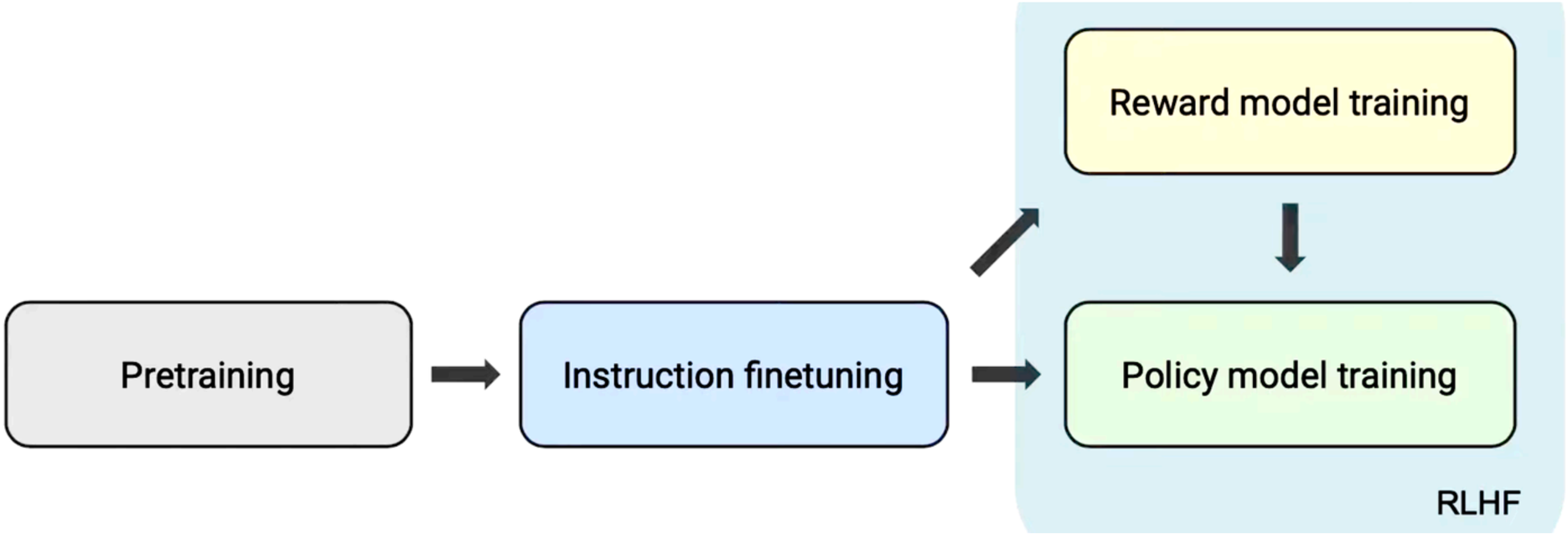
LLMs 2

Intro to NLP

Rahul Mishra
IIIT-Hyderabad
Apr 05 & 12, 2024

Overall Pipeline

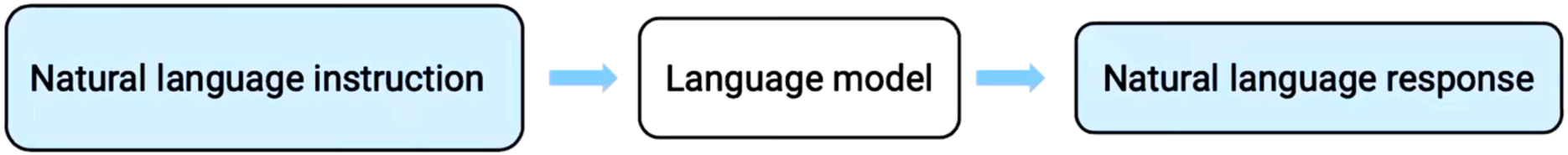


Instruction Fine-tuning

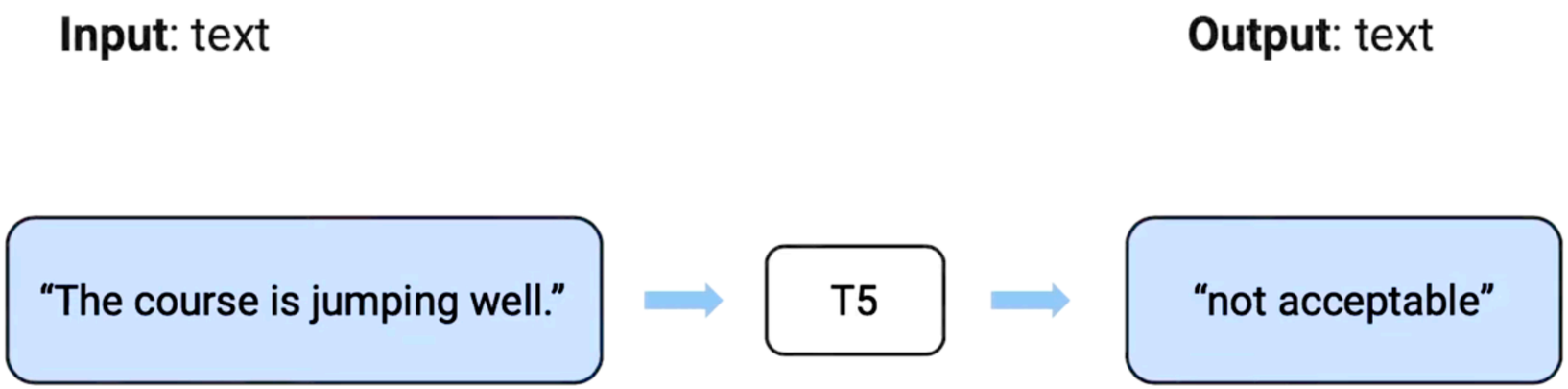
Instruction finetuning

*Frame **all** tasks in the form of*

natural language instruction to natural language response mapping



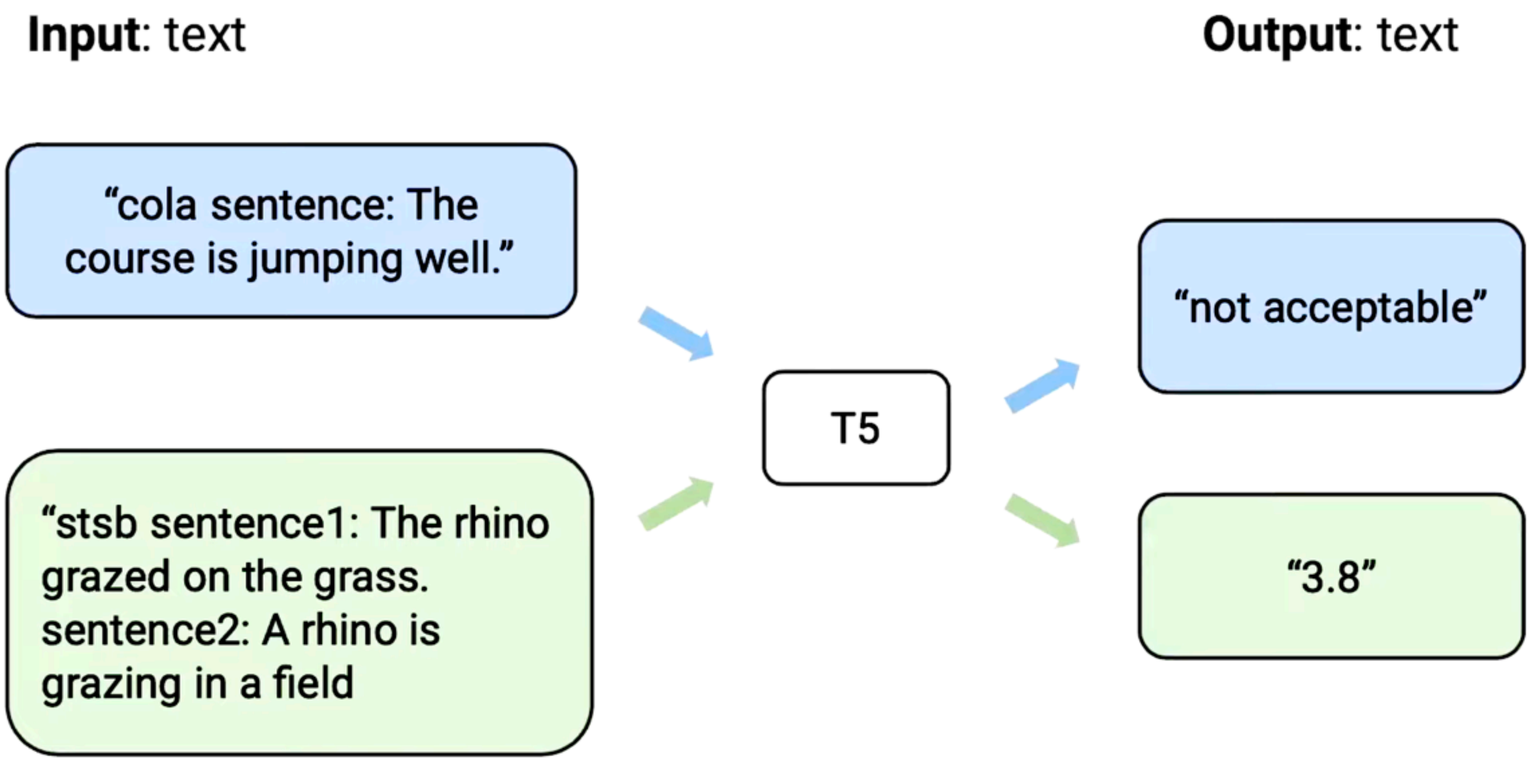
Example



Architecture is unified across tasks with *text-to-text* format

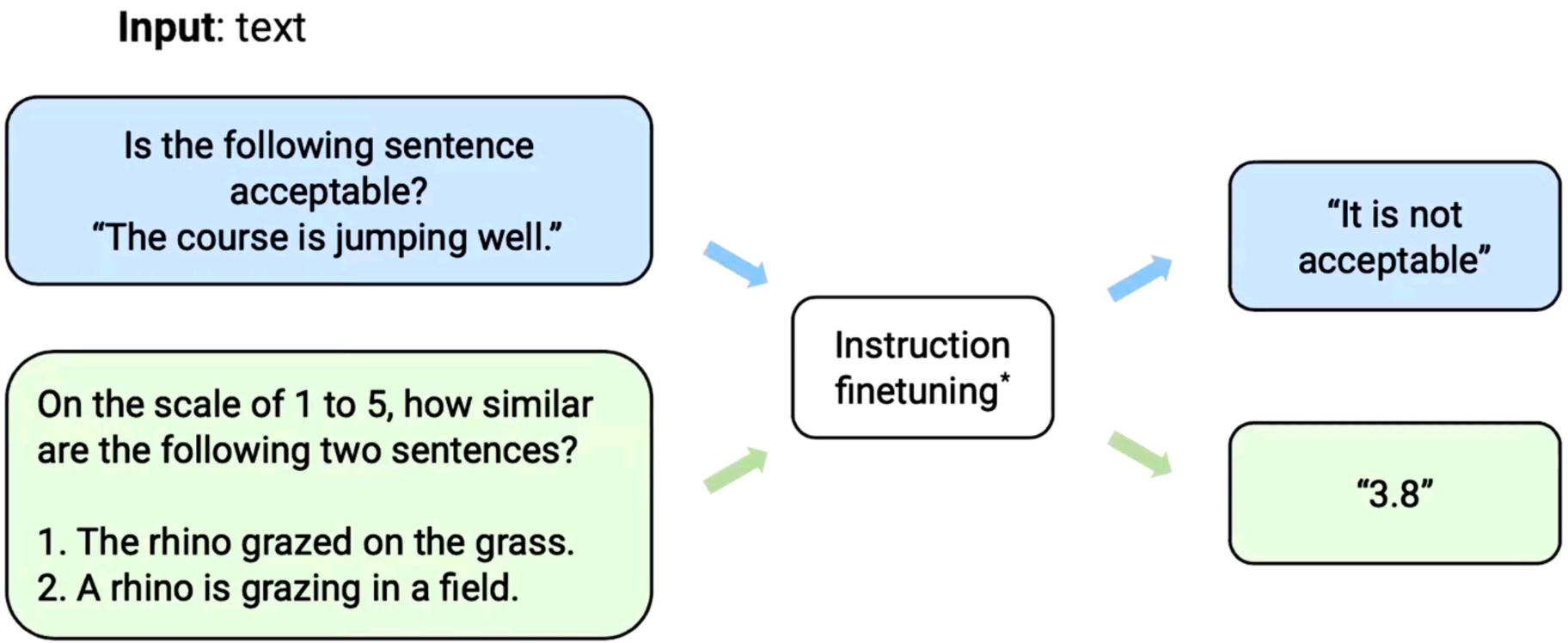
Raffel et al. (2019)

Multi-task Leanring



Tasks are not semantically related

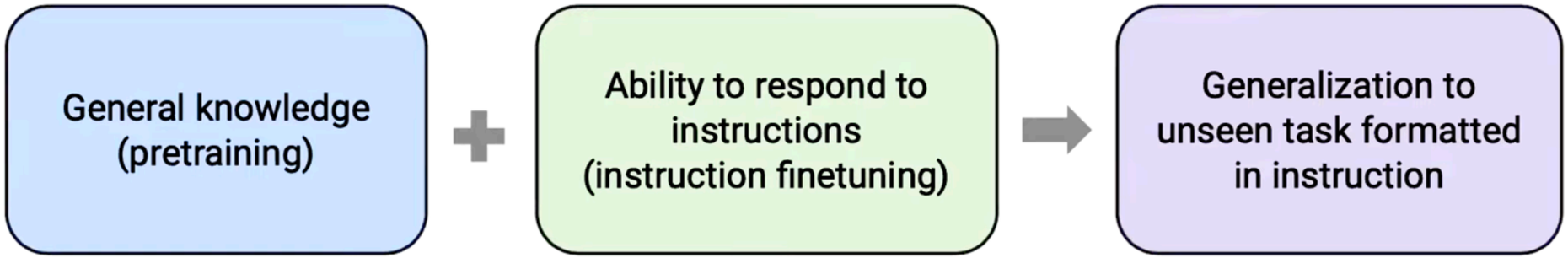
Instruction Finetuning



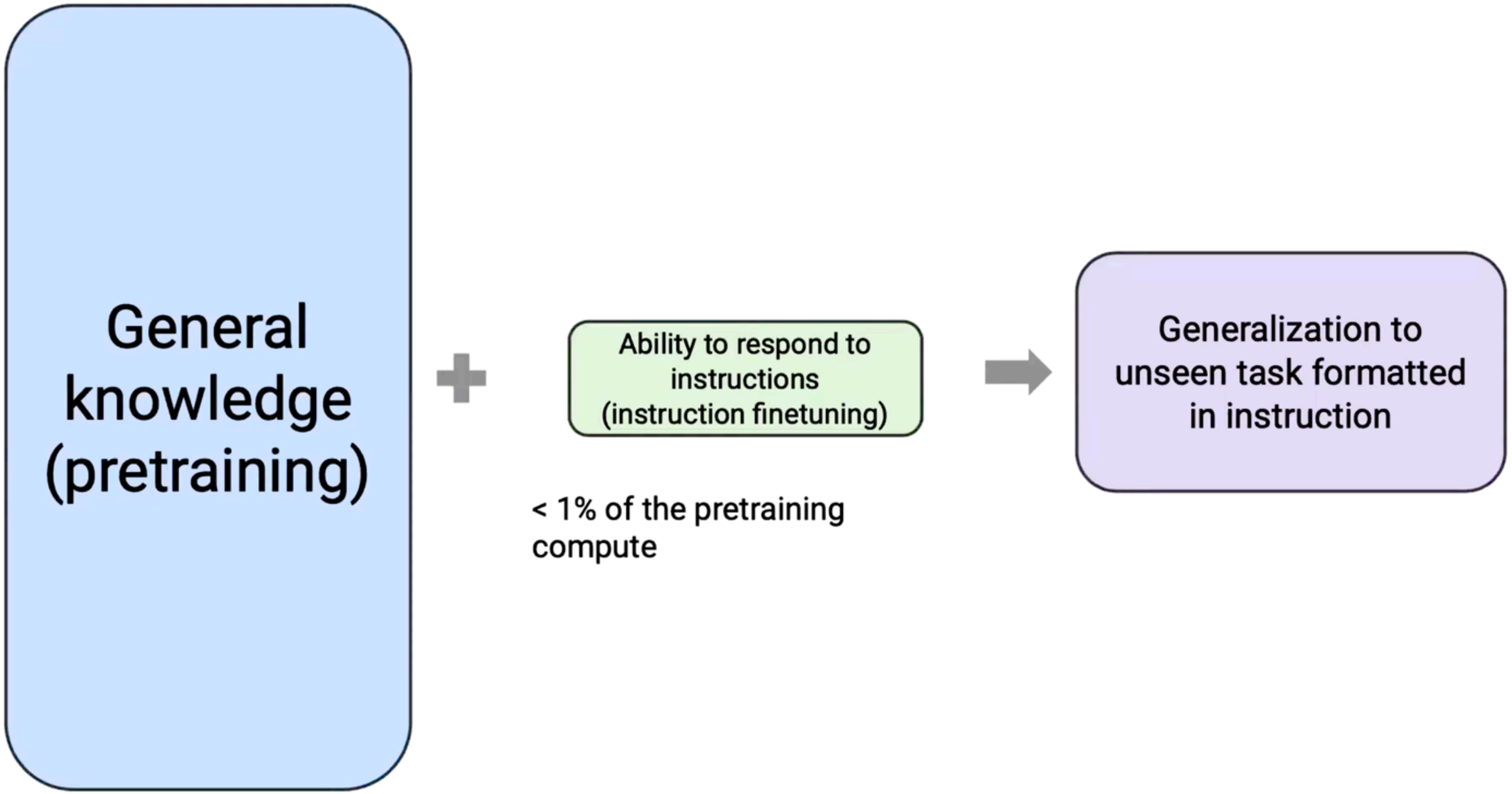
Tasks are unified. So for an unseen task, the model just needs to respond to the natural language instruction

*Wei et al. (2021), Sanh et al. (2021), Ouyang et al. (2022)

Overall scene as of now



Compute perspective



Alignment

Instruction fintuning: Align the models to follow instructions

Two “flavors” of instruction finetuning

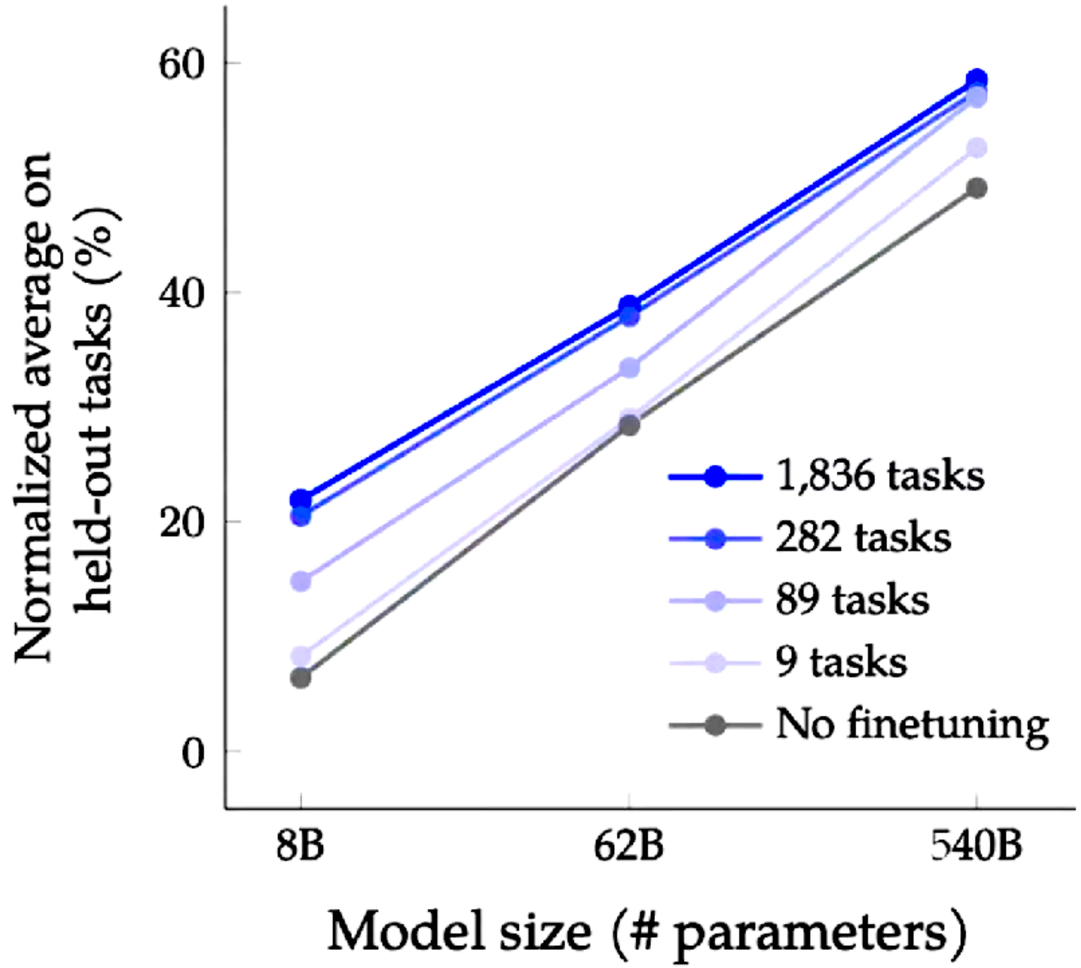
1. Instruction finetuning on a mixture of academic tasks
   a. Flan (Wei et al. 2021, Chung et al. 2021)
   b. T0 (Sanh et al. 2021)

2. Instruction finetuning on user prompts of language model APIs (e.g. GPT-3)
   a. InstructGPT (Ouyang et al. 2021)
   b. ChatGPT

Knobs of Instruction Finetuning

Number of tasks

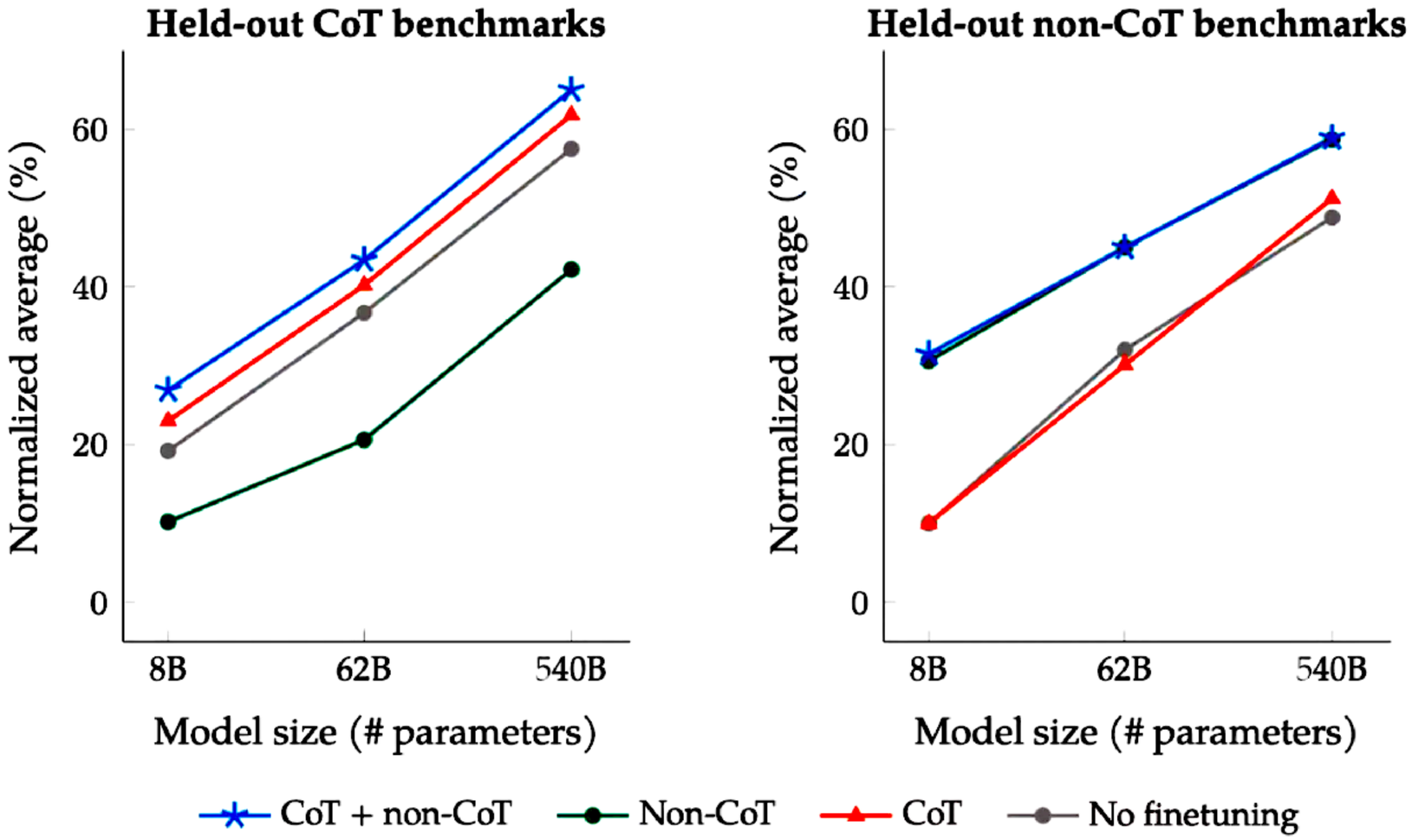
Task diversity



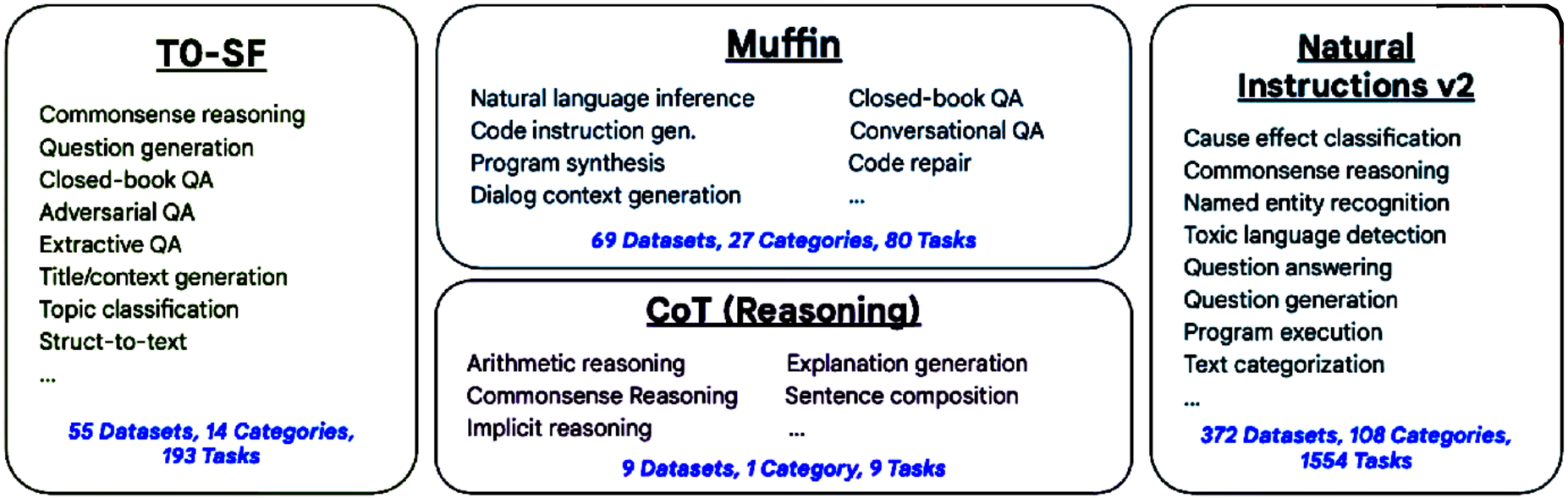
Knobs of Instruction Finetuning

Number of tasks

Task diversity



FLAN



- A **Dataset** is an original data source (e.g. SQuAD).
- A **Task Category** is unique task setup (e.g. the SQuAD dataset is configurable for multiple task categories such as extractive question answering, query generation, and context generation).
- A **Task** is a unique <dataset, task category> pair, with any number of templates which preserve the task category (e.g. query generation on the SQuAD dataset.)

Instruction finetuning on 1836 (!!) academic tasks

Chung et al. (2021)

Instruction Finetuning Benefits

Improve language understanding tasks

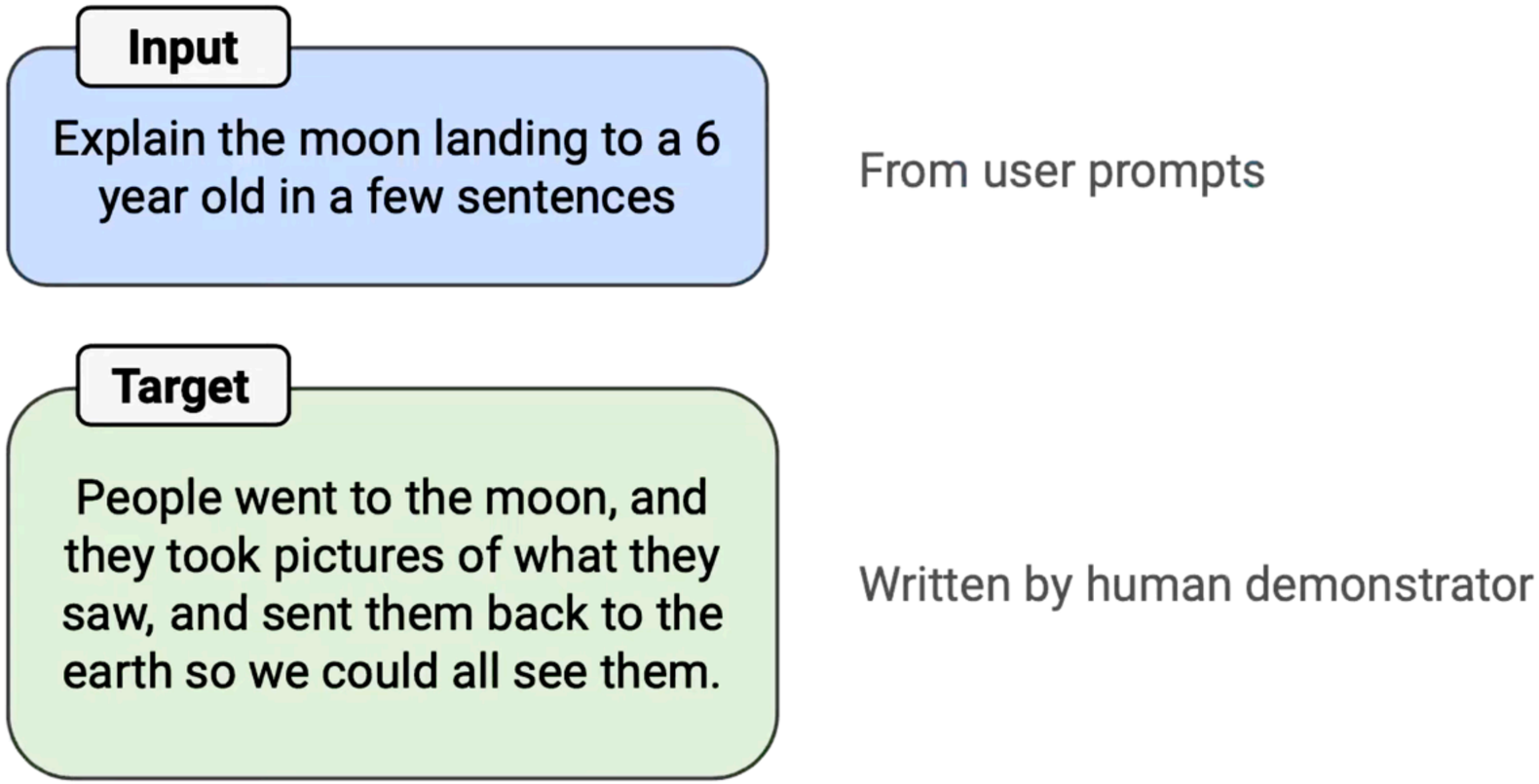
Cross-lingual transfer

Benefits wide range of model scales and types

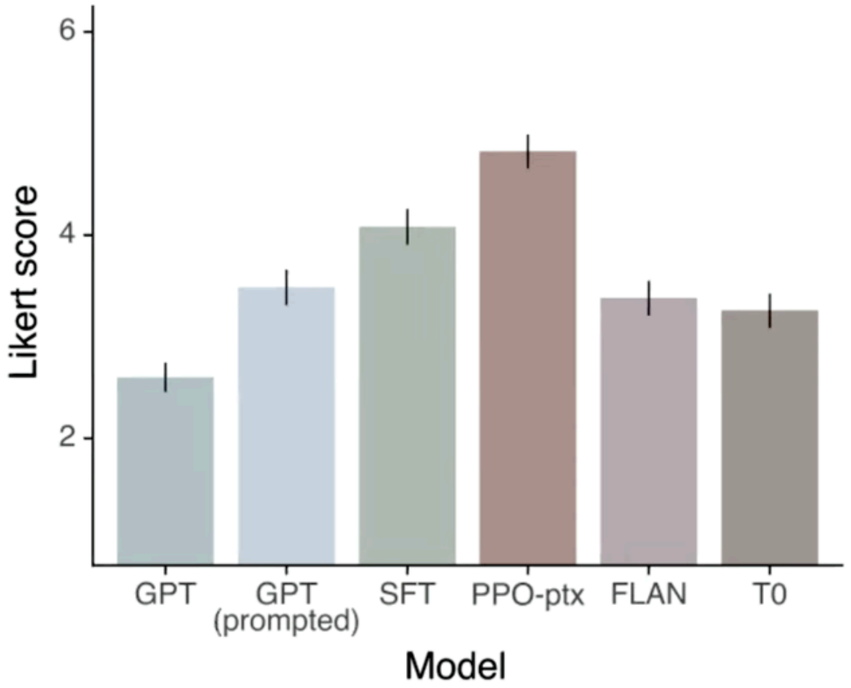
Provides a better initialization for “single-task” finetuning (Longpre et al. 2023)

What is the Problem?

InstructGPT (SFT)



SFT vs FLAN



SFT model trained on (user prompt, human demonstration) are preferred over FLAN (instruction finetuning on academic task) on user prompts

Ouyang et al. (2022)

Limitation of Instruction Finetuning

For a given input, the target is the single correct answer

In RL, this is called “behavior cloning”

Increasingly we want to teach models more abstract behaviors

Objective function of instruction finetuning seems to be the “bottleneck” of teaching these behaviors

The maximum likelihood objective is “predefined” function (i.e. no learnable parameter)

Can we parameterize the objective function and ***learn*** it?

RL to the rescue

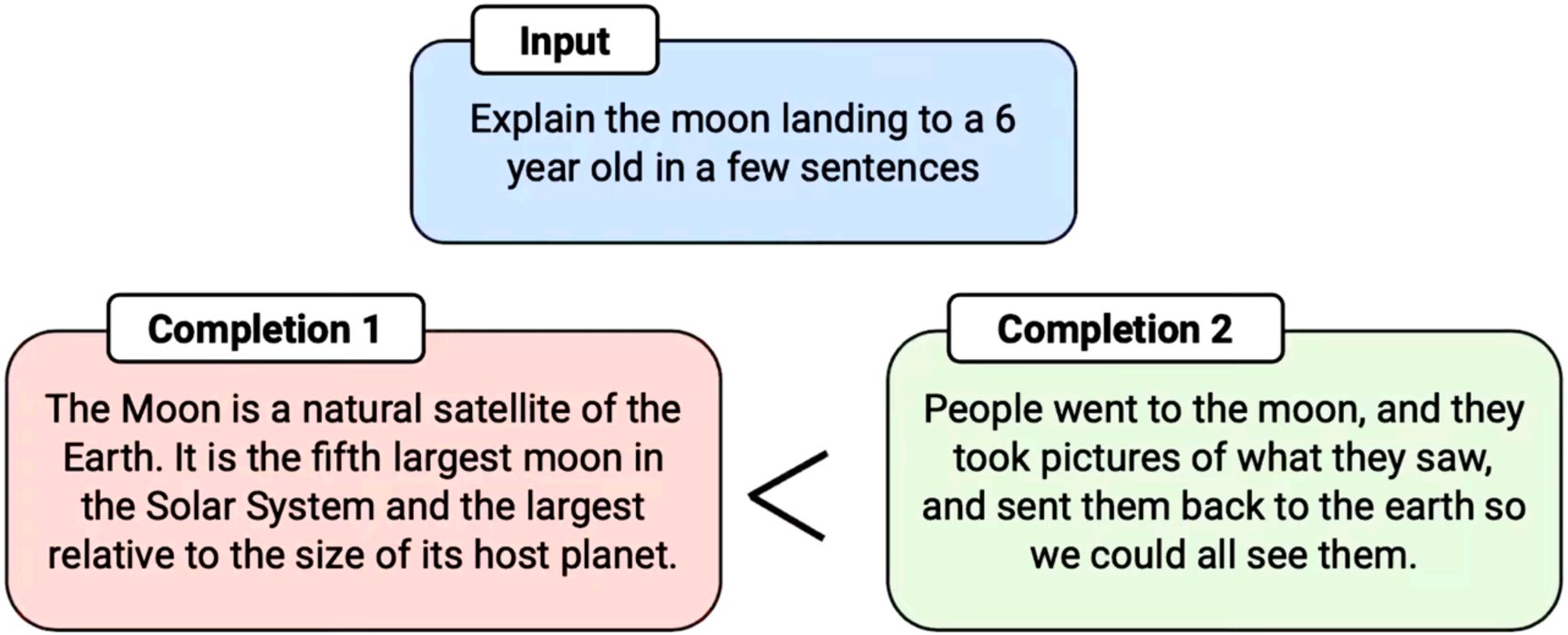
In RL, we try to maximize the expected reward function

$$\max_{\theta} \mathbb{E}[R]$$

Reward is the objective function. We can *learn* the reward: reward model.

We know how to do supervised learning with neural network well. Let's use neural net to represent the reward model.

Reward Model



Humans label which completion is preferred.

This setup aims to align models to the human preference

Scalar rewards?

Let $r(x, y; \phi)$ be the reward of completion y for a prompt x parameterized by ϕ

Remove the “unembedding” layer of the language model and let the model output scalar (e.g. add a linear projection layer) reward

Language model is used as an encoder, similar to BERT

Reward model objective

Let p_{ij} be the probability that completion y_i is better than completion y_j

Bradley–Terry model (1952): log odds that completion y_i is favored over y_j is modeled as difference in the rewards

$$\log \frac{p_{ij}}{1 - p_{ij}} = r(x, y_i; \phi) - r(x, y_j; \phi)$$

$$p_{ij} = \frac{e^{r(x,y_i;\phi) - r(x,y_j;\phi)}}{1 + e^{r(x,y_i;\phi) - r(x,y_j;\phi)}} = \sigma(r(x, y_i; \phi) - r(x, y_j; \phi))$$

$$\max_{\phi} \sum_{x,y_i,y_j \in D} \log p_{ij}$$

Observation

It doesn’t model *how much better* one completion is than the other.

It just models says how confident it is in one completion being better than another.

Potentially lots of room for improvements

Policy Model Objective

Once we have a reward model, we can use it in RL to learn the language model parameters that maximizes the expected reward

$$J(\theta) = \mathbb{E}_{(X,Y)\sim D_{\pi_\theta}}[r(X, Y; \phi)]$$

where $X = (X_1, \ldots, X_S)$ is the prompt and $Y = (Y_1, \ldots, Y_T)$ is the completion sampled from the policy model.

The optimization problem is then

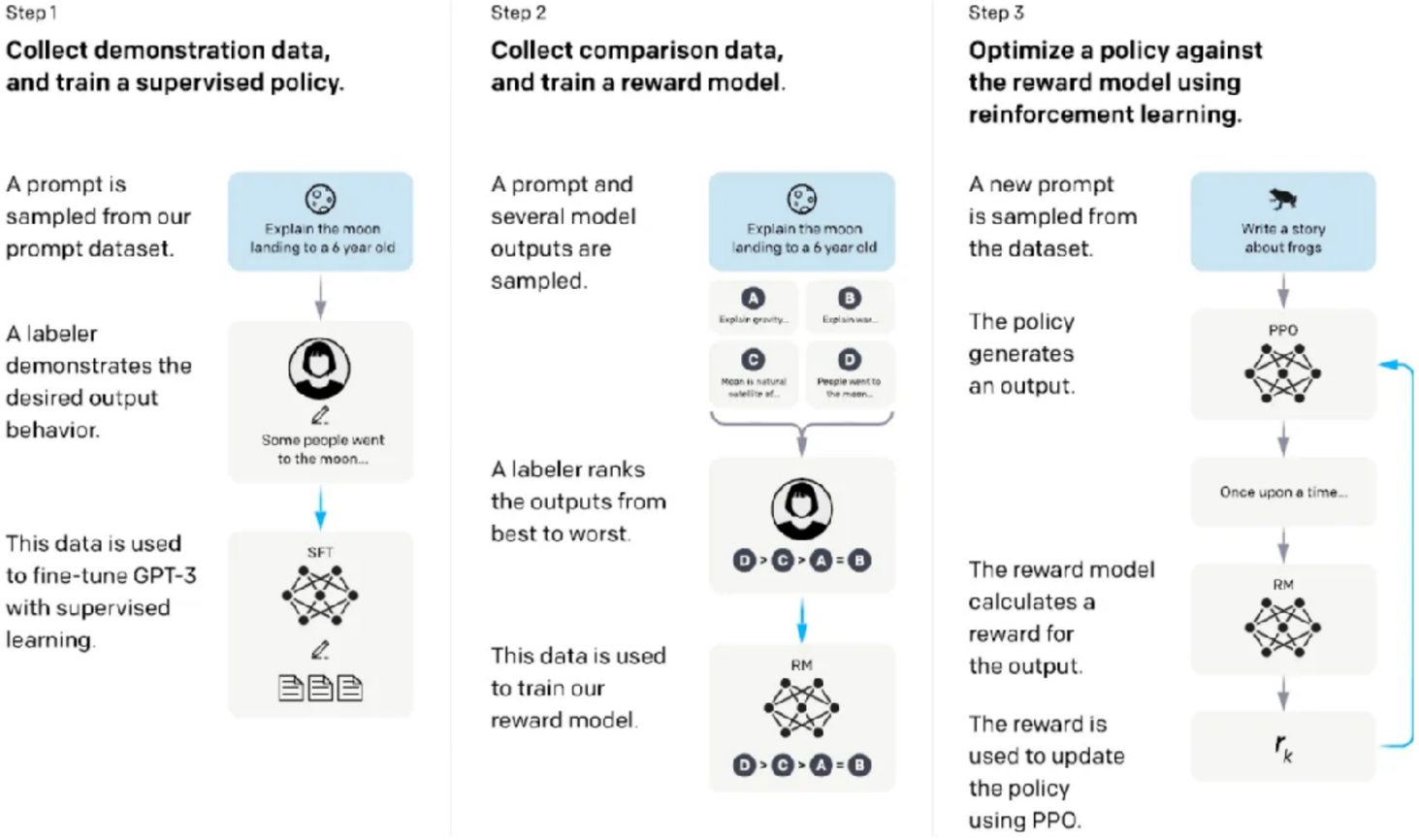
$$\max_\theta J(\theta) = \max_\theta \mathbb{E}_{(X,Y)\sim D_{\pi_\theta}}[r(X, Y; \phi)]$$

We use iterative algorithm such as gradient ascent to solve this

$$\theta := \theta + \alpha \nabla J(\theta)$$

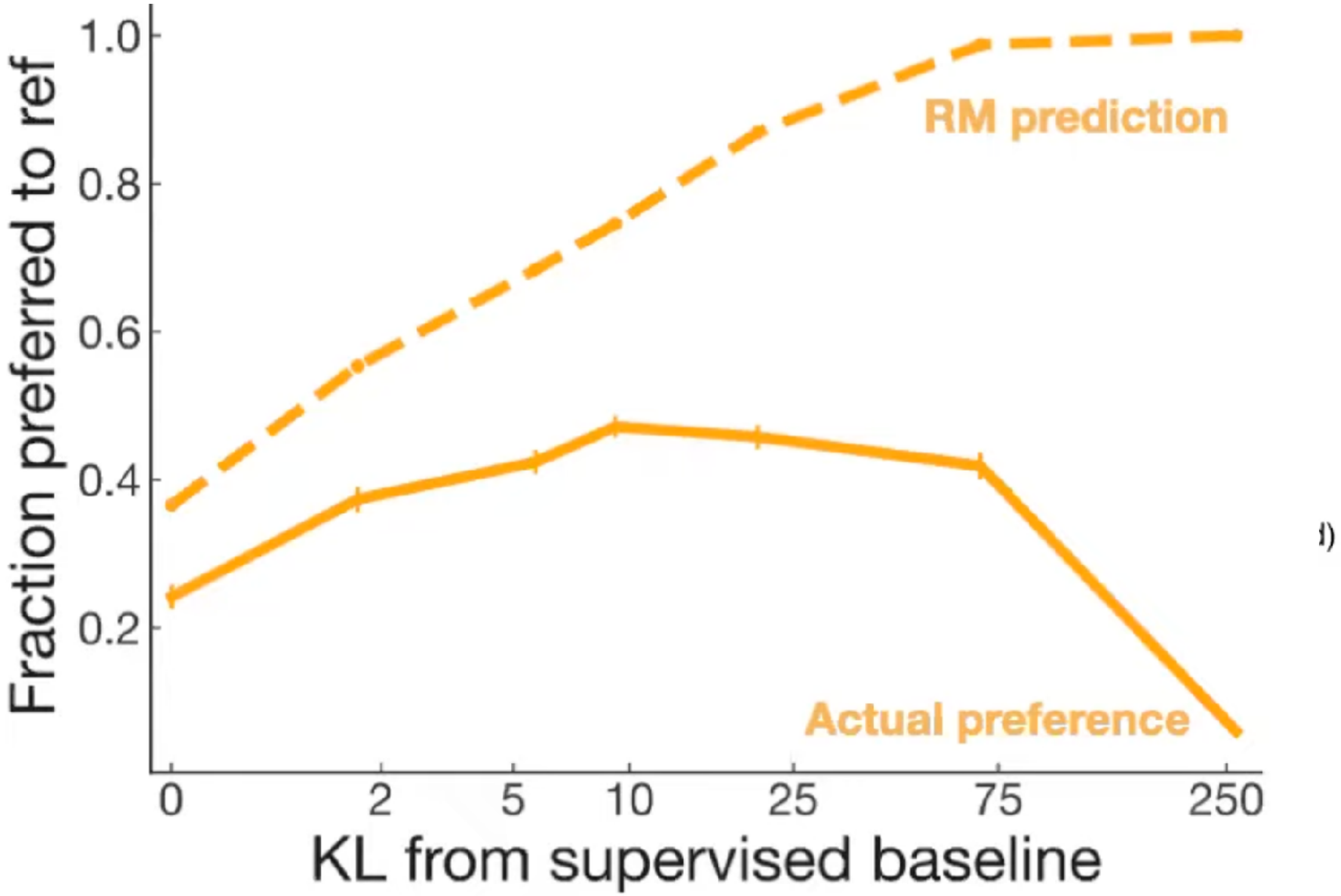
We can use policy gradient algorithms such as PPO to compute the gradient.

Policy Gradient Training



- A **pretrained language model**, i.e. GPT-3 here, is used as starting point. The **steps mainly follow Human Feedback Model.**
- **Step 1: Collect demonstration data**, and **train a supervised policy**. The labelers provide demonstrations of the desired behavior on the input prompt distribution. Then, a **pretrained GPT-3** model is **fine-tuned** on this data using **supervised learning.**
- **Step 2: Collect comparison data**, and **train a reward model.** A dataset of comparisons between model outputs is collected, where labelers indicate which output they prefer for a given input. Then, a reward model is trained to **predict the human-preferred output.**
- **Step 3: A policy is optimized against the reward model using PPO.** The **output of the RM** is used as a **scalar reward.** The supervised policy is **fine-tuned to optimize this reward** using the PPO algorithm.
- **Steps 2 and 3 can be iterated continuously;** more comparison data is collected on the current best policy, which is used to train a new RM and then a new policy.

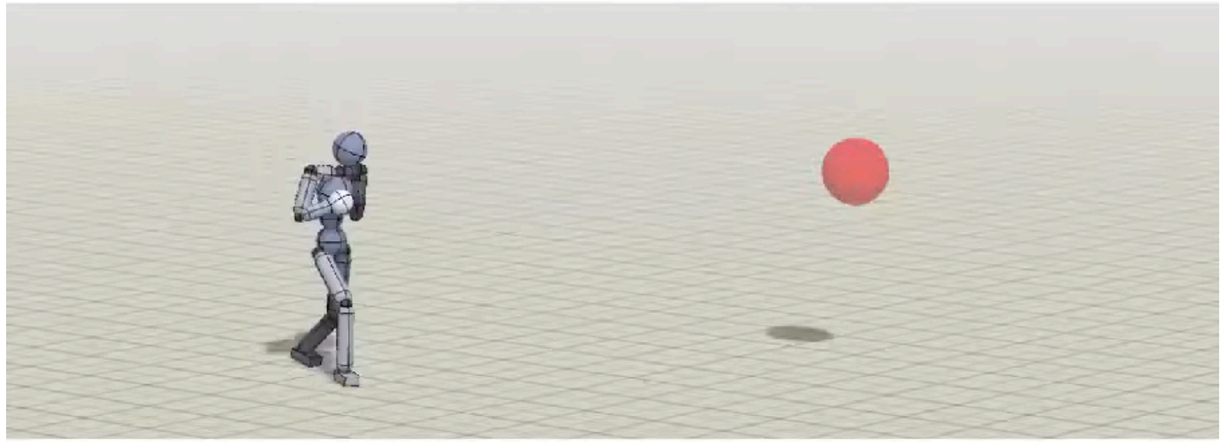
KL Penalty to prevent over optimisation



Example of over-optimisation

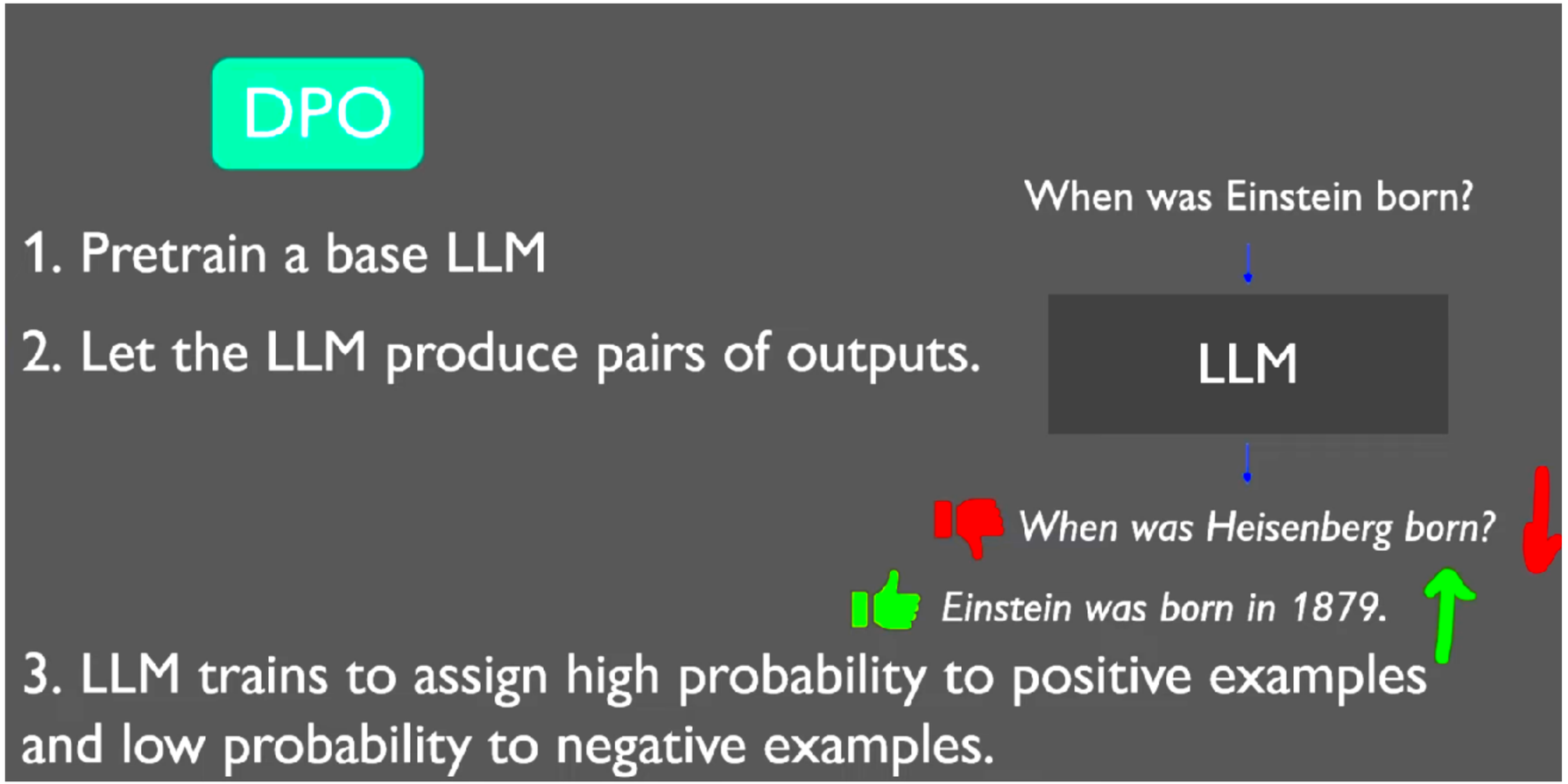
https://openai.com/research/faulty-reward-functions

Humanoid: Baseball Pitch - Throw



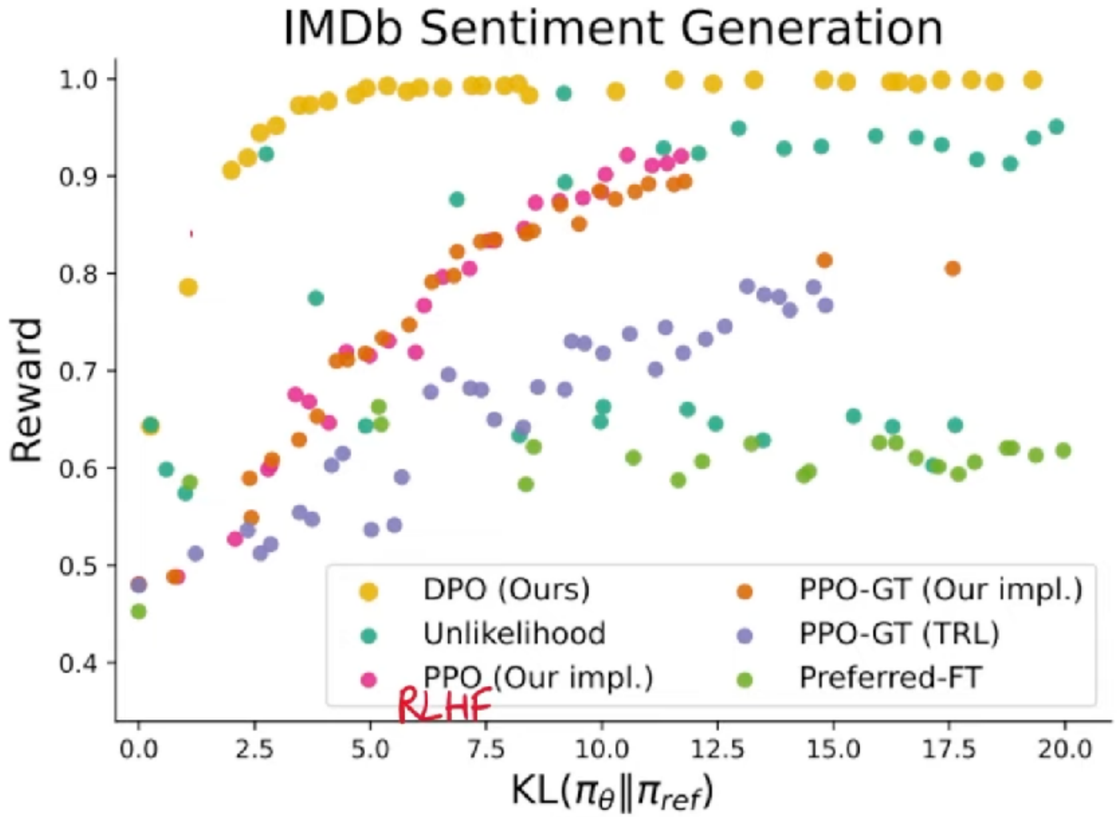
Throwing a ball to a target.

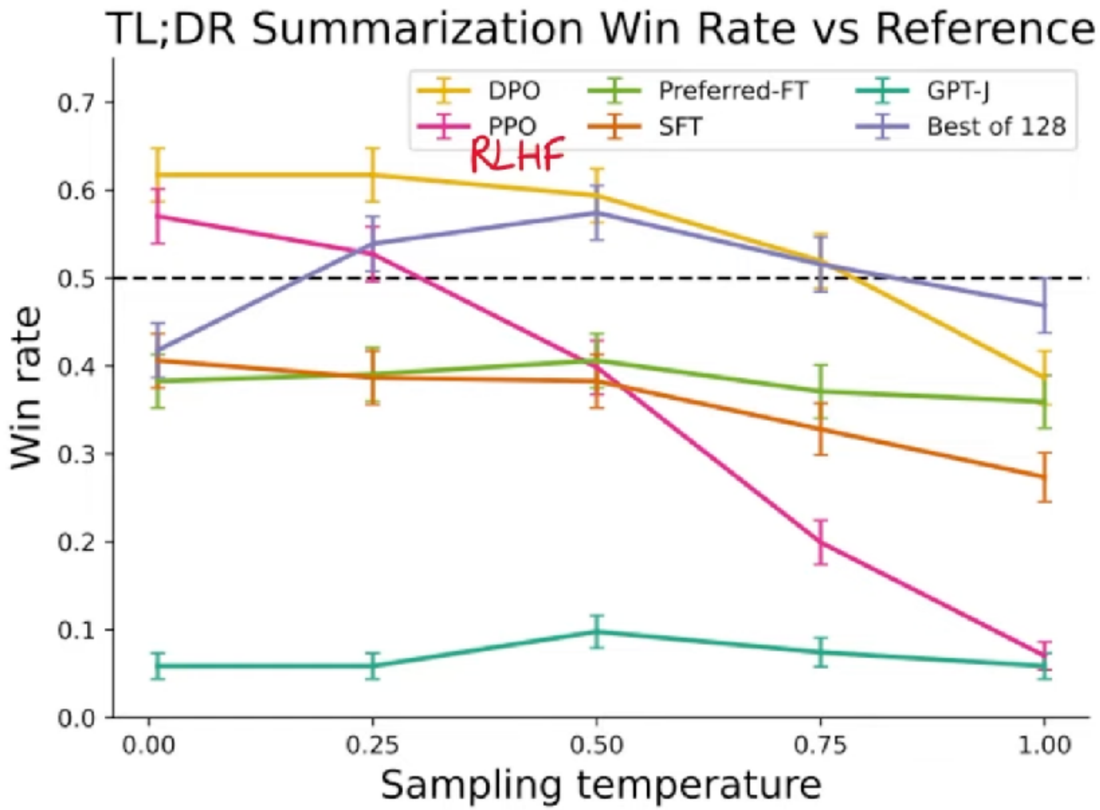
DPO: Direct Preference Optimisation



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

DPO: Direct Preference Optimisation





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