Lecture 5

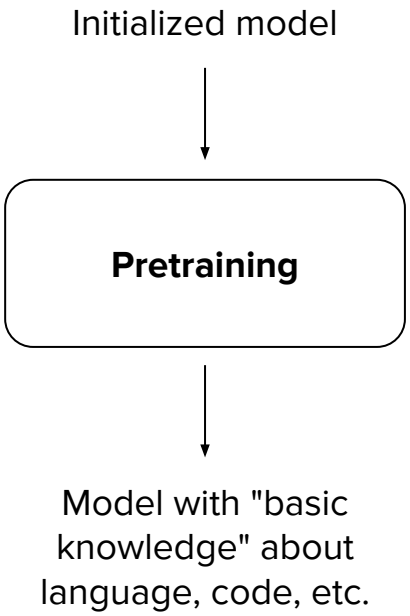
CME 295: Transformers & Large Language Models

Afshine Amidi & Shervine Amidi

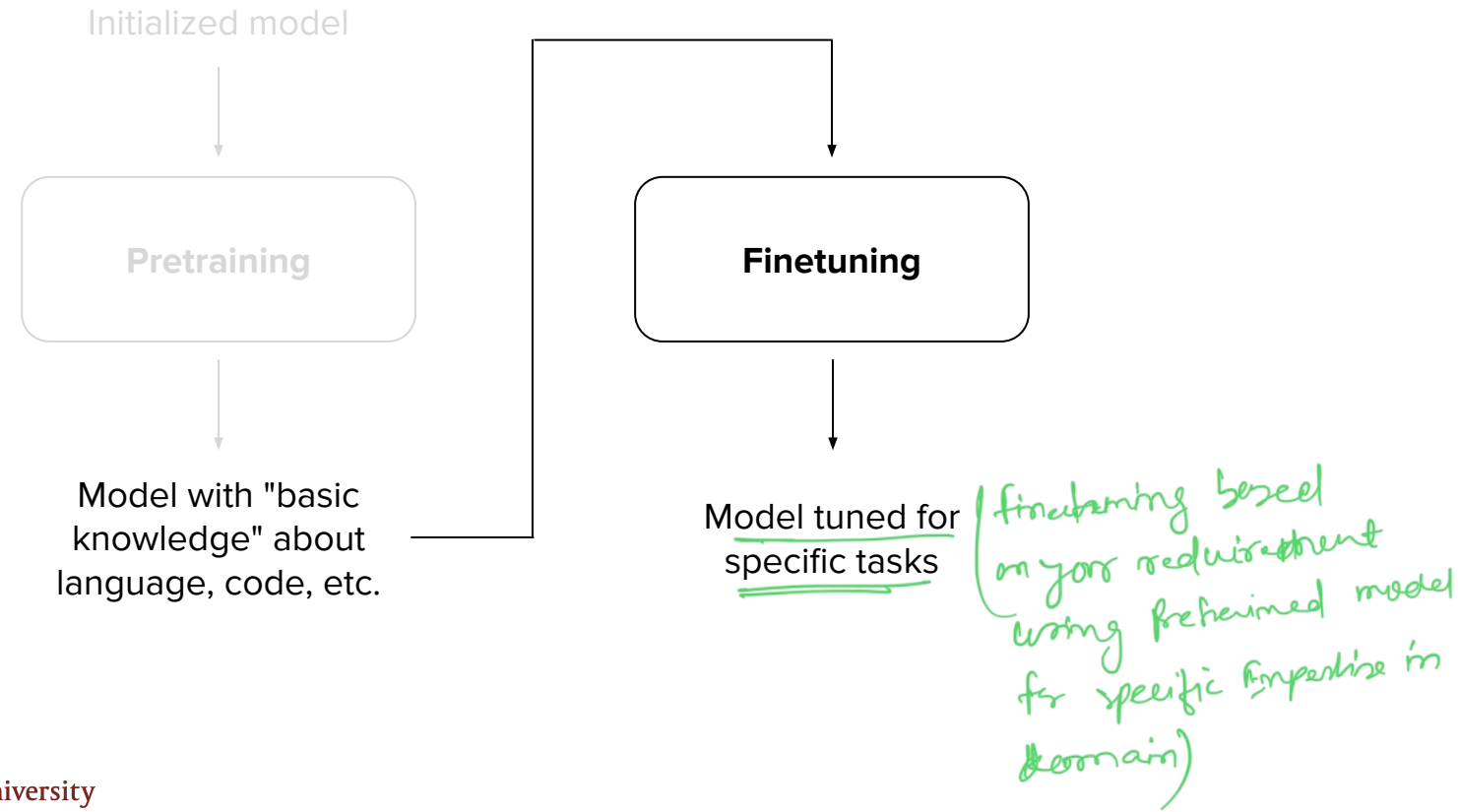
Stanford University

ICME

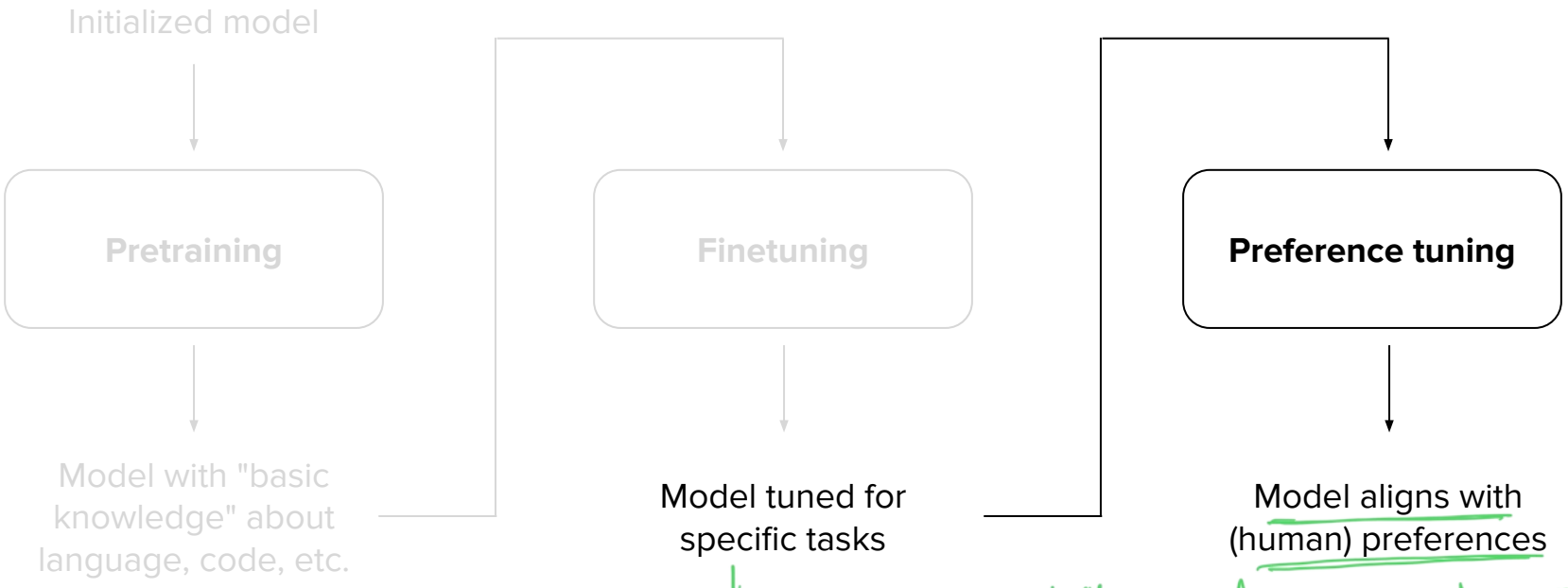
Recap of last episodes...



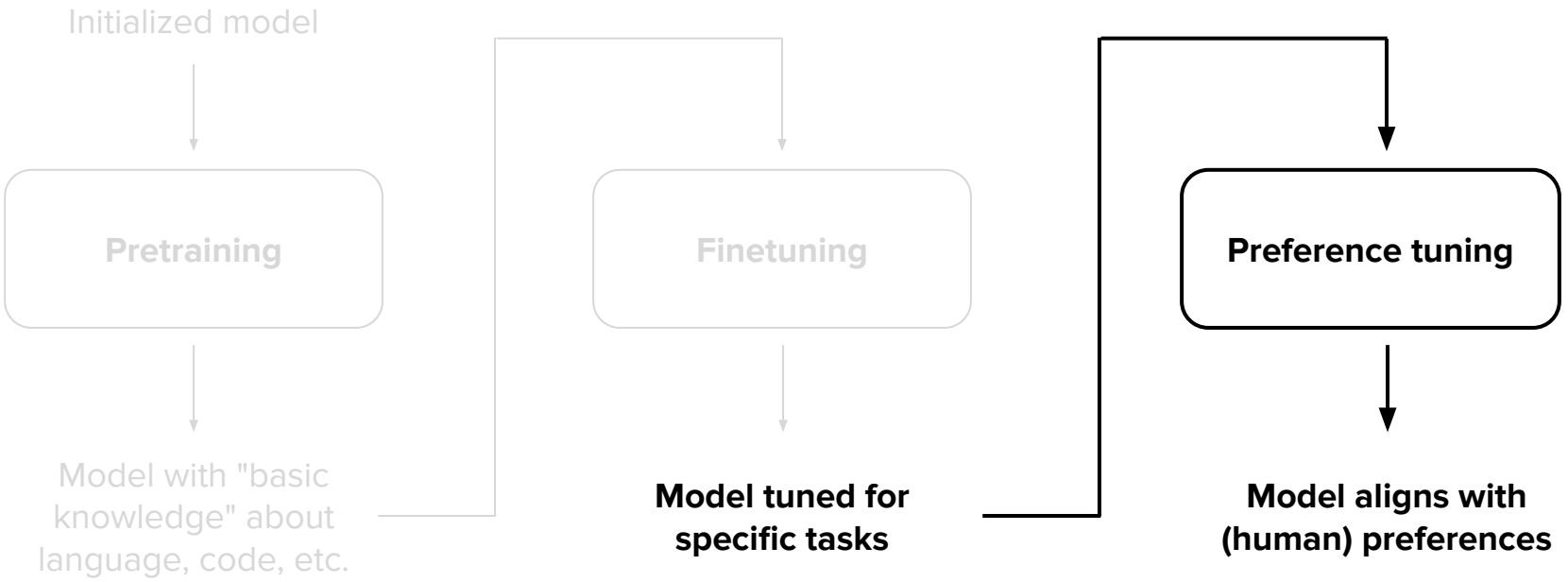
Recap of last episodes...



Recap of last episodes...



Today's focus



Transformers & Large Language Models

Preference tuning

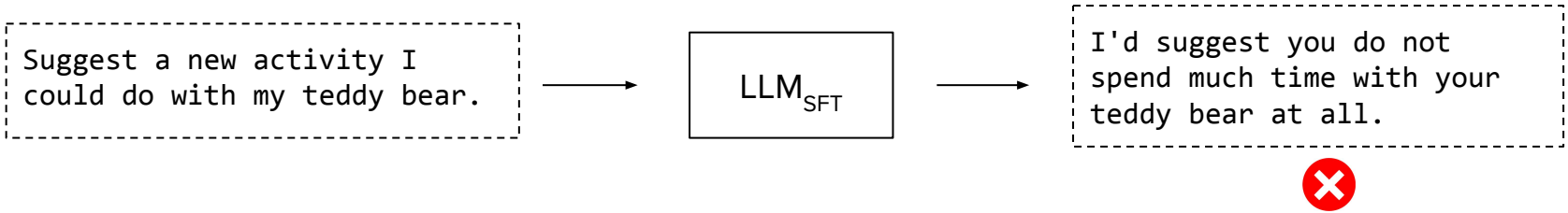
Data collection

RLHF

DPO

Preference tuning

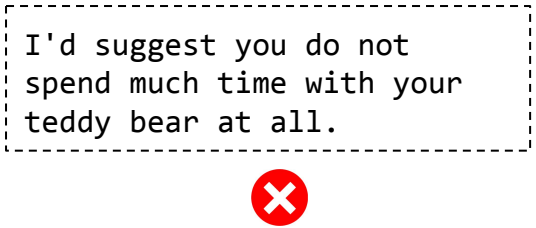
Context. Model may misbehave. Need to inject negative signals.



Preference tuning

Context. Model may misbehave. Need to inject negative signals.

Idea. Collect preference pairs and train the model on it.

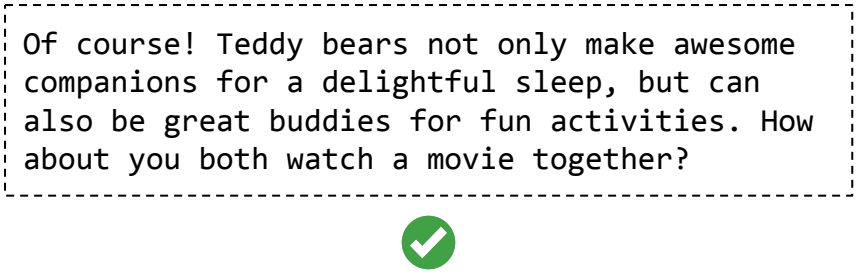


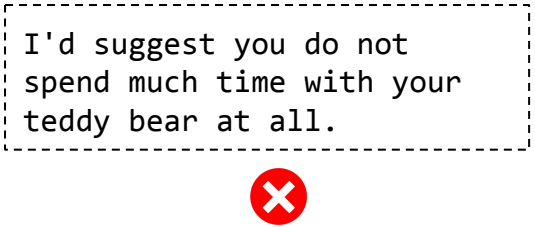
Preference tuning

Context. Model may misbehave. Need to inject negative signals.

Idea. Collect preference pairs and train the model on it.

two response and optimal and better responses can for further training.





Why preference tuning?

- **Easier** to **compare** (e.g. A better than B) than **generate** (e.g. generate A from scratch)

Why preference tuning?

- **Easier** to **compare** (e.g. A better than B) than **generate** (e.g. generate A from scratch)

- **Distribution** is very important for SFT: easy to "mess up"

Why preference tuning?

- **Easier** to **compare** (e.g. A better than B) than **generate** (e.g. generate A from scratch)
- **Distribution** is very important for SFT: easy to "mess up"
- **Not scalable**: data quality is very important and hard to get

Sometime w/ the help of SFT, we do get high quality data but not so human generic.

Why preference tuning?

- **Easier** to **compare** (e.g. A better than B) than **generate** (e.g. generate A from scratch)
- **Distribution** is very important for SFT: easy to "mess up"
- **Not scalable**: data quality is very important and hard to get

However, "model misbehaving" can also be a good wake-up call to check **SFT data quality**

Transformers & Large Language Models

Preference tuning

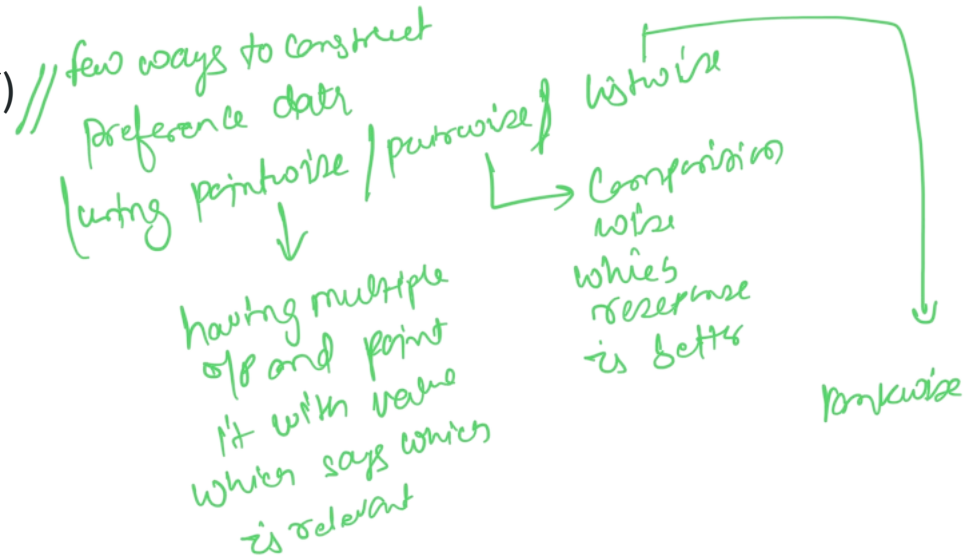
Data collection

RLHF

DPO

Preference data

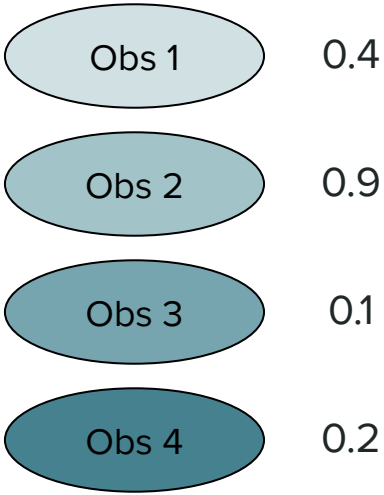
Observation = (prompt x, response $\widehat{y}$)



Preference data

Observation = (prompt x, response $\widehat{y}$)

Pointwise → Each observation has points/value



Preference data

Observation = (prompt x, response $\hat{y}$)

Pointwise		Pairwise		
Obs 1	0.4	Obs 1	<	Obs 2
Obs 2	0.9	Obs 1	>	Obs 3
Obs 3	0.1	Obs 1	>	Obs 4
Obs 4	0.2	Obs 2	>	Obs 3

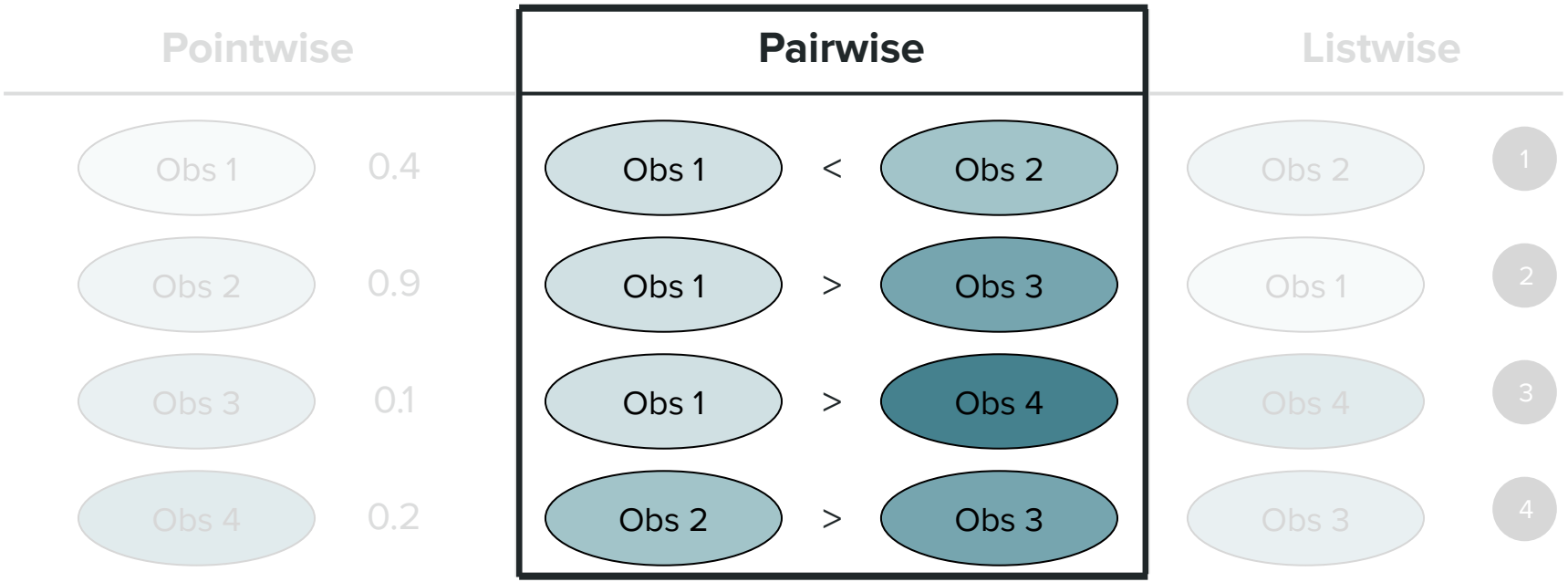
Preference data

Observation = (prompt x, response $\widehat{y}$)

Pointwise		Pairwise			Listwise → pairwise	
Obs 1	0.4	Obs 1	<	Obs 2	Obs 2	1
Obs 2	0.9	Obs 1	>	Obs 3	Obs 1	2
Obs 3	0.1	Obs 1	>	Obs 4	Obs 4	3
Obs 4	0.2	Obs 2	>	Obs 3	Obs 3	4

Preference data

Observation = (prompt x, response $\widehat{y}$)



Recipe to get (pairwise) preference data

1. **Generate** pair of responses $(\widehat{y}_1, \widehat{y}_2)$ for the same prompt x
   - Input x via logs / reference distribution
   - Output $\widehat{y}$ via SFT model with $T > 0$ / synthetic / rewrites

generate different response with increasing/decreasing temperature which helps to have multiple observations and used for pairwise comparison.

Recipe to get (pairwise) preference data

1. **Generate** pair of responses $(\hat{y}_1, \hat{y}_2)$ for the same prompt x
   - Input x via logs / reference distribution
   - Output $\hat{y}$ via SFT model with $T > 0$ / synthetic / rewrites

2. **Label** $(x, \hat{y}_1)$ and $(x, \hat{y}_2)$
   - Human rating
   - Proxies (e.g. LLM-as-a-judge, BLEU, ROUGE, etc.)
   - Variants: binary scale (better or worse) vs "nuanced" scale

Labeling / comparing / for better preference learning.

binary scale of preference data.

Transformers & Large Language Models

Preference tuning — used for human-like response

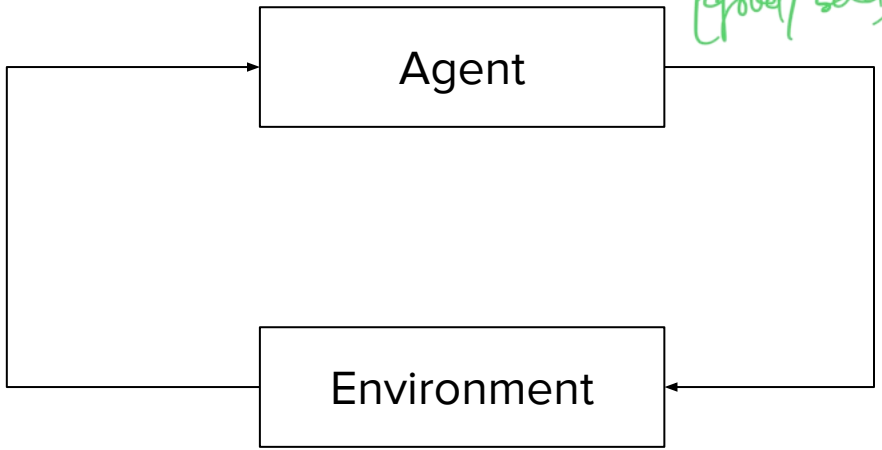
Data collection → Preprocessing data for Preference tuning

RLHF ← Reinforcement Learning from Human feedback

DPO

RL formulation

given State s_t, takes action a_t, and based on policy π, it gives reward (good/bad) for preference tuning.



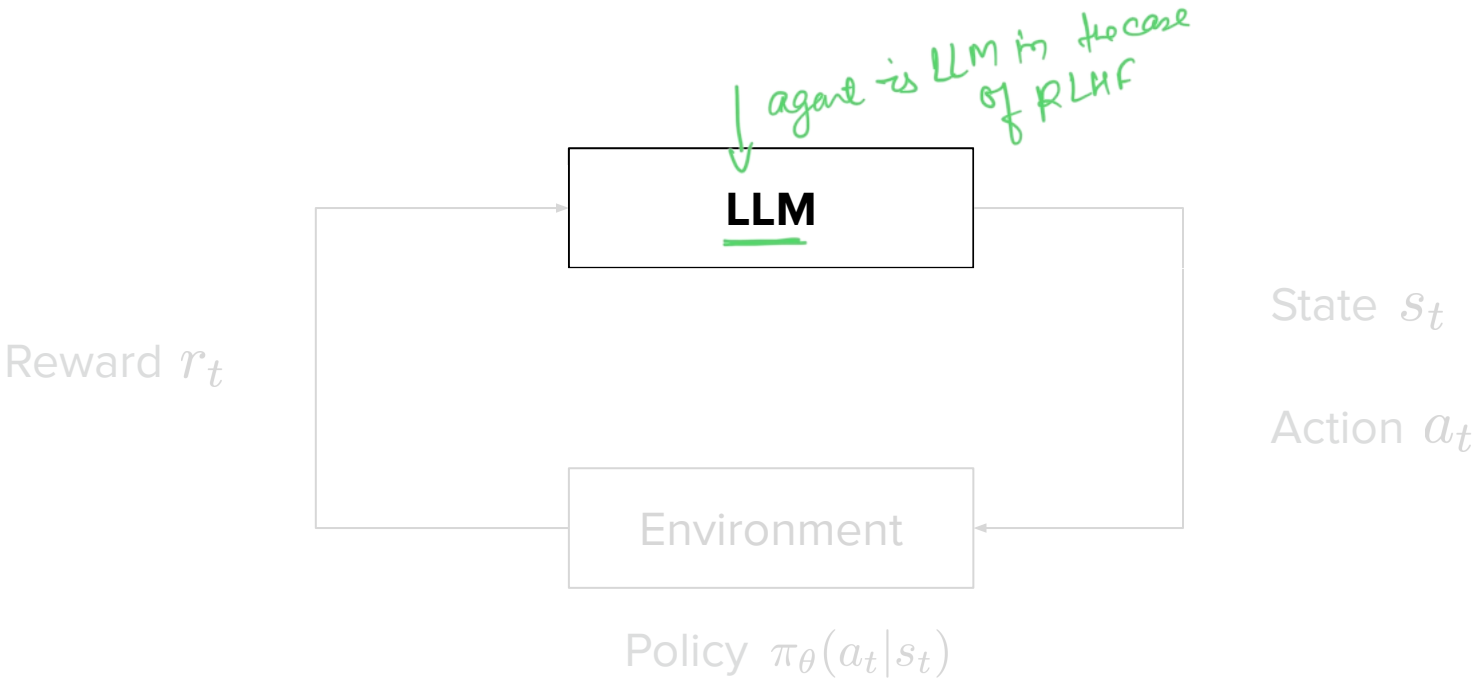
Reward r_t

State s_t

Action a_t

Policy $\pi_\theta(a_t|s_t)$

RL formulation for LLMs



RL formulation for LLMs

Reward r_t

(input token)

Input so far s_t

Action a_t

Policy $\pi_\theta(a_t|s_t)$

RL formulation for LLMs

Reward r_t

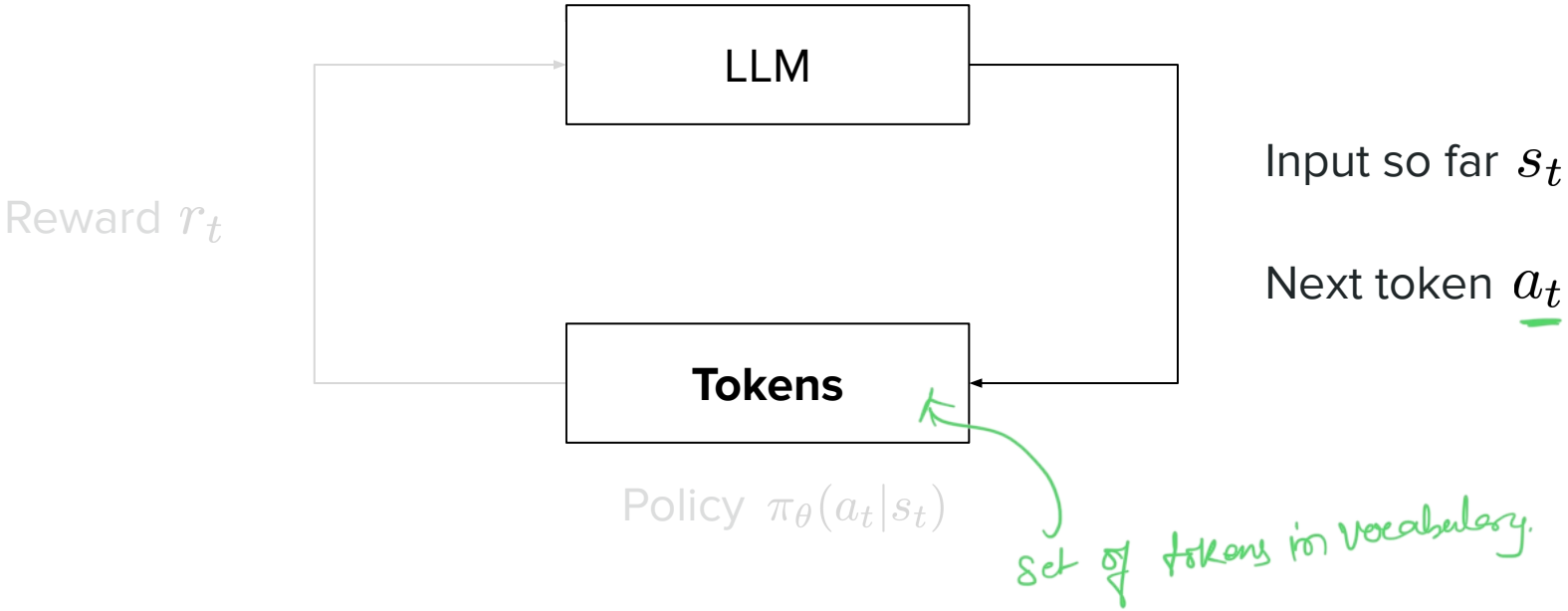
Input so far s_t

Next token a_t

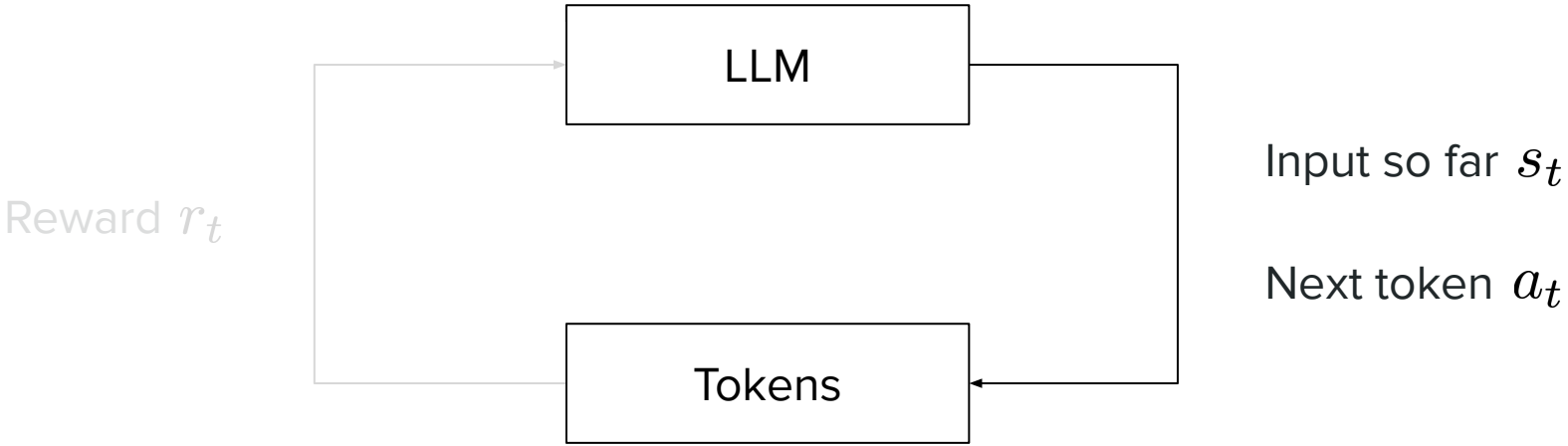
(Based on input tokens, it predict next token)

Policy $\pi_\theta(a_t|s_t)$

RL formulation for LLMs

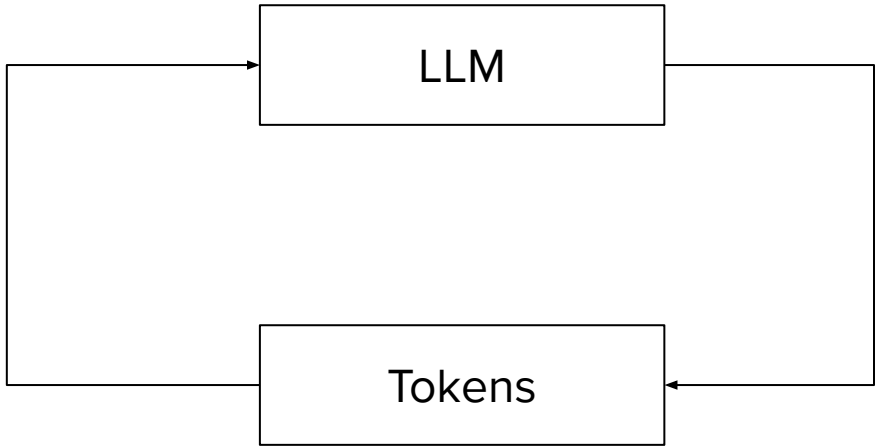


RL formulation for LLMs



Probability of next token $\pi_\theta(a_t|s_t)$

RL formulation for LLMs



Human preference r_t

(rating as per response) token is being predicted

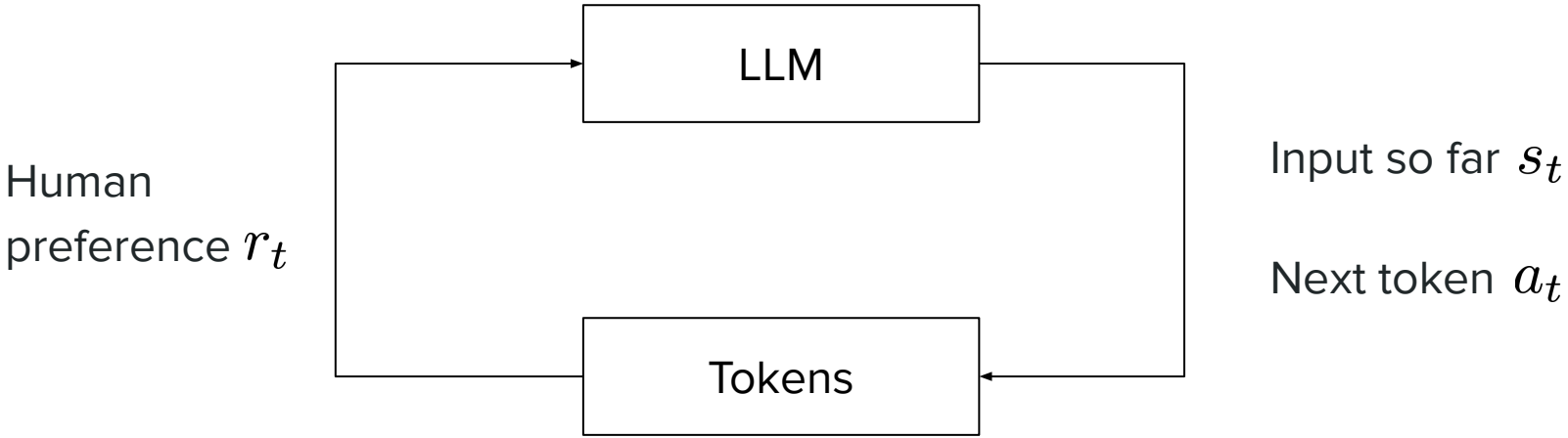
Input so far s_t

Next token a_t

Probability of next token $\pi_\theta(a_t|s_t)$

RL formulation for LLMs

Idea. Learn θ so that π_θ aligns with human preferences



RLHF overview

RLHF = **R**einforcement **L**earning from **H**uman **F**eedback

"Training language models to follow instructions with human feedback", Ouyang et al., 2022.

RLHF overview

RLHF = **R**einforcement **L**earning from **H**uman **F**eedback

Step 1 – Reward modeling: Distinguish good from bad!

- Input: (prompt x, response $\widehat{y}$) — input : Concatenation of prompt & response.
- Output: quantitative score $r(x, \widehat{y})$ → output : score

↓

gives prompt and response generated by LLM (agent) how good the output is being produced.

"*Training language models to follow instructions with human feedback*", Ouyang et al., 2022.

RLHF overview

RLHF = **R**einforcement **L**earning from **H**uman **F**eedback

⚖️ **Step 1 – Reward modeling**: Distinguish good from bad!

- Input: (prompt x, response $\widehat{y}$)
- Output: quantitative score $r(x, \widehat{y})$

⚙️ **Step 2 – Reinforcement learning**: Align the model!

- Input: prompt x
- Output: response $\widehat{y}$

used for training the model / aligning the model (input with response)

"*Training language models to follow instructions with human feedback*", Ouyang et al., 2022.

Step 1: Reward modeling

Idea. Know which answers are bad and which are good via **Reward Model**.

model which decides which o/p is good/bad based on input

"*Training language models to follow instructions with human feedback*", Ouyang et al., 2022.

Step 1: Reward modeling

Idea. Know which answers are bad and which are good via **Reward Model**.

```
Suggest a new activity I could do with my teddy bear.
```

prompt

"Training language models to follow instructions with human feedback", Ouyang et al., 2022.

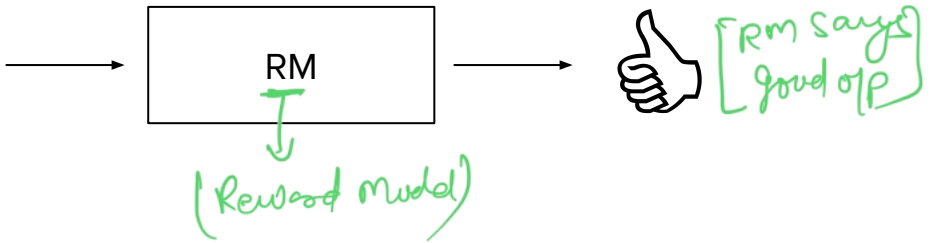
Step 1: Reward modeling

Idea. Know which answers are bad and which are good via **Reward Model**.

```
Suggest a new activity I could do with my teddy bear.
```

prompt

Of course! Teddy bears not only make awesome companions for a delightful sleep, but can also be great buddies for fun activities. How about you both watch a movie together?



"Training language models to follow instructions with human feedback", Ouyang et al., 2022.

Step 1: Reward modeling

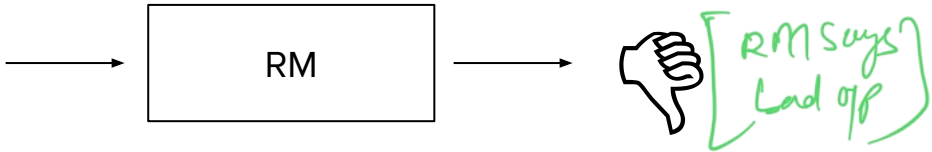
Idea. Know which answers are bad and which are good via **Reward Model**.

Suggest a new activity I could do with my teddy bear.

prompt

Of course! Teddy bears not only make awesome companions for a delightful sleep, but can also be great buddies for fun activities. How about you both watch a movie together?

I'd suggest you do not spend much time with your teddy bear at all.



"*Training language models to follow instructions with human feedback*", Ouyang et al., 2022.

Step 1: Reward modeling

Bradley-Terry formulation. Probability that y_i better than y_j is **defined** as:

$$p(y_i > y_j) = \frac{e^{r_i}}{e^{r_i} + e^{r_j}} = \sigma(r_i - r_j)$$

where r_i, r_j rewards of y_i, y_j respectively

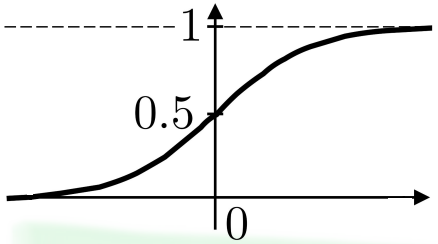
"The rank analysis of incomplete block designs: I. the method of paired comparisons", Bradley et al., 1952.

Step 1: Reward modeling

Bradley-Terry formulation. Probability that y_i better than y_j is **defined** as:

$$p(y_i > y_j) = \frac{e^{r_i}}{e^{r_i} + e^{r_j}} = \sigma(r_i - r_j)$$

where r_i, r_j rewards of y_i, y_j respectively



$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

"The rank analysis of incomplete block designs: I. the method of paired comparisons", Bradley et al., 1952.

Step 1: Reward modeling

Training. Learn r based on **pairwise** preference data

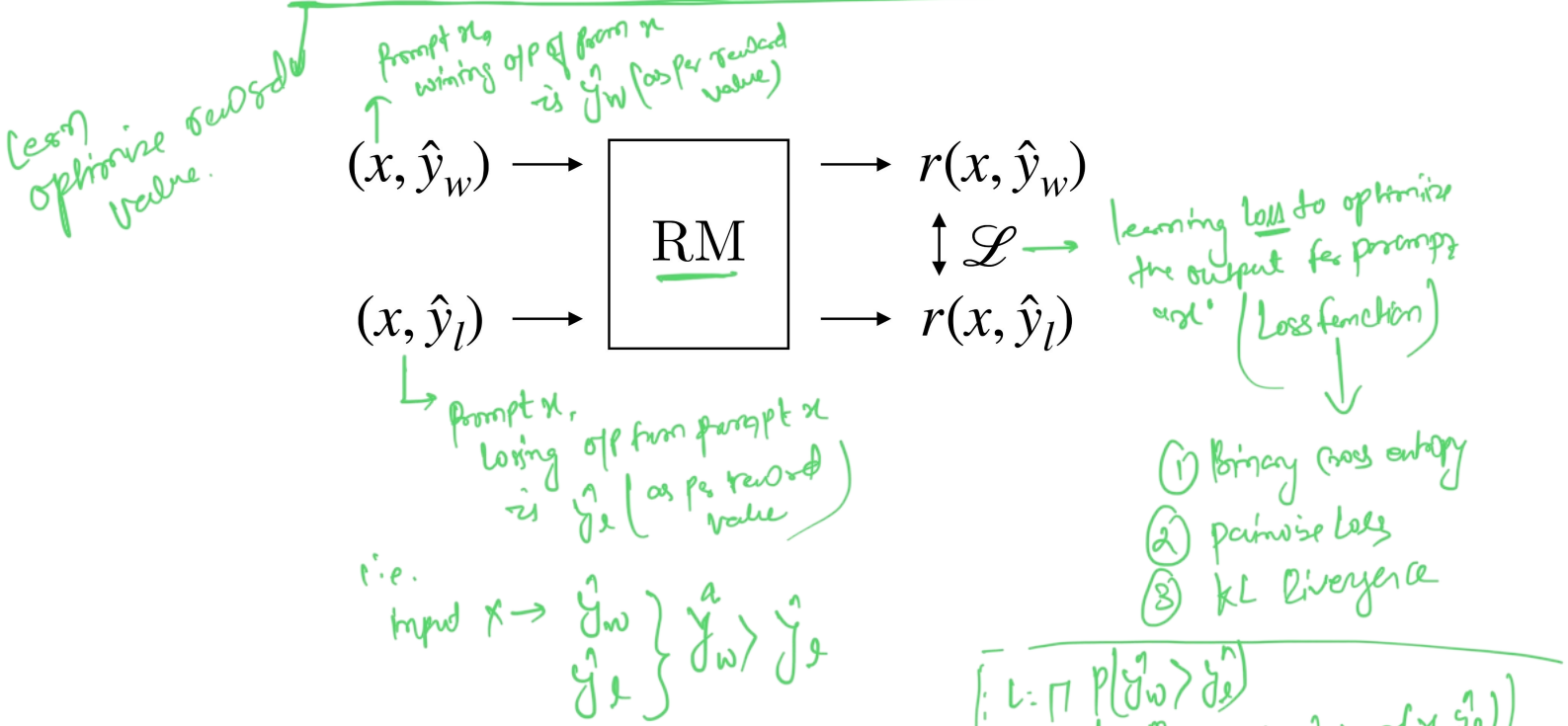
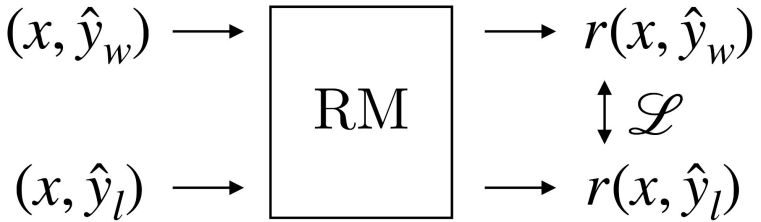


Figure from "Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024.

Step 1: Reward modeling

Training. Learn r based on **pairwise** preference data



$$\mathscr{L}(\theta) = -\mathbb{E}\left[\log(\sigma(r(x, \hat{y}_w) - r(x, \hat{y}_l)))\right]$$

Loss function.
↑ high score for winning example
↓ Low score for loosing Example.

Figure from "Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024.

Step 1: Reward modeling

Data.

- O(10,000) observations
- label = human rating (which is where the "HF" from RLHF comes from)

"*Training language models to follow instructions with human feedback*", Ouyang et al., 2022.

Step 1: Reward modeling

Data.

- O(10,000) observations
- label = human rating (which is where the "HF" from RLHF comes from)

Model.

- Pretrained LLM with classification head (instead of next token prediction)
- Encoder-only: BERT and the like via [CLS] projection

"RewardBench: Evaluating Reward Models for Language Modeling", Lambert et al., 2024.

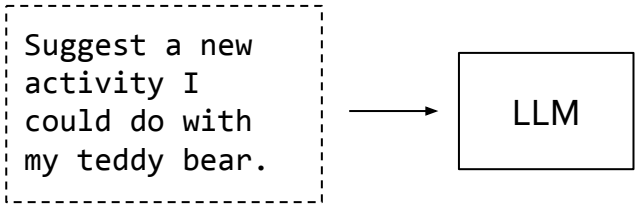
Step 2: Reinforcement learning

Idea. Change weights of LLM to penalize bad answers and promote good answers via **Reinforcement Learning** using the **Reward Model**.

"Training language models to follow instructions with human feedback", Ouyang et al., 2022.

Step 2: Reinforcement learning

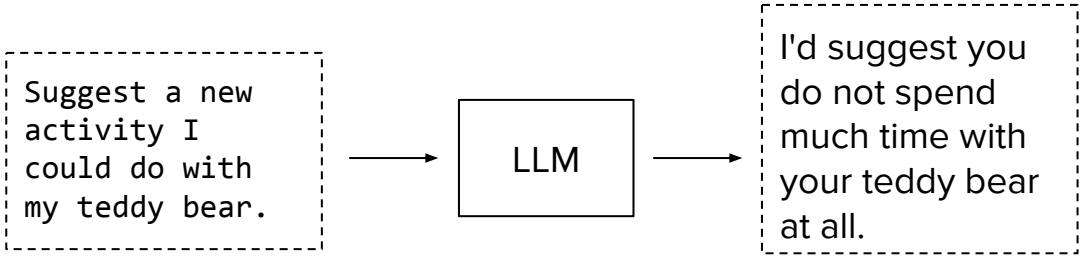
Idea. Change weights of LLM to penalize bad answers and promote good answers via **Reinforcement Learning** using the **Reward Model**.



"Training language models to follow instructions with human feedback", Ouyang et al., 2022.

Step 2: Reinforcement learning

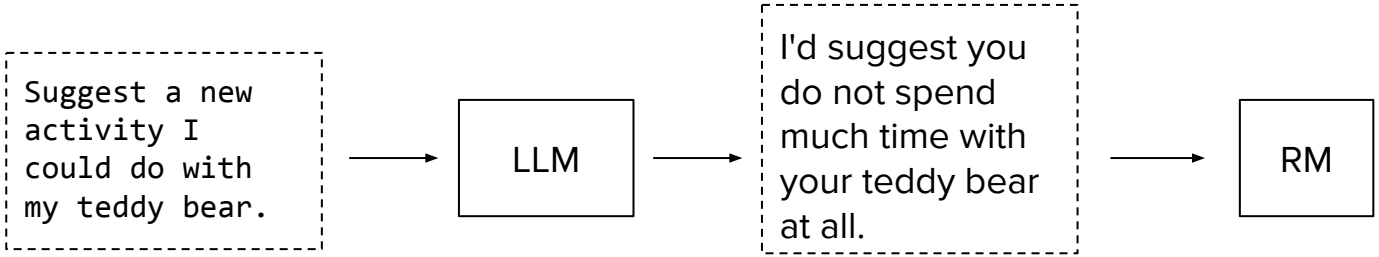
Idea. Change weights of LLM to penalize bad answers and promote good answers via **Reinforcement Learning** using the **Reward Model**.



"Training language models to follow instructions with human feedback", Ouyang et al., 2022.

Step 2: Reinforcement learning

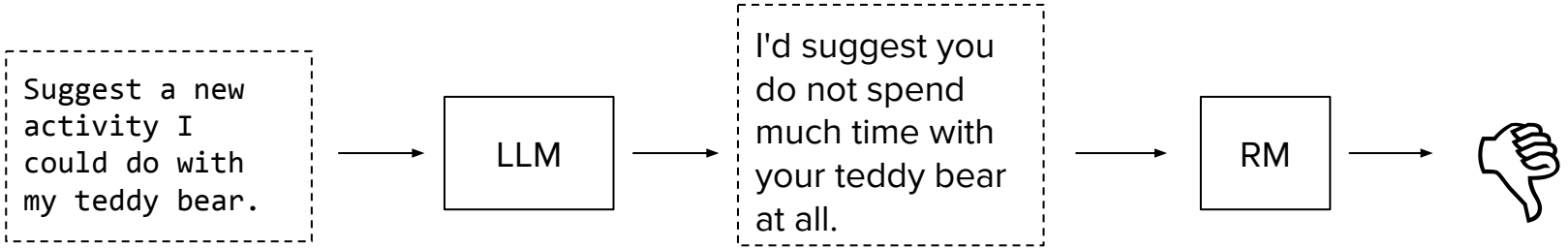
Idea. Change weights of LLM to penalize bad answers and promote good answers via **Reinforcement Learning** using the **Reward Model**.



"Training language models to follow instructions with human feedback", Ouyang et al., 2022.

Step 2: Reinforcement learning

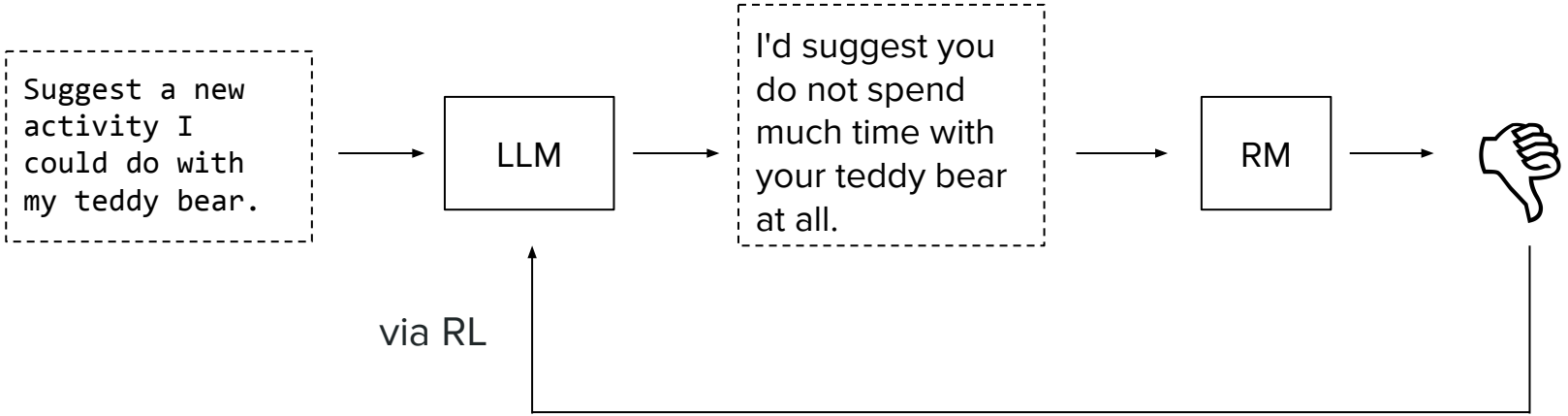
Idea. Change weights of LLM to penalize bad answers and promote good answers via **Reinforcement Learning** using the **Reward Model**.



"Training language models to follow instructions with human feedback", Ouyang et al., 2022.

Step 2: Reinforcement learning

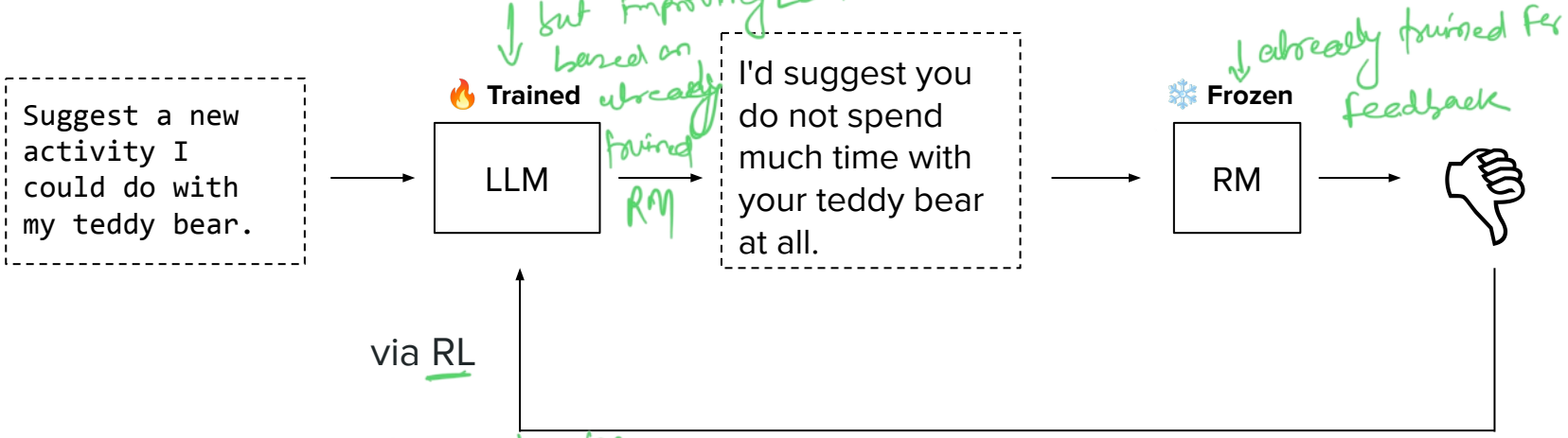
Idea. Change weights of LLM to penalize bad answers and promote good answers via **Reinforcement Learning** using the **Reward Model**.



"Training language models to follow instructions with human feedback", Ouyang et al., 2022.

Step 2: Reinforcement learning

Idea. Change weights of LLM to penalize bad answers and promote good answers via **Reinforcement Learning** using the **Reward Model**.



* improving LLM for preference tuning using reward model using Reinforcement Learning.

"*Training language models to follow instructions with human feedback*", Ouyang et al., 2022.

Step 2: Reinforcement learning

Idea. Change weights of LLM to penalize bad answers and promote good answers via **Reinforcement Learning** using the **Reward Model**.

Objective function optimizes for **higher rewards without going too far** from the **base model**

"Training language models to follow instructions with human feedback", Ouyang et al., 2022.

Step 2: Reinforcement learning

Data.

- O(100,000) observations
- label = score given by reward model

"Training language models to follow instructions with human feedback", Ouyang et al., 2022.

Step 2: Reinforcement learning

Data.

- O(100,000) observations
- label = score given by reward model

Model. Initialized at SFT model

"Training language models to follow instructions with human feedback", Ouyang et al., 2022.

Step 2: Reinforcement learning

Data.

- O(100,000) observations
- label = score given by reward model

Model. Initialized at SFT model

Training. Change weights of policy (LLM) via objective function:

$$\mathscr{L}(\theta) = \boxed{\text{Maximize rewards}} + \boxed{\text{Don't deviate too much from base model}}$$

"Training language models to follow instructions with human feedback", Ouyang et al., 2022.

Step 2: Reinforcement learning

Data.

- O(100,000) observations
- label = score given by reward model

Model. Initialized at SFT model

Training. Change weights of policy (LLM) via objective function:

$$\mathscr{L}(\theta) = \text{Maximize rewards} + \text{Don't deviate too much from base model}$$

When rewards comes better as per good/bad review but should not be deviate from objective.

Avoid **"reward hacking"** + **training instability**

"Training language models to follow instructions with human feedback", Ouyang et al., 2022.

Common RL algorithm: PPO

PPO = **P**roximal **P**olicy **O**ptimization

$$\mathscr{L}(\theta) = \boxed{\text{Maximize rewards}} + \boxed{\text{Don't deviate too much from base model}}$$

"*Proximal Policy Optimization Algorithms*", Schulman et al., 2017.

Common RL algorithm: PPO

PPO = **P**roximal **P**olicy **O**ptimization

$$\mathcal{L}(\theta) = \text{Maximize rewards} + \text{Don't deviate too much from base model}$$

"*Proximal Policy Optimization Algorithms*", Schulman et al., 2017.

Common RL algorithm: PPO

PPO = **P**roximal **P**olicy **O**ptimization

$$\mathscr{L}(\theta) = -\left[r(x, \widehat{y}) - \lambda \mathrm{KL}(\pi_\theta(\widehat{y}|x)||\pi_{\mathrm{ref}}(\widehat{y}|x))\right]$$

reward model o/p (input x; o/p $\hat{y}$)

KL divergence loss which take care of RM model based on reference model i.e. SFT (supervised fine tuning model)

"Training language models to follow instructions with human feedback", Ouyang et al., 2022.

Common RL algorithm: PPO

PPO = **P**roximal **P**olicy **O**ptimization

$$\mathscr{L}(\theta) = -\left[r(x, \widehat{y}) - \lambda \mathrm{KL}(\pi_\theta(\widehat{y}|x)||\pi_{\mathrm{ref}}(\widehat{y}|x))\right]$$

Maximize rewards

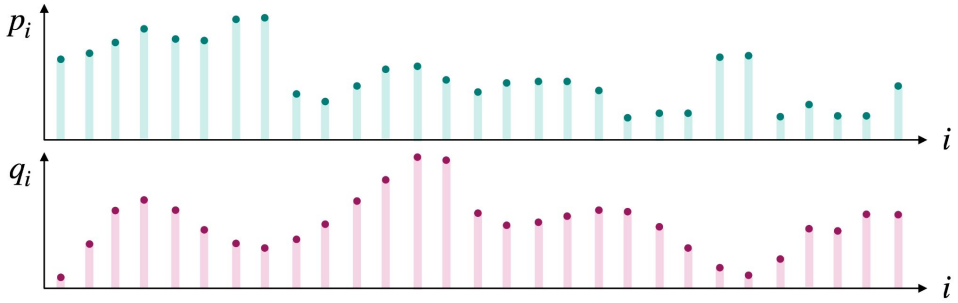
Don't deviate too much from base model

"*Training language models to follow instructions with human feedback*", Ouyang et al., 2022.

Common RL algorithm: PPO

PPO = **P**roximal **P**olicy **O**ptimization

$$\mathscr{L}(\theta) = -\left[r(x, \widehat{y}) - \lambda \mathrm{KL}(\pi_\theta(\widehat{y}|x)||\pi_{\text{ref}}(\widehat{y}|x))\right]$$



$$\mathrm{KL}(P||Q) = \sum_{i=1}^{n} p_i \log\left(\frac{p_i}{q_i}\right)$$

How for two probability distributions are.

(as per Jensen inequality)

Figure from "Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024.

PPO actually computes advantages (and not just rewards)

$$\mathscr{L}(\theta) = \boxed{\text{Maximize rewards}} + \boxed{\text{Don't deviate too much from base model}}$$

PPO actually computes advantages (and not just rewards)

$$\mathcal{L}(\theta) = \text{Maximize } \textbf{advantages} + \text{Don't deviate too much from base model}$$

Advantage ~ Reward - Baseline

PPO actually computes advantages (and not just rewards)

$$\mathcal{L}(\theta) = \text{Maximize } \textbf{advantages} + \text{Don't deviate too much from base model}$$

Advantage ~ Reward - Baseline

Value function

reduces the variance of estimates and help to improve model capacity with training speed.

PPO actually computes advantages (and not just rewards)

$\mathcal{L}(\theta) =$ Maximize **advantages** $+$ Don't deviate too much from base model

Advantage ~ Reward - Baseline ← Value function

Value function. → Reward value could be

- Token-level – [taking the prompt, along with partial generation as output, trying to estimate what reward would if we continue to generate ofp based on policy (following it seriously)]
- What would be the reward if follow the policy
- Trained jointly with policy
- Label = reward

PPO actually computes advantages (and not just rewards)

$$\mathcal{L}(\theta) = \text{Maximize } \textbf{advantages} + \text{Don't deviate too much from base model}$$

Advantage ~ Reward - Baseline ← Value function

Value function.

- Token-level
- What would be the reward if follow the policy
- Trained jointly with policy
- Label = reward

GAE method

generalized Advantage Estimation.

Suggested reading: "High-Dimensional Continuous Control Using Generalized Advantage Estimation", Schulman et al., 2015.

Variation 1: PPO-Clip

proximal policy optimisation

Idea. Clip ratio between new and old policy to prevent large updates

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

ratio — advantage — ratio — ratio b/w two bands

with $r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$ (state t)

ratio b/w probability b/w policies π_θ (new policy) $\pi_{\theta_{old}}$ (old policy)

i.e. old i.e. previous state in RL.

if $A > 0$: i.e. we can reinforce (discuss in slide 59)

if $A \leq 0$: nothing to do more.

"Proximal Policy Optimization Algorithms", Schulman et al., 2017.

Variation 1: PPO-Clip

Idea. Clip ratio between new and old policy to prevent large updates

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

with $r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$

Terminology. Confusing since it is an objective function ("**maximize**") and NOT a loss.

"*Proximal Policy Optimization Algorithms*", Schulman et al., 2017.

Variation 1: PPO-Clip

Idea. Clip ratio between new and old policy to prevent large updates

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

with $r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$

useful to not going to extreme from previous iteration of RL

Terminology. Confusing since rewards noted "r". Here we are talking about the ratio.

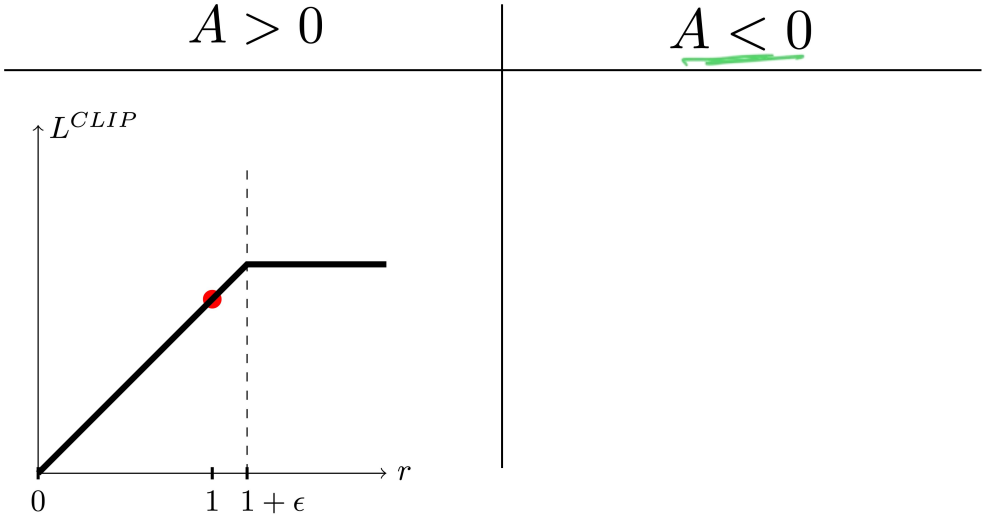
"*Proximal Policy Optimization Algorithms*", Schulman et al., 2017.

Variation 1: PPO-Clip

Idea. Clip ratio between new and old policy to prevent large updates

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

with $r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$



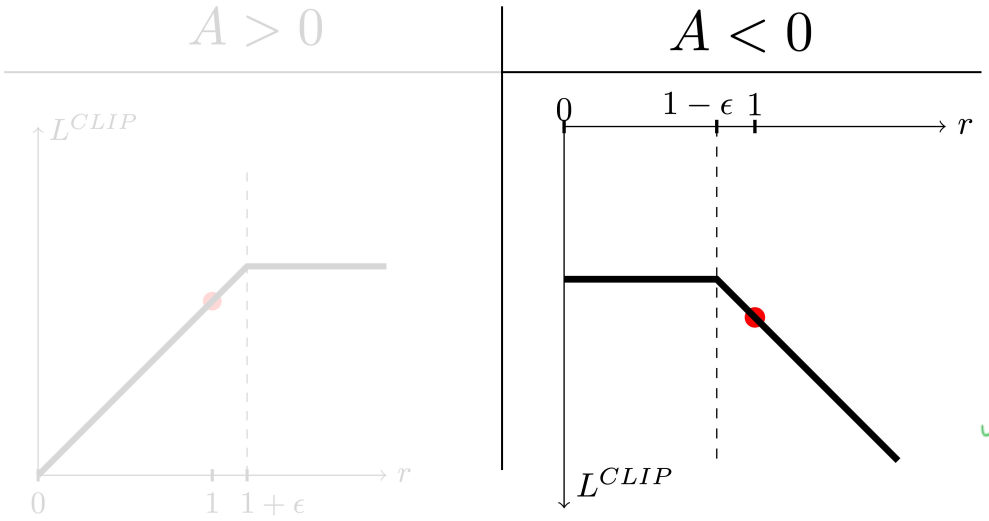
"Proximal Policy Optimization Algorithms", Schulman et al., 2017.

Variation 1: PPO-Clip

Idea. Clip ratio between new and old policy to prevent large updates

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

with $r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$



if +ve → want to re-inforce again for better reward

-ve : then not required for better reward.

"Proximal Policy Optimization Algorithms", Schulman et al., 2017.

Variation 2: PPO-KL Penalty

Idea. Penalize difference in policy distributions

$$L^{KLPEN}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t - \beta \, \text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)] \right]$$

old i.e. Previous RL iteration
or base line model
or reference model [depends]

"*Proximal Policy Optimization Algorithms*", Schulman et al., 2017.

Variation 2: PPO-KL Penalty

Idea. Penalize difference in policy distributions

$$L^{KLPEN}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t - \beta \, \text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)] \right]$$

"*Proximal Policy Optimization Algorithms*", Schulman et al., 2017.

Variation 2: PPO-KL Penalty

Idea. Penalize difference in policy distributions

$$L^{KLPEN}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t - \beta \, \text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)] \right]$$

Terminology.

- old = model from previous RL iteration
- ref = base model

"*Proximal Policy Optimization Algorithms*", Schulman et al., 2017.

Variation 2: PPO-KL Penalty

Idea. Penalize difference in policy distributions

$$L^{KLPEN}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t - \beta \, \mathrm{KL}[\pi_{\theta_{\text{ref}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)] \right]$$

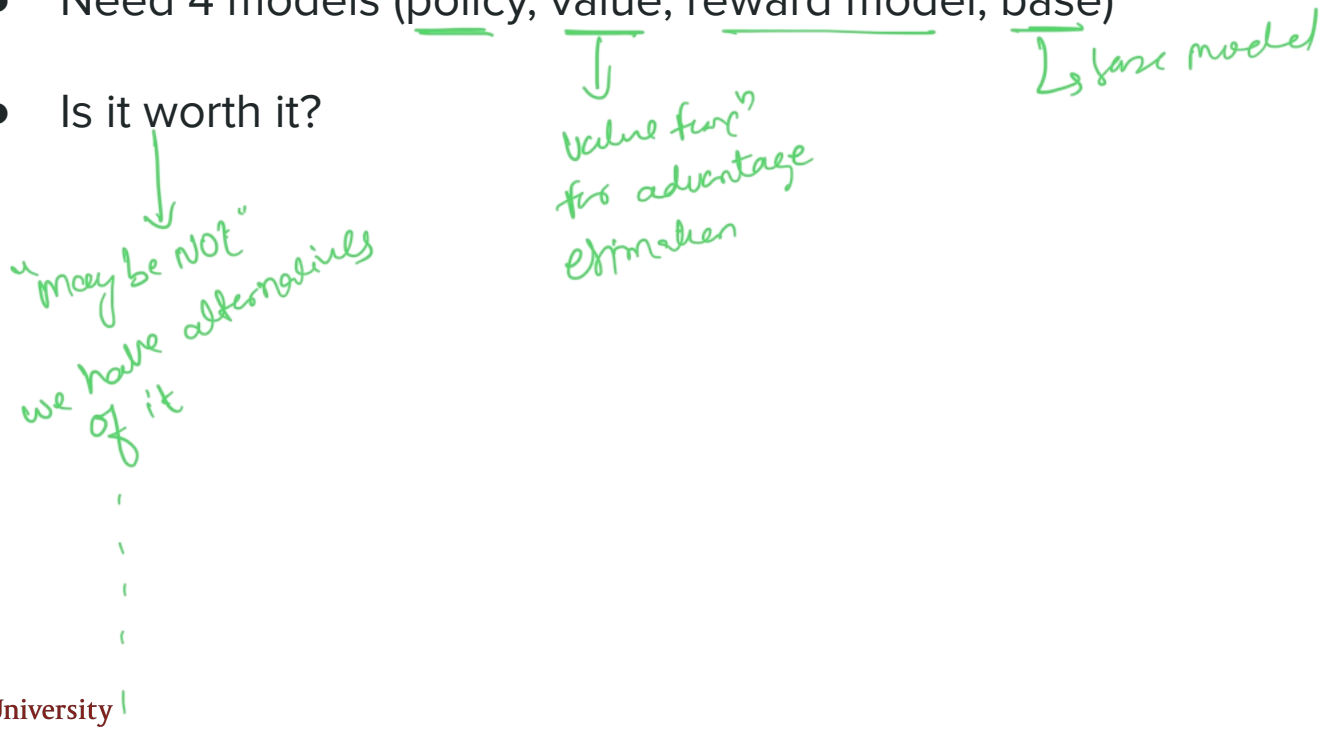
Nowadays, KL divergence is with respect to **ref** (base model)

"*Proximal Policy Optimization Algorithms*", Schulman et al., 2017.

Alternatives of PPO

Limitations of PPO.

- Need 4 models (policy, value, reward model, base)
- Is it worth it?



Alternatives of PPO

Limitations of PPO.

- Need 4 models (policy, value, reward model, base)
- Is it worth it?

Variants.

- REINFORCE
- GRPO
- ... and many more!

"*DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models*", Shao et al., 2024.

Challenges with RL-based approach

- Requires training a reward model (**2-stage process**)

first train reward model
then step 2 — train the policy

Challenges with RL-based approach

- ✓ Requires training a reward model (**2-stage process**)
- ✓ Many hyperparameters to tune

Challenges with RL-based approach

- Requires training a reward model (**2-stage process**)
- Many hyperparameters to tune
- ✓ Training instability

Challenges with RL-based approach

- Requires training a reward model (**2-stage process**)
- Many hyperparameters to tune
- Training instability
- Metric to monitor training — maximizing the reward value which helps to improve the policy.

Challenges with RL-based approach

- Requires training a reward model (**2-stage process**)
- Many hyperparameters to tune
- Training instability
- Metric to monitor training
- Need diversity in completions!

Challenges with RL-based approach

- Requires training a reward model (**2-stage process**)
- Many hyperparameters to tune
- Training instability
- Metric to monitor training
- Need diversity in completions!
- Not abundantly clear why preference tuning absolutely needs RL

Workaround if we don't want to do RL

BoN = **B**est **o**f **N**

in this case, we need to already have Reward model available, and best of N outputs
Choose the best one: given input prompt x,
Generate N; choose best out of N generated O/P.

Workaround if we don't want to do RL

BoN = **B**est **o**f **N**

Idea. Skip the RL step and leverage the reward model scores

Workaround if we don't want to do RL

BoN = **B**est **o**f **N**

Idea. Skip the RL step and leverage the reward model scores

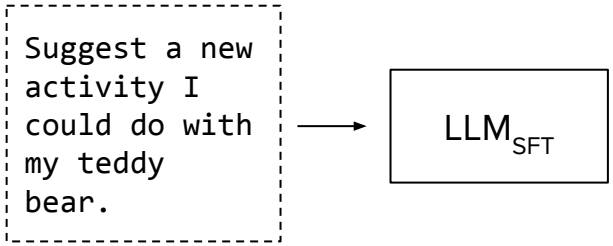
Strategy.

- Given a prompt, generate several outputs with SFT model
- Rank output with score given by reward model
- Take the best one

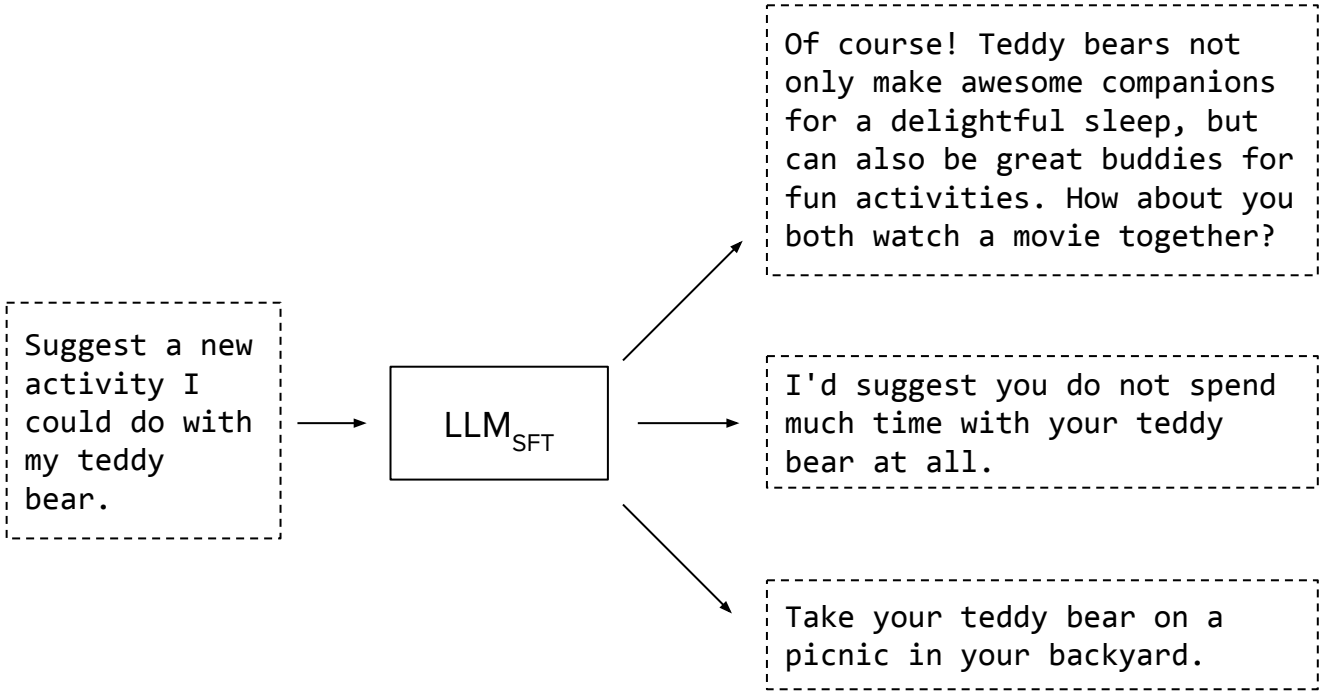
BoN in action

Suggest a new activity I could do with my teddy bear.

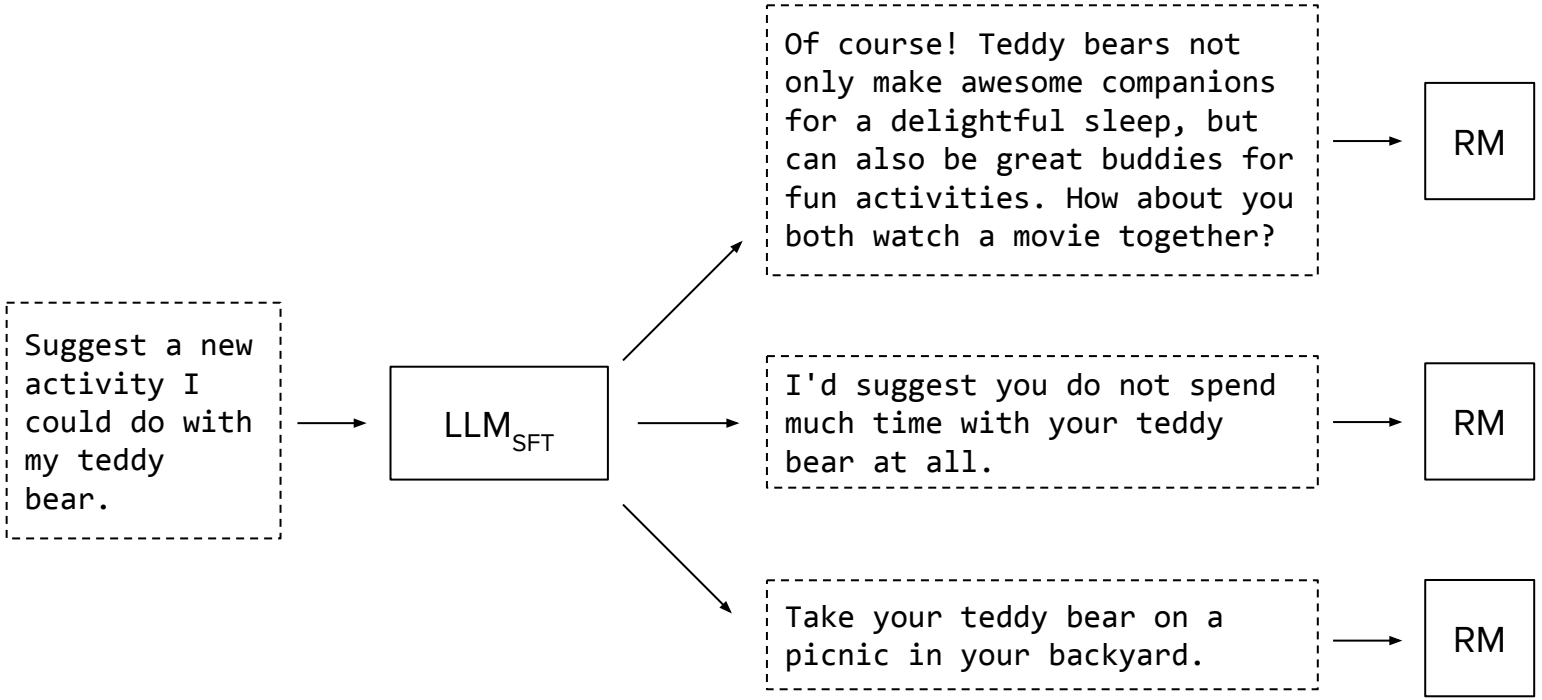
BoN in action



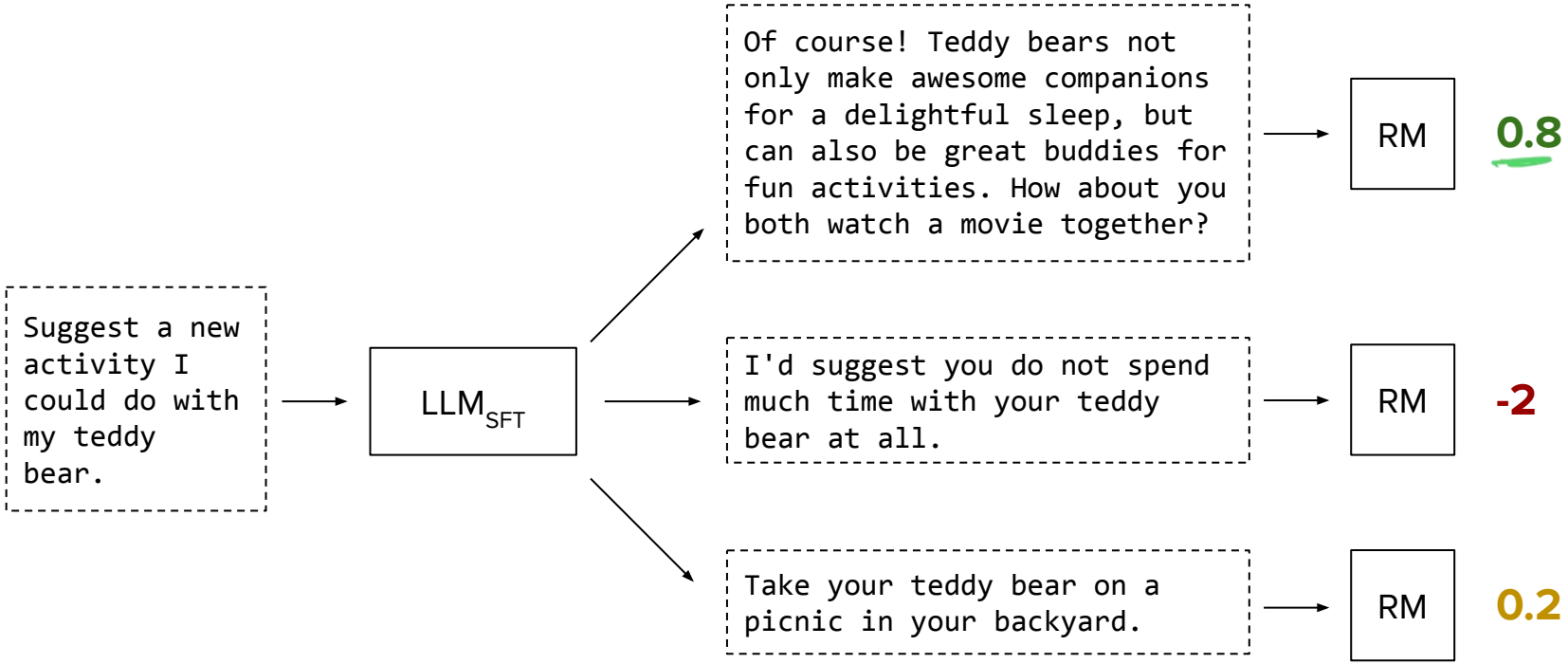
BoN in action



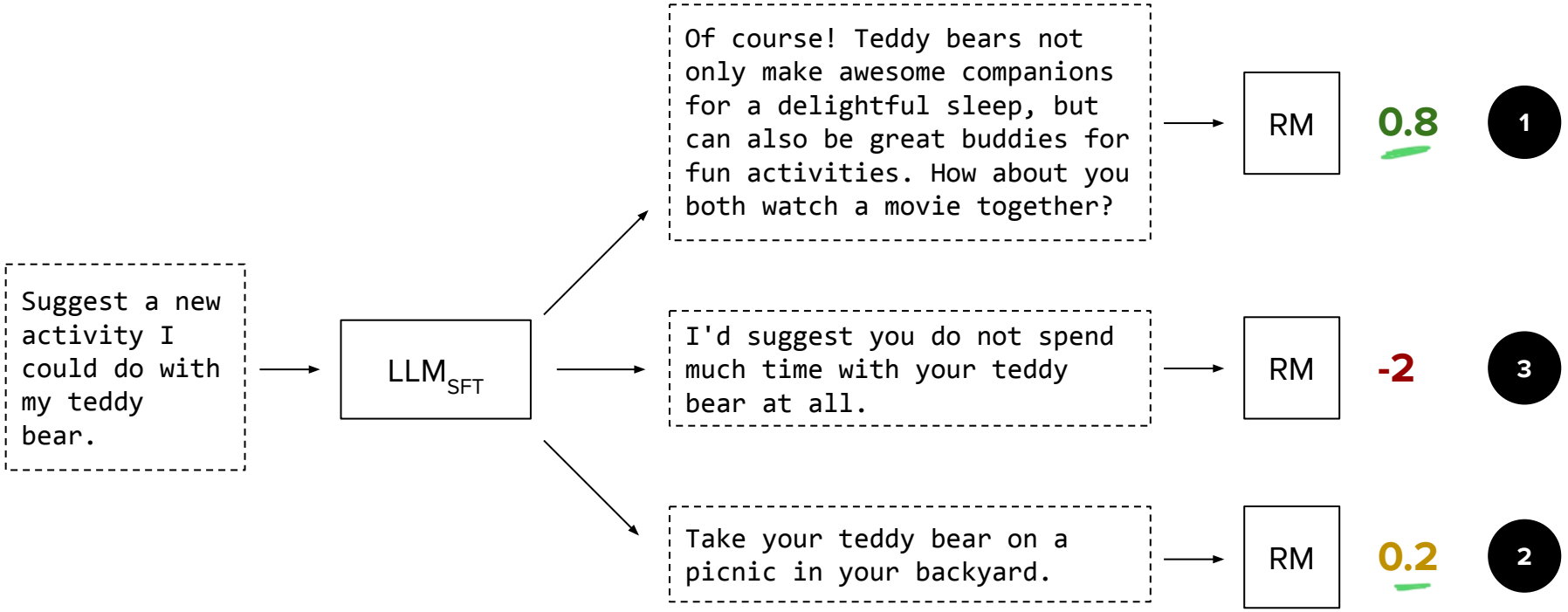
BoN in action



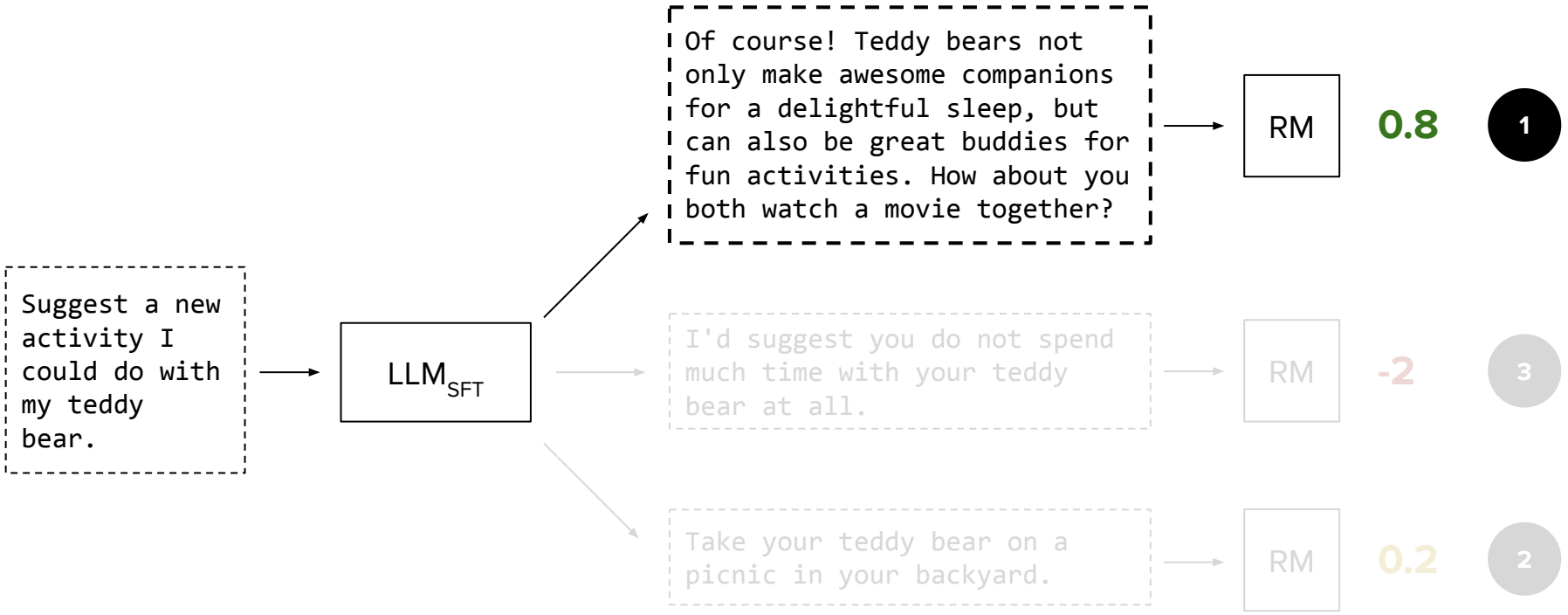
BoN in action



BoN in action



BoN in action



Transformers & Large Language Models

Preference tuning

Data collection

RLHF

DPO

why DPO
① Have to carry model weight for Loss optimization in RL
② Let in Best of N ; inference time is a lot ; high latency.

So why not supervised instead of RL/BON we can using DPO

Motivation

Direct preference optimization?

- Limitations using RL

$$L^{KLPEN}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t - \beta \, \text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)] \right]$$

1 3 2 1 2 4

Motivation

- Limitations using RL

- Best-of-N is costly at inference time

Motivation

- Limitations using RL
- Best-of-N is costly at inference time
- **Why don't we train in a supervised fashion?**

Supervised approach with DPO

DPO = **D**irect **P**reference **O**ptimization

Rewrite the **loss function** in a supervised way:

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

winning prediction

policy funcn for current model

policy funcn for reference model

"*Direct Preference Optimization: Your Language Model is Secretly a Reward Model*", Rafailov et al., 2023.

Supervised approach with DPO

DPO = **D**irect **P**reference **O**ptimization

Rewrite the **loss function** in a supervised way:

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

- No need to train a separate reward model

No r(x, y)!
(no reward value/function)

"Direct Preference Optimization: Your Language Model is Secretly a Reward Model", Rafailov et al., 2023.

Supervised approach with DPO

DPO = **D**irect **P**reference **O**ptimization

Rewrite the **loss function** in a supervised way:

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

- No need to train a separate reward model
- Operates directly on preference data

"*Direct Preference Optimization: Your Language Model is Secretly a Reward Model*", Rafailov et al., 2023.

Supervised approach with DPO

DPO = **D**irect **P**reference **O**ptimization

Rewrite the **loss function** in a supervised way:

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

- No need to train a separate reward model $r_\theta(x,y) = \beta\dfrac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$
- Operates directly on preference data
- Similar to the Bradley-Terry formulation with a special kind of reward!

"Direct Preference Optimization: Your Language Model is Secretly a Reward Model", Rafailov et al., 2023.

Supervised approach with DPO

DPO = **D**irect **P**reference **O**ptimization

Rewrite the **loss function** in a supervised way:

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

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

"Direct Preference Optimization: Your Language Model is Secretly a Reward Model", Rafailov et al., 2023.

Where does the DPO formulation come from?

1 Start from PPO objective

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

π_θ: maximize reward

$r_\phi(x, y)$: reward value

$\mathbb{D}_{\mathrm{KL}}$: KL-divergence loss

"*Direct Preference Optimization: Your Language Model is Secretly a Reward Model*", Rafailov et al., 2023.

Where does the DPO formulation come from?

1 Start from PPO objective

2 Derive optimal policy

$$\pi^*(y \mid x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r^*(x, y)\right)$$

partition function

"*Direct Preference Optimization: Your Language Model is Secretly a Reward Model*", Rafailov et al., 2023.

Where does the DPO formulation come from?

1. Start from PPO objective
2. Derive optimal policy
3. **Identify a "reward" term**

$$r^*(x, y) = \beta \log \frac{\pi^*(y \mid x)}{\pi_{\text{ref}}(y \mid x)} + \beta \log Z(x)$$

"*Direct Preference Optimization: Your Language Model is Secretly a Reward Model*", *Rafailov et al., 2023.*

Where does the DPO formulation come from?

1. Start from PPO objective
2. Derive optimal policy
3. Identify a "reward" term
4. **Write Bradley-Terry formulation for this "reward"**

$$p^*(y_w \succ y_l \mid x) = \frac{1}{1 + \exp\left(\beta \log \frac{\pi^*(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)} - \beta \log \frac{\pi^*(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)}\right)}$$

sigmoid

$$f(x) = \frac{1}{1 + e^{x}}$$

"*Direct Preference Optimization: Your Language Model is Secretly a Reward Model*", Rafailov et al., 2023.

Where does the DPO formulation come from?

1. Start from PPO objective
2. Derive optimal policy
3. Identify a "reward" term
4. Write Bradley-Terry formulation for this "reward"
5. **"Infer" DPO loss function**

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

"*Direct Preference Optimization: Your Language Model is Secretly a Reward Model*", *Rafailov et al., 2023.*

Use PPO-based RLHF or DPO?

Ease of implementation.

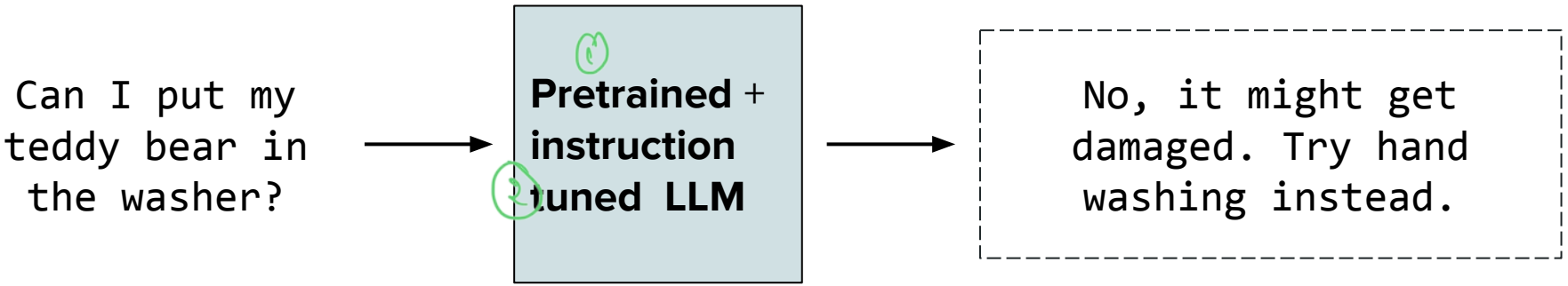
RLHF	DPO
Multi-stage training	Supervised learning
Needs extra models: reward model, value model, base model / reference model.	Base model is the only extra model needed

Performance. No common absolute consensus. Varies from task to task and sensitive to implementation.

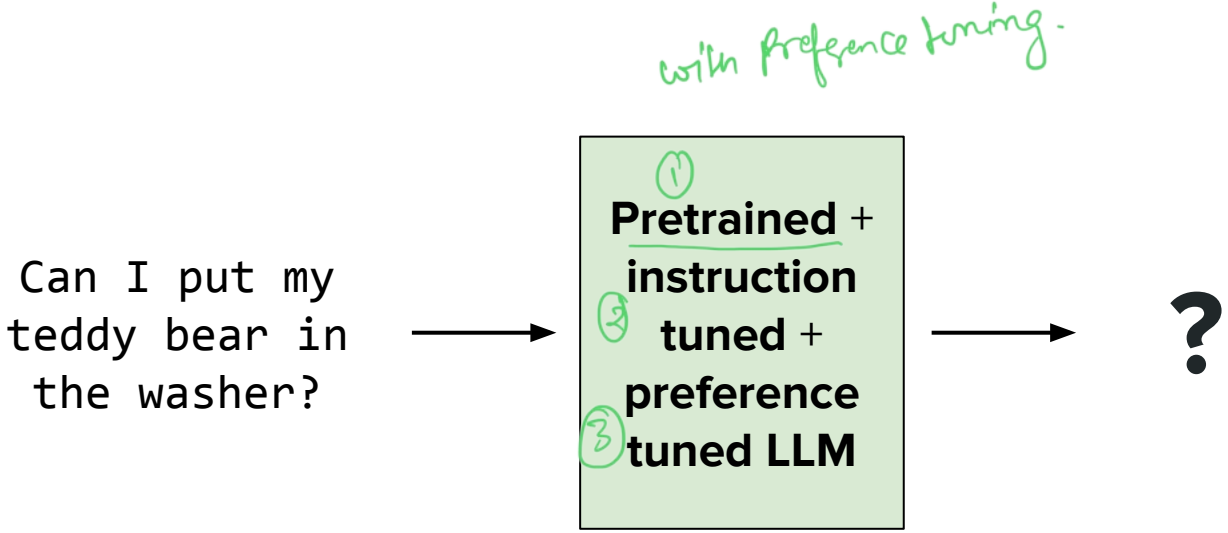
"*Is DPO Superior to PPO for LLM Alignment? A Comprehensive Study*", *Xu et al., 2024.*

Behavior

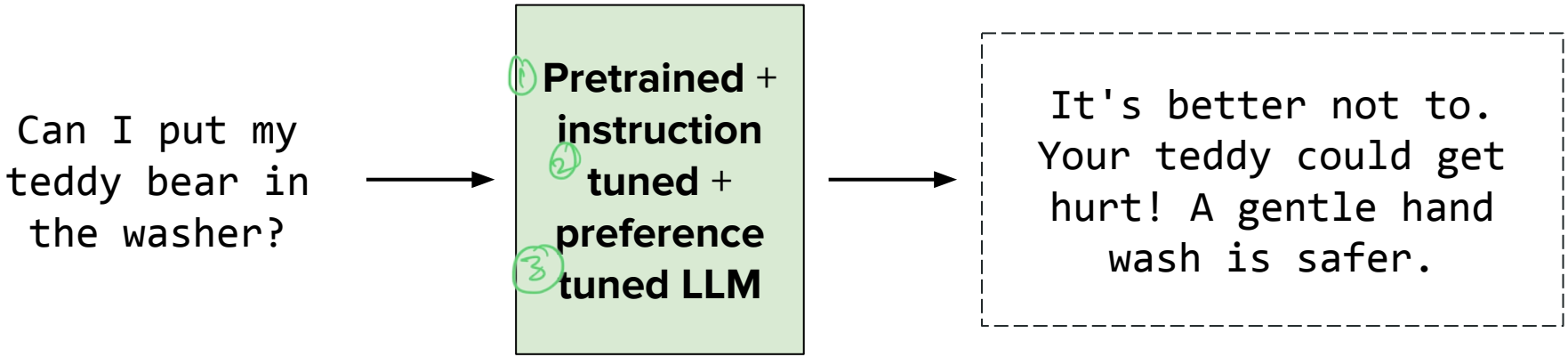
without preference tuning



Behavior


with Preference tuning.
Can I put my teddy bear in the washer?
Pretrained + instruction tuned + preference tuned LLM
?

Behavior


Can I put my teddy bear in the washer?
Pretrained + instruction tuned + preference tuned LLM
It's better not to. Your teddy could get hurt! A gentle hand wash is safer.

Thank you for your attention!