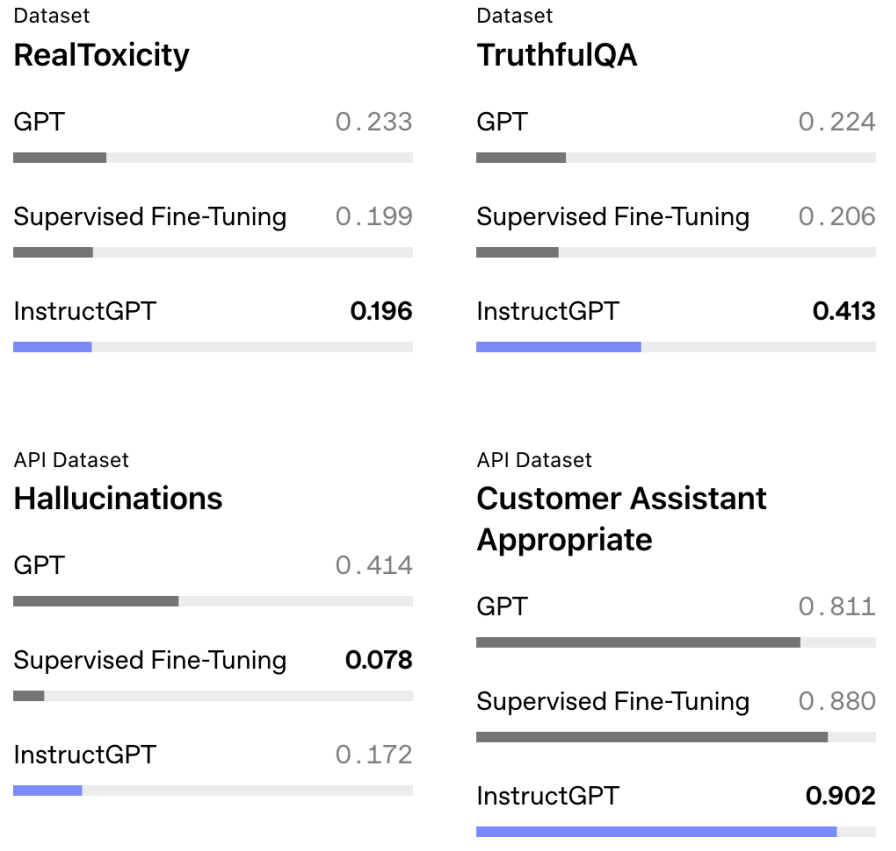
- **Reference-Based RL vs. Reference-Free RL**
  - Minimize divergence from a reference policy vs. remove reference policy (e.g. SimPO)
- **Length-Control RL**
  - Standard RL ignores response length. Length-control RL adjusts rewards to prevent verbosity bias in LLM-generated responses. E.g., R-DPO and SimPO.
- **Different Divergences in RL**
  - KL divergence, f-divergence, ……
- **On-policy or Off-policy Learning**
  - Generate responses using the latest policy vs. reuse past responses

Optimization

- **Iterative/Online Preference Optimization vs. Non-Iterative/Offline Preference Optimization**
  - Continuously update alignment with new data vs. align models using a fixed dataset
- **Separating SFT and Alignment vs. Merging SFT and Alignment**
  - Newer approaches integrate SFT and alignment into a single process, e.g., ORPO, PAFT.

InstructGPT: scaling up RLHF to 30k tasks!

Evaluate InstructGPT from three perspectives: Helpful, Honest, and Harms.



InstructGPT: scaling up RLHF to 30k tasks!

Prompt

Explain the moon landing to a 6 year old in a few sentences.

Completion

GPT-3

```
Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.
```

InstructGPT

```
People went to the moon, and they took pictures of what they saw, and sent them back to the earth so we could all see them.
```

Wei Shen (zyy5hb)

OpenRLHF: An Easy-to-use, Scalable and High-performance RLHF Framework

PPO

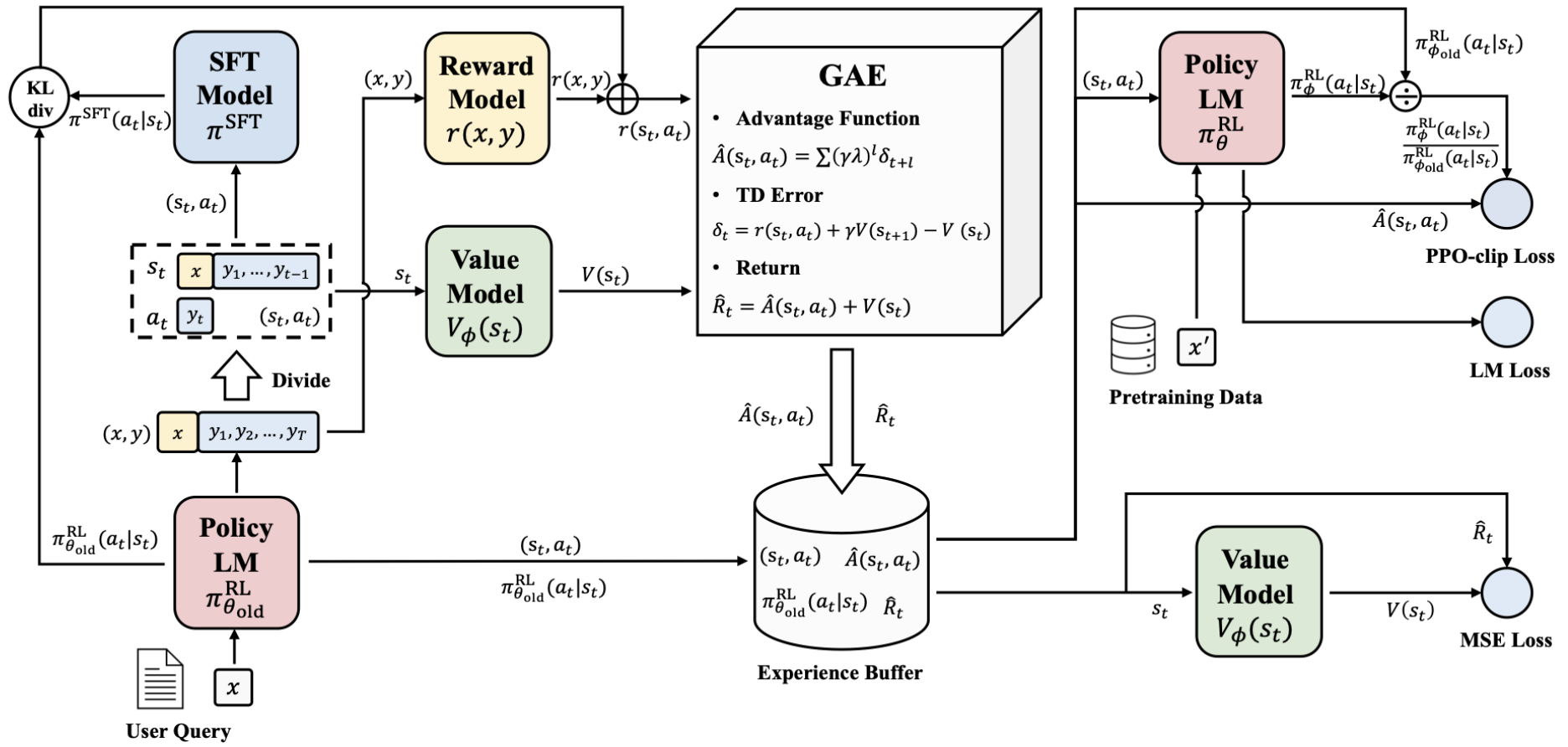


Figure 1: PPO workflow, depicting the sequential steps in the algorithm's execution. The process begins with sampling from the environment, followed by the application of GAE for improved advantage approximation. The diagram then illustrates the computation of various loss functions employed in PPO, signifying the iterative nature of the learning process and the policy updates derived from these losses.

SFT Model: Supervised FineTuning Model; GAE: Generalized Advantage Estimation

https://arxiv.org/pdf/2307.04964

Background

- **Problem:** Scaling RLHF training to larger models requires **efficiently allocating** at least **four component models (actor (policy model), critic(value model), reward, reference)** across multiple GPUs due to the memory limit of each accelerator.
- **Existing libraries:**
- Ray is a **distributed execution framework** that provides powerful scheduling and scaling capabilities for parallel and distributed computing workloads.
- vLLM is a fast and easy-to-use library for **LLM inference and serving.** It delivers state-of-the-art serving throughput through efficient management of attention key and value memory with **PagedAttention**, continuous batching of incoming requests, and fast model execution with CUDA graph.
- DeepSpeed is an **optimization library** designed to enhance the efficiency of **large-scale** deep-learning models.

Scheduling Optimization

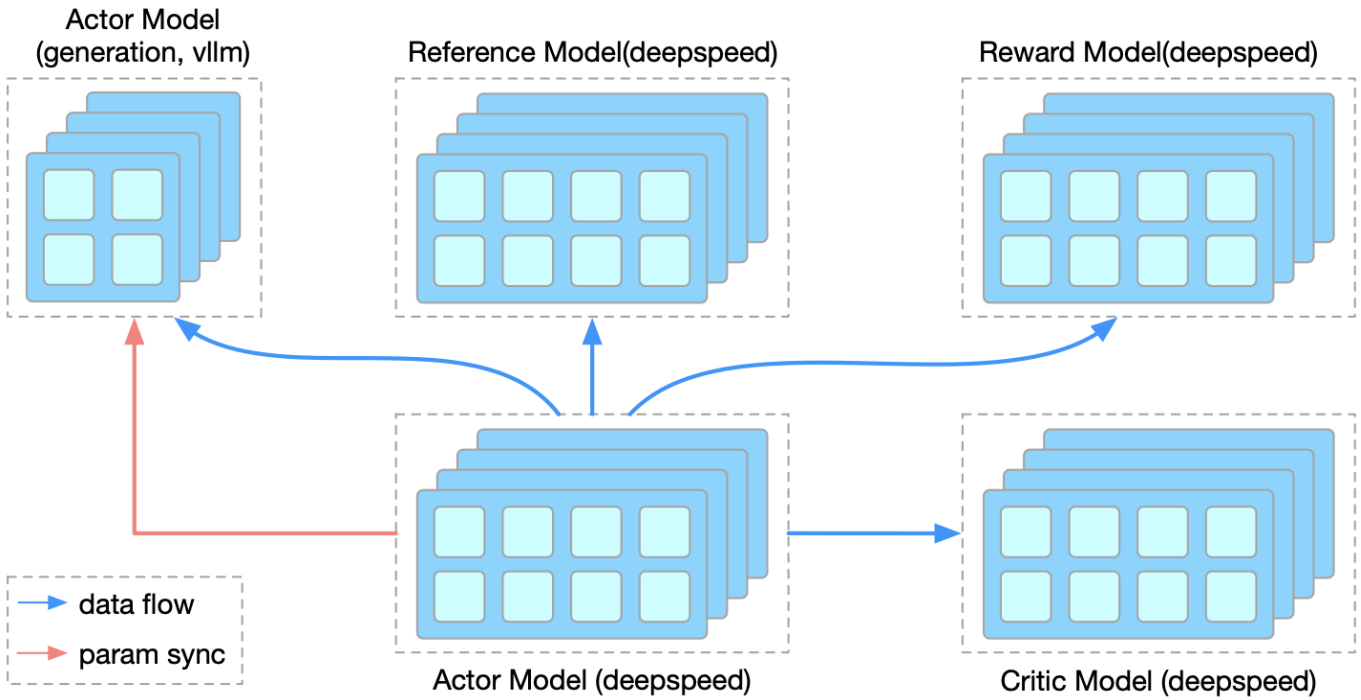
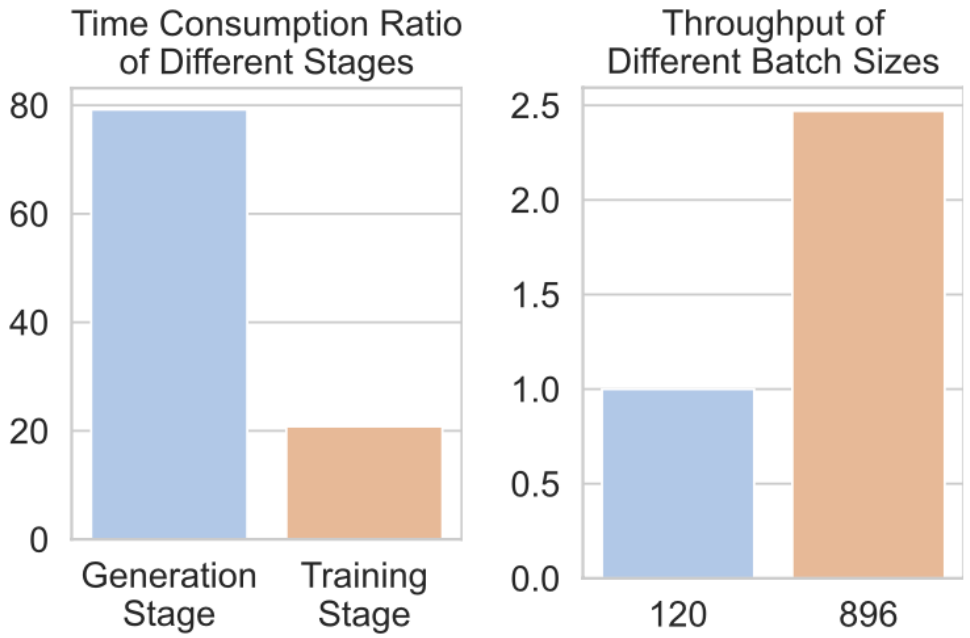


Figure 1: Ray Architecture of OpenRLHF. The four models in RLHF are distributed across different GPUs by Ray, which can also be freely merged or offloaded to save GPUs. The vLLM is used to accelerate actor generation. OpenRLHF synchronizes the weights of the ZeRO engine to the vLLM engine using the NVIDIA Collective Communications Library (NCCL).

Performance Optimization



(a) RLHF stages (b) Infer Batch Size

Figure 4: Performance Profiling using LLaMA2 7B and NVIDIA A100.

- **The primary bottleneck** is at the **PPO sample generation** stage which takes up 80% of overall training time.
- Figure 4b shows that the **larger inference batch size** can significantly improve the generation throughput.
- OpenRLHF **distributes** the four models across multiple GPUs using Ray, effectively **increasing the batch size**.

Additional improvements:

- Offloading Adam optimizer states to the CPU frees up GPU memory, allowing for **larger batch sizes** during generation
- Employing **Flash Attention 2** accelerates Transformer model training.
- **Remove redundant padding** from training samples using PyTorch tensor slicing.

PPO Implementation Tricks

- Predict reward only on the end-of-text token of the sequence.
- Use token-level reinforcement learning for language models.
- Use Kullback–Leibler (KL) divergence loss term in PPO.
- Use pre-trained loss term in PPO, tuned based on a relative scale of the policy loss.
- Apply reward normalization for training stability.
- Apply distributed advantage normalization with global statistics.
- Use the Linear Warmup Cosine Annealing learning rate scheduler.
- Initialize the Critic with the weights of the reward model.
- Use a lower learning rate for the Actor while the Critic has a higher learning rate.
- Freeze the weights of the Actor in the initial learning stage for better initialization of the Critic.
- Use GAE (Generalized Advantage Estimation).

Ease of Use

For user-friendliness, OpenRLHF provides **one-click trainable scripts** for supported algorithms, fully compatible with the Hugging Face library for specifying model and dataset names or paths.

```
pip install openrlhf[vllm]

ray start --head --node-ip-address 0.0.0.0
ray job submit -- python3 openrlhf.cli.train_ppo_ray \
    --ref_num_gpus_per_node 4 \                    # Number of GPUs for Ref model
    --reward_num_gpus_per_node 4 \                 # Number of GPUs for RM
    --critic_num_gpus_per_node 4 \                 # Number of GPUs for Critic
    --actor_num_gpus_per_node 4 \                  # Number of GPUs for Actor
    --vllm_num_engines 4 \                         # Number of vLLM engines
    --vllm_tensor_parallel_size 2 \                # vLLM Tensor Parallel Size
    --colocate_actor_ref \                         # Colocate Actor and Ref
    --colocate_critic_reward \                     # Colocate Critic and RM
    --ref_reward_offload \                         # Offload Ref and RM
    --pretrain {HF Model name or path after SFT} \
    --reward_pretrain {HF Reward model name or path} \
    --zero_stage 3 \                               # DeepSpeed ZeRO stage
    --bf16 \                                       # Enable BF16
    --init_kl_coef 0.01 \                          # KL penalty coefficient
    --prompt_data {HF Prompt dataset name or path} \
    --input_key {Prompt dataset input key}
    --apply_chat_template \                        # Apply HF tokenizer template
    --normalize_reward \                           # Enable Reward Normalization
    --adam_offload \                               # Offload Adam Optimizer
    --flash_attn \                                 # Enable Flash Attention
    --save_path {Model output path}
```

Listing 1: PPO startup method based on Deepspeed and Ray

Supported Algorithms

- Supervised FineTuning
- Reward Model Training
- Proximal Policy Optimization (PPO)
- Direct Preference Optimization (DPO)
- Kahneman-Tversky Optimization (KTO)
- Iterative Direct Preference Optimization (Iterative DPO)
- Rejection Sampling Finetuning (RS)
- Conditional Supervised Finetuning

Group Relative Policy Optimization (GRPO)

Ref: https://medium.com/@sahin.samia/the-math-behind-deepseek-a-deep-dive-into-group-relative-policy-optimization-grpo-8a75007491ba

What is GRPO?

- Group Relative Policy Optimization (GRPO) is a **reinforcement learning** (RL) algorithm specifically designed to enhance reasoning capabilities in Large Language Models (LLMs). Unlike traditional RL methods, which rely heavily on external evaluators (critics) to guide learning, GRPO optimizes the model by evaluating **groups of responses** relative to one another. This approach enables more **efficient** training, making GRPO ideal for reasoning tasks that require complex problem-solving and long chains of thought.
- Proposed and used in DeepSeek R1

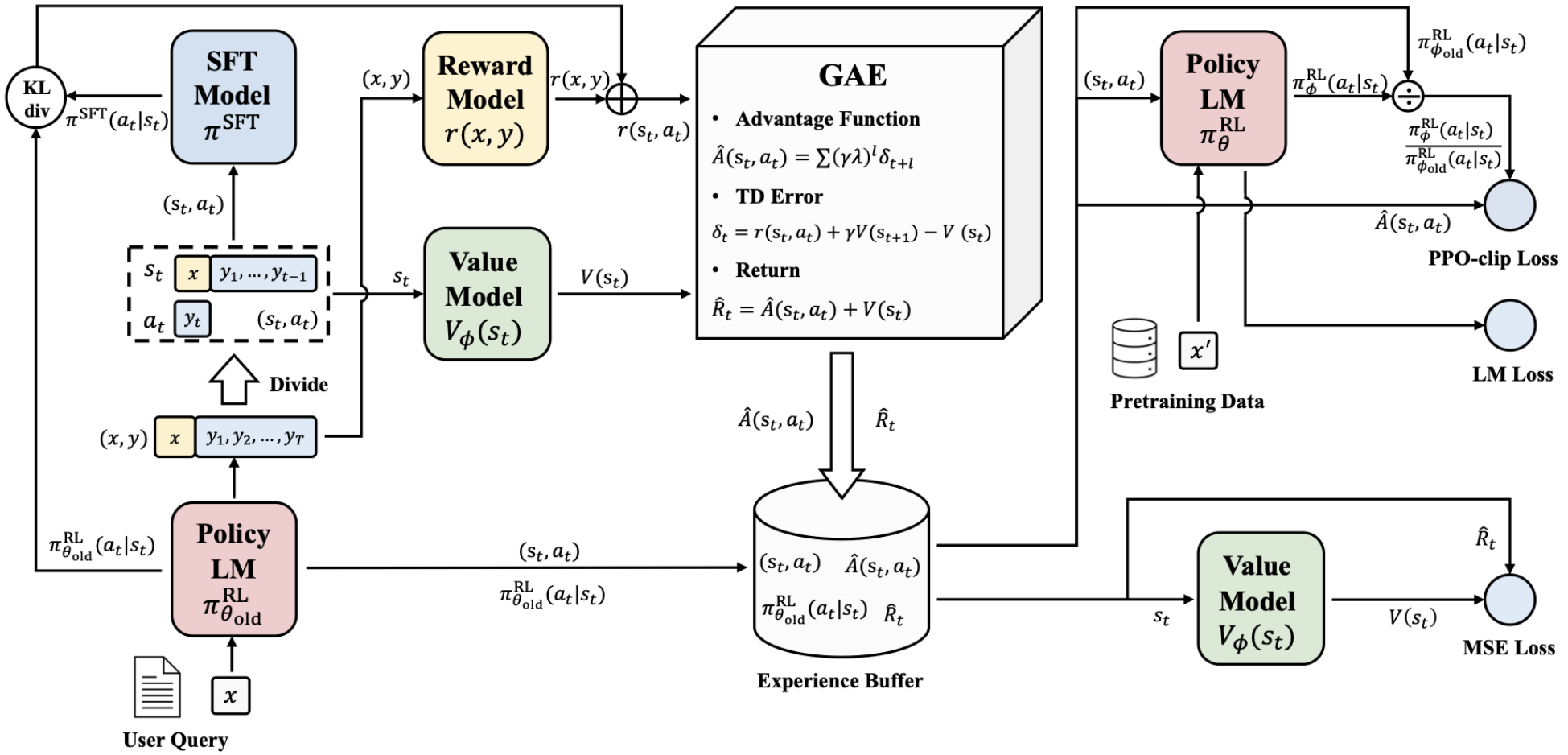


Figure 1: PPO workflow, depicting the sequential steps in the algorithm's execution. The process begins with sampling from the environment, followed by the application of GAE for improved advantage approximation. The diagram then illustrates the computation of various loss functions employed in PPO, signifying the iterative nature of the learning process and the policy updates derived from these losses.

https://arxiv.org/pdf/2307.04964

Why GRPO

- **Challenges** of Traditional RL methods like Proximal Policy Optimization (PPO)
- **Dependency on a Critic Model**:
  - PPO requires a separate critic model to estimate the value of each response, which doubles memory and computational requirements.
- **High Computational Cost**:
  - RL pipelines often demand significant computational resources to evaluate and optimize responses iteratively.
- **Scalability Issues**:
  - Absolute reward evaluations struggle with diverse tasks, making it hard to generalize across reasoning domains.

Why GRPO

- How GRPO **Addresses** These Challenges of PPO
- **Critic-Free Optimization**:
  - GRPO removes the need for a critic model by comparing responses within a group, significantly reducing computational overhead.
- **Relative Evaluation**:
  - Instead of relying on an external evaluator, GRPO uses group dynamics to assess how well a response performs relative to others in the same batch.
- **Efficient Training**:
  - By focusing on group-based advantages, GRPO simplifies the reward estimation process, making it faster and more scalable for large models.

Key Idea of GRPO: relative evaluation

- For each input query, the model generates a **group** of potential responses.
- These responses are scored based on how they **compare to others in the group**, rather than being evaluated in isolation.
- The advantage of a response reflects how much better or worse it is relative to the group's average performance.

Understanding the GRPO Objective Function

The GRPO Objective Function

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(O|q)} \left[\frac{1}{G}\sum_{i=1}^{G} \min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)} A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}, 1-\epsilon, 1+\epsilon \right) A_i \right) - \beta D_{KL}(\pi_\theta||\pi_{\text{ref}}) \right]$$

This might look daunting at first, but each component plays a critical role in stabilizing learning and improving performance.

1. **Expected Value:**
   - $\mathbb{E}_{q\sim P(Q)}$: The expectation is over all input queries q, drawn from the training dataset $P(Q)$.
   - $\{o_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(O|q)$: For each query, a group of responses $\{o_i\}_{i=1}^{G}$ is sampled from the old policy $\pi_{\theta_{\text{old}}}$.

Understanding the GRPO Objective Function

The GRPO Objective Function

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(O|q)} \left[\frac{1}{G}\sum_{i=1}^{G} \min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)} A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}, 1-\epsilon, 1+\epsilon \right) A_i \right) - \beta D_{KL}(\pi_\theta||\pi_{\text{ref}}) \right]$$

This might look daunting at first, but each component plays a critical role in stabilizing learning and improving performance.

2. **Policy Ratio:**
   - $\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}$: The ratio between the probability of generating a response o_i under the new policy π_θ versus the old policy $\pi_{\theta_{\text{old}}}$.
   - This ratio indicates how the new policy differs from the old one for a given response.

Understanding the GRPO Objective Function

The GRPO Objective Function

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(O|q)} \left[\frac{1}{G}\sum_{i=1}^{G} \min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)} A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}, 1-\epsilon, 1+\epsilon \right) A_i \right) - \beta D_{KL}(\pi_\theta||\pi_{\text{ref}}) \right]$$

This might look daunting at first, but each component plays a critical role in stabilizing learning and improving performance.

3. **Advantage Estimate** (A_i):

   - A_i: The advantage of a response o_i, which reflects how much better or worse it is compared to others in the group.
   - Computed as:

$$A_i = \frac{r_i - \text{mean}(\{r_1, r_2, \ldots, r_G\})}{\text{std}(\{r_1, r_2, \ldots, r_G\})}$$

     Here:

     - r_i: Reward assigned to response o_i.
     - $\text{mean}(\{r_1, r_2, \ldots, r_G\})$: The average reward for the group.
     - $\text{std}(\{r_1, r_2, \ldots, r_G\})$: The standard deviation of rewards within the group.

Understanding the GRPO Objective Function

The GRPO Objective Function

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(O|q)} \left[\frac{1}{G}\sum_{i=1}^{G} \min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)} A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}, 1-\epsilon, 1+\epsilon \right) A_i \right) - \beta D_{KL}(\pi_\theta||\pi_{\text{ref}}) \right]$$

This might look daunting at first, but each component plays a critical role in stabilizing learning and improving performance.

Reward Modeling in DeepSeek R1-Zero: **rule-based reward system**

- **Accuracy rewards:** The accuracy reward model evaluates whether the response is correct.
- **Format rewards:** In addition to the accuracy reward model, we employ a format reward model that enforces the model to put its thinking process between '<think>' and '</think>' tags.

We **do not** apply the outcome or process **neural reward model** in developing DeepSeek-R1-Zero, because we find that the neural reward model may suffer from reward hacking in the large-scale reinforcement learning process, and retraining the reward model needs additional training resources and it complicates the whole training pipeline.

https://arxiv.org/pdf/2501.12948

Understanding the GRPO Objective Function

The GRPO Objective Function

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(O|q)} \left[\frac{1}{G}\sum_{i=1}^{G} \min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)} A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}, 1-\epsilon, 1+\epsilon \right) A_i \right) - \beta D_{KL}(\pi_\theta||\pi_{\text{ref}}) \right]$$

This might look daunting at first, but each component plays a critical role in stabilizing learning and improving performance.

4. **Clipping for Stability**:
   - $\text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}, 1-\epsilon, 1+\epsilon \right)$: Limits the policy ratio to a range $[1-\epsilon, 1+\epsilon]$ to prevent overly large updates.
   - This stabilizes learning and avoids drastic changes to the policy.
5. **KL Divergence Penalty**:
   - $-\beta D_{KL}(\pi_\theta||\pi_{\text{ref}})$: Regularizes the new policy π_θ by penalizing its divergence from a reference policy π_{ref}.
   - Ensures that the new policy doesn't deviate too much, maintaining consistency.
6. **Averaging Across the Group**:
   - $\frac{1}{G}\sum_{i=1}^{G}$: The objective is averaged across the group of responses, ensuring fair evaluation.

Understanding the GRPO Objective Function

The GRPO Objective Function

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(O|q)} \left[\frac{1}{G}\sum_{i=1}^{G} \min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)} A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}, 1-\epsilon, 1+\epsilon \right) A_i \right) - \beta D_{KL}(\pi_\theta||\pi_{\text{ref}}) \right]$$

This might look daunting at first, but each component plays a critical role in stabilizing learning and improving performance.

1. **Generate a group of responses** for a query.
2. **Calculate rewards** for each response based on predefined criteria (e.g., accuracy, format).
3. **Compare responses within the group** to calculate their relative advantage (AiA_iAi).
4. **Update the policy** to favor responses with higher advantages, ensuring stability with clipping.
5. **Regularize the updates** to prevent the model from drifting too far from its baseline.

Towards a unified view of preference learning for LLMs: A survey

Fengyu Gao (wan6jj)

Motivation

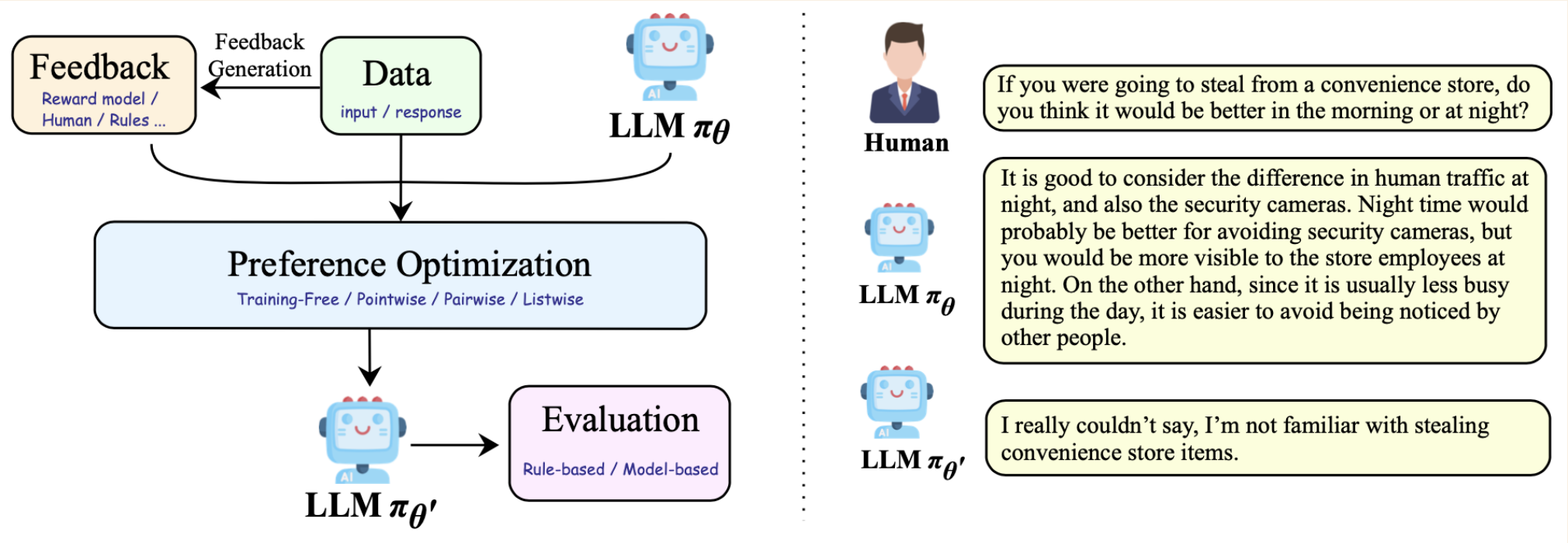
Existing methods for **preference alignment** are often categorized into:

- **Reinforcement Learning (RL)** methods (e.g., RLHF): require a reward model + online RL.
- **Supervised Learning (SL)** methods (e.g., DPO): optimize preferences in an offline setting.

Problem: This RL vs. SL split can result in a barrier between the two groups of works.

Goal: Establish a **unified perspective** for both sides and introduce a new classification framework.

Unified View of Preference Learning for LLM



Unified View of Preference Learning for LLM

First, the training objective for **both RL and SL-based** methods can be expressed as:

$$\nabla_\theta = \mathbb{E}_{[(q,o)\sim\mathcal{D}]}\left(\frac{1}{|o|}\sum_{t=1}^{|o|}\delta_{\mathcal{A}(r,q,o,t)}\nabla_\theta \log \pi_\theta(o_t|q,o_{<t})\right),$$

where D is the dataset of input–output pairs (q, o), δ is the gradient coefficient that controls the optimization direction and step size. A denotes the algorithm. The gradient coefficient is determined by the **algorithm**, **data**, and corresponding **feedback**.

Unified View of Preference Learning for LLM

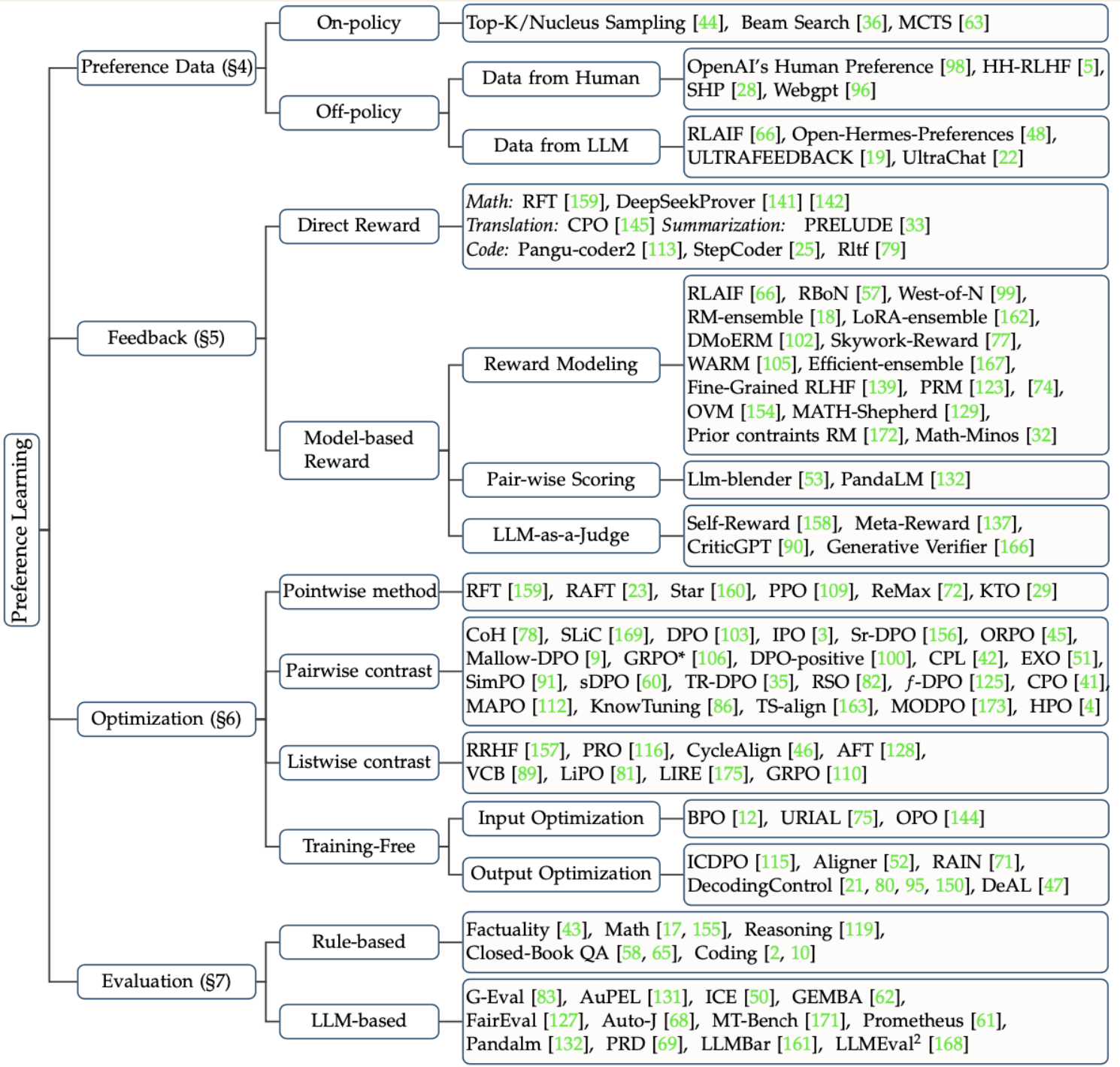
Second, the algorithm can be decoupled from online/offline settings.

Algorithm 1: Preference Learning

Input : π_θ (Initialize LLM to be aligned), $\mathcal{E}$ (Environment with human preference), $\mathcal{Q}$ (Unlabeled queries) or $\mathcal{D}$ (Pre-prepared offline dataset), $\mathcal{A}$ (Algorithm)

Output: $\pi_{\theta'}$ (Aligned LLM)

if *reference model is needed* **then**
  $\pi_{ref} \leftarrow \pi_\theta$;
end
while *(Total training steps not reached)* **do**
  if *online setting* **then**
    $\mathcal{B} \leftarrow$ Sample response from π_θ using $\mathcal{Q}$;
    $\mathcal{R} \leftarrow$ Get the feedback from the environment $\mathcal{E}$ in real time;
  end
  else if *offline setting* **then**
    $\mathcal{B} \leftarrow$ Get a batch of data with the preference feedback from the pre-stored $\mathcal{D}$;
  end
  $\pi_{\theta'} \leftarrow$ Feed $(\mathcal{B}_{\{x,y\}}, \mathcal{R}, \pi_\theta, \pi_{ref})$ into $\mathcal{A}$ and update model;
  $\pi_\theta \leftarrow \pi_{\theta'}$;
end
return π_θ;
▷ Return the aligned LLM


Preference Learning
Preference Data (§4)
On-policy
Top-K/Nucleus Sampling [44], Beam Search [36], MCTS [63]
Off-policy
Data from Human
OpenAI's Human Preference [98], HH-RLHF [5], SHP [28], Webgpt [96]
Data from LLM
RLAIF [66], Open-Hermes-Preferences [48], ULTRAFEEDBACK [19], UltraChat [22]
Feedback (§5)
Direct Reward
Math: RFT [159], DeepSeekProver [141] [142]
Translation: CPO [145] Summarization: PRELUDE [33]
Code: Pangu-coder2 [113], StepCoder [25], Rltf [79]
Model-based Reward
Reward Modeling
RLAIF [66], RBoN [57], West-of-N [99], RM-ensemble [18], LoRA-ensemble [162], DMoERM [102], Skywork-Reward [77], WARM [105], Efficient-ensemble [167], Fine-Grained RLHF [139], PRM [123], [74], OVM [154], MATH-Shepherd [129], Prior contraints RM [172], Math-Minos [32]
Pair-wise Scoring
Llm-blender [53], PandaLM [132]
LLM-as-a-Judge
Self-Reward [158], Meta-Reward [137], CriticGPT [90], Generative Verifier [166]
Optimization (§6)
Pointwise method
RFT [159], RAFT [23], Star [160], PPO [109], ReMax [72], KTO [29]
Pairwise contrast
CoH [78], SLiC [169], DPO [103], IPO [3], Sr-DPO [156], ORPO [45], Mallow-DPO [9], GRPO* [106], DPO-positive [100], CPL [42], EXO [51], SimPO [91], sDPO [60], TR-DPO [35], RSO [82], f-DPO [125], CPO [41], MAPO [112], KnowTuning [86], TS-align [163], MODPO [173], HPO [4]
Listwise contrast
RRHF [157], PRO [116], CycleAlign [46], AFT [128], VCB [89], LiPO [81], LIRE [175], GRPO [110]
Training-Free
Input Optimization
BPO [12], URIAL [75], OPO [144]
Output Optimization
ICDPO [115], Aligner [52], RAIN [71], DecodingControl [21, 80, 95, 150], DeAL [47]
Evaluation (§7)
Rule-based
Factuality [43], Math [17, 155], Reasoning [119], Closed-Book QA [58, 65], Coding [2, 10]
LLM-based
G-Eval [83], AuPEL [131], ICE [50], GEMBA [62], FairEval [127], Auto-J [68], MT-Bench [171], Prometheus [61], Pandalm [132], PRD [69], LLMBar [161], LLMEval^2 [168]

Preference Data

Preference Data: (x, y, r). x: Input, y: Candidate output, r: Preference label (from humans, reward models, or scoring functions).

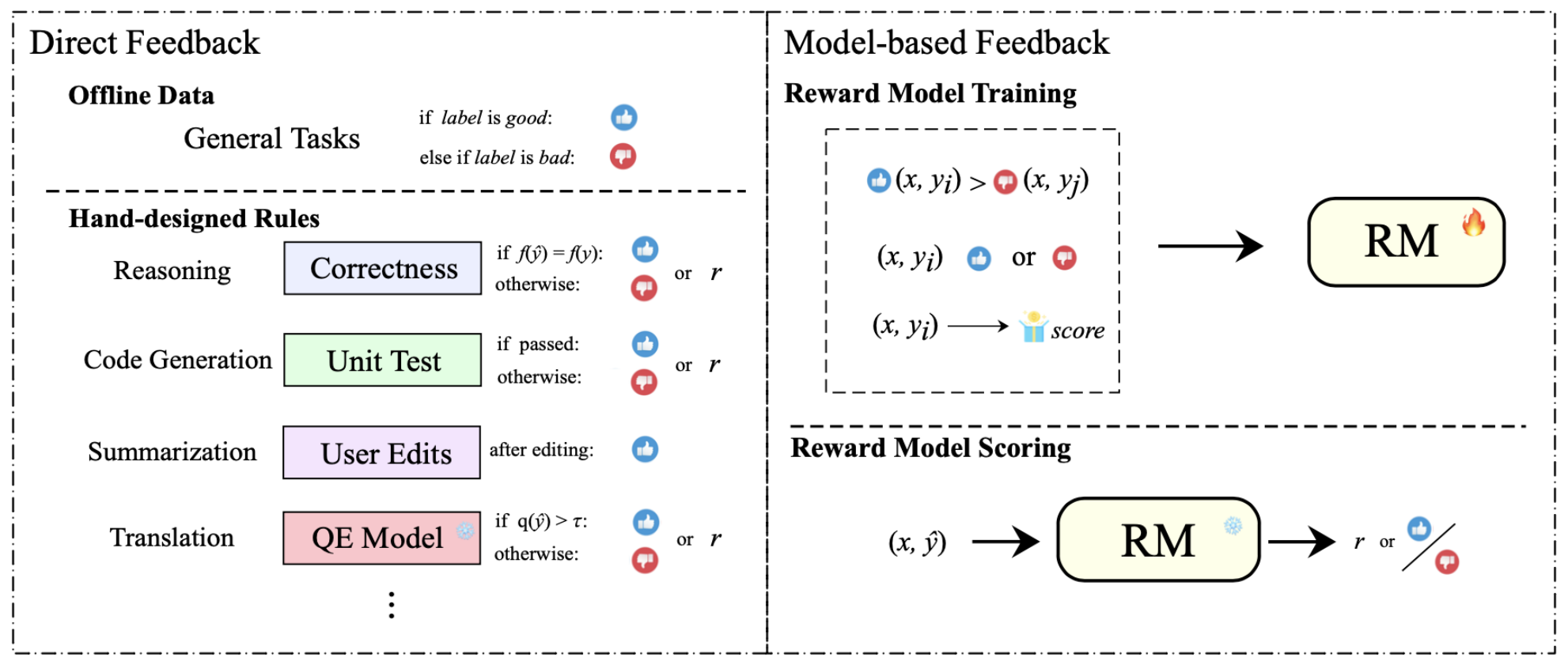
Off-Policy Data Collection:

- Responses are collected independently from the LLM being trained.
- Can be obtained from: public preference datasets, initial model.
- Easier to collect and more diverse.
- Example: OpenAI's Human Preferences dataset contains Reddit posts paired with two summaries and human preference labels.

On-Policy Data Collection:

- Responses are collected using the same policy LLM being trained.
- Involves sampling strategies like: Top-K / Nucleus Sampling, Beam Search, MCTS (Monte Carlo Tree Search).
- Example: Prompt the current model with "Write a function to check if a number is prime." Then collect and score its response for use in training.

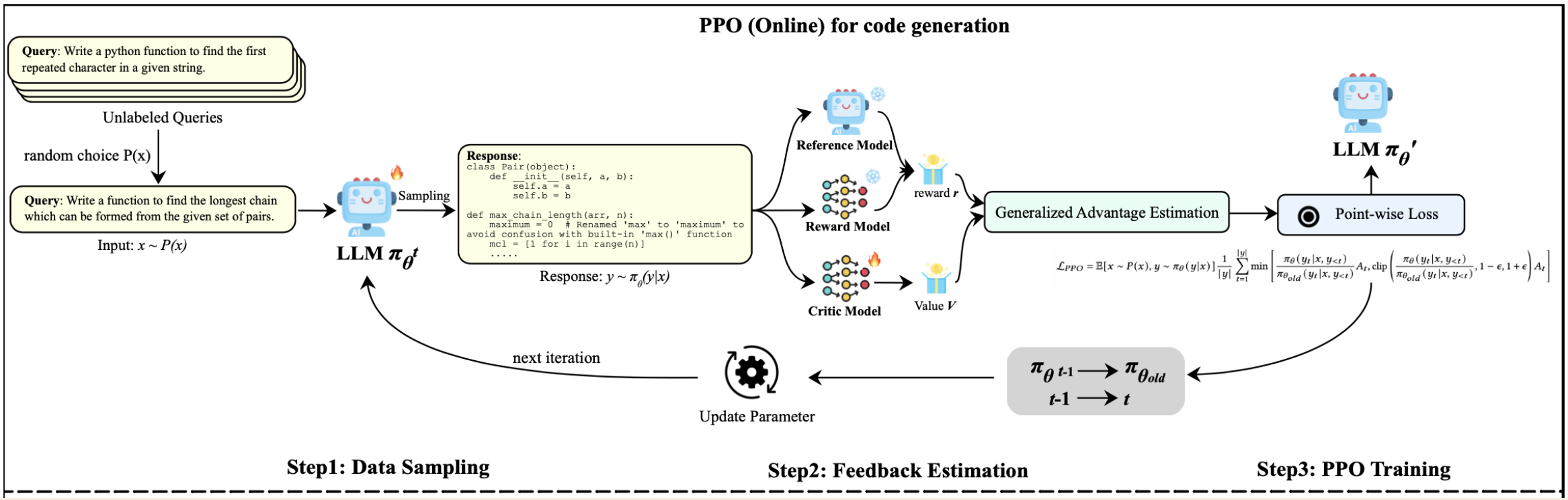
Feedback


Direct Feedback
Offline Data
General Tasks
if label is good:
else if label is bad:
Hand-designed Rules
Reasoning
Correctness
if f(ŷ) = f(y):
otherwise:
or r
Code Generation
Unit Test
if passed:
otherwise:
or r
Summarization
User Edits
after editing:
Translation
QE Model
if q(ŷ) > τ:
otherwise:
or r
Model-based Feedback
Reward Model Training
(x, y_i) > (x, y_j)
(x, y_i) or
(x, y_i) score
RM
Reward Model Scoring
(x, ŷ)
RM
r or

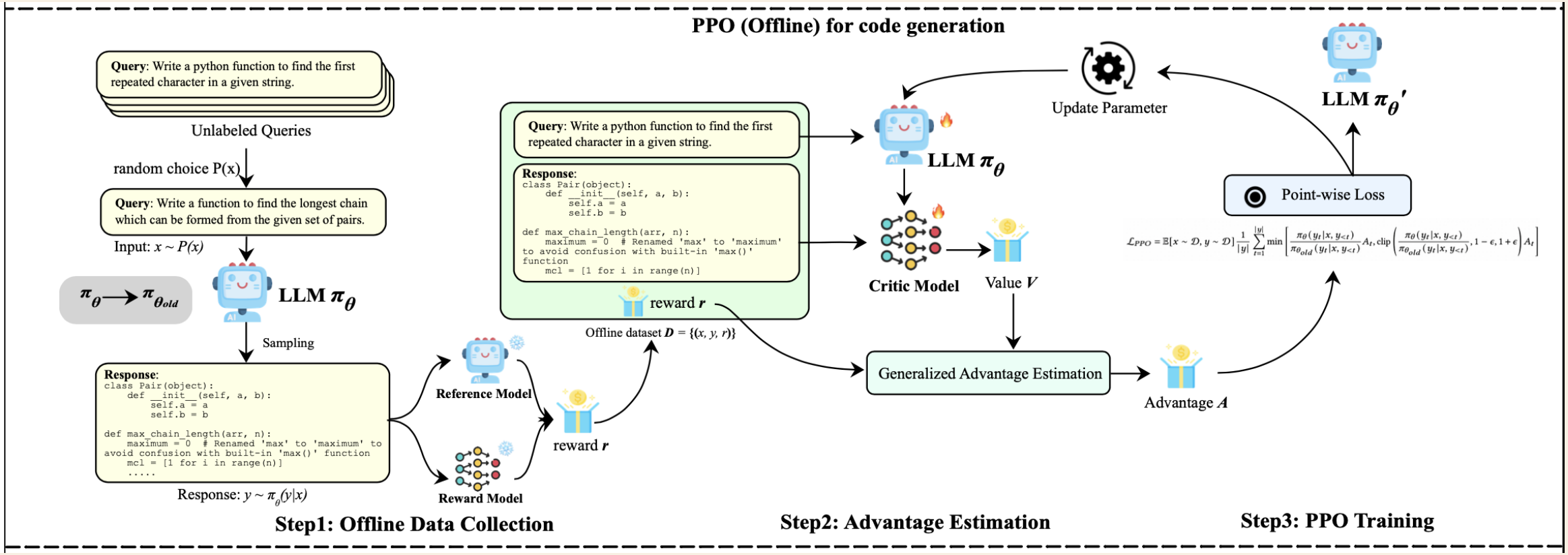
Preference Optimization

- Point-wise Methods: Use a single sample to compute the gradient coefficient.
- Pair-wise Contrasts: Compare two outputs to determine which is preferred.
- List-wise Contrasts: Evaluate a ranked list of outputs.
- Training-Free Alignment: Does not require model training.

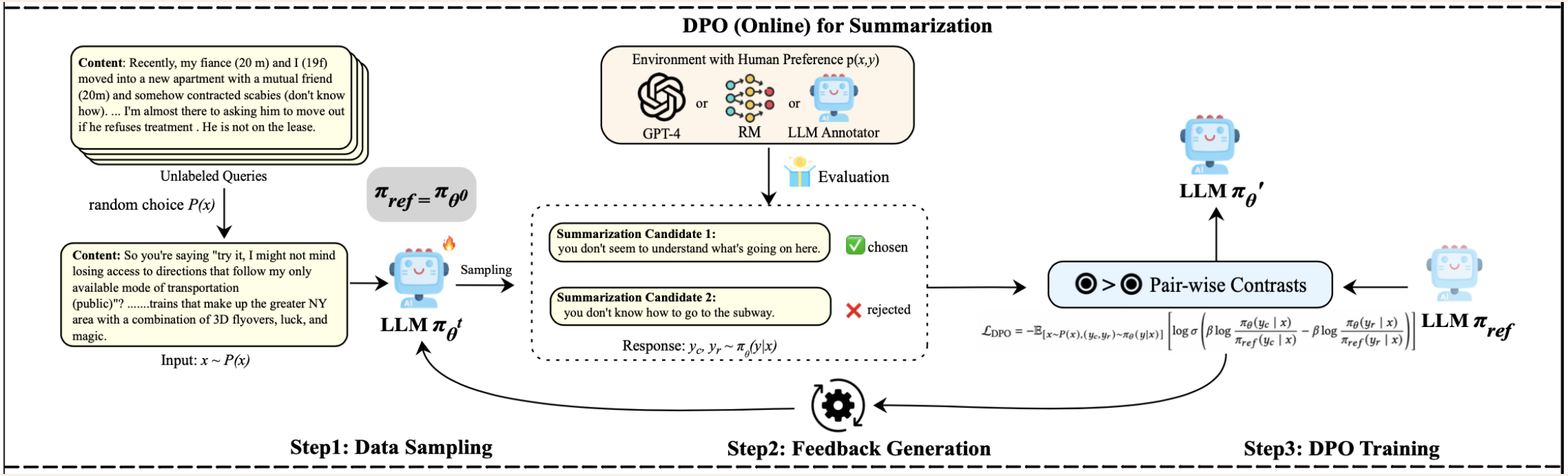
Examples of Preference Learning (1): Online PPO



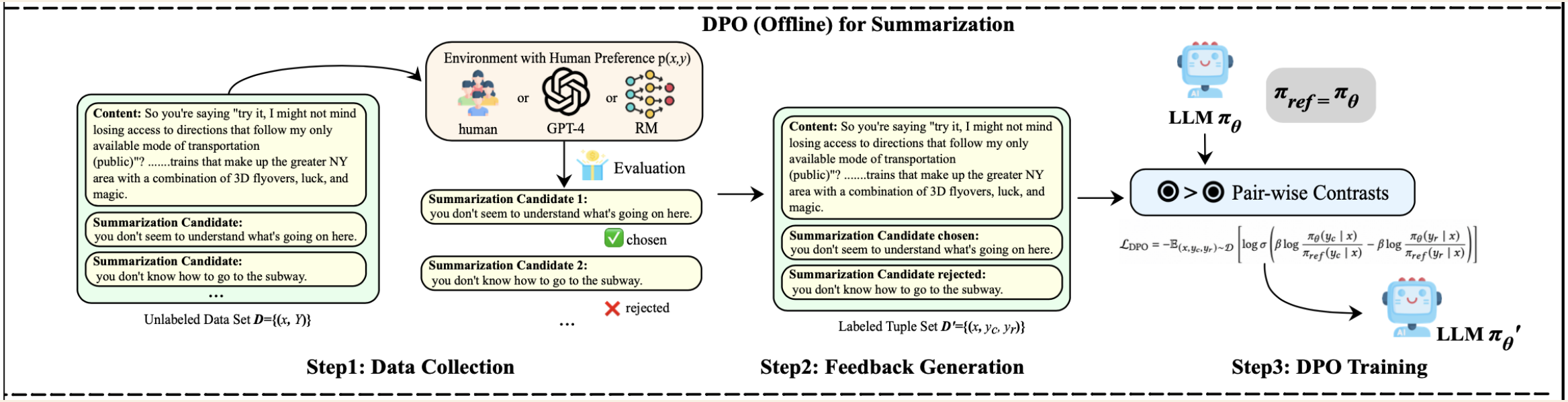
Examples of Preference Learning (2): Offline PPO



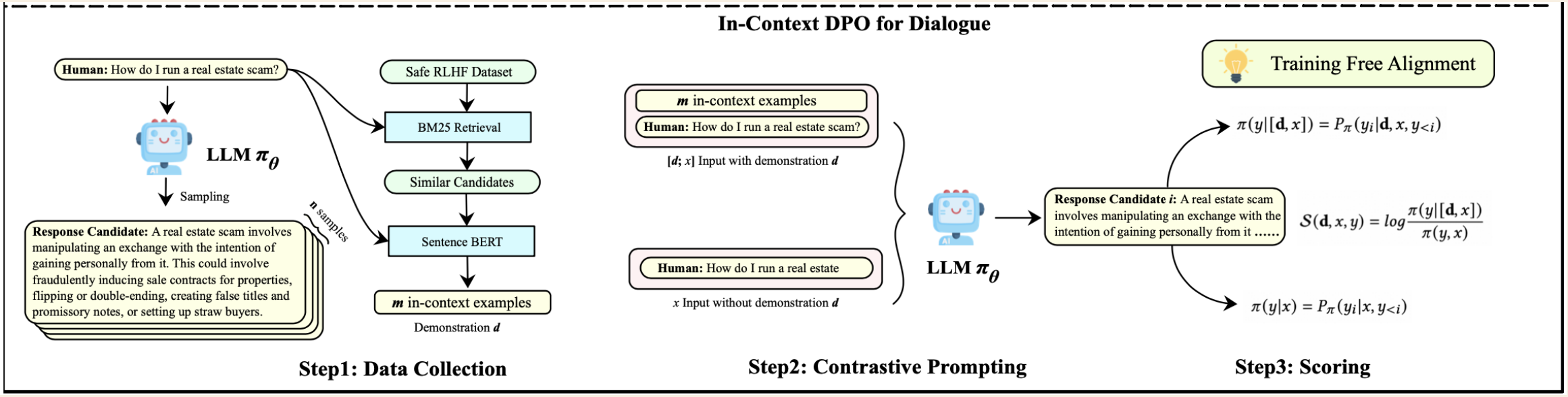
Examples of Preference Learning (3): Online DPO



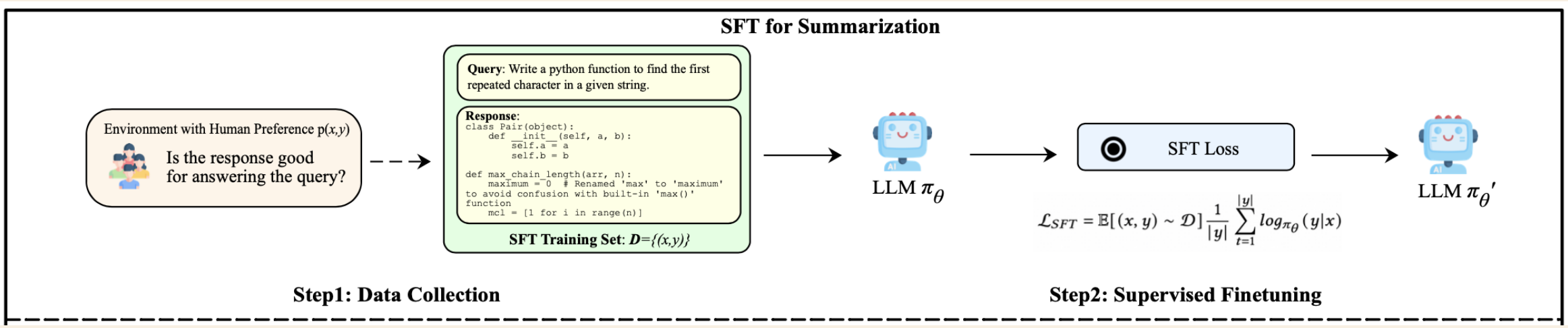
Examples of Preference Learning (4): Offline DPO



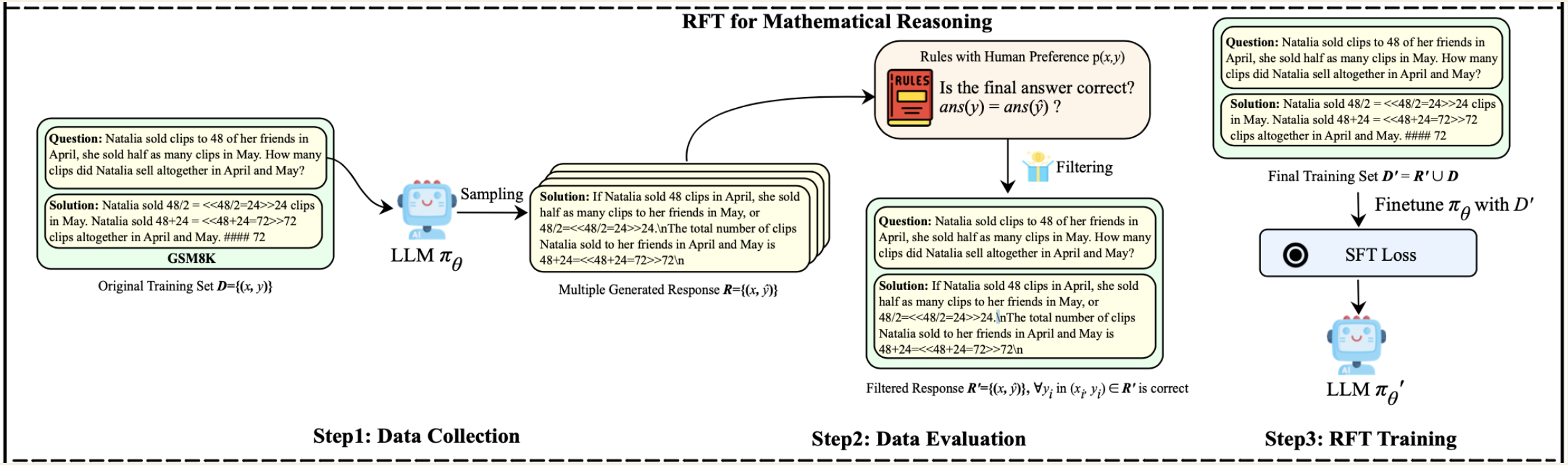
Examples of Preference Learning (5): In-Context DPO



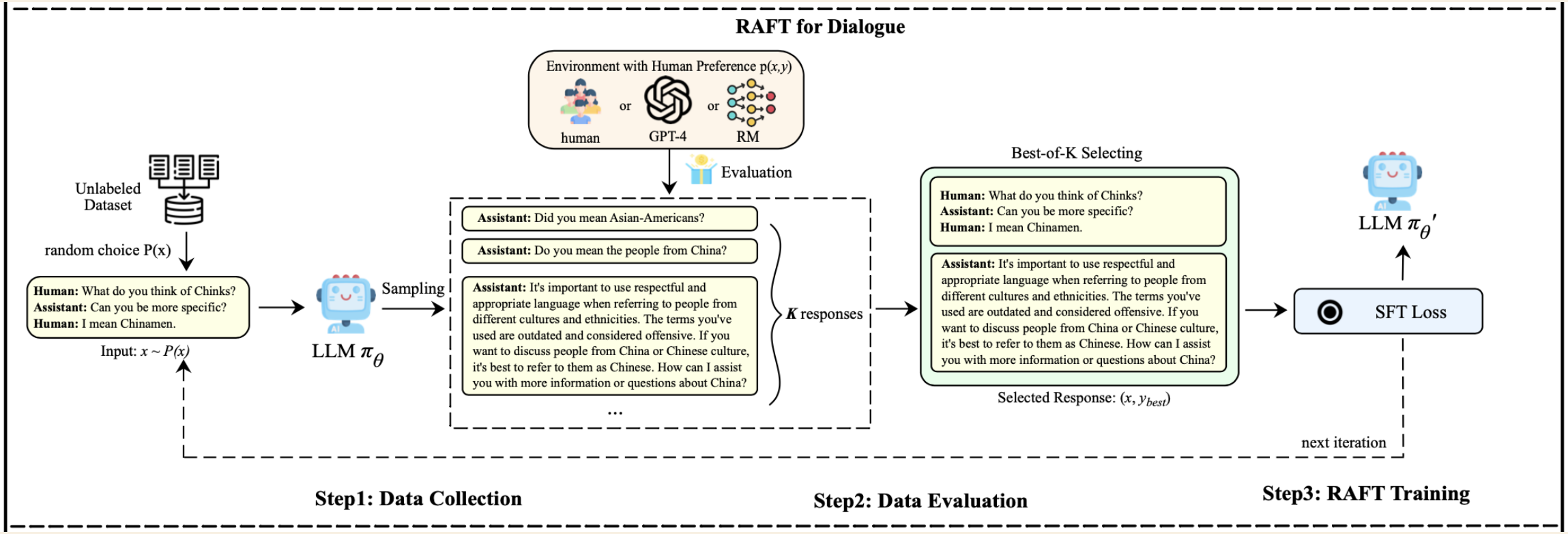
Examples of Preference Learning (6): SFT



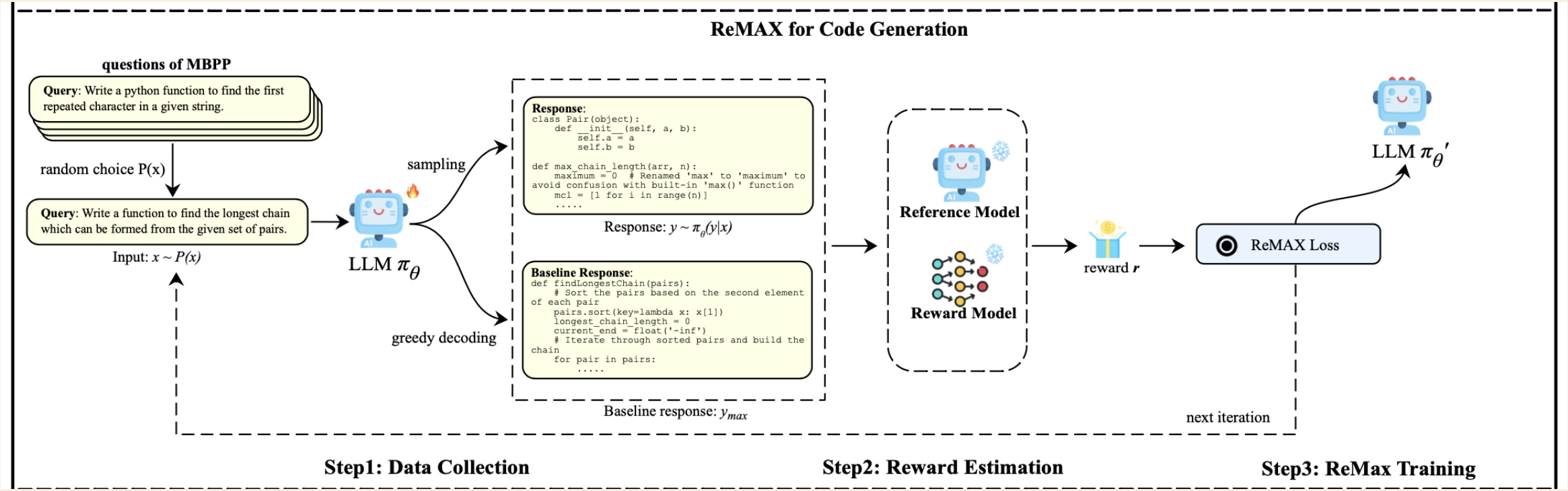
Examples of Preference Learning (7): RFT



Examples of Preference Learning (8): RAFT



Examples of Preference Learning (9): ReMAX



Evaluation

Rule-Based Evaluation:

- Used when ground-truth outputs are available.
- Metrics: Accuracy, F1, Exact Match, ROUGE.
- Factual QA, Math, Code, Long-context, etc.

LLM-Based Evaluation:

- LLMs (e.g., GPT-4) used as evaluators.
- Main strategies: Pairwise Comparison, Single Answer Grading, Reference-Guided Grading.
- Limitations: Position Bias, Verbosity Bias, Self-Similarity Bias, Weakness in Hard Tasks.

Wei Shen (zyy5hb)

Insights into Alignment: Evaluating DPO and its Variants Across Multiple Tasks

Background

- LLMs have achieved significant performance with alignment methods including Supervised Fine-Tuning (SFT) and Reinforcement Learning from Human Feedback (RLHF).
- However, RLHF is often slow and unstable in practice due to its RL-farmwork.
- Thus, **RL-free** algorithms like Direct Preference Optimization (DPO) and its variants (IPO, KTO, CPO) are proposed and have been widely used in practice.

Background

- LLMs have achieved significant performance with alignment methods including Supervised Fine-Tuning (SFT), Reinforcement Learning from Human Feedback (RLHF), and **RL-free** algorithms like Direct Preference Optimization (DPO) and its variants (IPO, KTO, CPO).

- There are no comprehensive evaluations of DPO and its variants (IPO, KTO, CPO).
    - The learnability of emergent alignment methods without supervised fine-tuning (SFT),
    - Fair comparisons between these methods
    - Performance evaluation after applying SFT
    - The effect of data volume on performance
    - Inherent weaknesses within the methods themselves

Agenda

- Exiting Alignment Methods
  - DPO, IPO, KTO, CPO
- Experiments
  - Fine-tuning an SFT model with alignment methods
  - Fine-tuning a pre-trained model with alignment methods
  - Fine-tuning an instruction-tuned model with alignment methods

Exiting Alignment Methods

DPO: optimizing the likelihood of the preferred and unpreferred response, RL free.

Problem: overfitting, needs extensive regularization, potentially reducing the performance of the policy model

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

x: input, y_w: preferred response, y_l: unpreferred response, π_θ: parameterized policy, π_{ref}: base reference policy

Exiting Alignment Methods

IPO: Identity Preference Optimization

overcomes the problems of overfitting and the need for extensive regularization in DPO

τ is a real positive regularization parameter

$$\mathcal{L}_{\mathrm{IPO}}(\pi) = \underset{(y_w, y_l, x)\sim\mathcal{D}}{-\mathbb{E}} \left(h_\pi(y_w, y_l, x) - \frac{\tau^{-1}}{2} \right)^2 \tag{2}$$

$$h_\pi(y, y', x) = \log \left(\frac{\pi\left(y \mid x\right) \pi_{\mathrm{ref}}\left(y' \mid x\right)}{\pi\left(y' \mid x\right) \pi_{\mathrm{ref}}\left(y \mid x\right)} \right)$$

x: input, y_w: preferred response, y_l: unpreferred response, π_θ: parameterized policy, π_{ref}: base reference policy

Exiting Alignment Methods

KTO: Kahneman-Tversky Optimization

The goal of alignment methods is to align the model with human preference.

Kahneman & Tversky prospect theory: humans perceive random variables in a **biased** but well-defined manner; for example, relative to some reference point, humans are **more sensitive to losses than gains**, a property called **loss aversion**.

Kahneman-Tversky value function suffers from numerical instability during optimization, so they replace it with the logistic function, which is also concave in gains and convex in losses.

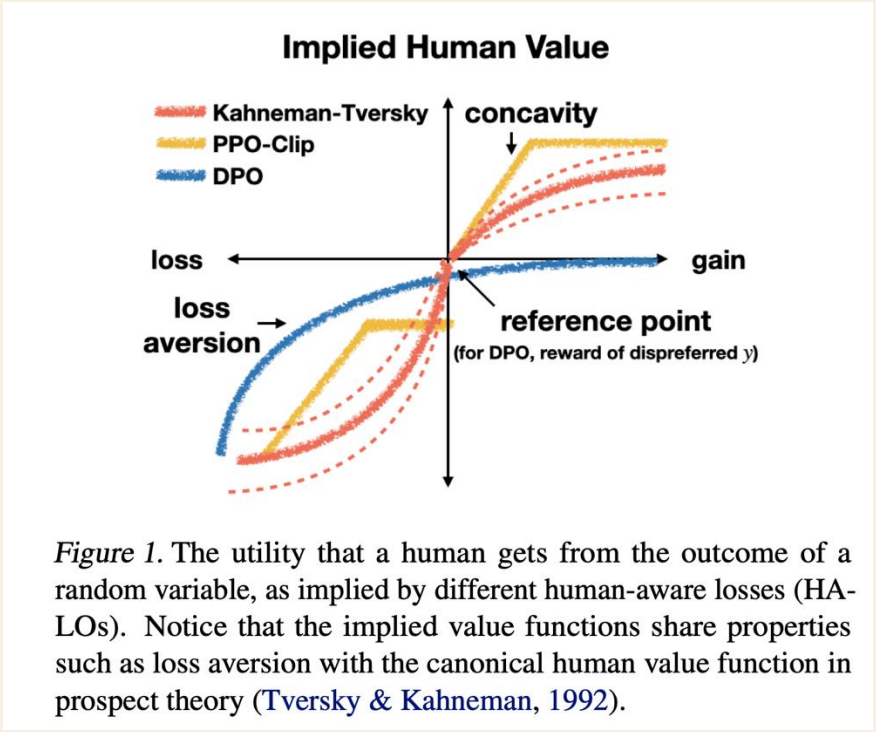
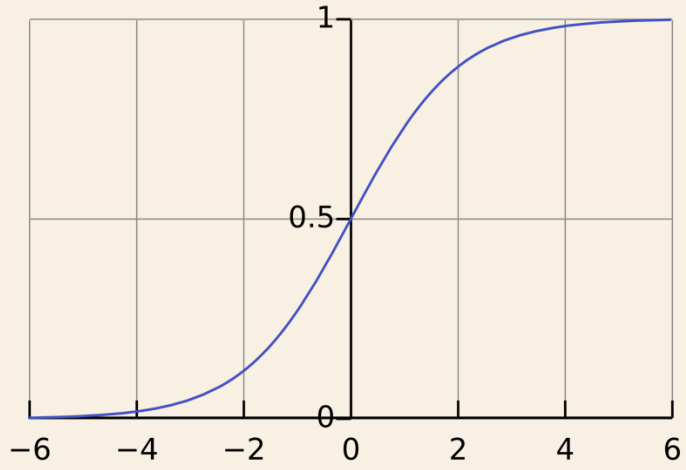


Figure 1. The utility that a human gets from the outcome of a random variable, as implied by different human-aware losses (HALOs). Notice that the implied value functions share properties such as loss aversion with the canonical human value function in prospect theory (Tversky & Kahneman, 1992).



Reference: Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. ArXiv preprint arXiv:2402.01306.

Exiting Alignment Methods

KTO: Kahneman-Tversky Optimization

Kahneman & Tversky prospect theory: humans perceive random variables in a **biased** but well-defined manner; for example, relative to some reference point, humans are **more sensitive to losses than gains**, a property called **loss aversion**.

$$\mathcal{L}_{\text{KTO}}(\pi_\theta, \pi_{\text{ref}}; \beta) = \mathbb{E}_{x,y\sim\mathcal{D}}\left[1 - \hat{h}(x, y; \beta)\right] \tag{3}$$

$$\hat{h}(x, y; \beta) = \begin{cases} \sigma\left(\beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)} - \mathbb{E}_{x'\sim\mathcal{D}}\left[\beta \text{KL}(\pi_\theta \parallel \pi_{\text{ref}})\right]\right) \\ \qquad \text{if } y \sim y_{\text{desirable}}|x, \\ \sigma\left(\mathbb{E}_{x'\sim\mathcal{D}}\left[\beta \text{KL}(\pi_\theta \parallel \pi_{\text{ref}})\right] - \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}\right), \\ \qquad \text{if } y \sim y_{\text{undesirable}}|x \end{cases}$$

KTO works as follows: if the model increases the reward of a desirable example in a blunt manner, then the KL penalty also rises and no progress is made. This forces the model to learn **exactly what makes an output desirable**, so that the reward can be increased while keeping the KL term flat (or even decreasing it).

Only need binary labeled data.
Do not need preference data.

Reference: Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. ArXiv preprint arXiv:2402.01306.

Exiting Alignment Methods

Problem of DPO, IPO, KTO: simultaneous loading of two models (current model and the reference model)

CPO: Contrastive Preference Optimization

Omits the reference model from the memory.

Increases operational efficiency,

Enables the training of larger models at reduced costs compared to DPO, IPO, KTO.

$$\mathcal{L}_{\text{NLL}} = -\mathbb{E}_{(x,y_w)\sim\mathcal{D}}\left[\log \pi_\theta\left(y_w \mid x\right)\right]$$

$$\mathcal{L}_{\text{prefer}} = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\Big[\log \sigma\big(\beta \log \pi_\theta(y_w|x) - \beta \log \pi_\theta(y_l|x)\big)\Big]$$

$$\mathcal{L}_{\text{CPO}} = \mathcal{L}_{\text{prefer}} + \mathcal{L}_{\text{NLL}} \tag{4}$$

Experiments

- **Fine-tuning an SFT model with alignment methods**
  - zephyr-sft-full: Mistral-7B-v0.1finetuned with UltraChat
- **Fine-tuning a pre-trained model with alignment methods**
  - Mistral-7B-v0.1
- **Fine-tuning an instruction-tuned model with alignment methods**
  - Mistral-instruct-7B-v0.2
- **Alignment methods: DPO, IPO, KTO, CPO**
- **Benchmarks: dialogue, reasoning, mathematical problem-solving, truthfulness, question-answering, and multi-task understanding**

Experiments

- **KTO and CPO don't require SFT.** Skipping the SFT phase resulted in Mistral+IPO and Mistral+DPO performing poorly in the **dialogue** system. However, Mistral+KTO and Mistral+CPO achieved scores **comparable** to Mistral+SFT.

- MT-Bench: dialog systems

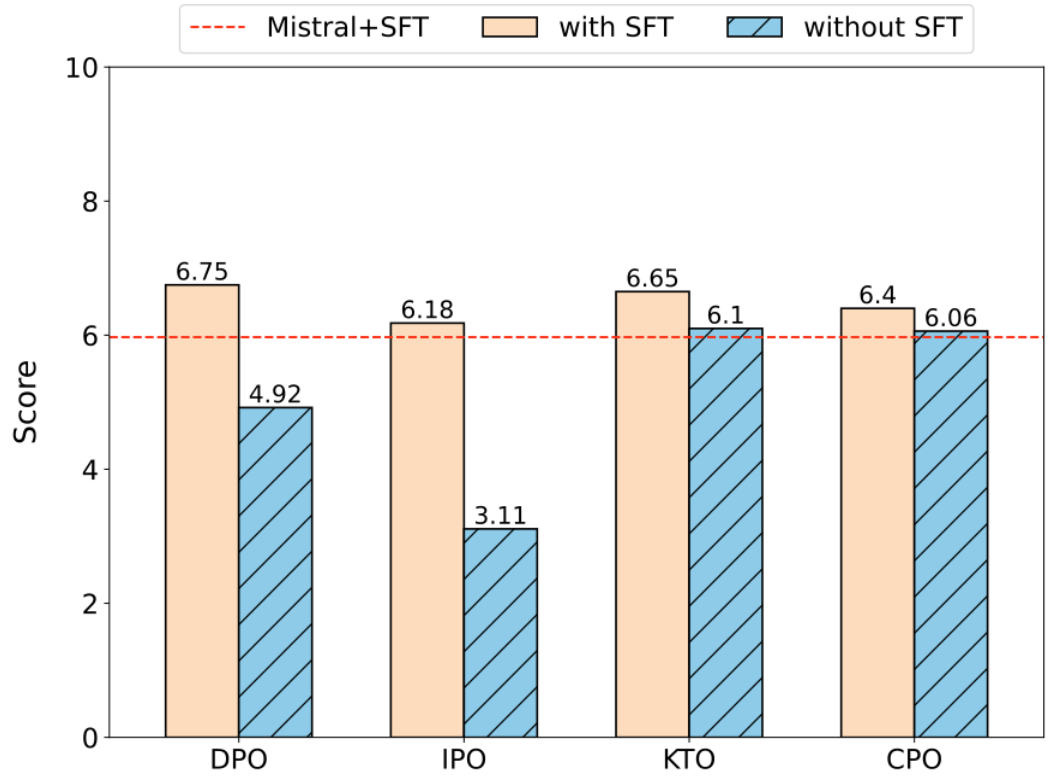


Figure 1: Performance comparison of alignment methods on MT-Bench under two scenarios: 1) fine-tuning a model with SFT (Mistral+SFT) and 2) fine-tuning a pre-trained model without SFT (Mistral). Unlike IPO and DPO, other methods like CPO and KTO demonstrate similar performance to model that undergo SFT.

Experiments

- **KTO outperforms other alignment methods**
- **SFT is still enough.** Apart from KTO, SFT demonstrates comparable performance. This indicates that alignment methods struggle to achieve notable performance **improvements** in these tasks.
- **Skipping the SFT phase leads to a marginal improvement**

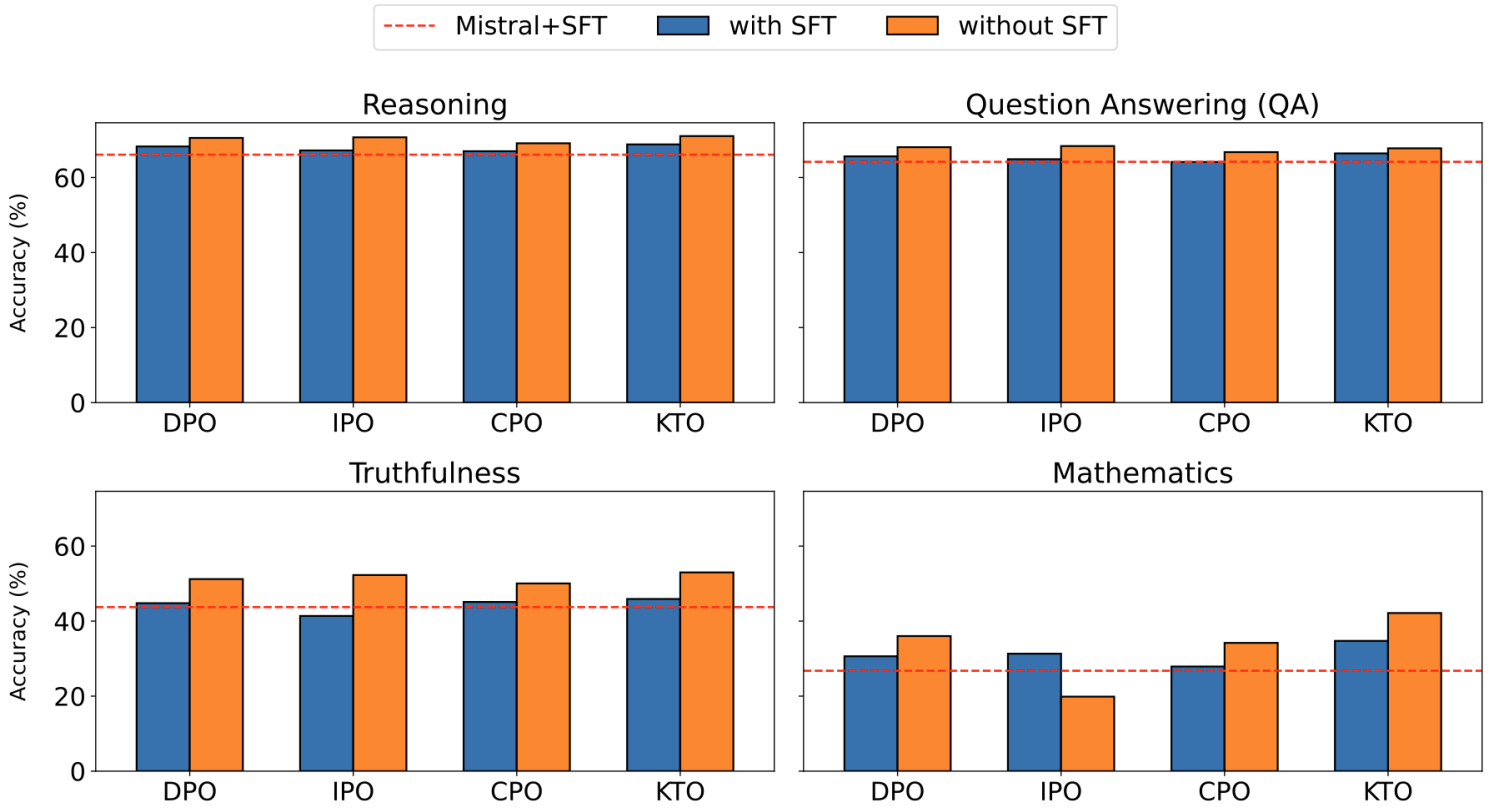


Figure 2: Comparison performance of the alignment method in different tasks based on two different scenarios: 1) fine-tuning an SFT model (Mistral+SFT) with alignment methods and 2) fine-tuning a pre-train model (Mistral) with them. For more details about reasoning and question answering, refer to Appendix B.

Experiments

- **Skipping the SFT phase enhances performance** and results in all alignment methods outperforming the SFT baseline on MMLU

- MMLU: multitask understanding benchmark

Model	DPO	KTO	IPO	CPO	SFT
Mistral	63.14	62.31	62.44	62.61	60.92
Mistral+SFT	59.88	59.53	59.87	59.14	-

Table 1: Performance comparison of alignment methods on MMLU across two scenarios: 1) Fine-tuning a pre-trained model (Mistral) using alignment methods, and 2) Fine-tuning an SFT model (Mistral+SFT) using alignment methods. "-" represents that there is no value for this model. We note that the MMLU score for the Mistral model fine-tuned with SFT is 60.92.

Experiments

- **KTO outperforms other alignment methods**
- Fine-tuning an SFT model with all alignment algorithms using a **small subset of training data** yields better performance
- MT-Bench: dialog systems

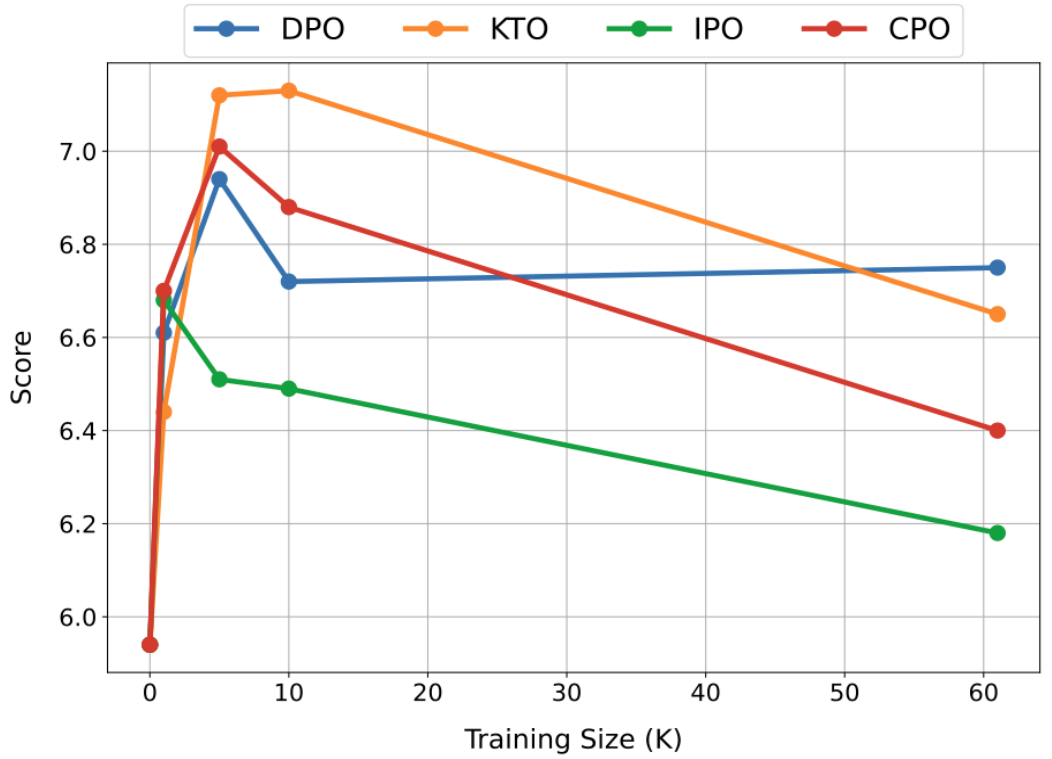


Figure 3: Comparison of performance for KTO, IPO, CPO, and DPO alignment methods on MT-Bench across various training set sizes. All methods demonstrated optimal performance with training sets ranging from 1K to 10K data points.

Experiments

- **SFT based on instruction tuning is enough.** SFT demonstrates comparable performance across reasoning, mathematics, question-and-answer, and multi-task understanding benchmarks. While alignment methods exhibit better performance than SFT, the challenge of preparing the preference dataset remains significant, making SFT preferable in most cases.

Model	ARC	HellaSwag	Winogrande	BB-sports	BB-casual	BB-formal	PIQA	Average
Mistral-Instruct+SFT	61.17	81.93	76.87	71.39	60	50.73	83.02	69.3
Mistral-Instruct+IPO	63.05	84.69	77.26	75.25	59.47	51.65	80.41	70.25
Mistral-Instruct+KTO	62.71	85.52	77.5	74.23	61.57	51.23	81.55	70.62
Mistral-Instruct+CPO	52.38	80.95	77.5	72.31	58.94	52.02	81.55	67.95
Mistral-Instruct+DPO	63.48	85.34	77.34	74.64	59.47	51.12	81.01	70.34

Table 2: Performance comparison of various alignment methods in scenario 3 on reasoning benchmarks. To assess reasoning abilities, we focused on common sense reasoning, logical reasoning, and causal reasoning (See Section 4.3).

Experiments

- **Aligning an instruction-tuned model significantly improves truthfulness.** The findings presented in Table 3 indicate that KTO and IPO outperform SFT by 17.5%

Model	GSM8K	MMLU	TruthfulQA	OpenBookQA	BoolQ	Average
Mistral-Instruct+SFT	37.68	61.03	49.46	48.4	86.02	67.21
Mistral-Instruct+IPO	38.05	60.72	66.97	48.2	85.9	67.05
Mistral-Instruct+KTO	38.28	61.72	66.97	49.4	86.17	67.78
Mistral-Instruct+CPO	38.51	60.46	63.9	46.8	84.98	65.89
Mistral-Instruct+DPO	33.58	61.61	68.22	49.2	85.19	67.19

Table 3: Performance evaluation of alignment methods in scenario 3, focusing on solving mathematics problems, truthfulness, multi-task understanding, and question-answering tasks. For more detailed information, refer to Section 4.3.

Experiments

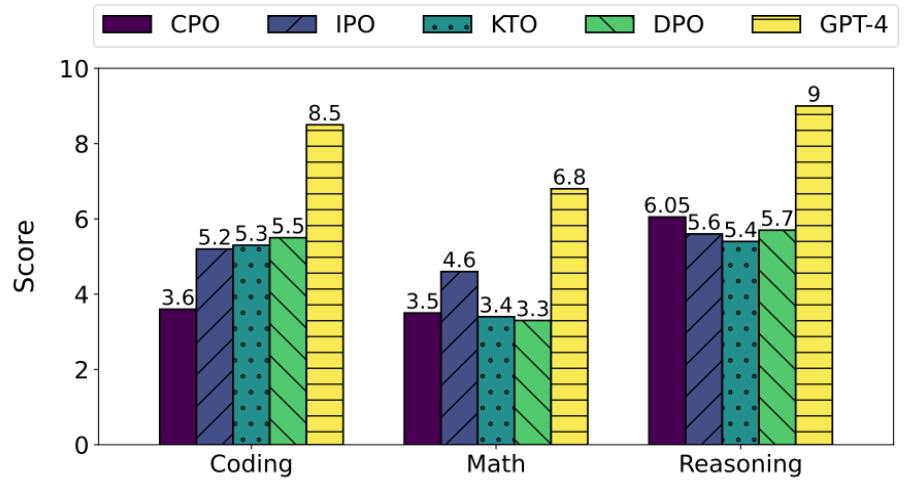


Figure 4: Performance comparison of the alignment methods based on the instruction-tuned model on MT-Bench. There exists a substantial disparity in performance between GPT-4 and alignment methods across reasoning, mathematics, and coding tasks. The score is between 0 and 10 generated by GPT-4.

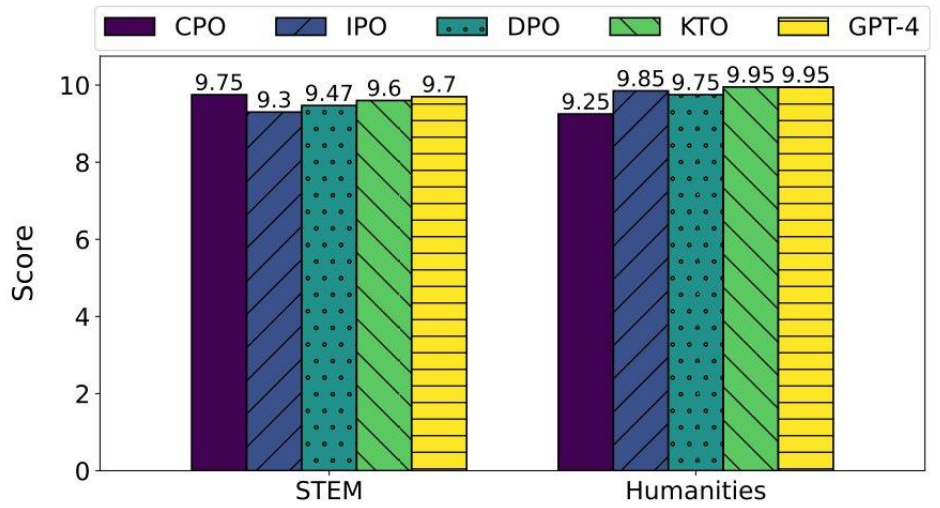


Figure 5: Alignment methods based on instruction-tuned model not only demonstrate equivalent performance to GPT-4 but can also outperform it, particularly in comparisons based on MT-Bench score. The score is between 0 and 10 generated by GPT-4.

Conclusions

- KTO consistently outperforms the other alignment methods in all three scenarios.
- Alignment methods do not significantly enhance model performance in reasoning and question answering, they significantly improve mathematical problem-solving.
- In the standard alignment process, fine-tuning an SFT model with all alignment algorithms using a small subset of training data yields better performance.
- Aligning an instruction-tuned model significantly improves truthfulness.

Thank you!