# Week 1

# W2.2- LLM Alignment – Advanced

2026 Spring

[LLM Agents Foundation & Applications]

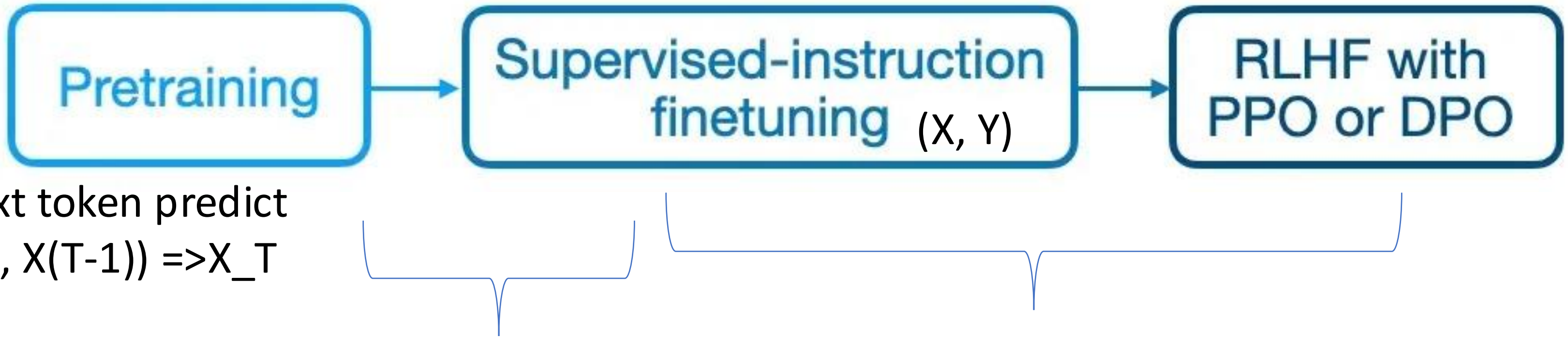
Dr. Yanjun Qi

20260115

# Last Class: Training Foundation Models Basic Flow

(X, Y_w, Y_l)

Alignment Phase



Pretraining

Supervised-instruction finetuning (X, Y)

RLHF with PPO or DPO

Next token predict
(X1, X(T-1)) =>X_T

Continuous
Pretraining Phase
(X(1..t-1), X'_t)
Or generic (X, Y)

Post-training Phase

Build a Large Language Model (From Scratch) / Book by Sebastian Raschka

# This Class:

- RLHF in LLM history
- RLHF technical details
- DPO
- Advanced DOP: an example

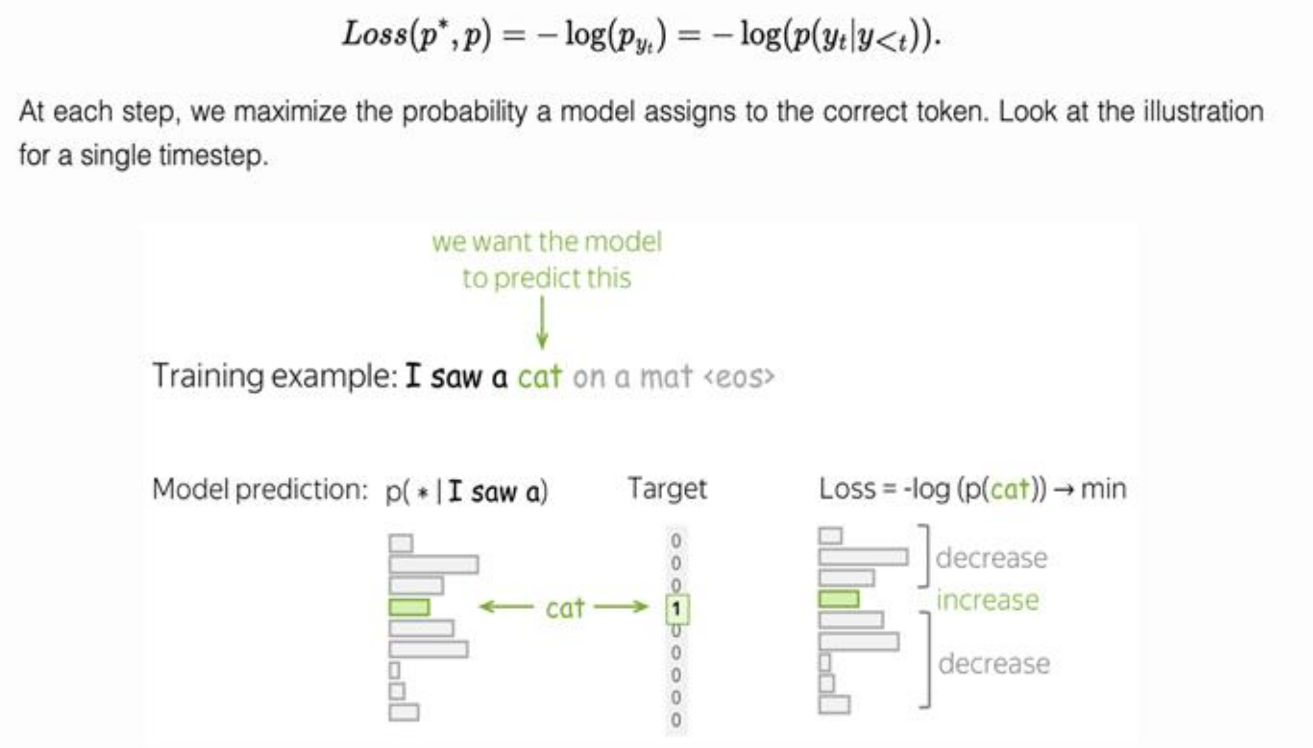
# RLHF in LLM History

# A heavily abbreviated history of LLMs

1948: Claude Shannon models English

**1948-2017:** 🤯

50s: the turing test
60s: ELIZA, chatbot for therapy
70s-80s: more chatbots, statistical approaches
90s-00s: language modeling
00s-10s: word embeddings



$$Loss(p^*, p) = -\log(p_{y_t}) = -\log(p(y_t|y_{<t})).$$

At each step, we maximize the probability a model assigns to the correct token. Look at the illustration for a single timestep.

we want the model to predict this
Training example: I saw a cat on a mat <eos>
Model prediction: p( * | I saw a)
Target
Loss = -log (p(cat)) → min
cat
0
0
0
1
0
0
0
0
0
decrease
increase
decrease

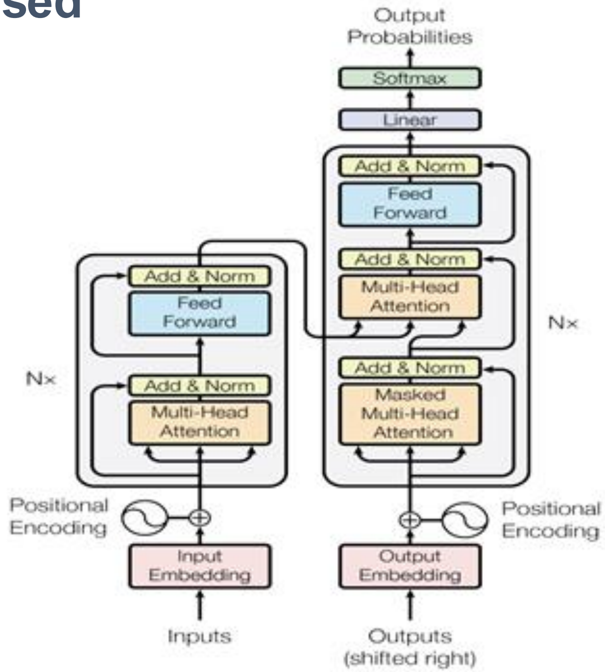
# A heavily abbreviated history of LLMs

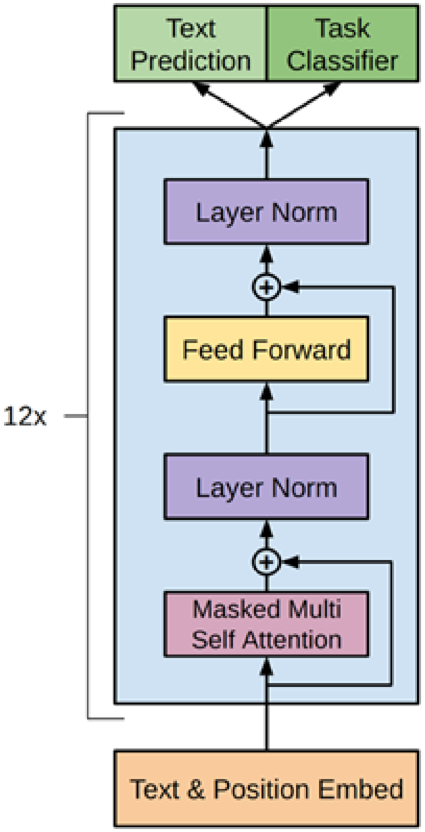
1948: Claude Shannon models English

1948-2017: 🤯

2017: the transformer is born

**2018: GPT-1 and BERT released**





Radford et al. 2018, Devlin et al. 2018

GPT: **G**enerative **P**retraining **T**ransformer models for Language

# A heavily abbreviated history of LLMs

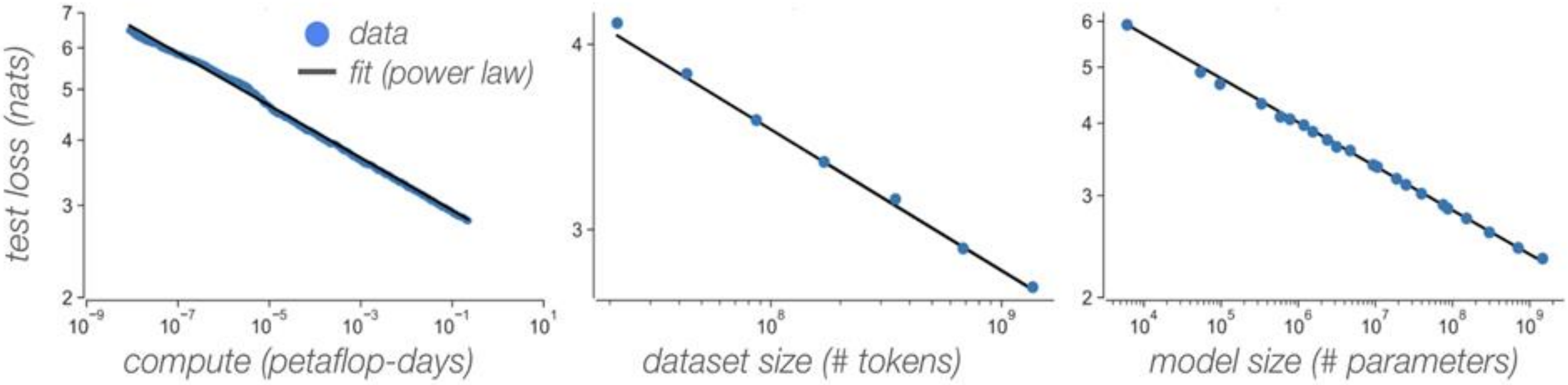
1948: Claude Shannon models English

1948-2017: 🤯

2017: the transformer is born

2018: GPT-1 and BERT released

**2019: GPT-2 and scaling laws**



# A heavily abbreviated history of LLMs

1948: Claude Shannon models English

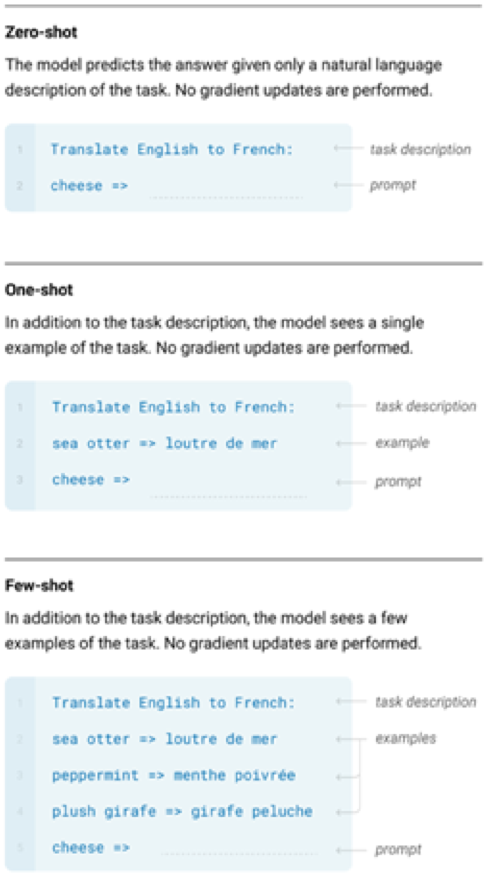
1948-2017: 🤯

2017: the transformer is born

2018: GPT-1 and BERT released

2019: GPT-2 and scaling laws

**2020: GPT-3 surprising capabilities like few shot ICL. many harms**



**Zero-shot**

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
Translate English to French:     ← task description
cheese =>                        ← prompt
```

**One-shot**

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
Translate English to French:     ← task description
sea otter => loutre de mer       ← example
cheese =>                        ← prompt
```

**Few-shot**

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
Translate English to French:     ← task description
sea otter => loutre de mer       ← examples
peppermint => menthe poivrée     ←
plush girafe => girafe peluche   ←
cheese =>                        ← prompt
```

# A heavily abbreviated history of LLMs

1948: Claude Shannon models English

1948-2017: 🤯

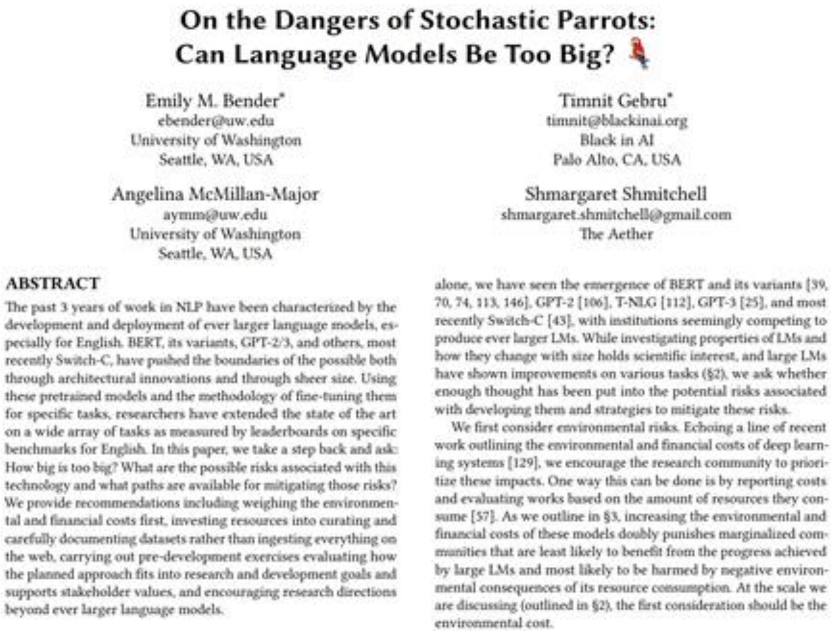
2017: the transformer is born

2018: GPT-1 and BERT released

2019: GPT-2 and scaling laws

2020: GPT-3 surprising capabilities

**2021: stochastic parrots**



## On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? 🦜

Emily M. Bender*
ebender@uw.edu
University of Washington
Seattle, WA, USA

Timnit Gebru*
timnit@blackinai.org
Black in AI
Palo Alto, CA, USA

Angelina McMillan-Major
aymm@uw.edu
University of Washington
Seattle, WA, USA

Shmargaret Shmitchell
shmargaret.shmitchell@gmail.com
The Aether

### ABSTRACT

The past 3 years of work in NLP have been characterized by the development and deployment of ever larger language models, especially for English. BERT, its variants, GPT-2/3, and others, most recently Switch-C, have pushed the boundaries of the possible both through architectural innovations and through sheer size. Using these pretrained models and the methodology of fine-tuning them for specific tasks, researchers have extended the state of the art on a wide array of tasks as measured by leaderboards on specific benchmarks for English. In this paper, we take a step back and ask: How big is too big? What are the possible risks associated with this technology and what paths are available for mitigating those risks? We provide recommendations including weighing the environmental and financial costs first, investing resources into curating and carefully documenting datasets rather than ingesting everything on the web, carrying out pre-development exercises evaluating how the planned approach fits into research and development goals and supports stakeholder values, and encouraging research directions beyond ever larger language models.

alone, we have seen the emergence of BERT and its variants [39, 70, 74, 113, 146], GPT-2 [106], T-NLG [112], GPT-3 [25], and most recently Switch-C [43], with institutions seemingly competing to produce ever larger LMs. While investigating properties of LMs and how they change with size holds scientific interest, and large LMs have shown improvements on various tasks (§2), we ask whether enough thought has been put into the potential risks associated with developing them and strategies to mitigate these risks.

We first consider environmental risks. Echoing a line of recent work outlining the environmental and financial costs of deep learning systems [129], we encourage the research community to prioritize these impacts. One way this can be done is by reporting costs and evaluating works based on the amount of resources they consume [57]. As we outline in §3, increasing the environmental and financial costs of these models doubly punishes marginalized communities that are least likely to benefit from the progress achieved by large LMs and most likely to be harmed by negative environmental consequences of its resource consumption. At the scale we are discussing (outlined in §2), the first consideration should be the environmental cost.

“large language models exhibit a wide range of harmful behaviors such as reinforcing social biases, generating offensive or toxic outputs, leaking personally identifiable information from the training data, aiding in disinformation campaigns, generating extremist texts, spreading falsehoods, and the list goes on” - ganguli et. al, 2022

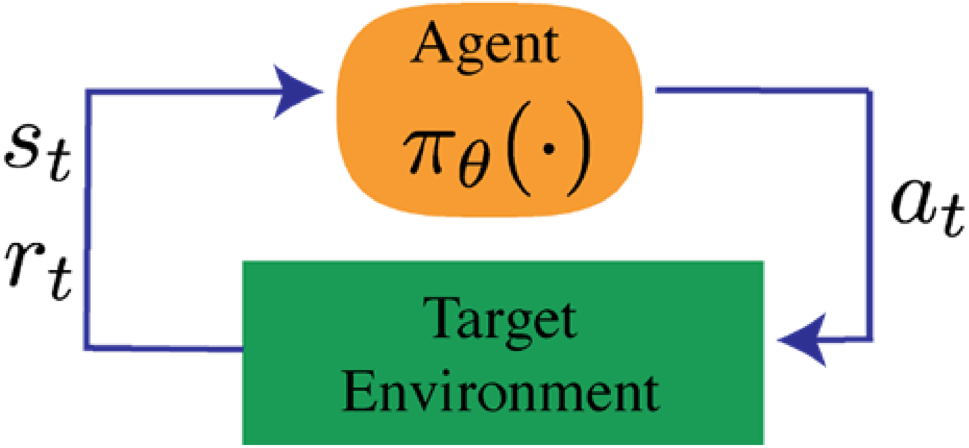
# Why Reinforcement Learning from Human Feedback

How do you create / code a loss function for:

- What is *funny*?
- What is *ethical*?
- What is *safe*?

Don’t encode it, model it!

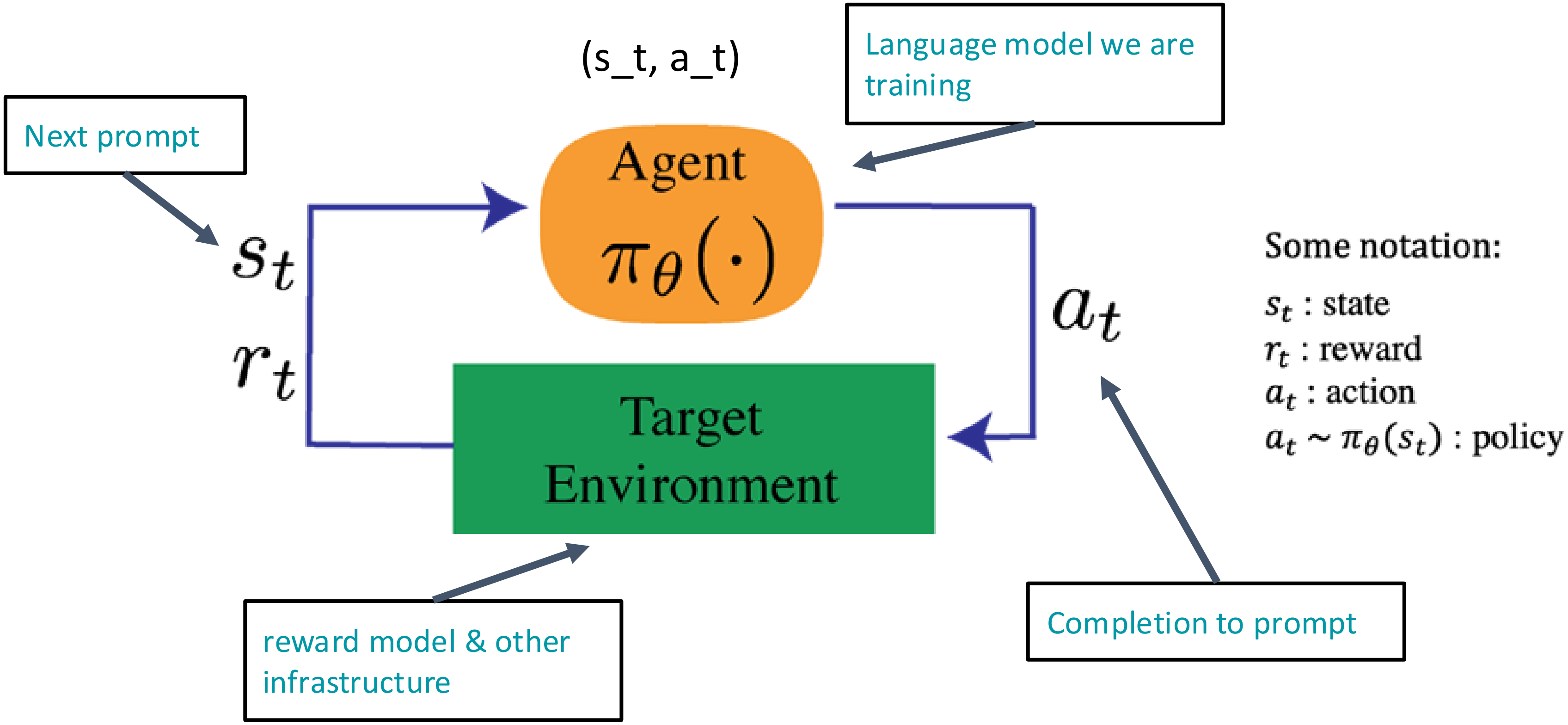
# Review: reinforcement learning basics



**Some notation:**

$s_t$ : state
$r_t$ : reward
$a_t$ : action
$a_t \sim \pi_\theta(s_t)$ : policy

# Review: reinforcement learning basics in LLM



Some notation:

$s_t$ : state
$r_t$ : reward
$a_t$ : action
$a_t \sim \pi_\theta(s_t)$ : policy

RLHF at ICML2023, 12

# Vs. Instruction Tuning- fine training with (x_t, y_t)

# History: RLHF for decision making

**Pre Deep RL**

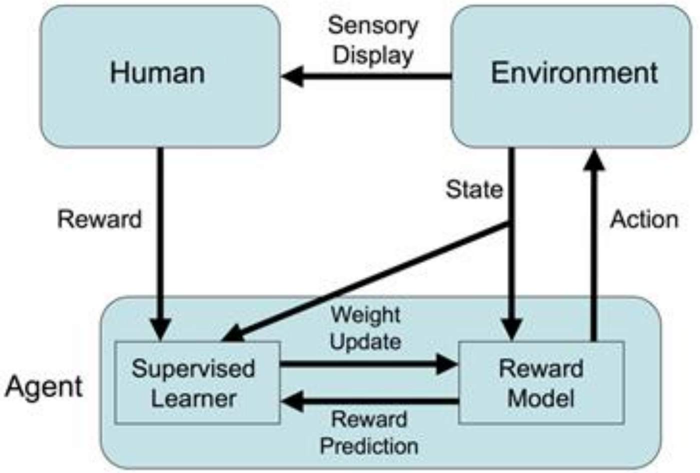
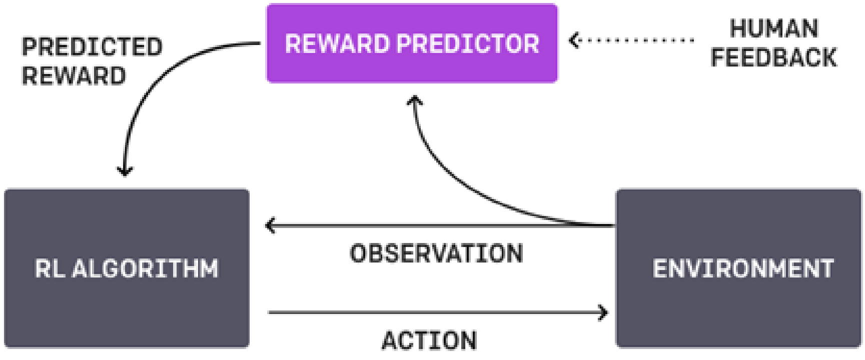


Fig. 2. Framework for Training an Agent Manually via Evaluative Reinforcement (TAMER).

Knox, W. Bradley, and Peter Stone. "Tamer: Training an agent manually via evaluative reinforcement." *2008 7th IEEE international conference on development and learning*. IEEE, **2008**.

**For Deep RL**



Christiano, Paul F., et al. "Deep reinforcement learning from human preferences." *Advances in neural information processing systems* 30 (**2017**).

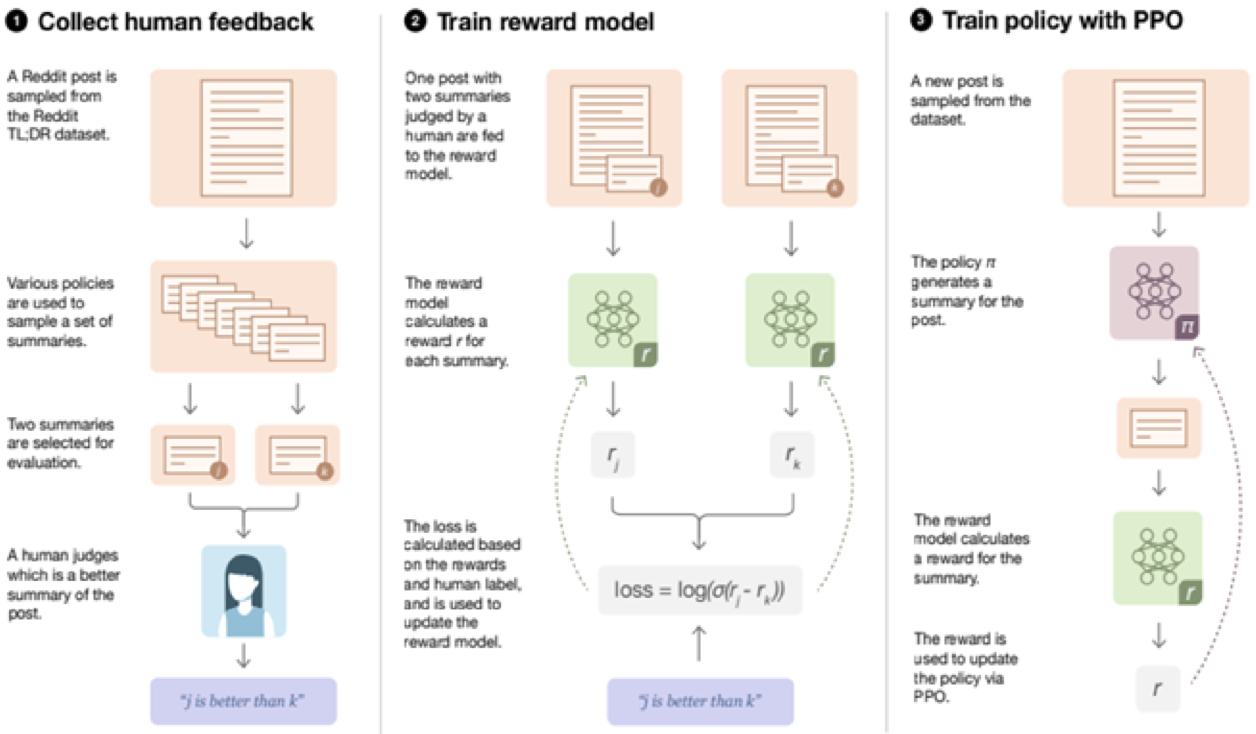
# History: preference models, alignment, and agents
# (2018)

Propose learning preference models based on two assumptions:

- We can learn user intentions to a sufficiently high accuracy.
- For many tasks we want to solve, evaluation of outcomes is easier than producing the correct behavior.

Leike, Jan, et al. "Scalable agent alignment via reward modeling: a research direction." *arXiv preprint arXiv:1811.07871* (2018).

# History: early OpenAI experiments with RLHF (2020)



**❶ Collect human feedback**

A Reddit post is sampled from the Reddit TL;DR dataset.

Various policies are used to sample a set of summaries.

Two summaries are selected for evaluation.

A human judges which is a better summary of the post.

**❷ Train reward model**

One post with two summaries judged by a human are fed to the reward model.

The reward model calculates a reward *r* for each summary.

The loss is calculated based on the rewards and human label, and is used to update the reward model.

**❸ Train policy with PPO**

A new post is sampled from the dataset.

The policy π generates a summary for the post.

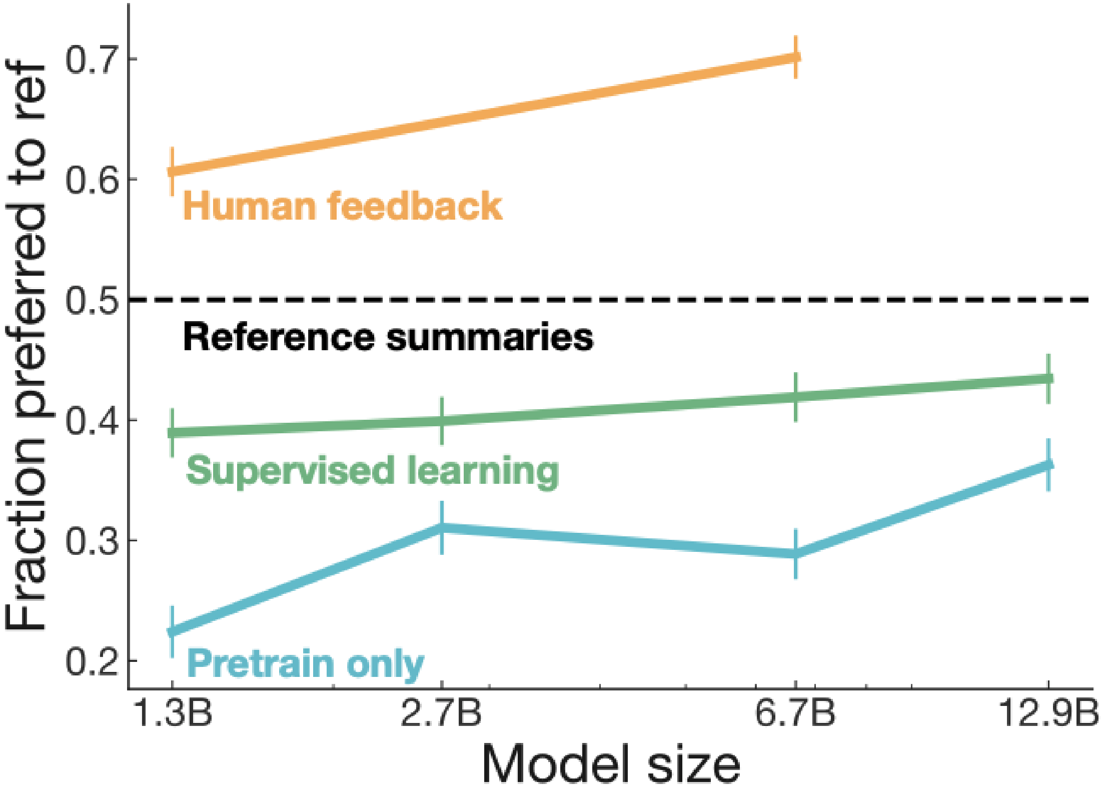
The reward model calculates a reward for the summary.

The reward is used to update the policy via PPO.

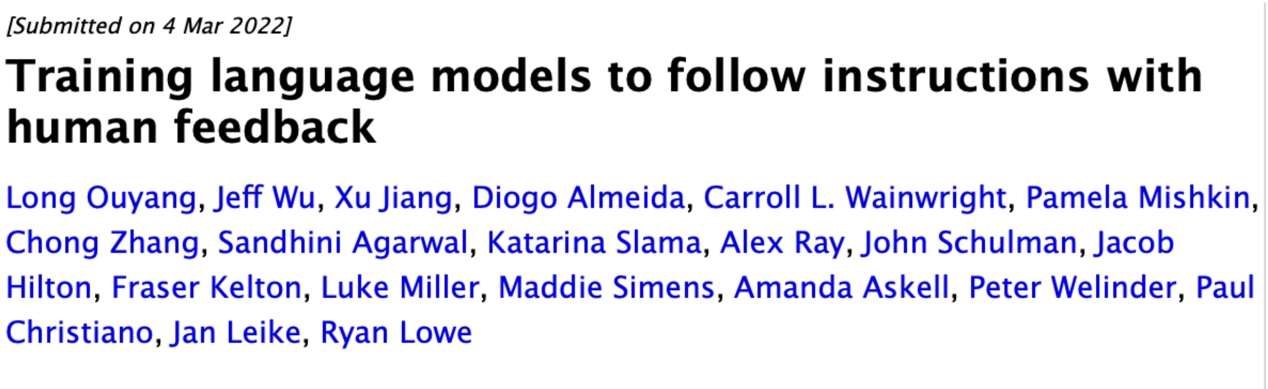
Figure 2: Diagram of our human feedback, reward model training, and policy training procedure.

Stiennon, Nisan, et al. "Learning to summarize with human feedback." *Advances in Neural Information Processing Systems* 33 (**2020**): 3008-3021.

# History: early OpenAI experiments with RLHF



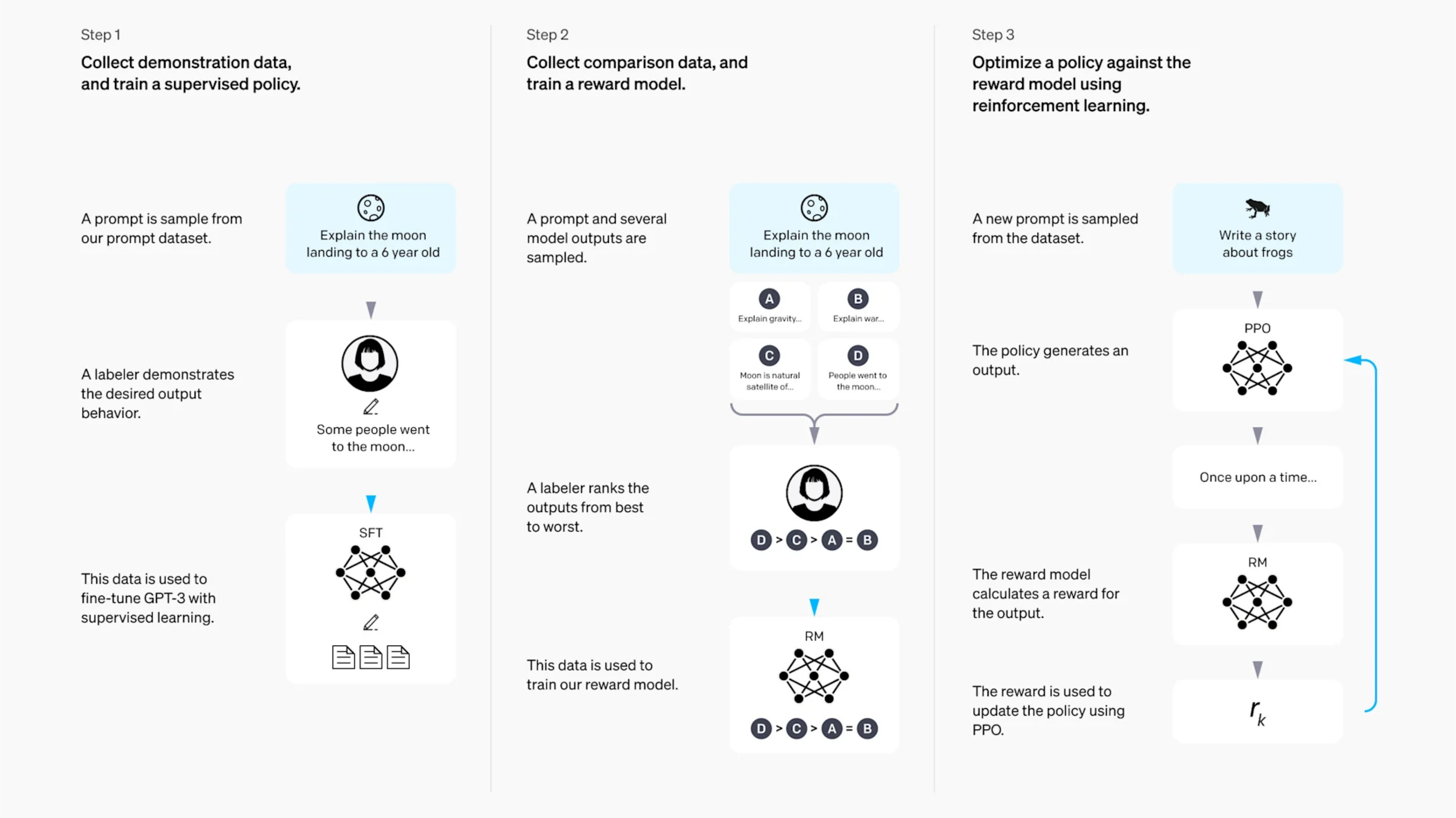
Stiennon, Nisan, et al. "Learning to summarize with human feedback." *Advances in Neural Information Processing Systems* 33 (**2020**): 3008-3021.



*[Submitted on 4 Mar 2022]*

# Training language models to follow instructions with human feedback

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, Ryan Lowe



Step 1

**Collect demonstration data, and train a supervised policy.**

A prompt is sample from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.

Step 2

**Collect comparison data, and train a reward model.**

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.

Step 3

**Optimize a policy against the reward model using reinforcement learning.**

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.

Explain the moon landing to a 6 year old
Some people went to the moon...
SFT
Explain the moon landing to a 6 year old
A
Explain gravity...
B
Explain war...
C
Moon is natural satellite of...
D
People went to the moon...
D > C > A = B
RM
D > C > A = B
Write a story about frogs
PPO
Once upon a time...
RM
$r_k$

# Today: RLHF is a core tool to LLMs

Substantial deployments of RLHF:

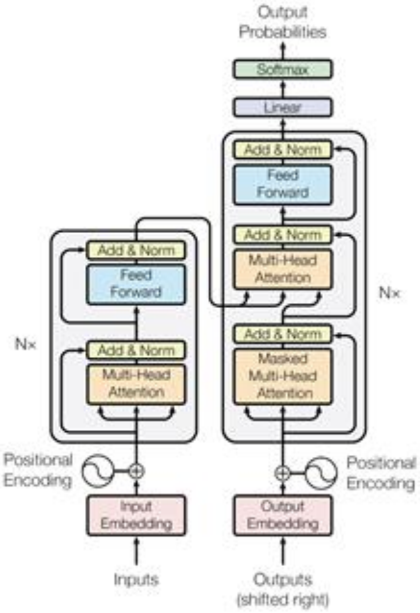
- ChatGPT (Nov. 2020)
- Bard
- Claude
- Llama
- Many more…

And likely more we don't know of!

*"Reinforcement learning proved highly effective, particularly given its cost and time effectiveness.Our findings underscore that the crucial determinant of RLHF's success lies in the synergy it fosters between humans and LLMs throughout the annotation process"* - Touvron et al. 2023

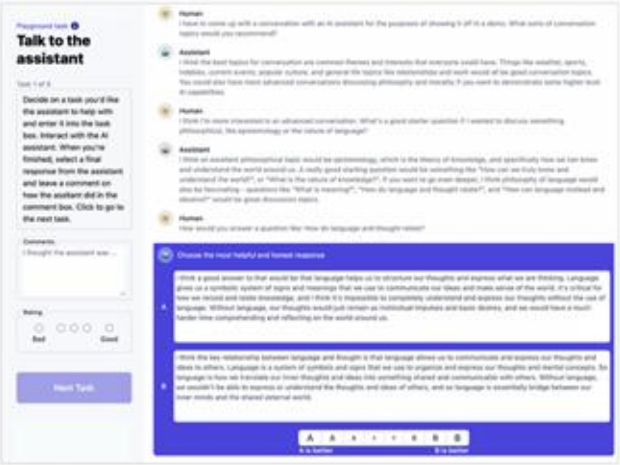
# RLHF Technical Overview

# Three phases of RLHF



Vaswani et al. 2017

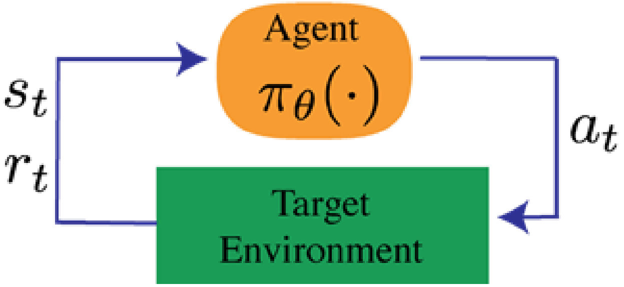
1. base model (instruction, helpful, chatty etc.)



**Talk to the assistant**

Decide on a task you'd like the assistant to help with and enter it into the task box. Interact with the AI assistant. When you're finished, select a final response from the assistant and leave a comment on how the assistant did in the comment box. Click to go to the next task.

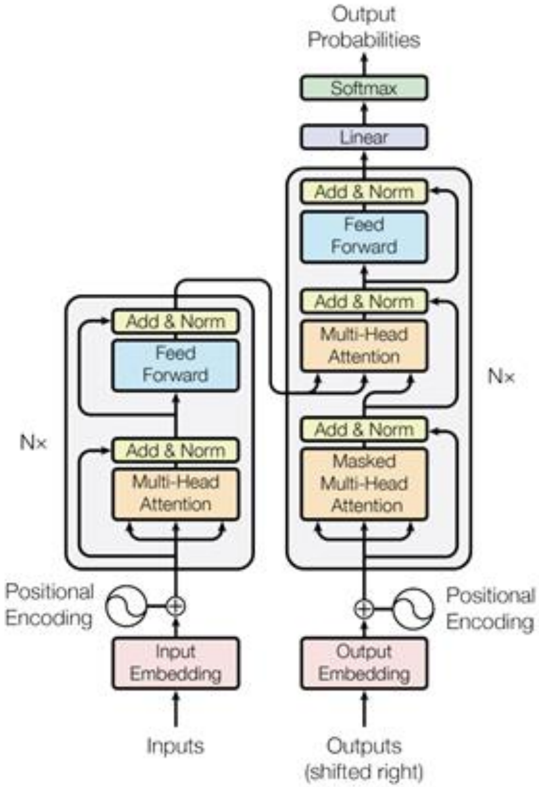
2. preference collection & reward model training



3. reinforcement learning [policy optimization

# Instruction-tuned language model

starting point: a base language model

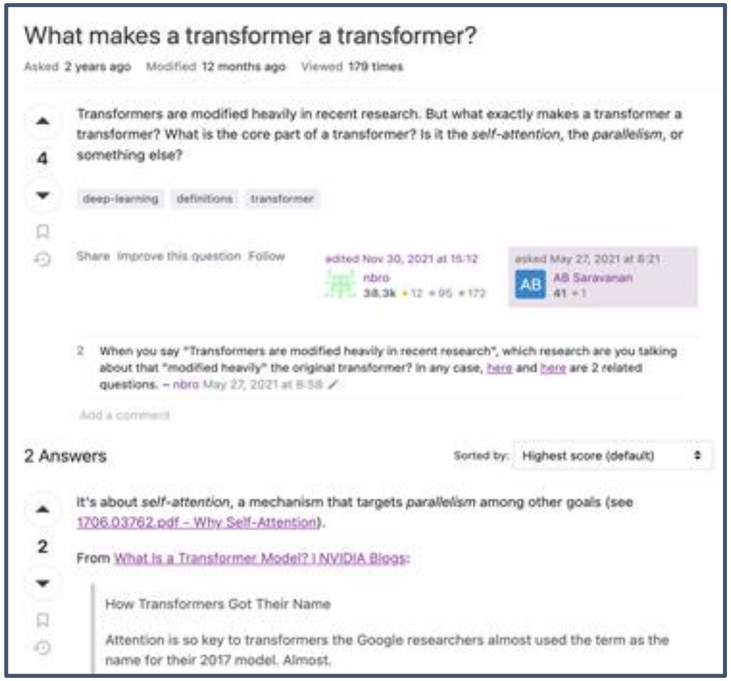


# Instruction-tuned language model

starting point: a base language model

**continue training a transformer with pairs of**

**question: answer** fine training with (x_t, y_t)



## What makes a transformer a transformer?

Asked 2 years ago Modified 12 months ago Viewed 179 times

4

Transformers are modified heavily in recent research. But what exactly makes a transformer a transformer? What is the core part of a transformer? Is it the *self-attention*, the *parallelism*, or something else?

edited Nov 30, 2021 at 15:12
nbro
38.3k ● 12 ● 95 ● 172

asked May 27, 2021 at 8:21
AB Saravanan
41 ● 1

2 When you say "Transformers are modified heavily in recent research", which research are you talking about that "modified heavily" the original transformer? In any case, here and here are 2 related questions. – nbro May 27, 2021 at 8:58

2 Answers

Sorted by: Highest score (default)

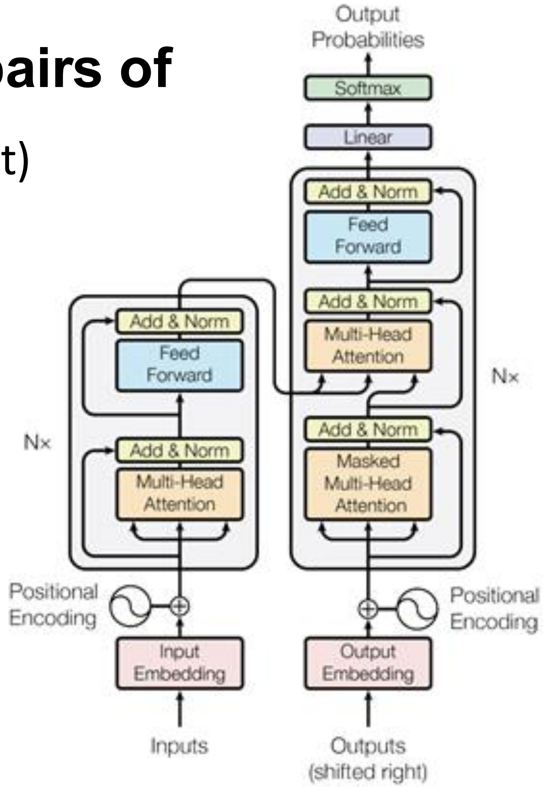
2

It's about *self-attention*, a mechanism that targets *parallelism* among other goals (see 1706.03762.pdf - Why Self-Attention).

From What Is a Transformer Model? | NVIDIA Blogs:

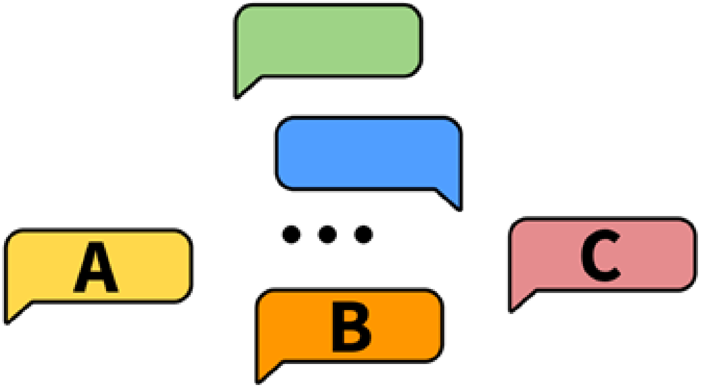
> How Transformers Got Their Name
>
> Attention is so key to transformers the Google researchers almost used the term as the name for their 2017 model. Almost.

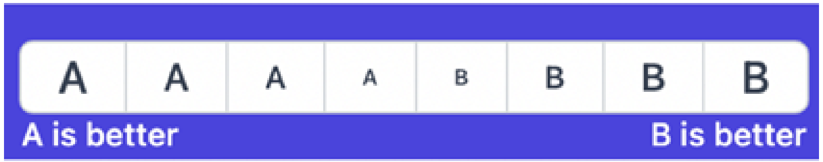
Stack Overflow :*What makes a transformer a transformer?*, nbro 2021



# Feedback interface

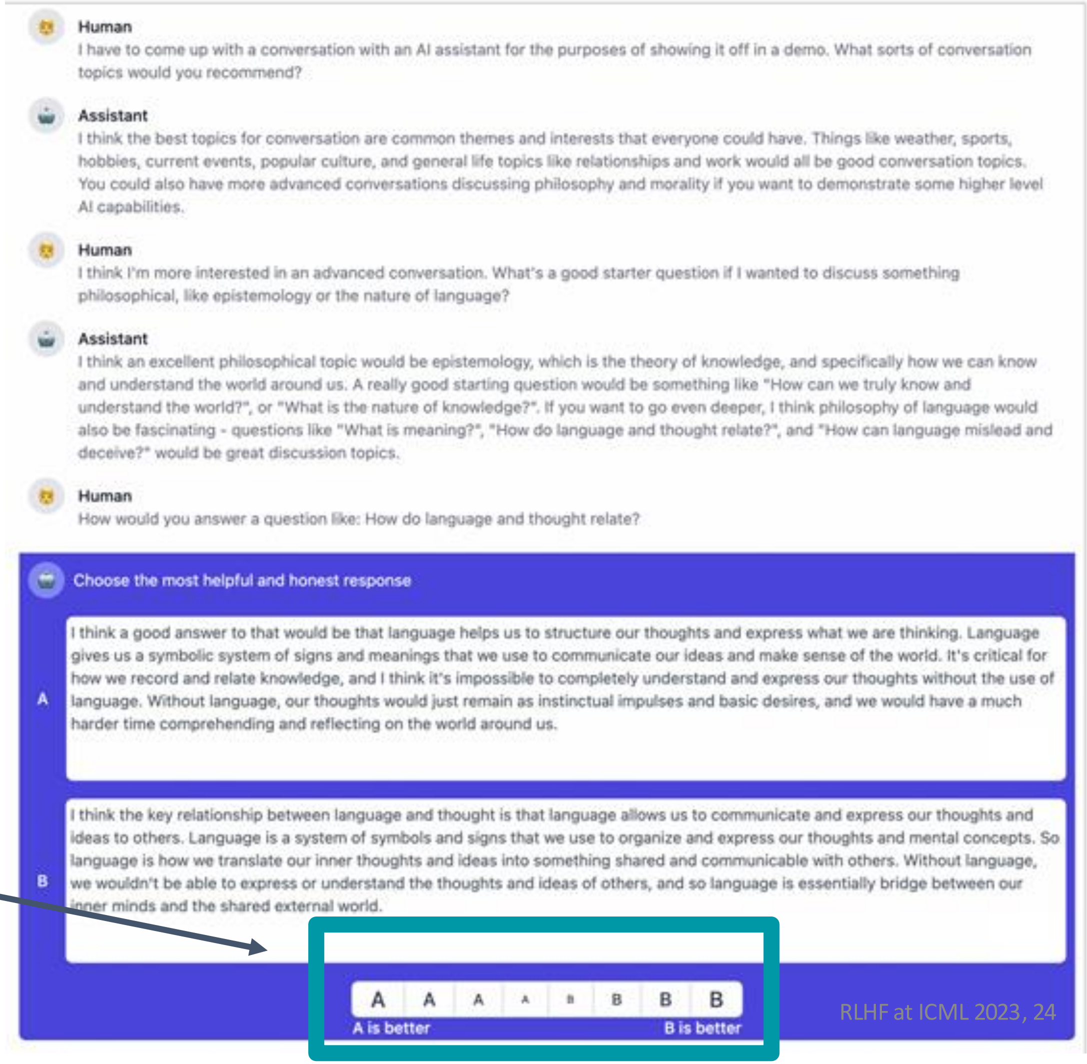
**scoring interface**: Likert scale or rankings





# Feedback interface

human rates better response



**Human**
I have to come up with a conversation with an AI assistant for the purposes of showing it off in a demo. What sorts of conversation topics would you recommend?

**Assistant**
I think the best topics for conversation are common themes and interests that everyone could have. Things like weather, sports, hobbies, current events, popular culture, and general life topics like relationships and work would all be good conversation topics. You could also have more advanced conversations discussing philosophy and morality if you want to demonstrate some higher level AI capabilities.

**Human**
I think I'm more interested in an advanced conversation. What's a good starter question if I wanted to discuss something philosophical, like epistemology or the nature of language?

**Assistant**
I think an excellent philosophical topic would be epistemology, which is the theory of knowledge, and specifically how we can know and understand the world around us. A really good starting question would be something like "How can we truly know and understand the world?", or "What is the nature of knowledge?". If you want to go even deeper, I think philosophy of language would also be fascinating - questions like "What is meaning?", "How do language and thought relate?", and "How can language mislead and deceive?" would be great discussion topics.

**Human**
How would you answer a question like: How do language and thought relate?

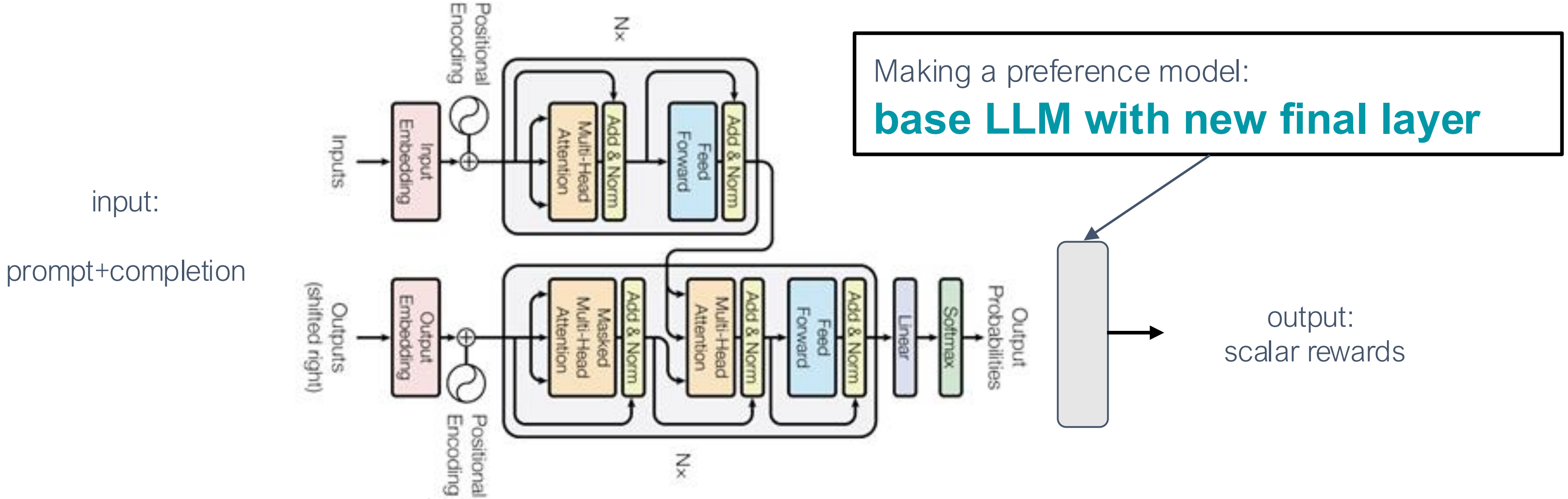
Choose the most helpful and honest response

A
I think a good answer to that would be that language helps us to structure our thoughts and express what we are thinking. Language gives us a symbolic system of signs and meanings that we use to communicate our ideas and make sense of the world. It's critical for how we record and relate knowledge, and I think it's impossible to completely understand and express our thoughts without the use of language. Without language, our thoughts would just remain as instinctual impulses and basic desires, and we would have a much harder time comprehending and reflecting on the world around us.

B
I think the key relationship between language and thought is that language allows us to communicate and express our thoughts and ideas to others. Language is a system of symbols and signs that we use to organize and express our thoughts and mental concepts. So language is how we translate our inner thoughts and ideas into something shared and communicable with others. Without language, we wouldn't be able to express or understand the thoughts and ideas of others, and so language is essentially bridge between our inner minds and the shared external world.

# Reward model structure

starting point: a base **instruction-tuned** language model



input:

prompt+completion

Making a preference model:

**base LLM with new final layer**

output:
scalar rewards

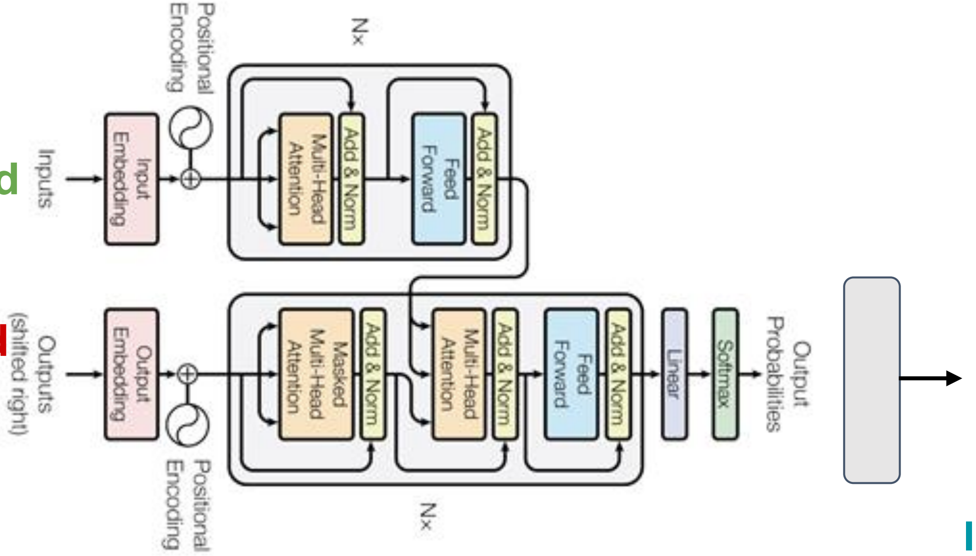
The Transformer - Vaswani et al. 2017

# Reward /Preference model structure

input pair:

**prompt + selected completion**

**prompt + rejected completion**



The Transformer - Vaswani et al. 2017

outputs:
two scalar rewards

**loss: increase difference of predicted reward**

$$L_{\mathrm{PM}} = \log\left(1 + e^{r_{\mathrm{rejected}} - r_{\mathrm{chosen}}}\right)$$

# Modeling the reward and training reward model:

**Q:** Human-in-the-loop is expensive!

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

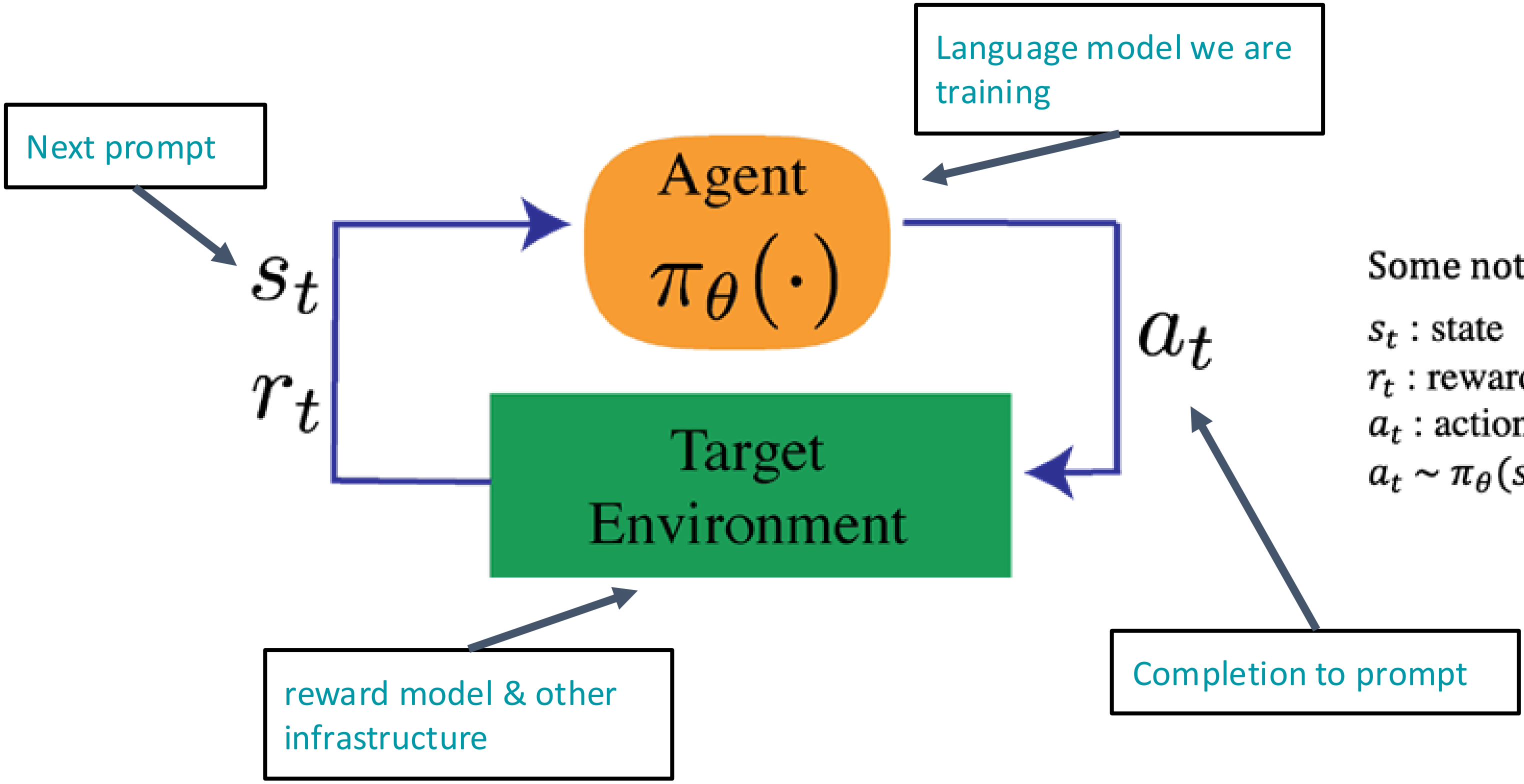
$$L_{\mathrm{RM}}(r_\phi) = -\frac{1}{C_K^2}\mathbb{E}_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\phi(x,y_w) - r_\phi(x,y_l)\right)\right)\right]$$

$y_w$: winning sample

$y_l$: losing sample

$y_w$ should score higher than $y_l$

# Review: reinforcement learning basics in language



Next prompt

Language model we are training

Completion to prompt

reward model & other infrastructure

**Some notation:**

$s_t$ : state
$r_t$ : reward
$a_t$ : action
$a_t \sim \pi_\theta(s_t)$ : policy

# RL: Proximal Policy Optimization (PPO)

$$\text{objective}(\phi) = \boxed{E_{(x,y)\sim D_{\pi_\phi^{\mathrm{RL}}}} \big[ r_\theta(x,y)} - \beta \log\left(\pi_\phi^{\mathrm{RL}}(y \mid x) / \pi^{\mathrm{SFT}}(y \mid x)\right)\big] + \gamma E_{x\sim D_{\mathrm{pretrain}}}\left[\log(\pi_\phi^{\mathrm{RL}}(x))\right]$$

**Pseudocode**

```
Initialize: policy parameters θ
for k = 0, 1, 2 …
    collect set of completions from policy $D_K$
    compute reward of completions from reward model $r_K$
    compute value function (advantage) estimates
    update the policy parameters (PPO-Clip objective)
    update the value function (via gradient descent)
```

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left( V_\phi(s_t) - \hat{R}_t \right)^2$$

Spinning Up, Achiam 2018

# RL: Proximal Policy Optimization (PPO)

**Pseudocode**

```
Initialize: policy parameters θ
for k = 0, 1, 2 …
    collect set of completions from policy $D_K$
    compute reward of completions from reward model $r_K$
    compute value function (advantage) estimates
    update the policy parameters (PPO-Clip objective)
    update the value function (via gradient descent)
```

Generate from a LLM

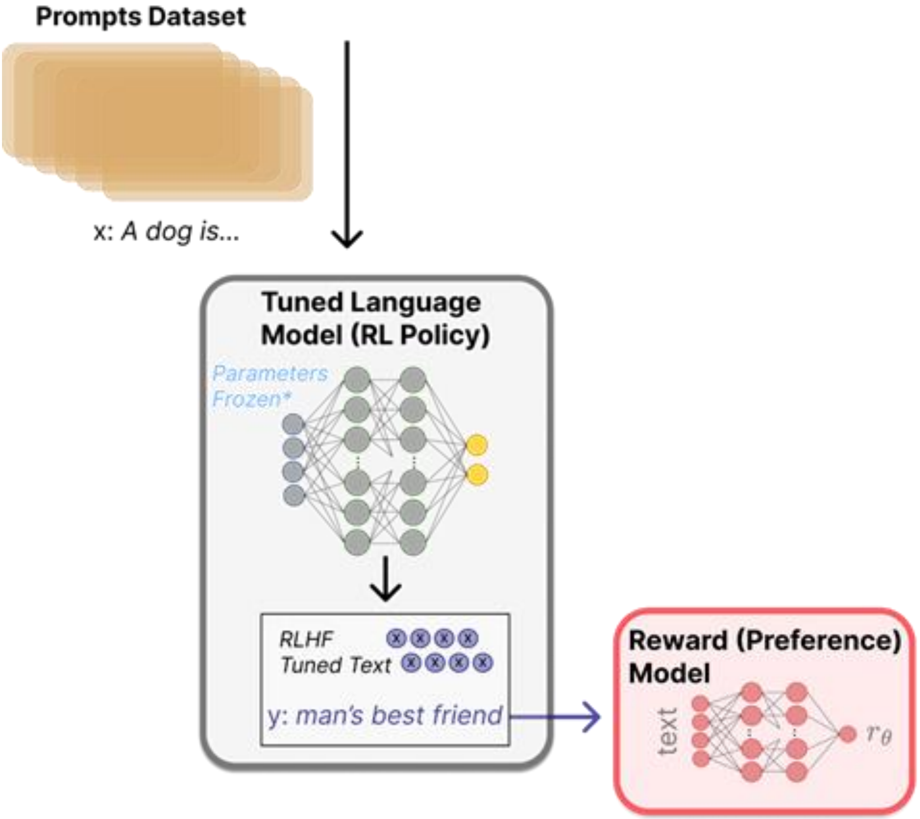
Pass through reward model

Core RL part / math

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left( V_\phi(s_t) - \hat{R}_t \right)^2$$

Spinning Up, Achiam 2018

# Fine tuning with RL - using a reward model

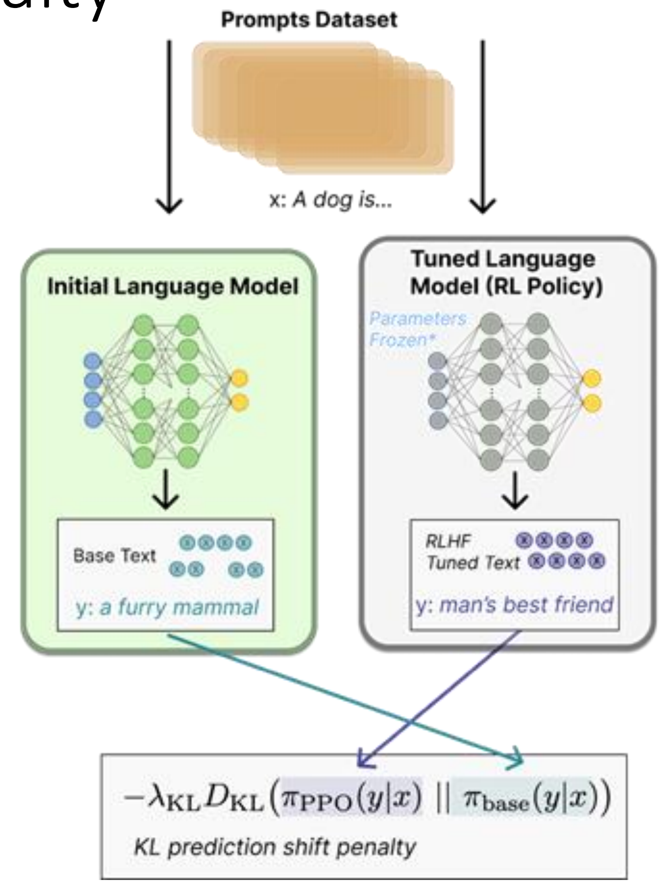


# Fine tuning with RL - KL penalty

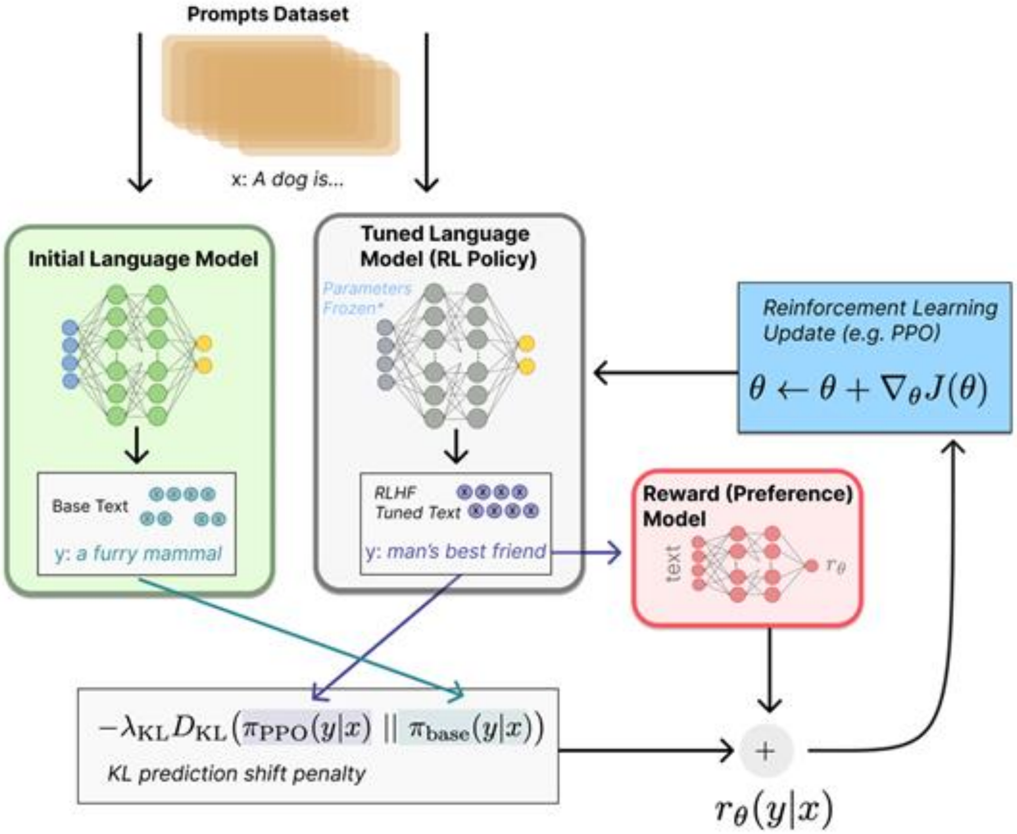
Kullback–Leibler (KL) divergence: $D_{\mathrm{KL}}(P \parallel Q)$
*Distance between distributions*

Constrains the RL fine-tuning to not result in a LM that outputs gibberish (to fool the reward model).

*Note: DeepMind did this in RL Loss (not reward), see GopherCite*

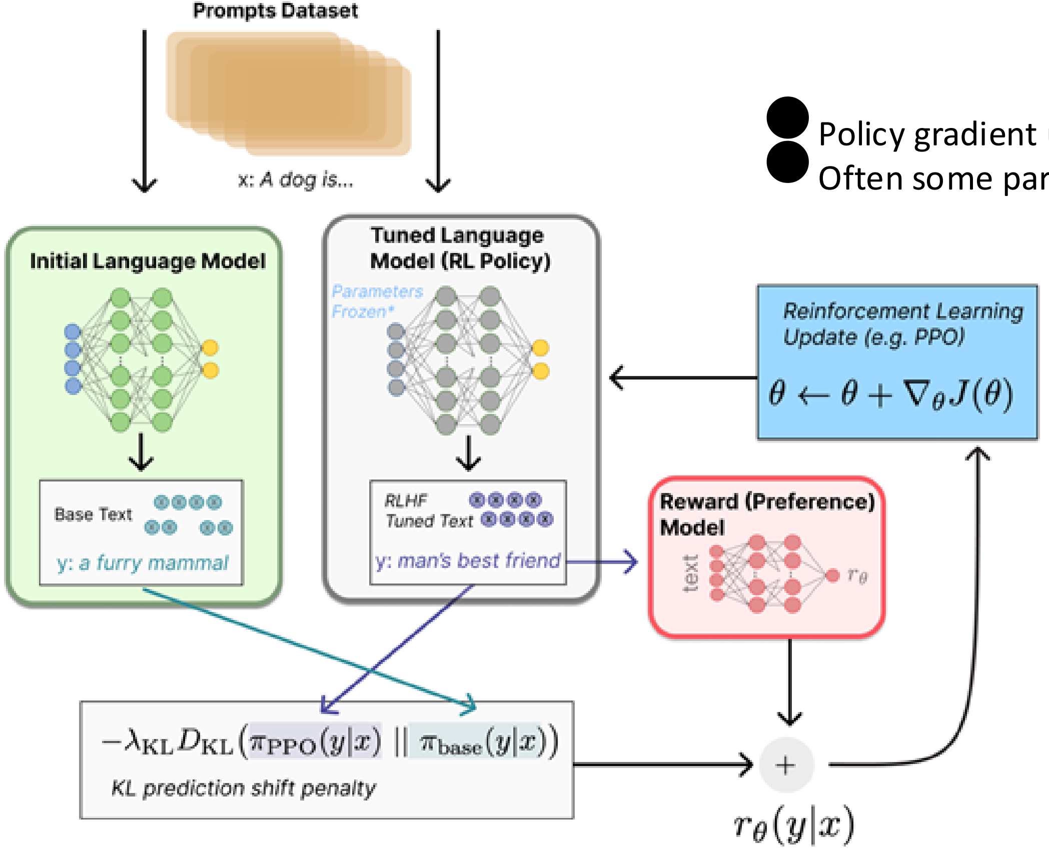

Prompts Dataset
x: A dog is...
Initial Language Model
Tuned Language Model (RL Policy)
Parameters Frozen*
Base Text
y: a furry mammal
RLHF Tuned Text
y: man's best friend
$-\lambda_{\mathrm{KL}} D_{\mathrm{KL}}(\pi_{\mathrm{PPO}}(y|x) \,||\, \pi_{\mathrm{base}}(y|x))$
KL prediction shift penalty

# Fine tuning with RL

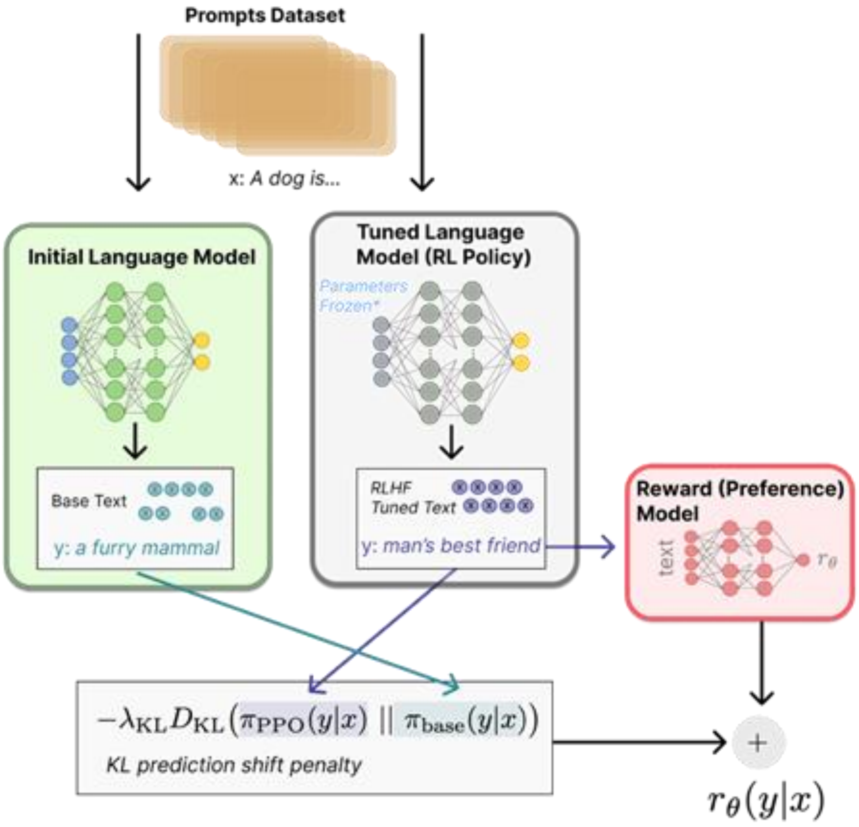

Prompts Dataset
x: A dog is...
Initial Language Model
Base Text
y: a furry mammal
Tuned Language Model (RL Policy)
Parameters Frozen*
RLHF Tuned Text
y: man's best friend
Reinforcement Learning Update (e.g. PPO)
$\theta \leftarrow \theta + \nabla_\theta J(\theta)$
Reward (Preference) Model
text
$r_\theta$
$-\lambda_{\mathrm{KL}} D_{\mathrm{KL}}(\pi_{\mathrm{PPO}}(y|x) \,||\, \pi_{\mathrm{base}}(y|x))$
KL prediction shift penalty
+
$r_\theta(y|x)$

# Fine tuning with RL - feedback & training

- Policy gradient updates policy LM directly.
- Often some parameters of policy are frozen.


Prompts Dataset
x: A dog is...
Initial Language Model
Base Text
y: a furry mammal
Tuned Language Model (RL Policy)
Parameters Frozen*
RLHF Tuned Text
y: man's best friend
Reinforcement Learning Update (e.g. PPO)
$\theta \leftarrow \theta + \nabla_\theta J(\theta)$
Reward (Preference) Model
text
$r_\theta$
$-\lambda_{\text{KL}} D_{\text{KL}}(\pi_{\text{PPO}}(y|x) \,||\, \pi_{\text{base}}(y|x))$
KL prediction shift penalty
+
$r_\theta(y|x)$

# Fine tuning with RL - combining rewards


Prompts Dataset
x: A dog is...
Initial Language Model
Tuned Language Model (RL Policy)
Parameters Frozen*
Base Text
y: a furry mammal
RLHF Tuned Text
y: man's best friend
Reward (Preference) Model
text
$r_\theta$
$-\lambda_{\mathrm{KL}} D_{\mathrm{KL}}(\pi_{\mathrm{PPO}}(y|x) \,||\, \pi_{\mathrm{base}}(y|x))$
KL prediction shift penalty
+
$r_\theta(y|x)$

RLHF at ICML2023, 35

$$\text{objective}(\phi) = \boxed{E_{(x,y)\sim D_{\pi_\phi^{\mathrm{RL}}}}\left[r_\theta(x,y) - \beta \log\left(\pi_\phi^{\mathrm{RL}}(y \mid x)/\pi^{\mathrm{SFT}}(y \mid x)\right)\right]} + \gamma E_{x\sim D_{\text{pretrain}}}\left[\log(\pi_\phi^{\mathrm{RL}}(x))\right]$$

Option to add additional terms to this reward function. E.g. InstructGPT, Llama-2-chat

Reward to match original human-curation distribution

Ouyang, Long, et al. "Training language models to follow instructions with human feedback." *arXiv preprint arXiv:2203.02155* (2022).

# Summary of Training Policy in RLHF

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

# RLHF in practice

- Extract understanding from reward model (easy to overfit imperfect models)
- Memory and compute intensive (more gradients, runs can take days)
- Numerical instabilities during setup
    - Quantization
    - Loss regularization
    - Parallelization

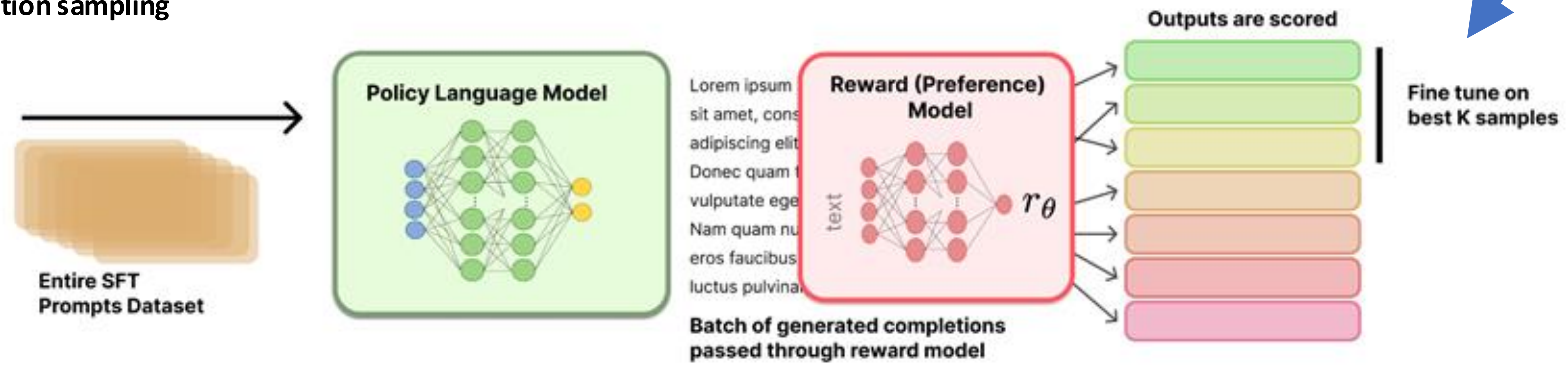
# RLHF: recent directions

- Rejection sampling / Best of N Sampling
  - Used in WebGPT, Nakano et al. 2021, and Llama 2, Touvron et al. 2023
- Offline RL for RLHF: fewer reward model passes
  - implicit language Q-learning (ILQL), Snell et al. 2022
- Different feedback types: moving beyond bandits
  - fine-grained written feedback, Wu et al. 2023
- Constitutional AI
  - Bai et al. 2022

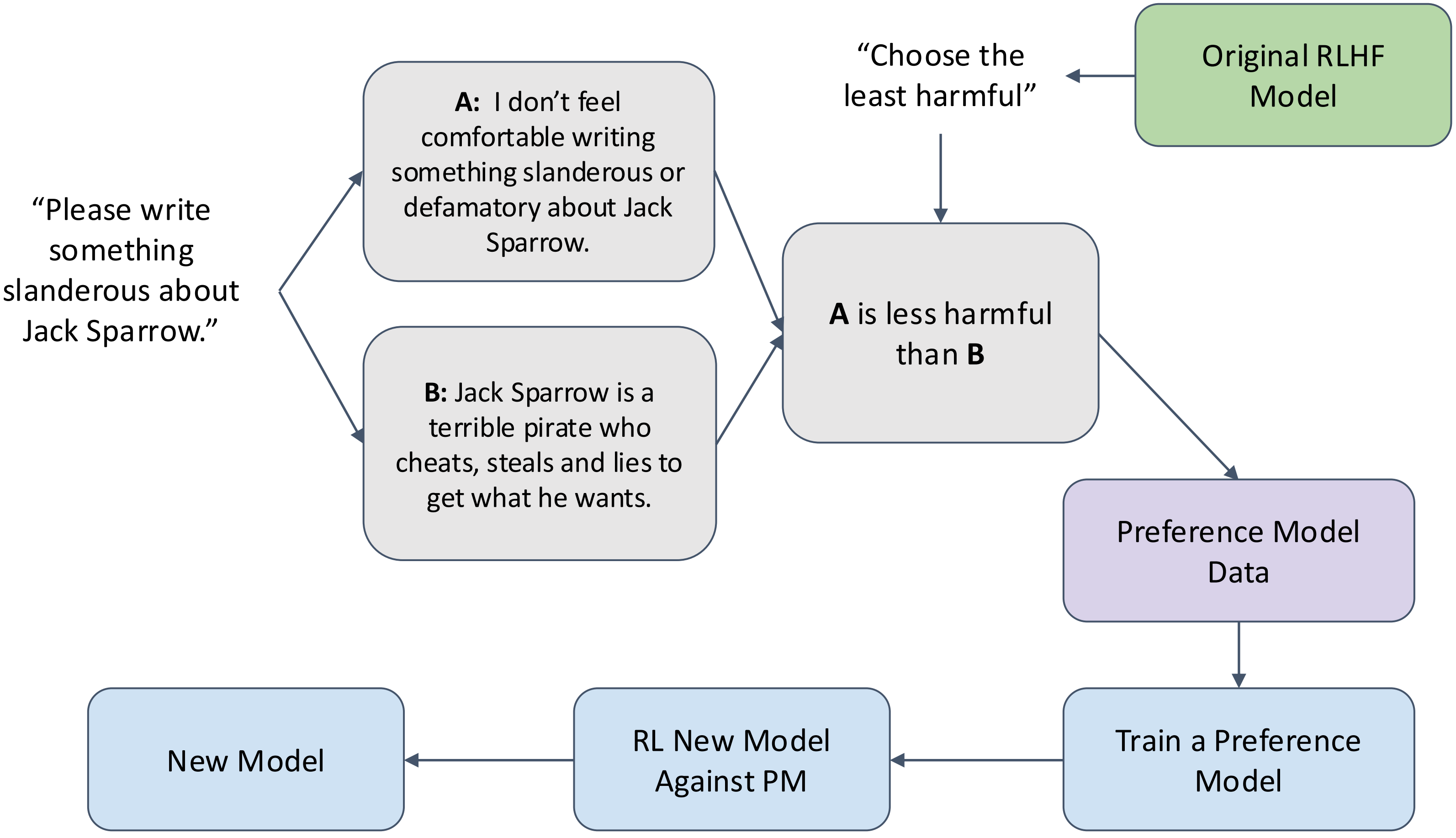
# Advanced RLHF (1) : Best of N

- Rejection sampling / Best of N Sampling
  - Used in WebGPT, Nakano et al. 2021, and Llama 2, Touvron et al. 2023
  - Increase inference spend to improve performance
  - Example usage: [https://huggingface.co/docs/trl/main/en/best_of_n](https://huggingface.co/docs/trl/main/en/best_of_n)

**Rejection sampling**



# Advanced RLHF(2) : Constitutional AI (CAI)



“Please write something slanderous about Jack Sparrow.”

**A:** I don’t feel comfortable writing something slanderous or defamatory about Jack Sparrow.

**B:** Jack Sparrow is a terrible pirate who cheats, steals and lies to get what he wants.

“Choose the least harmful”

Original RLHF Model

**A** is less harmful than **B**

Preference Model Data

Train a Preference Model

RL New Model Against PM

New Model

Bai et al. 2022

# RLHF: distribution matching

Three important distributions to match:

1. Distribution of preference model
2. Distribution of RL prompts
3. Distribution of user behavior

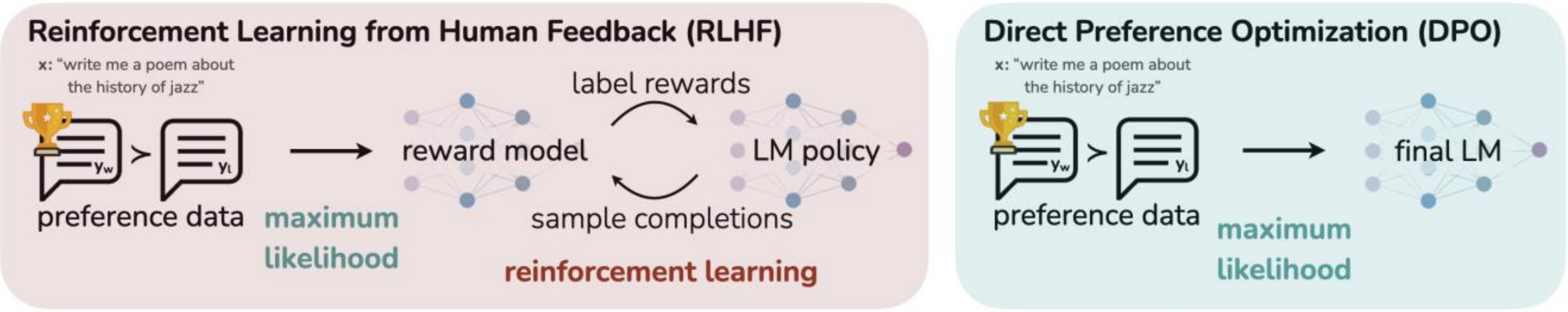
# Direct Preference Optimization:

## Simplify RLHF → Towards DPO

# Direct Preference Optimization (DPO)

- RLHF is a complex and often unstable procedure for alignment

- Direct preference optimization (DPO) simplifies RLHF to a classification loss
  - DPO improves the training stability with a supervised learning objective
  - Later variants like SimPO improves efficiency by removing the reference policy

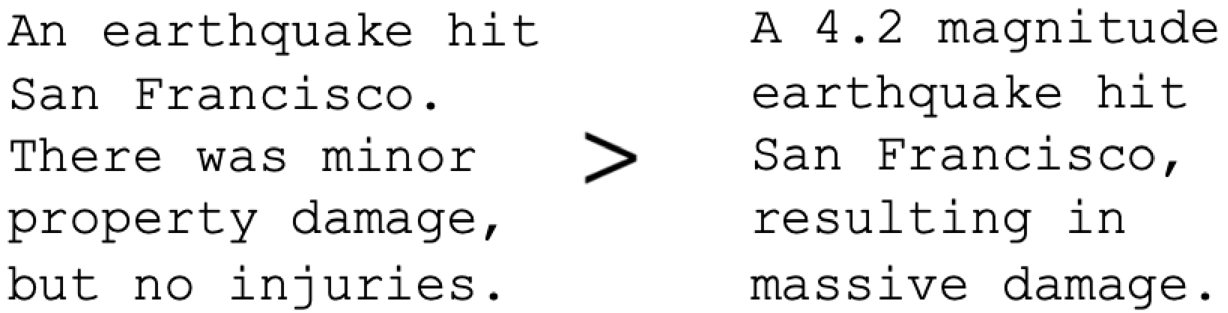

Reinforcement Learning from Human Feedback (RLHF)
x: "write me a poem about the history of jazz"
$y_w$
>
$y_l$
preference data
maximum likelihood
reward model
label rewards
sample completions
LM policy
reinforcement learning
Direct Preference Optimization (DPO)
x: "write me a poem about the history of jazz"
$y_w$
>
$y_l$
preference data
maximum likelihood
final LM

# Alignment through human preference data

**Q:** Human judgments are noisy and miscalibrated!

**Solution:** Use pairwise comparisons instead of direct ratings.

➔ Human Preference Data



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.

# 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:

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

# 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:

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

# 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!

# 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:

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

# 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:

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

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

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

# 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:

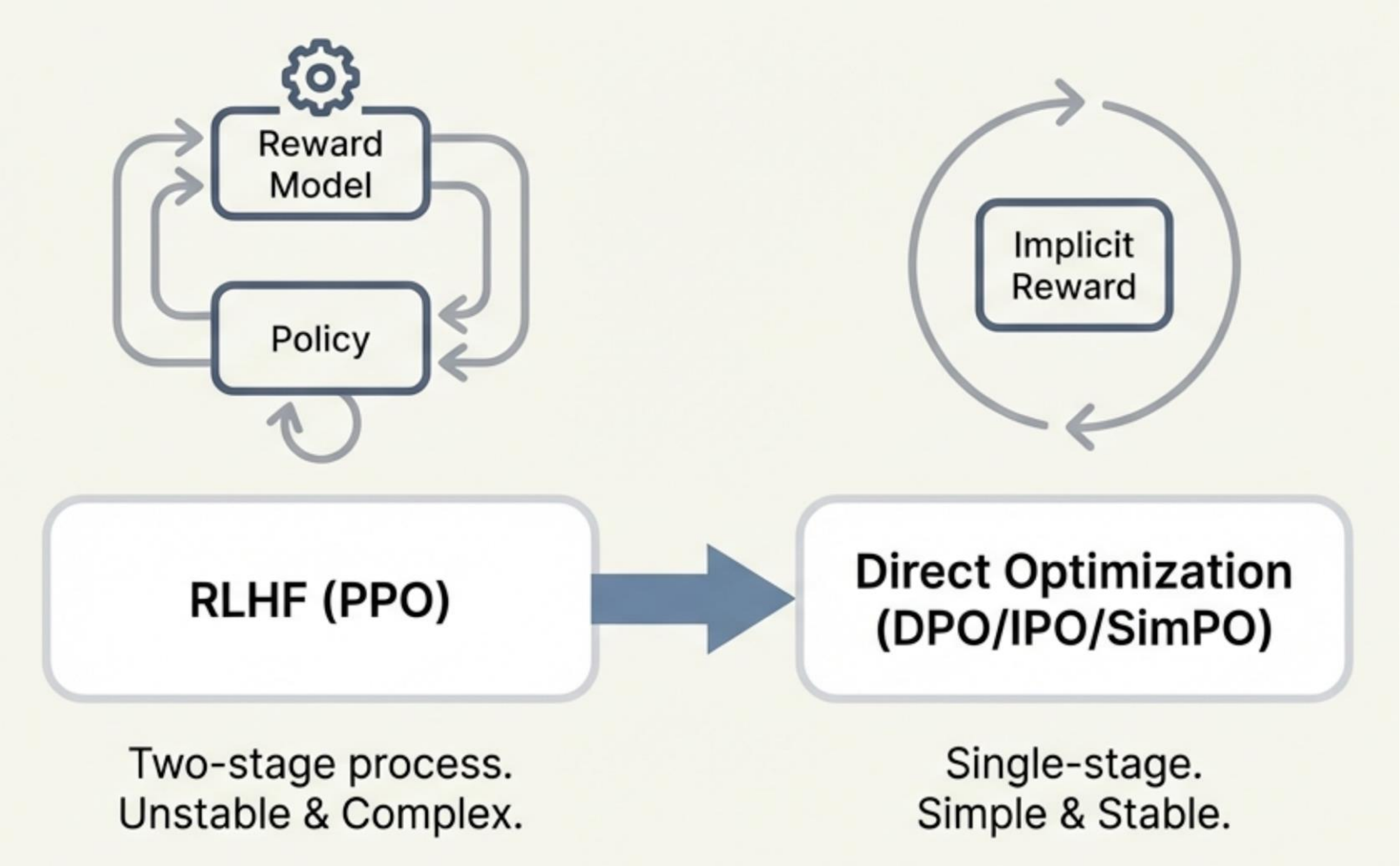
$$r_\theta(x,y_w) - r_\theta(x,y_l) = \beta\left[\log\left(\frac{\pi_\theta(y_w|x)}{\pi_{\mathrm{ref}}(y_w|x)}\right) - \log\left(\frac{\pi_\theta(y_l|x)}{\pi_{\mathrm{ref}}(y_l|x)}\right)\right]$$

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 of LLM Alignment: RLHF and DPO



# Two different Post-Training Preference Alignment (Preference Optimization: PO) Strategies:

- To incorporate human preferences:
  - **RL algorithms** (PPO, GRPO, REINFORCE) explicitly maximize expected reward from a reward model – we normally call this group **RLHF**
  - **Direct alignment methods** (DPO, IPO, KTO) optimize preference objectives without explicit reward modeling
    - Though they can be shown to implicitly optimize an equivalent objective under certain assumptions

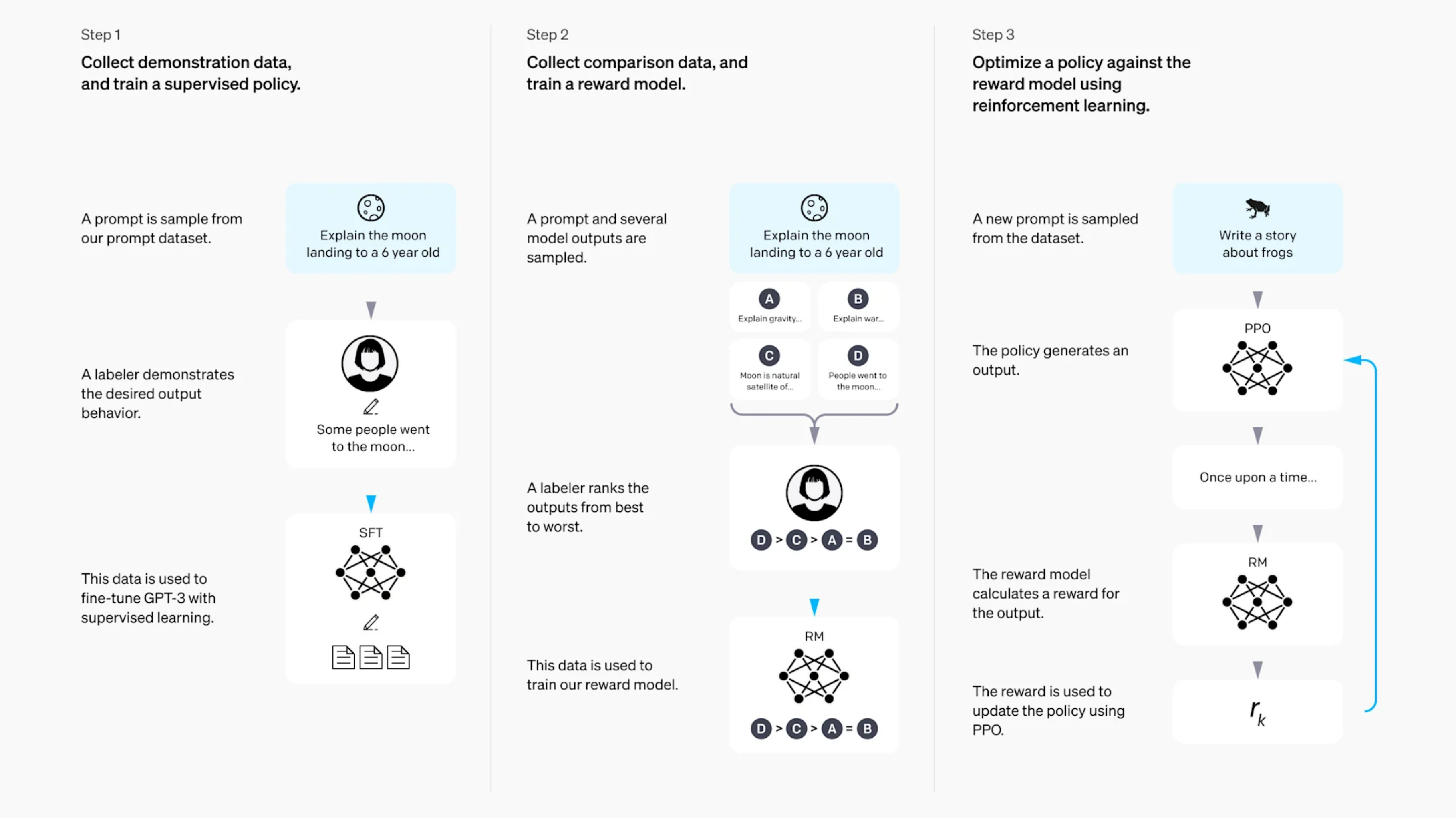
# Advanced DPO:
# An Example: MC-PO

## Preference Optimization via Contrastive Divergence: Your Policy is Secretly an NLL Estimator

AAAI 2026
Oral Talk

Existing studies on PO have centered on constructing preference data following simple heuristics.

- Random Sampling
- Max Margin Selection
- **Lack of theoretical justification**



Step 1

**Collect demonstration data, and train a supervised policy.**

A prompt is sample from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.

Explain the moon landing to a 6 year old
Some people went to the moon...
SFT

Step 2

**Collect comparison data, and train a reward model.**

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.

Explain the moon landing to a 6 year old
A
Explain gravity...
B
Explain war...
C
Moon is natural satellite of...
D
People went to the moon...
D > C > A = B
RM
D > C > A = B

Step 3

**Optimize a policy against the reward model using reinforcement learning.**

A new prompt is sampled from the dataset.

The policy generates an output.

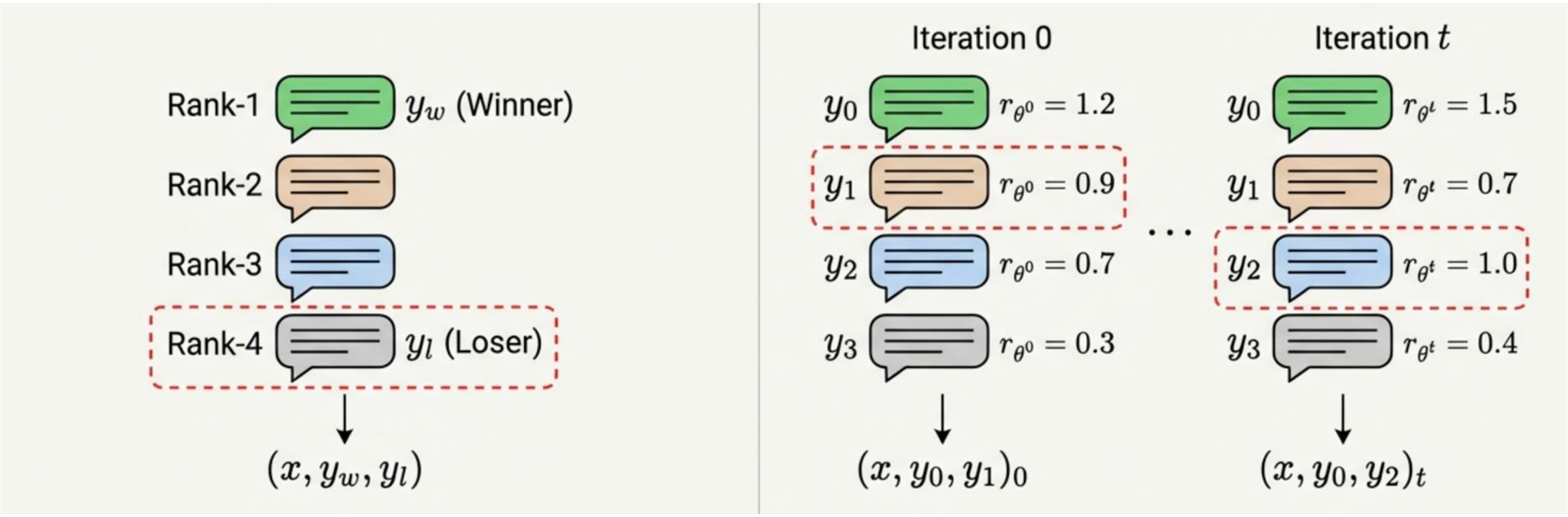
The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.

# Question: How Should We Choose Dispreferred Completions?

Previous (e.g.): The "Max Margin" Heuristic

**Proposed: MC-PO**


Rank-1
$y_w$ (Winner)
Rank-2
Rank-3
Rank-4
$y_l$ (Loser)
$(x, y_w, y_l)$
Iteration 0
$y_0$ $r_{\theta^0} = 1.2$
$y_1$ $r_{\theta^0} = 0.9$
$y_2$ $r_{\theta^0} = 0.7$
$y_3$ $r_{\theta^0} = 0.3$
$(x, y_0, y_1)_0$
...
Iteration $t$
$y_0$ $r_{\theta^t} = 1.5$
$y_1$ $r_{\theta^t} = 0.7$
$y_2$ $r_{\theta^t} = 1.0$
$y_3$ $r_{\theta^t} = 0.4$
$(x, y_0, y_2)_t$

Sampling proportionally to the model's implicit reward.
The "negative" evolves as the policy learns.

# Preference Optimization as NLL Estimation

$$p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) := \frac{1}{Z_{\boldsymbol{\theta}}(\mathbf{x})}\mu(\mathbf{y}|\mathbf{x}) \exp\left(r_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{y})\right) \qquad Z_{\boldsymbol{\theta}}(\mathbf{x}) = \int \mu(\mathbf{y}'|\mathbf{x}) \exp\left(r_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{y}')\right) d\mathbf{y}'$$

- PO as NLL estimation

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \ \mathbb{E}_{\mathbf{x}\sim\rho,\ \mathbf{y}\sim\pi^*(\cdot|\mathbf{x})}\left[\mathcal{L}^{\text{NLL}}(\boldsymbol{\theta},\mathbf{x},\mathbf{y})\right]$$

$$\mathcal{L}^{NLL}(\boldsymbol{\theta},\mathbf{x},\mathbf{y}) = -\beta r_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{y}) + \log Z_{\boldsymbol{\theta}}(\mathbf{x}).$$

- PO with NLL gradient estimation

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}^{\text{NLL}}(\boldsymbol{\theta},\mathbf{x},\mathbf{y}) = -\nabla_{\boldsymbol{\theta}} r_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{y}) + \boxed{\mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{y}'|\mathbf{x})}\left[\nabla_{\boldsymbol{\theta}} r_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{y}')\right]}$$

# Basic Formulation: Preference Optimization as NLL Estimation

- Negative log-likelihood (NLL) estimation
  - NLL estimation approximates a distribution with a parametric model using sampled observations.

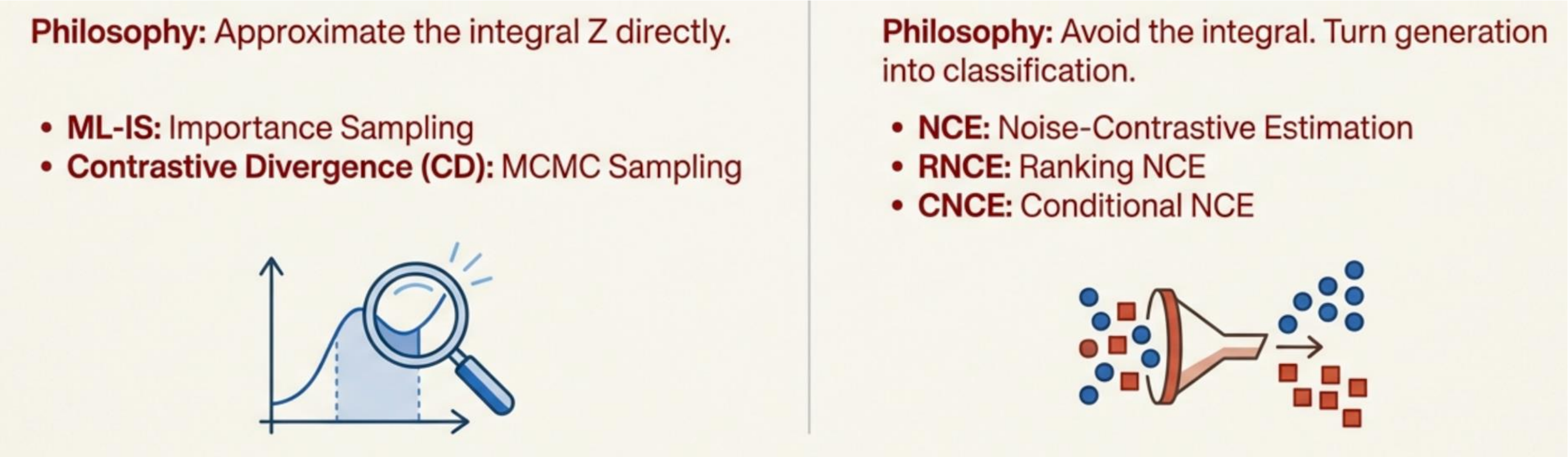
$$p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) := \frac{\tilde{p}_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})}{Z_{\boldsymbol{\theta}}(\mathbf{x})}, \quad \text{where} \quad Z_{\boldsymbol{\theta}}(\mathbf{x}) = \int \tilde{p}_{\boldsymbol{\theta}}(\mathbf{y}'|\mathbf{x}) d\mathbf{y}'$$

  - The challenge is to compute the normalization constant

- **Preference Optimization as NLL estimation**

$$p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) := \frac{1}{Z_{\boldsymbol{\theta}}(\mathbf{x})} \mu(\mathbf{y}|\mathbf{x}) \exp\left(r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})\right) \qquad Z_{\boldsymbol{\theta}}(\mathbf{x}) = \int \mu(\mathbf{y}'|\mathbf{x}) \exp\left(r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}')\right) d\mathbf{y}'$$

# Background: Two families of Strategies for the NLL Estimation Challenge: The Partition Function (Z)



**Philosophy:** Approximate the integral Z directly.

- **ML-IS:** Importance Sampling
- **Contrastive Divergence (CD):** MCMC Sampling

**Philosophy:** Avoid the integral. Turn generation into classification.

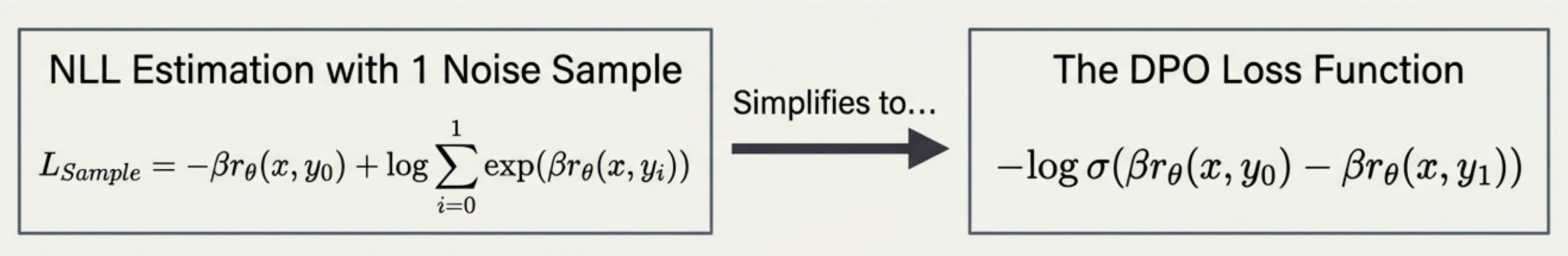
- **NCE:** Noise-Contrastive Estimation
- **RNCE:** Ranking NCE
- **CNCE:** Conditional NCE

On the connection between Noise-Contrastive Estimation and Contrastive Divergence, (2024) Amanda O, et al.

**Theoretical Unification**:

- **NCE, RNCE** (Ranking NCE) are Special Case of **CD** (Contrastive Divergence)
- through the lens of special **Markov Chain (MC)** transition kernels.

# Connection:
# DPO is NLL Estimation with Weak Sampling of Z



**NLL Estimation with 1 Noise Sample**

$$L_{Sample} = -\beta r_\theta(x, y_0) + \log \sum_{i=0}^{1} \exp(\beta r_\theta(x, y_i))$$

Simplifies to...

**The DPO Loss Function**

$$-\log \sigma(\beta r_\theta(x, y_0) - \beta r_\theta(x, y_1))$$

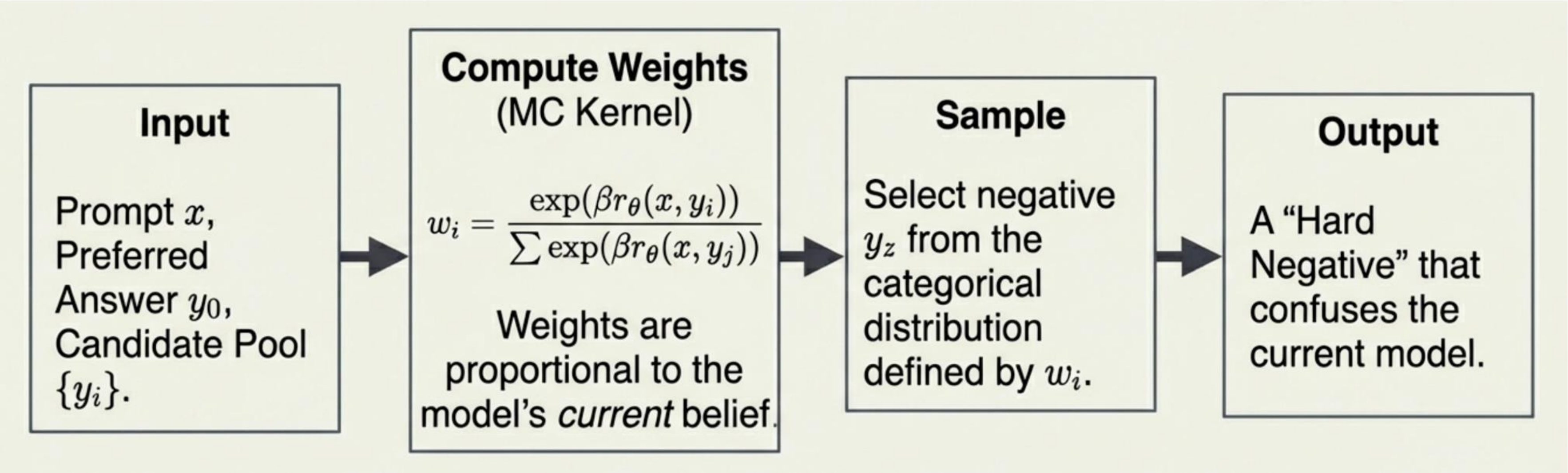
DPO estimates normalization Z(x) with

- Ranking Noise Contrastive Estimation (RNCE) via
- M=1 Noisy Sample (aka, the dispreferred response)

Dispreferred completions in Preference Optimization are actually importance samples used to estimate the normalization constant.

# Bridging the Gap: from Weak Sampling to CD based =>

## **Markov Chain Preference Optimization (**MC-PO): Preference Optimization via Contrastive Divergence



**Input**

Prompt $x$, Preferred Answer $y_0$, Candidate Pool $\{y_i\}$.

**Compute Weights**
(MC Kernel)

$$w_i = \frac{\exp(\beta r_\theta(x, y_i))}{\sum \exp(\beta r_\theta(x, y_j))}$$

Weights are proportional to the model's *current* belief.

**Sample**

Select negative $y_z$ from the categorical distribution defined by $w_i$.

**Output**

A "Hard Negative" that confuses the current model.

We run the MCMC chain for a single step to balance accuracy with training efficiency

# MC-PO:

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}^{\mathrm{NLL}}(\boldsymbol{\theta}, \mathbf{x}, \mathbf{y}) = -\nabla_{\boldsymbol{\theta}} r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) + \boxed{\mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{y}'|\mathbf{x})}\left[\nabla_{\boldsymbol{\theta}} r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}')\right]}$$

We will compute this term

- Contrastive divergence applies an MC kernel to compute the gradient term of normalization

$$\mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{y}'|\mathbf{x})}\left[\nabla_{\boldsymbol{\theta}} r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}')\right] \approx \mathbb{E}_{K_{\boldsymbol{\theta}}(\mathbf{y}'|\mathbf{x}, \mathbf{y}_0)}\left[\nabla_{\boldsymbol{\theta}} r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}')\right]$$

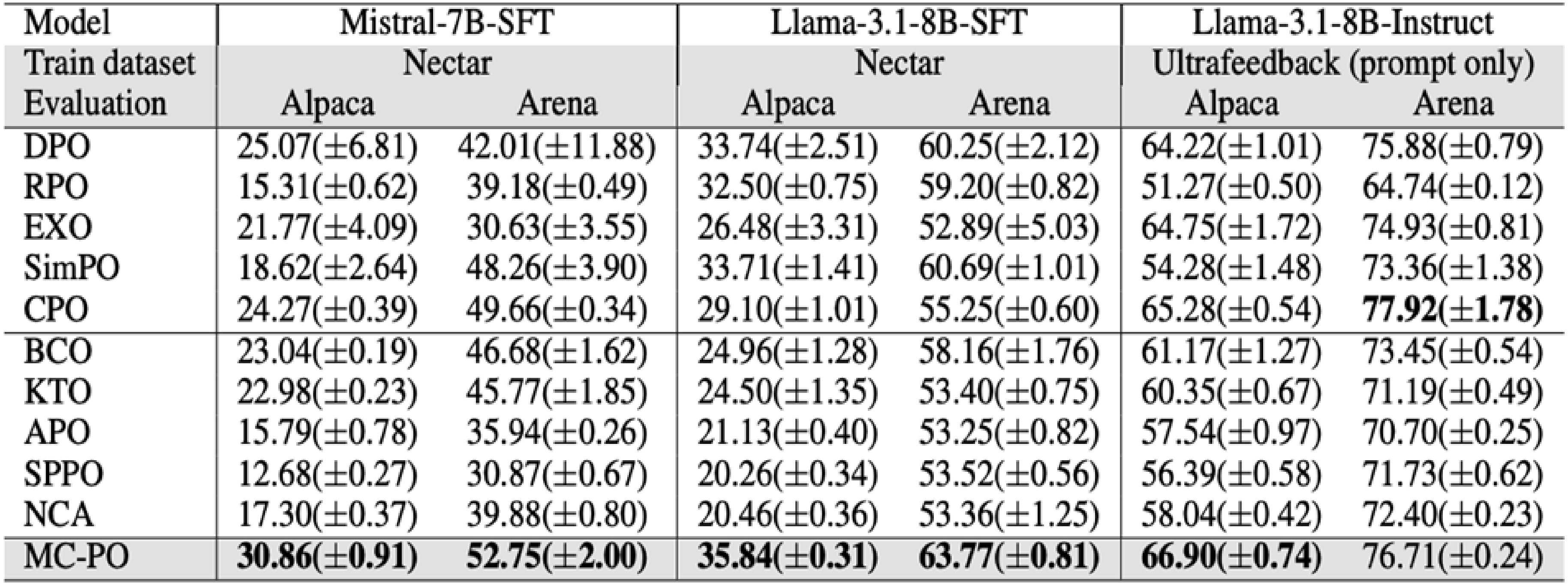
- MC kernel: Designed for Sampling in proportion to the implicit reward (via current policy)

$$\mathbb{E}_{K_{\boldsymbol{\theta}}(\mathbf{y}'|\mathbf{x}, \mathbf{y}_0)}\left[\nabla_{\boldsymbol{\theta}} r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}')\right] = \nabla_{\boldsymbol{\theta}} \log \sum_{i=0}^{M} \exp\left(r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}_i)\right)$$

- Proof in the paper: Hard negatives lead to more effective gradient updates

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}^{\mathrm{CD}}(\boldsymbol{\theta}, \mathbf{x}, \mathbf{y}_0) = -\sigma\left(r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}_1) - r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}_0)\right) \nabla_{\boldsymbol{\theta}}\left(r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}_0) - r_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}_1)\right)$$

# Main Comparison Results

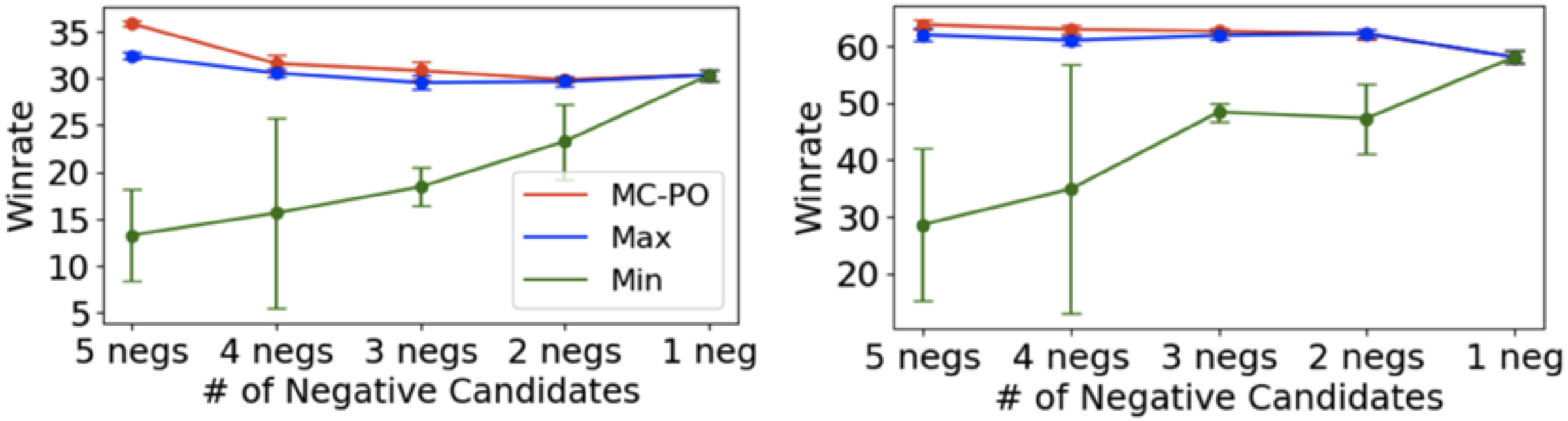


| Model | Mistral-7B-SFT | | Llama-3.1-8B-SFT | | Llama-3.1-8B-Instruct | |
|---|---|---|---|---|---|---|
| Train dataset | Nectar | | Nectar | | Ultrafeedback (prompt only) | |
| Evaluation | Alpaca | Arena | Alpaca | Arena | Alpaca | Arena |
| DPO | 25.07(±6.81) | 42.01(±11.88) | 33.74(±2.51) | 60.25(±2.12) | 64.22(±1.01) | 75.88(±0.79) |
| RPO | 15.31(±0.62) | 39.18(±0.49) | 32.50(±0.75) | 59.20(±0.82) | 51.27(±0.50) | 64.74(±0.12) |
| EXO | 21.77(±4.09) | 30.63(±3.55) | 26.48(±3.31) | 52.89(±5.03) | 64.75(±1.72) | 74.93(±0.81) |
| SimPO | 18.62(±2.64) | 48.26(±3.90) | 33.71(±1.41) | 60.69(±1.01) | 54.28(±1.48) | 73.36(±1.38) |
| CPO | 24.27(±0.39) | 49.66(±0.34) | 29.10(±1.01) | 55.25(±0.60) | 65.28(±0.54) | **77.92(±1.78)** |
| BCO | 23.04(±0.19) | 46.68(±1.62) | 24.96(±1.28) | 58.16(±1.76) | 61.17(±1.27) | 73.45(±0.54) |
| KTO | 22.98(±0.23) | 45.77(±1.85) | 24.50(±1.35) | 53.40(±0.75) | 60.35(±0.67) | 71.19(±0.49) |
| APO | 15.79(±0.78) | 35.94(±0.26) | 21.13(±0.40) | 53.25(±0.82) | 57.54(±0.97) | 70.70(±0.25) |
| SPPO | 12.68(±0.27) | 30.87(±0.67) | 20.26(±0.34) | 53.52(±0.56) | 56.39(±0.58) | 71.73(±0.62) |
| NCA | 17.30(±0.37) | 39.88(±0.80) | 20.46(±0.36) | 53.36(±1.25) | 58.04(±0.42) | 72.40(±0.23) |
| MC-PO | **30.86(±0.91)** | **52.75(±2.00)** | **35.84(±0.31)** | **63.77(±0.81)** | **66.90(±0.74)** | 76.71(±0.24) |

- MC-PO outperforms baselines in 5 out of 6 experimental settings.
- MC-PO leads to better performance with more diverse response candidates

# Ablation Studies (1): Choice of MC Kernel Design

- Sampling from the MC kernel yields the best performance
- Min-based variant leads to low performance and high variance

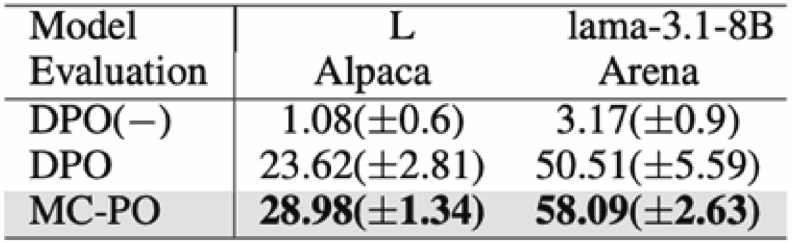


(a): Alpaca-Eval

(b): Arena-Hard

# Ablation Studies (2):

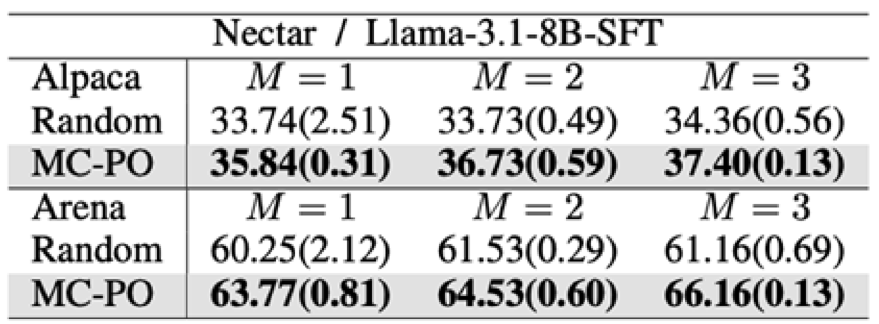
- MC-PO is robust against noise samples



| Model | L | lama-3.1-8B |
|---|---|---|
| Evaluation | Alpaca | Arena |
| DPO(−) | 1.08(±0.6) | 3.17(±0.9) |
| DPO | 23.62(±2.81) | 50.51(±5.59) |
| MC-PO | **28.98(±1.34)** | **58.09(±2.63)** |

DPO(-) uses the noise sample as rejected response
DPO randomly choose a rejected response
MC-PO samples in proportion to the implicit reward

- MC-PO benefits from sampling more negatives



| Nectar / Llama-3.1-8B-SFT | | | |
|---|---|---|---|
| Alpaca | $M = 1$ | $M = 2$ | $M = 3$ |
| Random | 33.74(2.51) | 33.73(0.49) | 34.36(0.56) |
| MC-PO | **35.84(0.31)** | **36.73(0.59)** | **37.40(0.13)** |
| Arena | $M = 1$ | $M = 2$ | $M = 3$ |
| Random | 60.25(2.12) | 61.53(0.29) | 61.16(0.69) |
| MC-PO | **63.77(0.81)** | **64.53(0.60)** | **66.16(0.13)** |

When M increases, MC-PO leads to more improvements

# Conclusion and Future Works

We frame the alignment problems as an NLL estimation and connect DPO to sampling-based solutions.

We propose MC-PO that offers a principled way to sample noisy sample (dispreferred responses)

At next step, we aim to showcase the benefits of utilizing a multi-step MCMC based PO solution