Lecture 5

CME 295: Transformers & Large Language Models


Afshine Amidi & Shervine Amidi


Stanford University

ICME

Recap of last episodes...

Initialized model

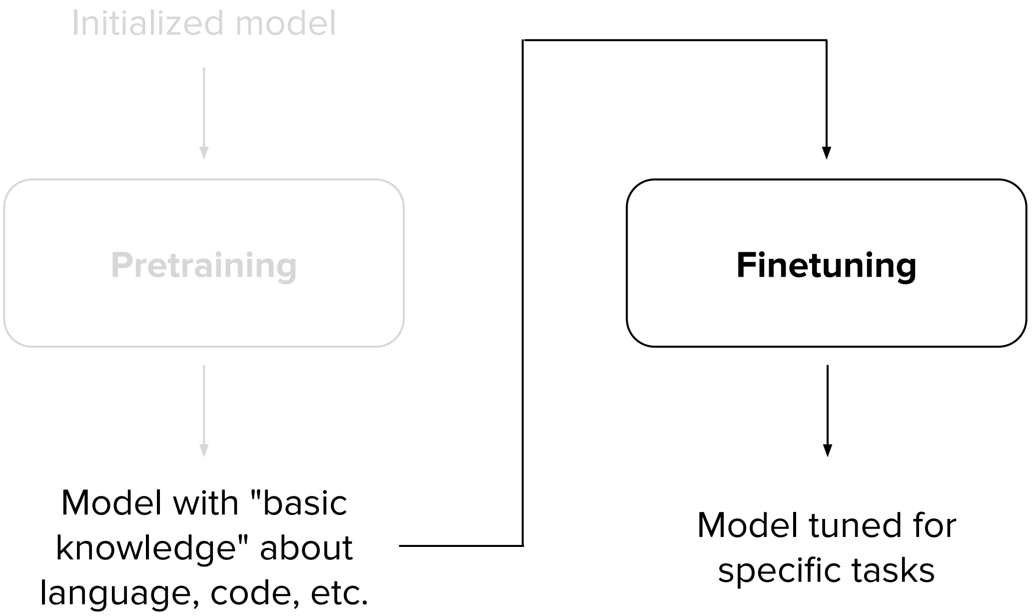
↓

Pretraining

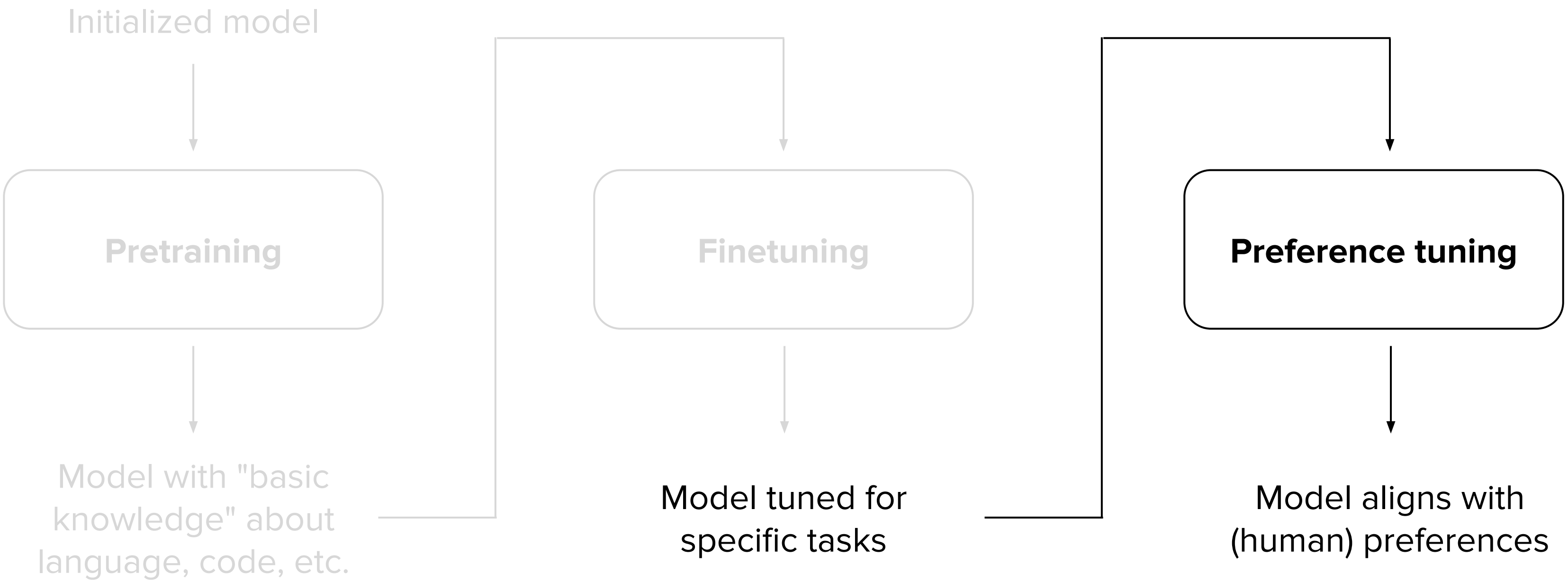
↓

Model with "basic knowledge" about language, code, etc.

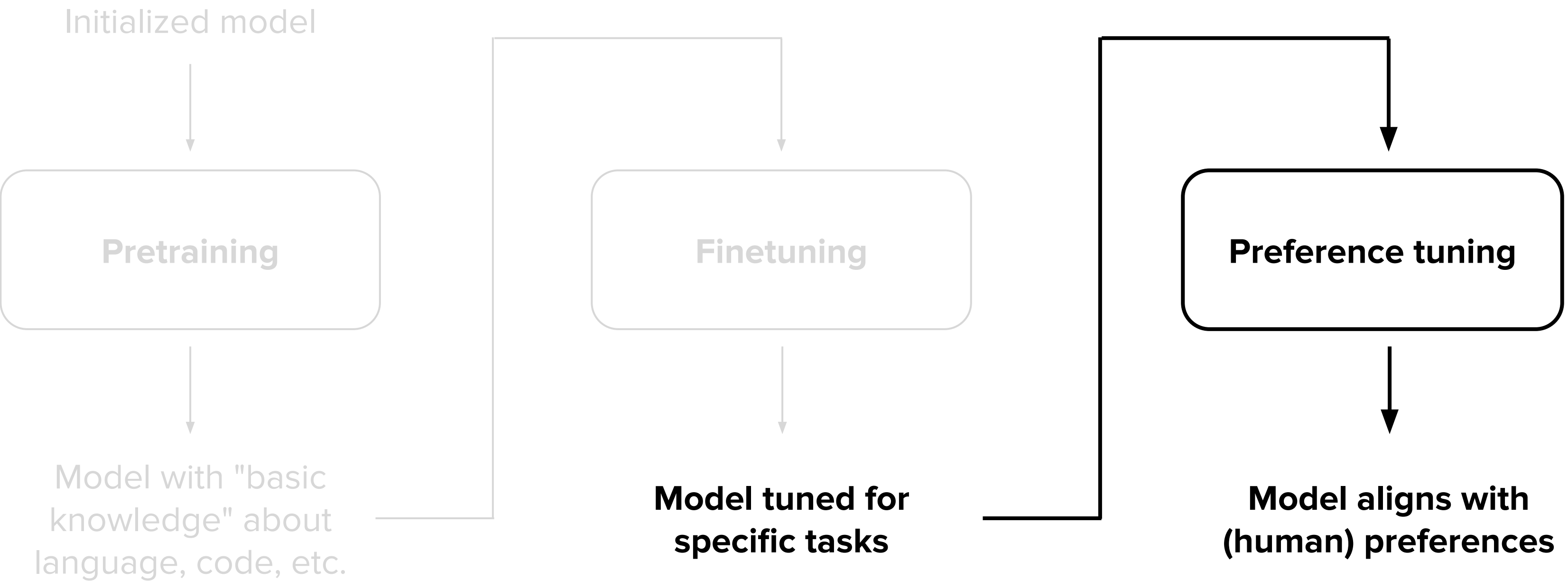
Recap of last episodes...



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.

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

$\longrightarrow$ $\boxed{LLM_{SFT}}$ $\longrightarrow$

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

Preference tuning

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

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

→ LLM_{SFT} →

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

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

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

Preference tuning

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

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

→ LLM_{SFT} →

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

✗

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

```
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.
```

✗

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

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

Obs 1	0.4
Obs 2	0.9
Obs 3	0.1
Obs 4	0.2

Preference data

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

Pointwise

Obs 1	0.4
Obs 2	0.9
Obs 3	0.1
Obs 4	0.2

Pairwise

Obs 1	<	Obs 2
Obs 1	>	Obs 3
Obs 1	>	Obs 4
Obs 2	>	Obs 3

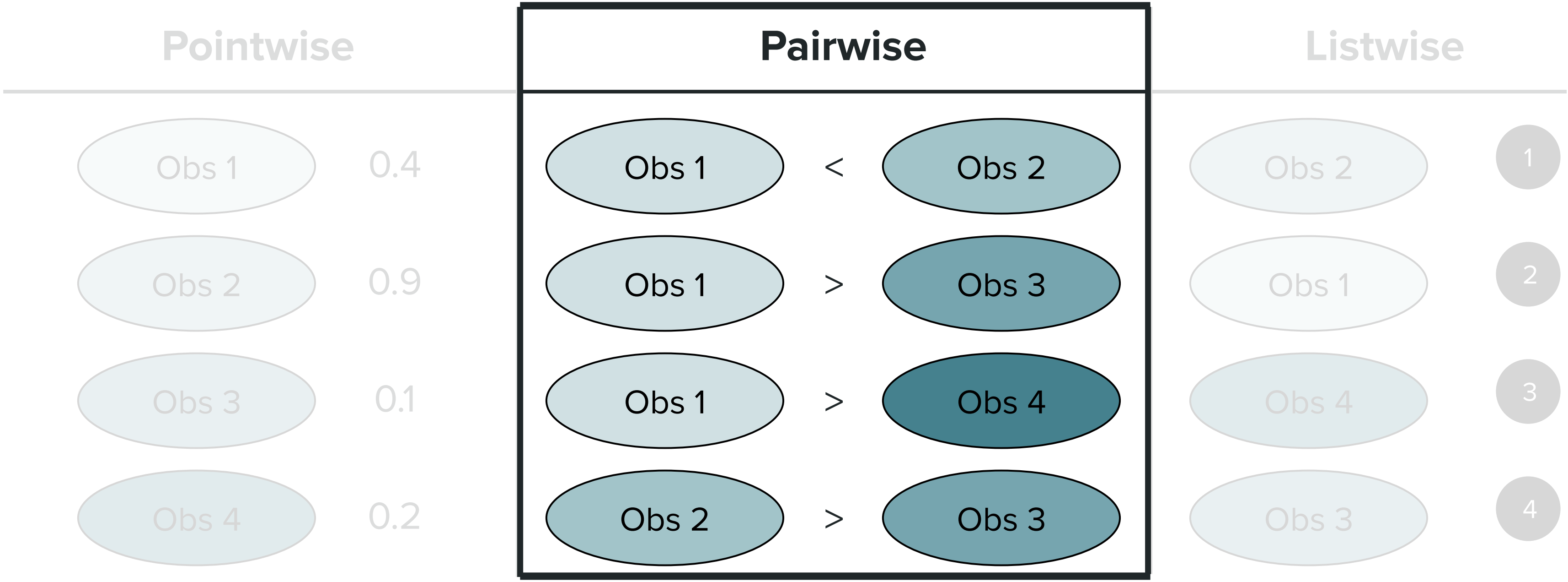
Preference data

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

Pointwise		Pairwise			Listwise	
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

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

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

Transformers & Large Language Models

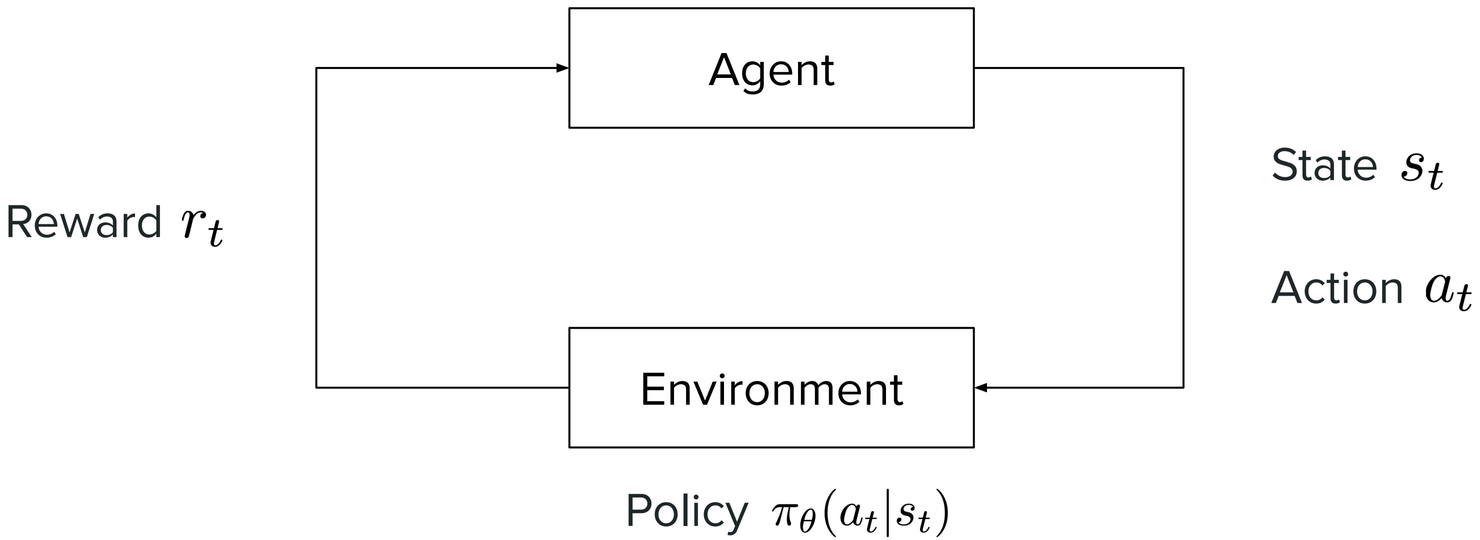
Preference tuning

Data collection

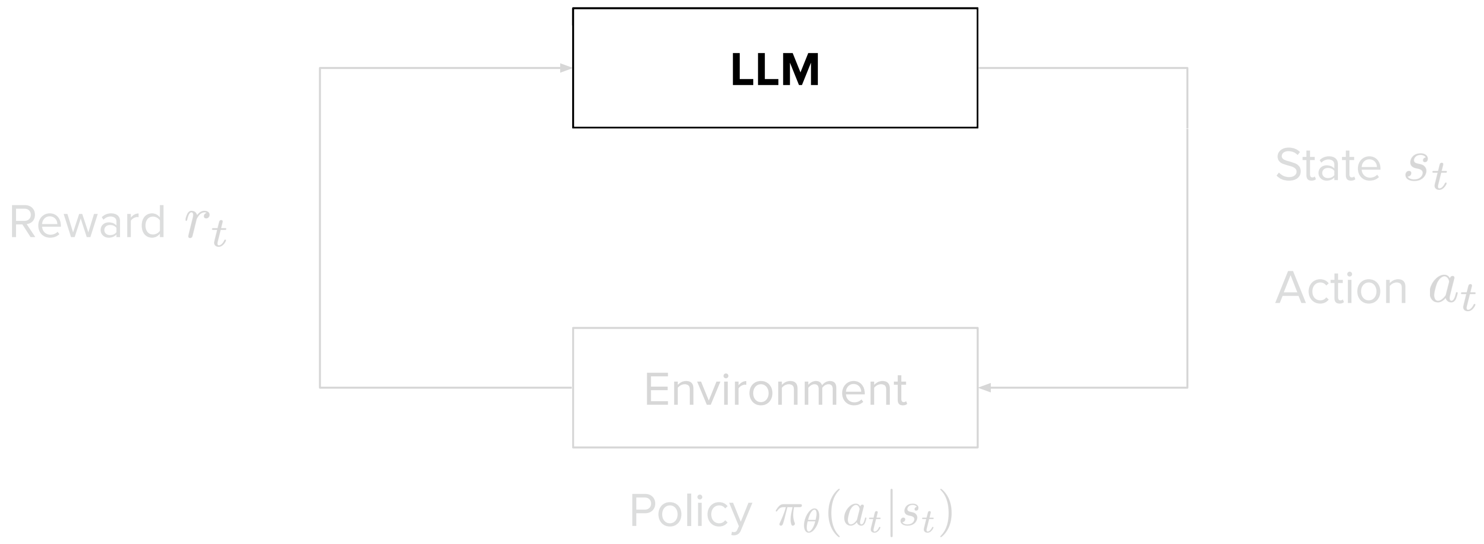
RLHF

DPO

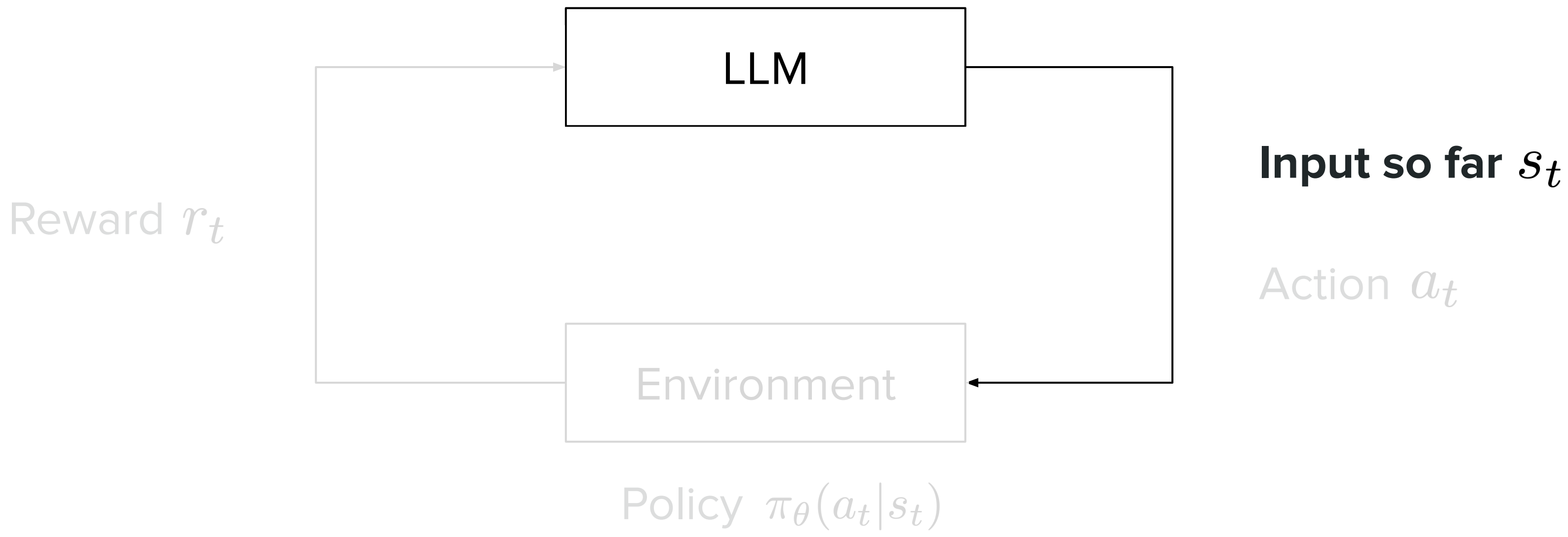
RL formulation



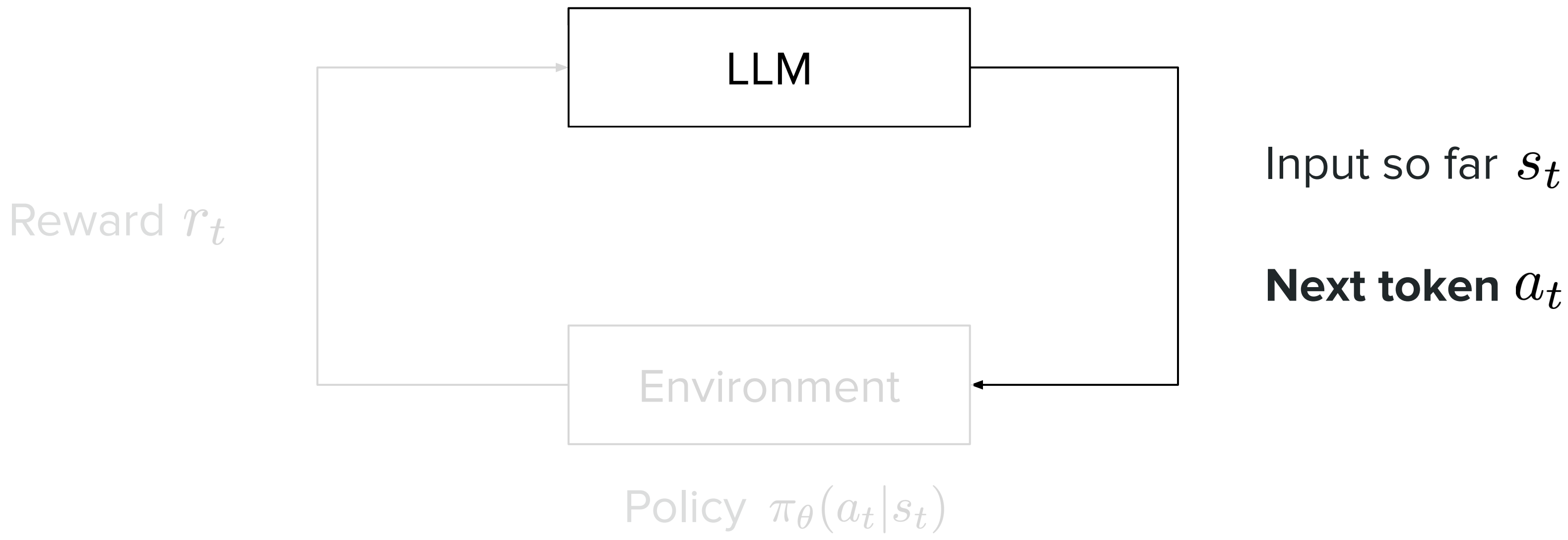
RL formulation for LLMs



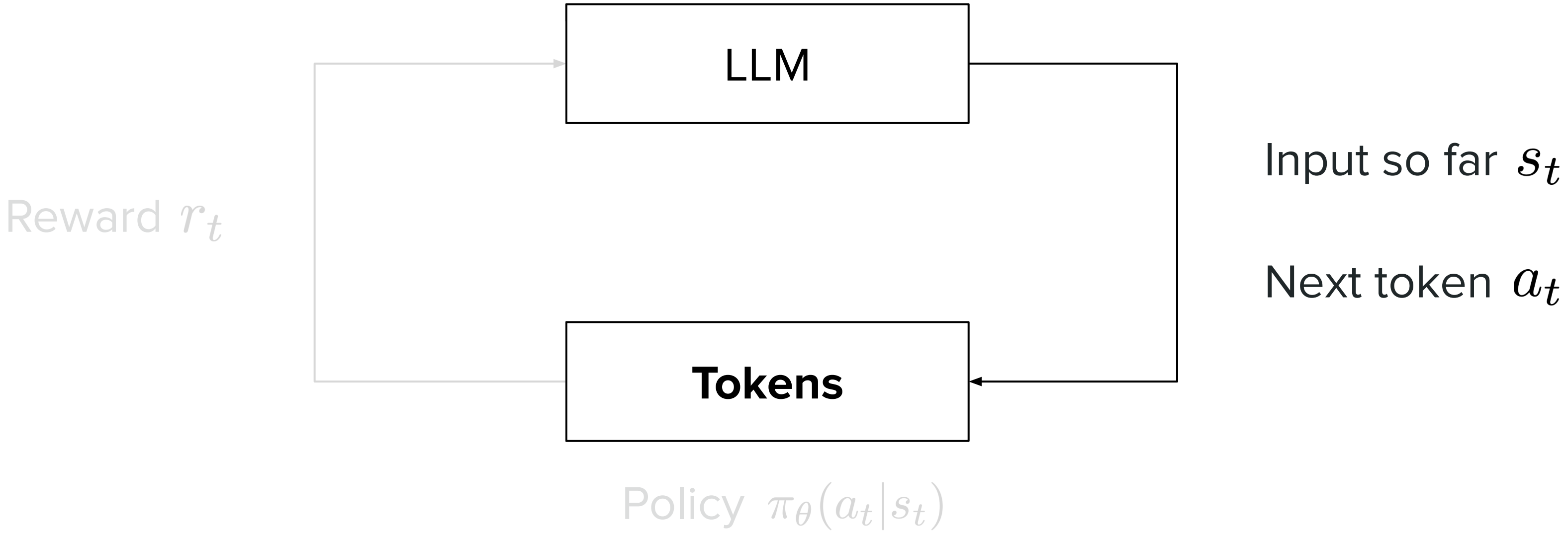
RL formulation for LLMs



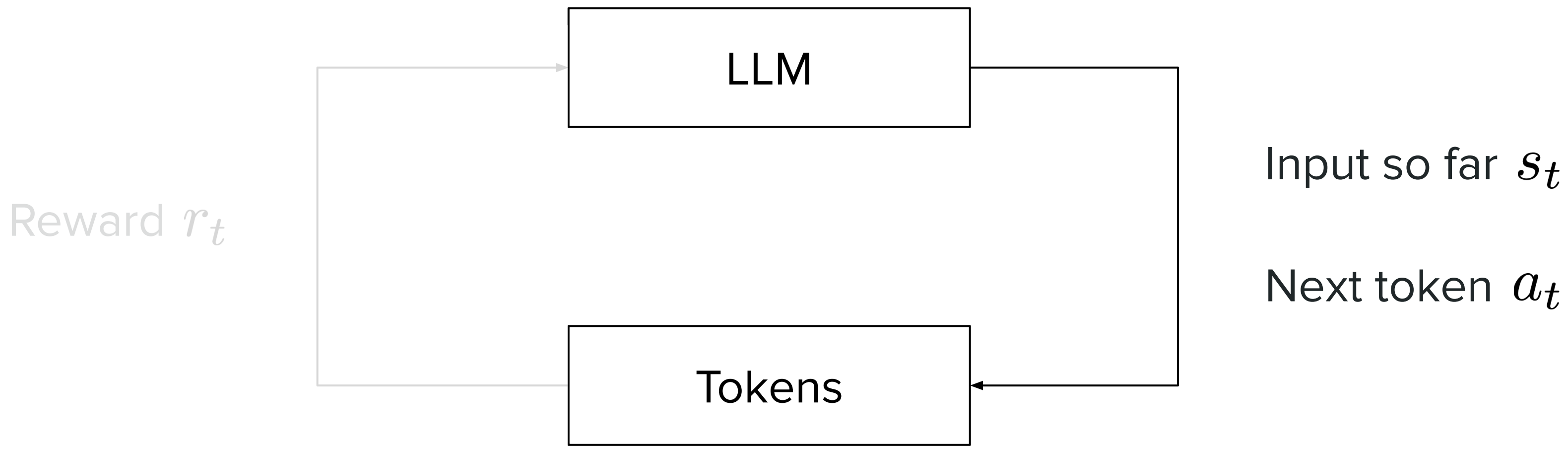
RL formulation for LLMs



RL formulation for LLMs

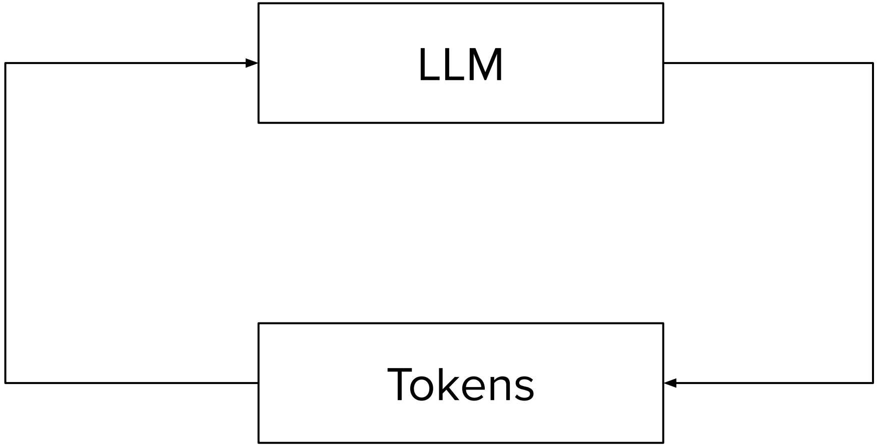


RL formulation for LLMs



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

RL formulation for LLMs



Human preference r_t

Input so far s_t

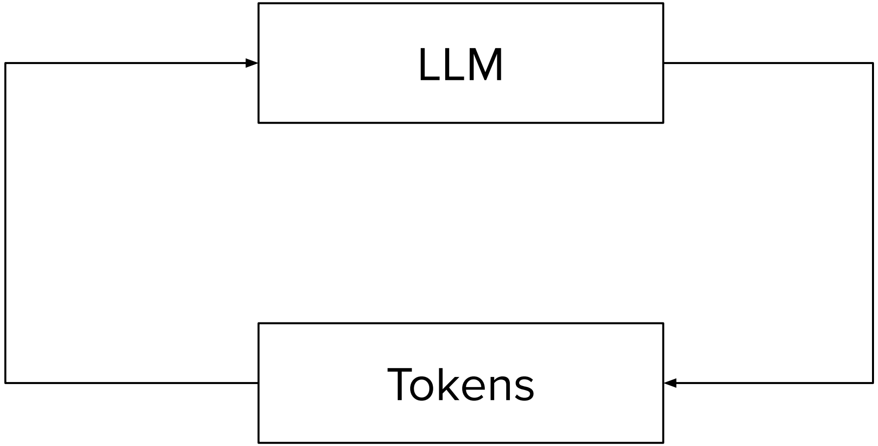
Next token a_t

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

Human preference r_t



Input so far s_t

Next token a_t

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

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

"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 $\widehat{y}$

"*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**.

"*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? → RM →

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

"*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:

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

where r_i, r_j rewards of y_i, y_j respectively

Stanford University

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

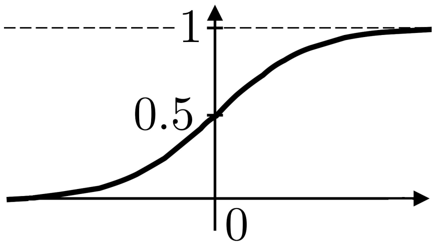
ICME

Step 1: Reward modeling

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

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

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

$$
\begin{tikzcd}
(x, \hat{y}_w) \arrow[r] & \boxed{\mathrm{RM}} \arrow[r] & r(x, \hat{y}_w) \arrow[d, leftrightarrow, "\mathscr{L}"] \\
(x, \hat{y}_l) \arrow[ru] & & r(x, \hat{y}_l)
\end{tikzcd}
$$

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

$$\begin{array}{ccccc} (x,\hat{y}_w) & \longrightarrow & & \longrightarrow & r(x,\hat{y}_w) \\ & & \boxed{\text{RM}} & & \updownarrow \mathscr{L} \\ (x,\hat{y}_l) & \longrightarrow & & \longrightarrow & r(x,\hat{y}_l) \end{array}$$

$$\boxed{\mathscr{L}(\theta) = -\mathbb{E}\Big[\log(\sigma(r(x,\widehat{y}_w) - r(x,\widehat{y}_l)))\Big]}$$

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.

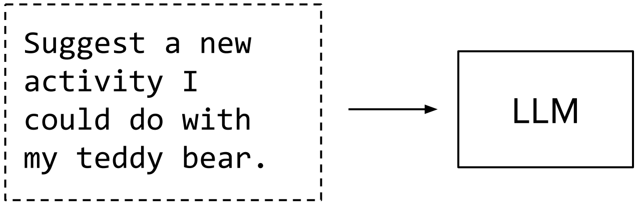
Step 2: Reinforcement learning

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

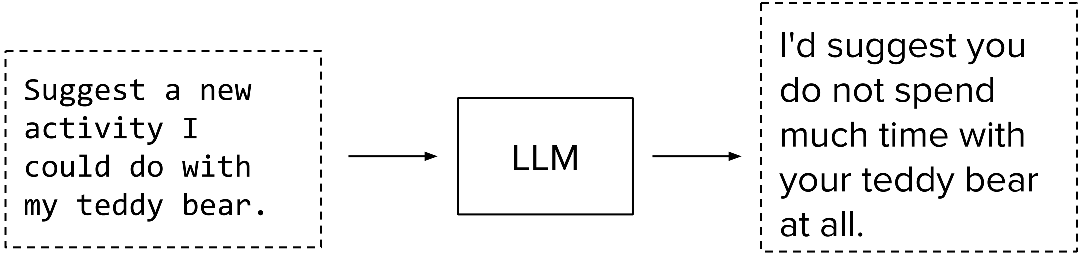
Idea. Change weights of LLM to penalize bad answers and promote good answers via **Reinforcement Learning** using the **Reward Model**.



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

Step 2: Reinforcement learning

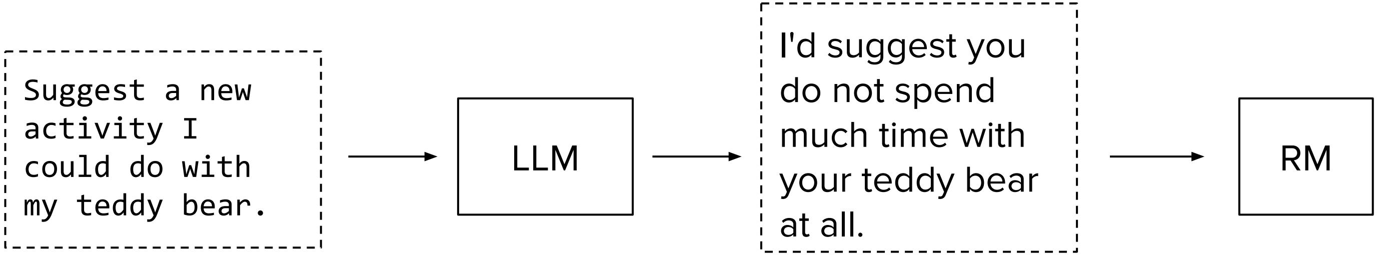
Idea. Change weights of LLM to penalize bad answers and promote good answers via **Reinforcement Learning** using the **Reward Model**.



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

Step 2: Reinforcement learning

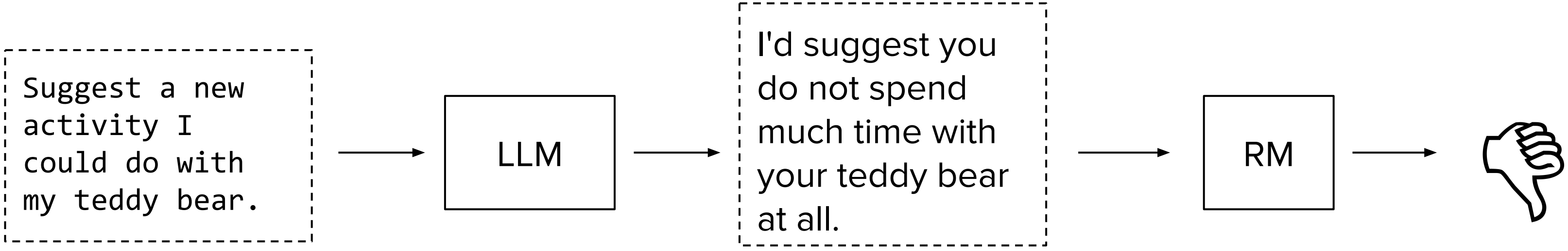
Idea. Change weights of LLM to penalize bad answers and promote good answers via **Reinforcement Learning** using the **Reward Model**.



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

Step 2: Reinforcement learning

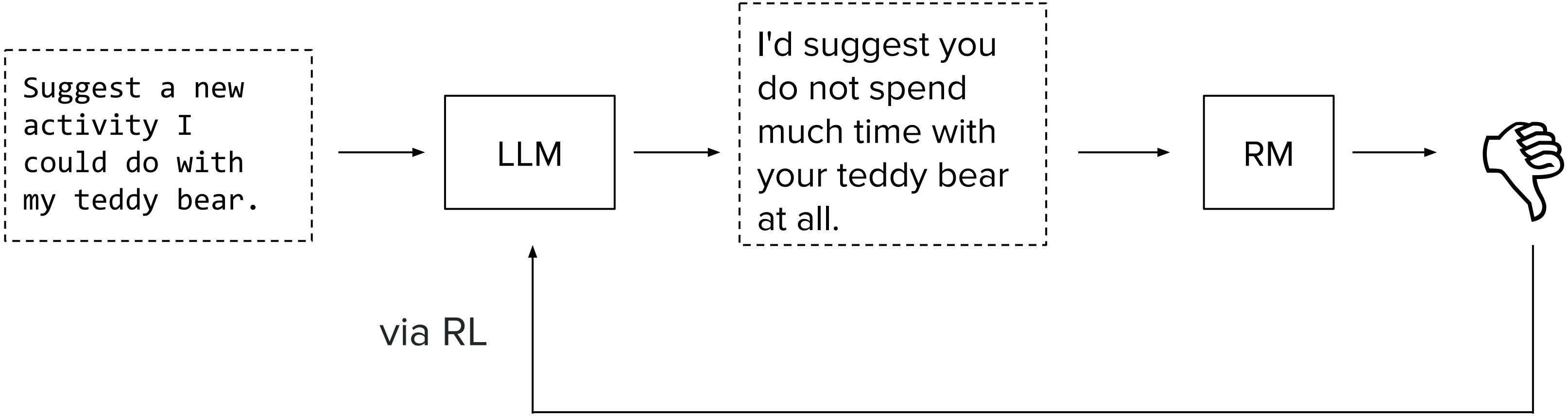
Idea. Change weights of LLM to penalize bad answers and promote good answers via **Reinforcement Learning** using the **Reward Model**.



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

Step 2: Reinforcement learning

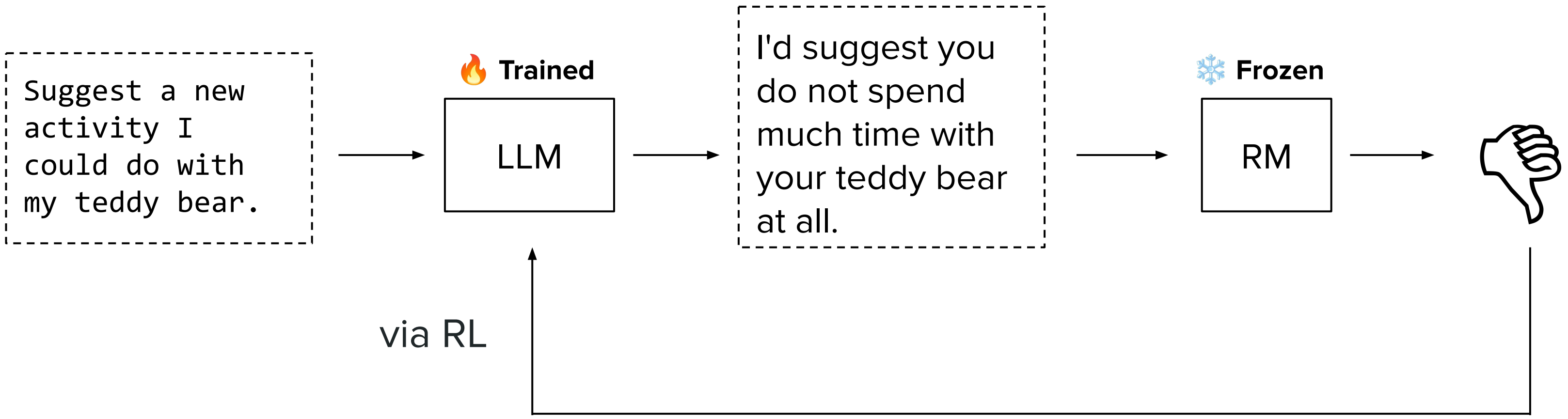
Idea. Change weights of LLM to penalize bad answers and promote good answers via **Reinforcement Learning** using the **Reward Model**.



"*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**.



"*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**

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

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

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.*

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

$\mathscr{L}(\theta) =$ Maximize rewards $+$ 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

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

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

Common RL algorithm: PPO

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

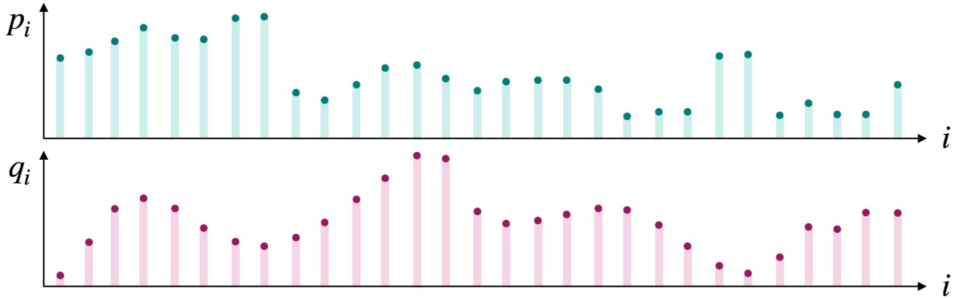
$$\mathscr{L}(\theta) = -\Big[\underbrace{r(x, \widehat{y})}_{\text{Maximize rewards}} - \underbrace{\lambda \mathrm{KL}(\pi_\theta(\widehat{y}|x) || \pi_{\mathrm{ref}}(\widehat{y}|x))}_{\text{Don't deviate too much from base model}}\Big]$$

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

Common RL algorithm: PPO

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

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



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

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)

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

Advantage ~ Reward - Baseline

PPO actually computes advantages (and not just rewards)

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

Advantage ~ Reward - Baseline

Value function

PPO actually computes advantages (and not just rewards)

$\mathscr{L}(\theta) =$ Maximize **advantages** $+$ 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

PPO actually computes advantages (and not just rewards)

$\mathscr{L}(\theta) =$ Maximize **advantages** $+$ 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

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

Variation 1: PPO-Clip

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

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

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

Variation 1: PPO-Clip

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

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

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

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

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

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

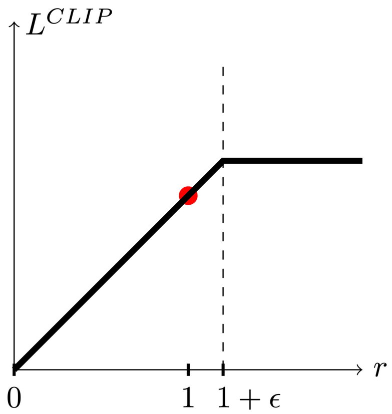
Terminology. Confusing since rewards noted "r".
Here we are talking about the ratio.

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

Variation 1: PPO-Clip

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

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

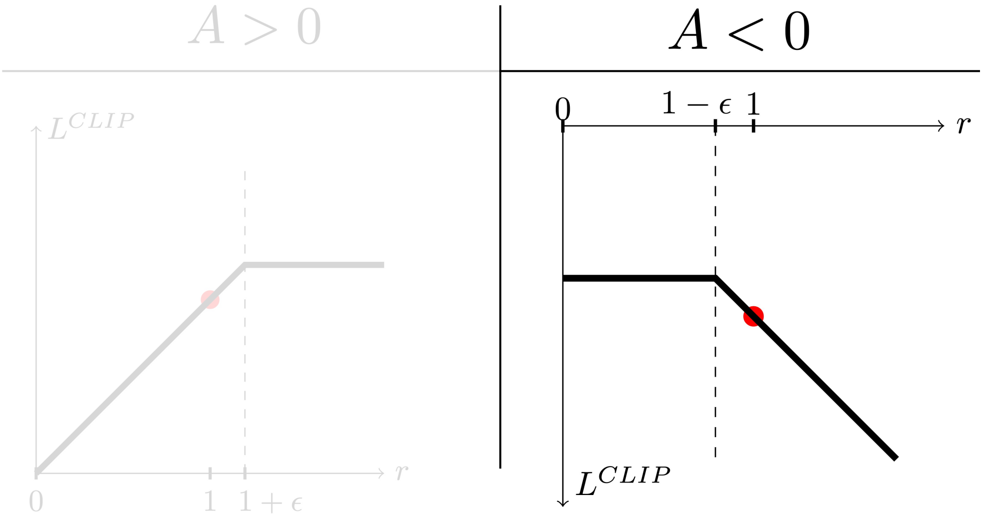
$A > 0$	$A < 0$
	

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

Variation 1: PPO-Clip

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

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



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

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

Variation 2: PPO-KL Penalty

Idea. Penalize difference in policy distributions

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

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

Variation 2: PPO-KL Penalty

Idea. Penalize difference in policy distributions

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

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

Variation 2: PPO-KL Penalty

Idea. Penalize difference in policy distributions

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

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**)

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

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 **of N**

Workaround if we don't want to do RL

BoN = **B**est **of** **N**

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

Workaround if we don't want to do RL

BoN = **B**est **of N**

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

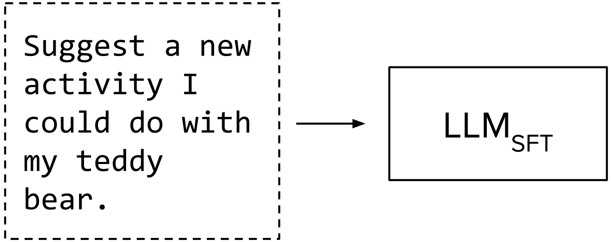
Strategy.

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

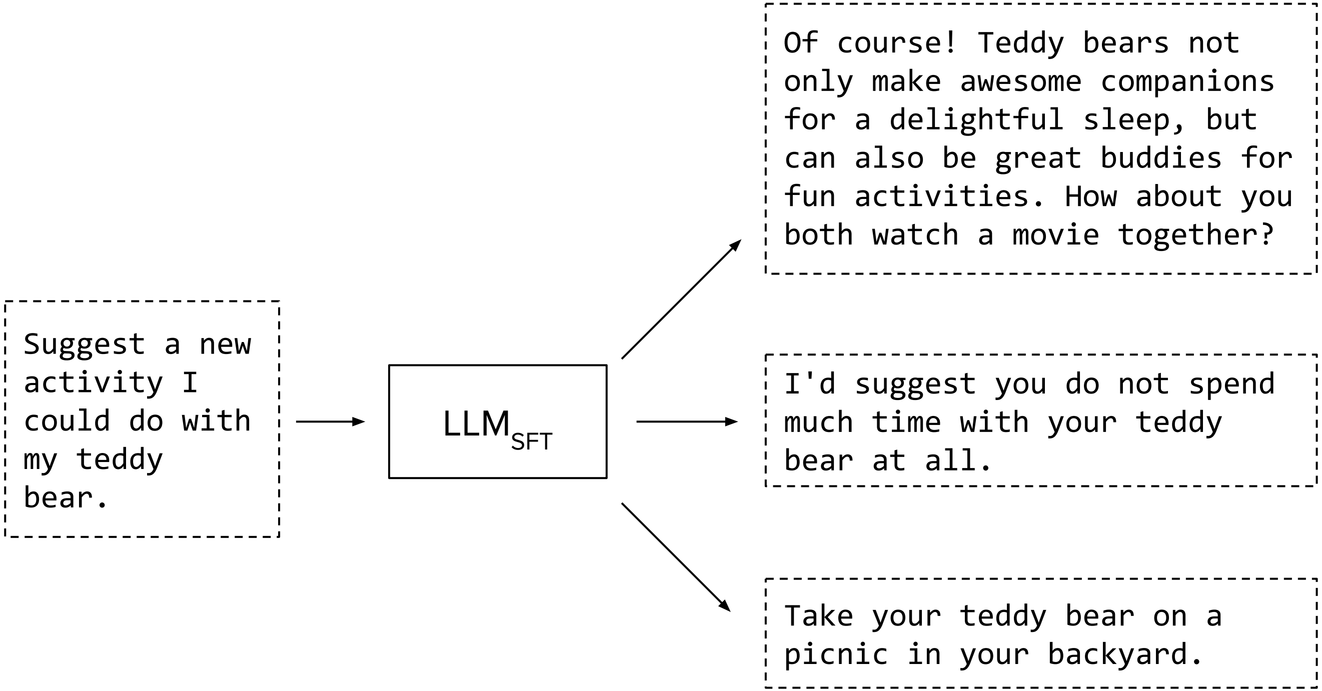
BoN in action

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

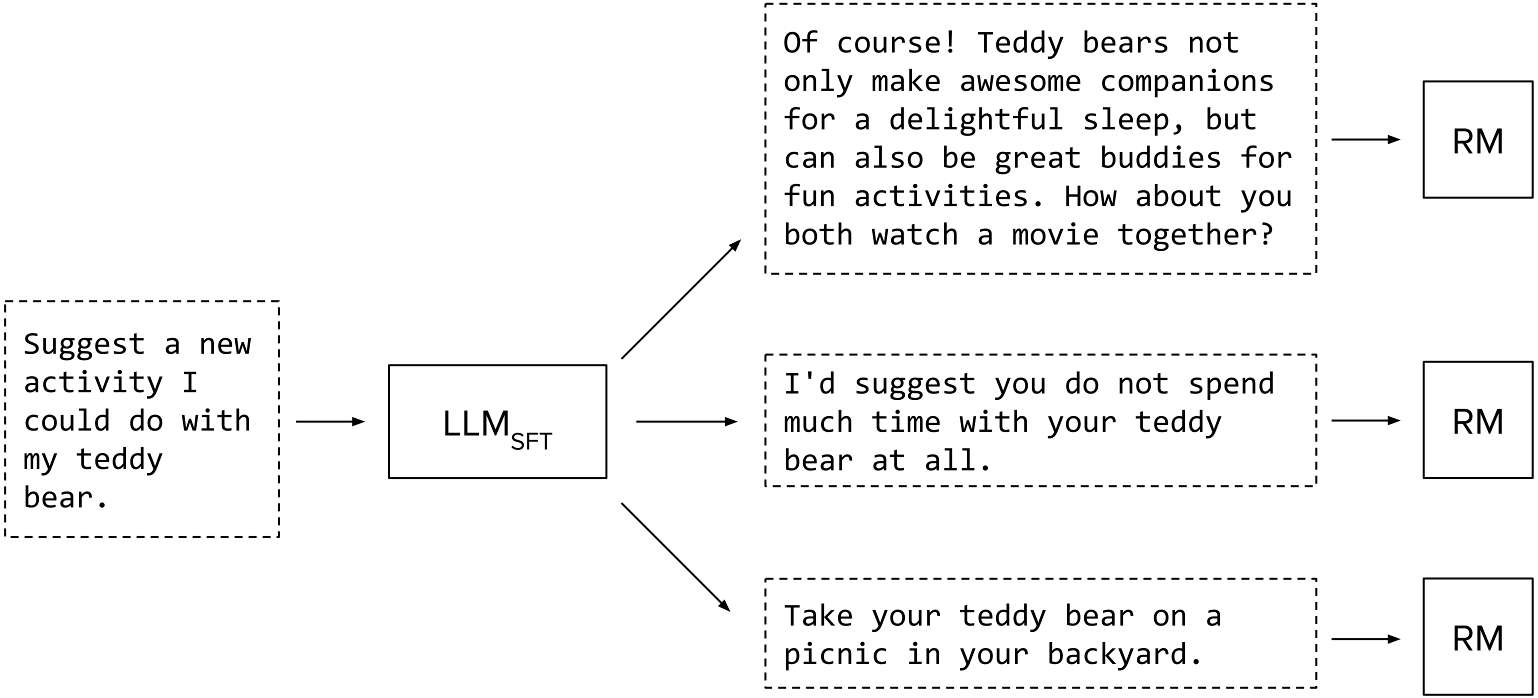
BoN in action



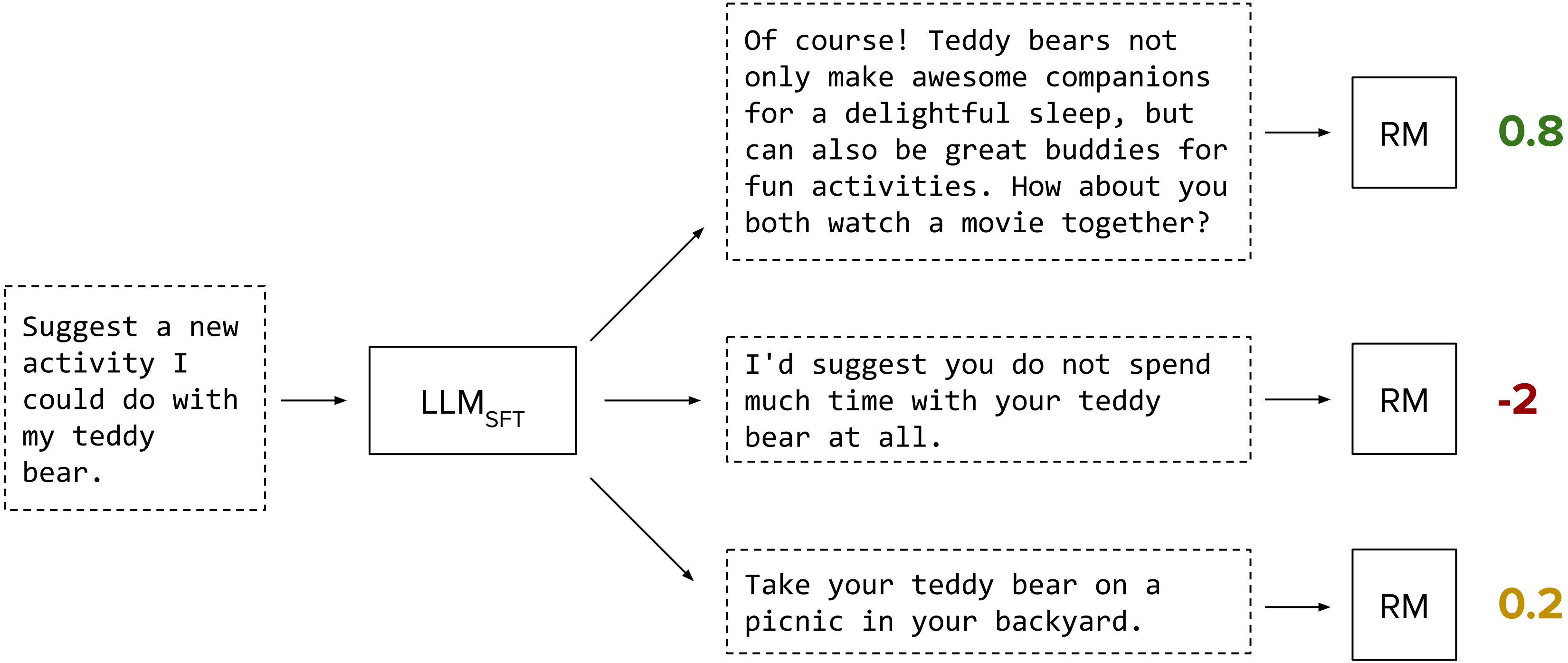
BoN in action



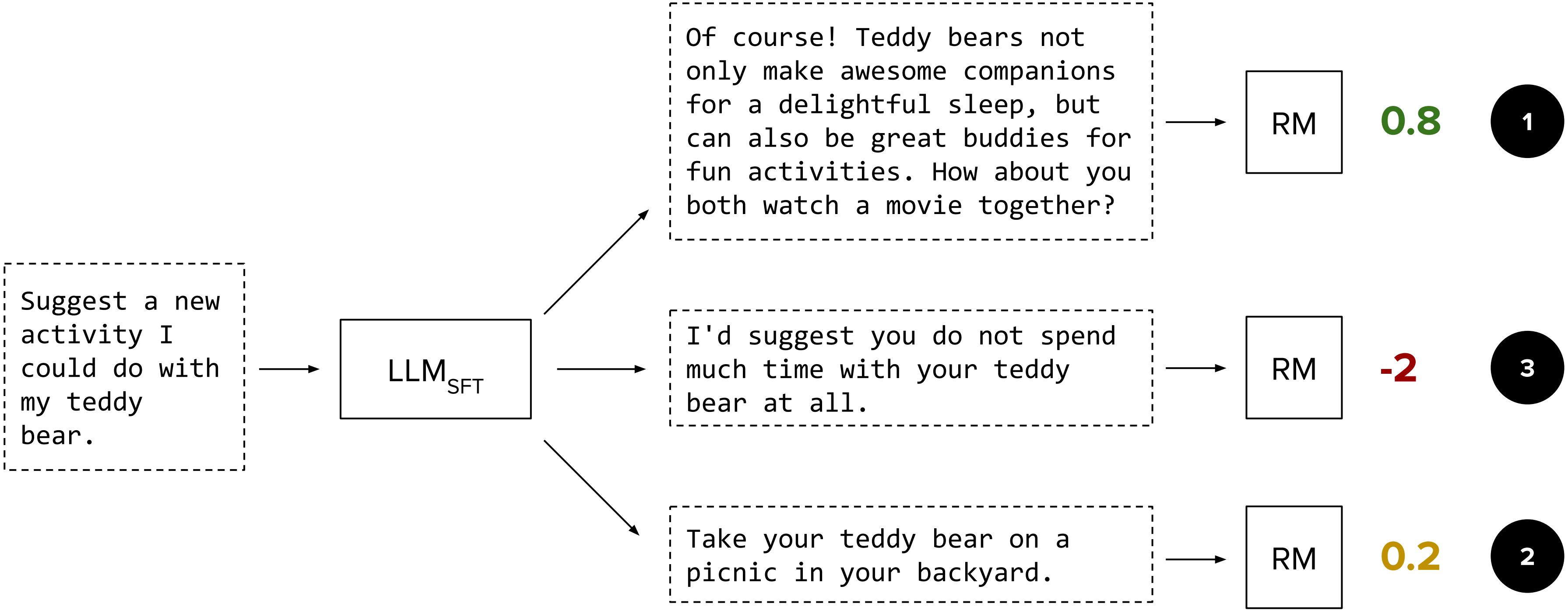
BoN in action



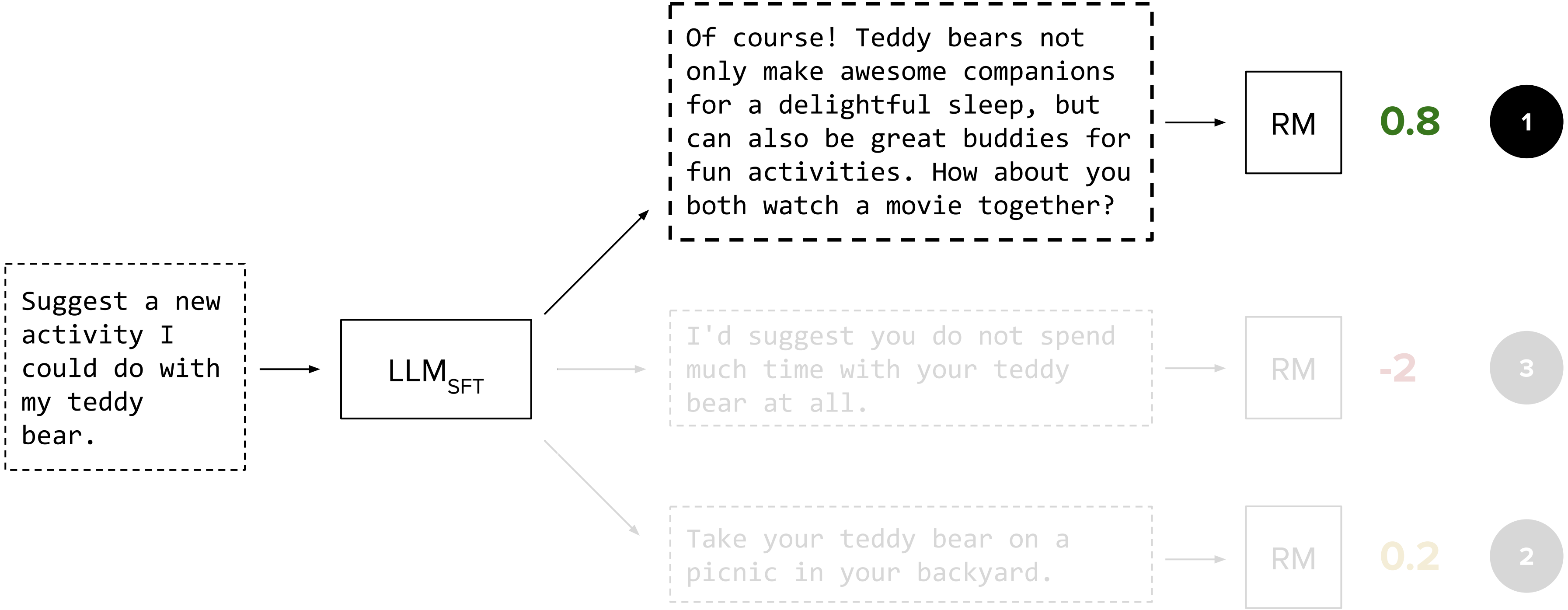
BoN in action



BoN in action



BoN in action



Transformers & Large Language Models

Preference tuning

Data collection

RLHF

DPO

Stanford University

ICME

Motivation

- Limitations using RL

$$L^{KLPEN}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\overset{1}{\boxed{\pi_\theta(a_t \mid s_t)}}}{\underset{2}{\boxed{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}}}\overset{3}{\underset{4}{\boxed{\hat{A}_t}}} - \beta\,\mathrm{KL}[\overset{2}{\boxed{\pi_{\theta_{\text{old}}}(\cdot \mid s_t)}}, \overset{1}{\boxed{\pi_\theta(\cdot \mid s_t)}}]\right]$$

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:

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

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

Supervised approach with DPO

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

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

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

No r(x, y)!

- No need to train a separate reward 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:

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

- 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:

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

- No need to train a separate reward model

$$r_\theta(x, y) = \beta \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$$

- Operates directly on preference data
- Similar to the Bradley-Terry formulation with a special kind of reward!

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**

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

Where does the DPO formulation come from?

1 Start from PPO objective

2 **Derive optimal policy**

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

"*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"**

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

"*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
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 • Needs extra models: reward model, value model, base model	• Supervised learning • Base model is the only extra model needed

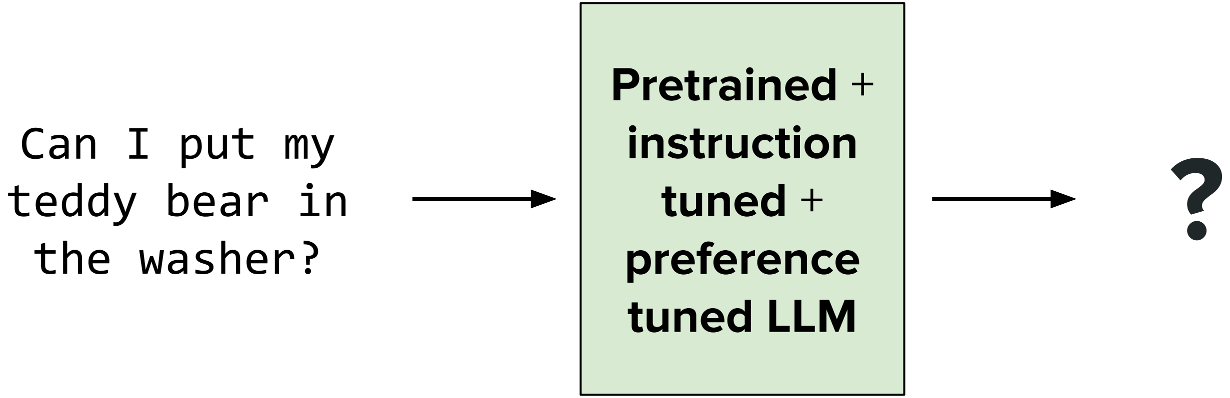
Performance. No common absolute consensus. Varies from task to task and sensitive to implementation.

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

Behavior

Can I put my teddy bear in the washer? → **Pretrained** + **instruction tuned LLM** → No, it might get damaged. Try hand washing instead.

Behavior



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!