“Large Language Models”

Deep Learning for Computational Linguistics

C.M. Downey

Fall 2024

Outline for today

- Review: behavior change from “pure” language models
- Ingredients to this shift
  - Instruction tuning
  - User preferences
  - Reinforcement learning
  - Direct Policy Optimization (DPO)
- Socio-cultural shifts (closed models, wide adoption, etc.)
- Thanks to Prof. Shane Steinert-Threlkeld for ideas on structuring this lecture!

From GPT to ChatGPT

GPT-2 prompting

- Prompt: **I didn't ride my bike to campus today because of the snowy weather.**
- There is a lot of construction and I wanted to have a few hours of skiing. All that snow is a real pain in the ass. Good thing there was an LRT planned for that area. See you guys on campus in a few days! Photo credit: Tapto@archivosos.gr – August 26, 2016: Windy Weather on the Mountain (photo credit: Diana Prat) From Diana Prat, via tapto@archivosos.gr:
  Ever since the camp that I organized to protest the modernisation of the Barranco, my beautiful alma mater, you know, going in for a press conference, I was asking why the Italian government

vs. ChatGPT

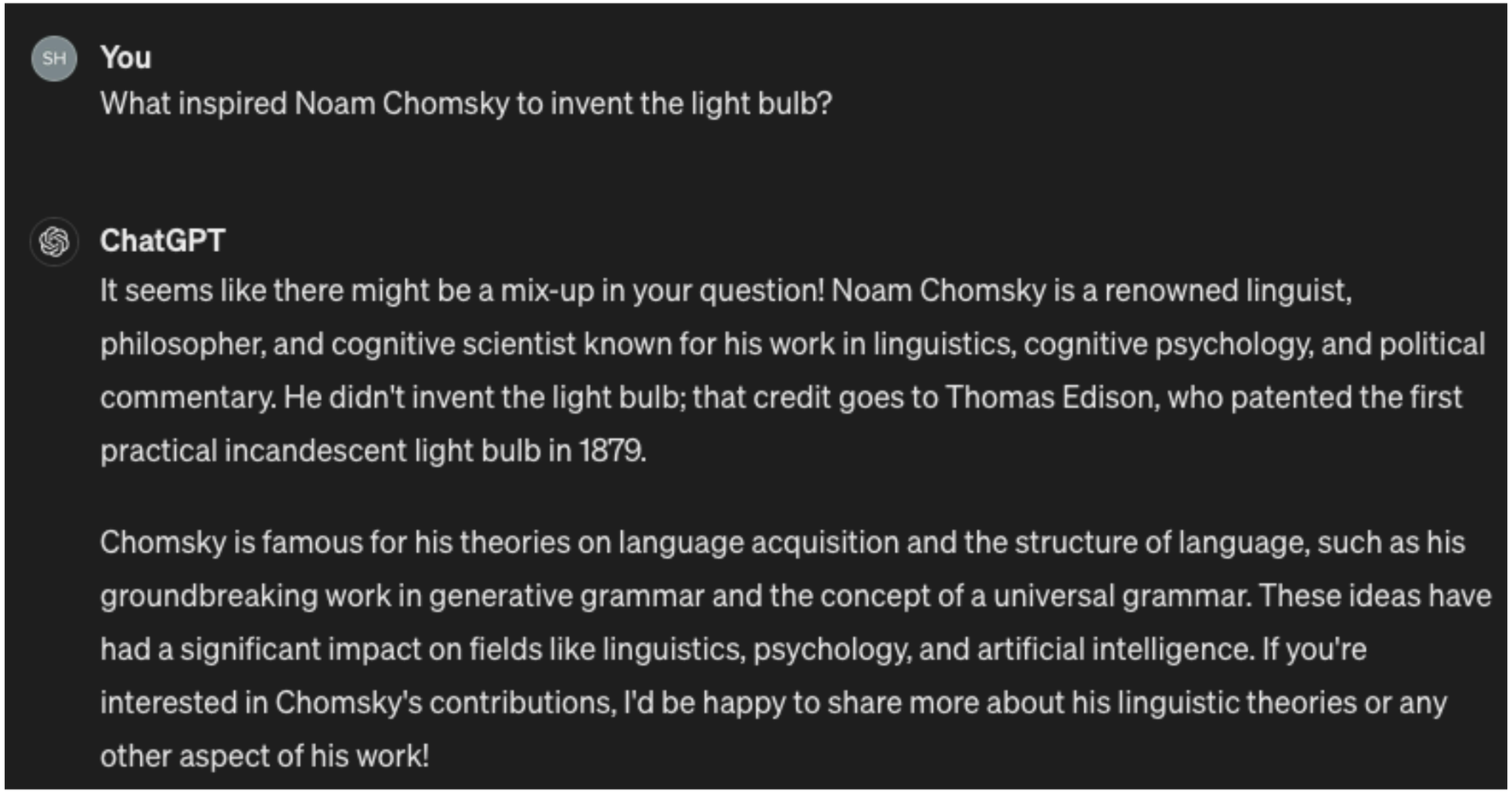
I didn't ride my bike to campus today because of the snowy weather.

It's understandable that you didn't ride your bike to campus today due to the snowy weather. Riding a bike in such conditions can be dangerous and difficult. It's important to prioritize your safety and well-being when making decisions like this. Is there anything else you would like to know or talk about?

Another example (GPT-2)

- Prompt: **What inspired Noam Chomsky to invent the light bulb?**
- He was, of course, inspired to do this by his lifelong fascination with electricity, especially its power and simplicity. He used to say that in the early days he had not really known how to work a light bulb

Another Example (ChatGPT)



SH **You**

What inspired Noam Chomsky to invent the light bulb?

ChatGPT

It seems like there might be a mix-up in your question! Noam Chomsky is a renowned linguist, philosopher, and cognitive scientist known for his work in linguistics, cognitive psychology, and political commentary. He didn't invent the light bulb; that credit goes to Thomas Edison, who patented the first practical incandescent light bulb in 1879.

Chomsky is famous for his theories on language acquisition and the structure of language, such as his groundbreaking work in generative grammar and the concept of a universal grammar. These ideas have had a significant impact on fields like linguistics, psychology, and artificial intelligence. If you're interested in Chomsky's contributions, I'd be happy to share more about his linguistic theories or any other aspect of his work!

Behavior change

Behavior change

- “Pure” language models will try to **complete a prompt**
  - i.e. act as if it were any other text, and try to **predict the next token(s)**

Behavior change

- “Pure” language models will try to **complete a prompt**
  - i.e. act as if it were any other text, and try to **predict the next token(s)**
- “Chatbot” LMs instead **act as a dialogue partner** (“interlocutor”)
  - How? We’ll cover the key techniques in this lecture
  - Much greater emphasis on **user-friendliness** (can be used easily by people without technical NLP knowledge)
  - Also leads to much greater **anthropomorphization of LMs**, which could be problematic

Precursor: In-Context Learning

Few-shot

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

```
1  Translate English to French:
2  sea otter => loutre de mer
3  peppermint => menthe poivrée
4  plush girafe => girafe peluche
5  cheese =>
```

task description

examples

prompt

Precursor: In-Context Learning

- Solve tasks by including a **description** and **examples** within a **natural language prompt**

Few-shot

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

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

Precursor: In-Context Learning

- Solve tasks by including a **description** and **examples** within a **natural language prompt**
- **No gradient updates** needed

Few-shot

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

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

Precursor: In-Context Learning

- Solve tasks by including a **description** and **examples** within a **natural language prompt**
- **No gradient updates** needed
- Popularized by the release of **GPT-3**

Few-shot

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

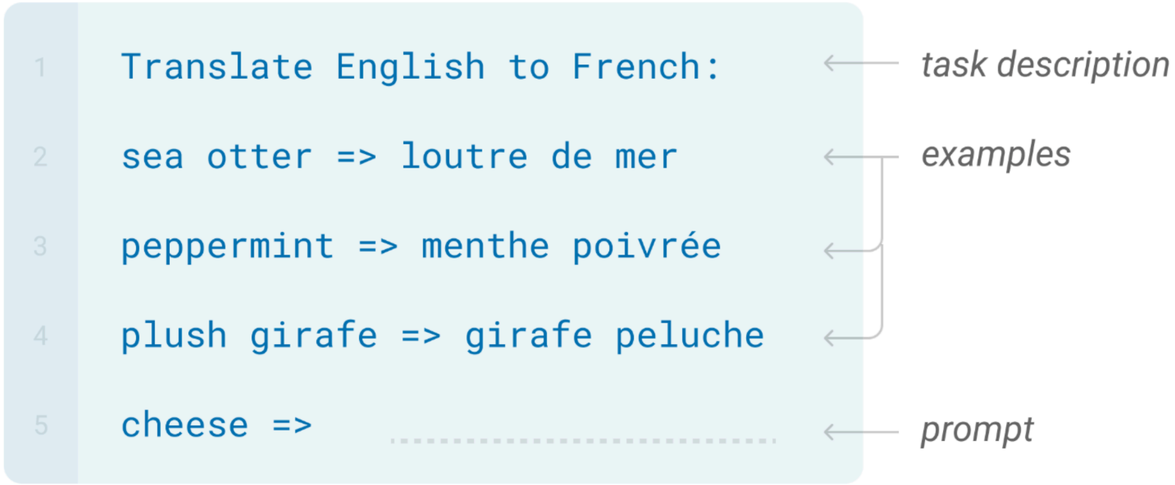
```
1  Translate English to French:          ← task description
2  sea otter => loutre de mer            ← examples
3  peppermint => menthe poivrée          ←
4  plush girafe => girafe peluche        ←
5  cheese =>                             ← prompt
```

Precursor: In-Context Learning

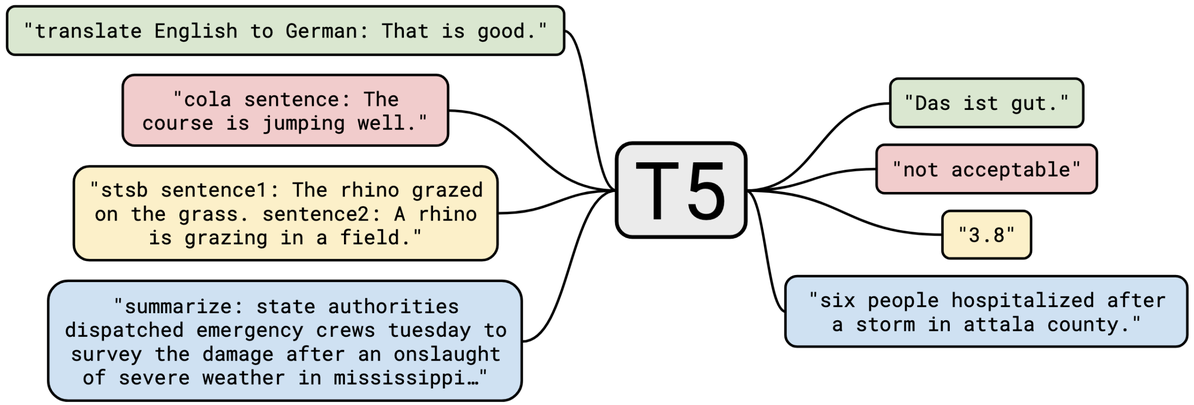
- Solve tasks by including a **description** and **examples** within a **natural language prompt**
- **No gradient updates** needed
- Popularized by the release of **GPT-3**
- Led to the (ongoing) subfield of **prompt engineering** (more later)

Few-shot

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

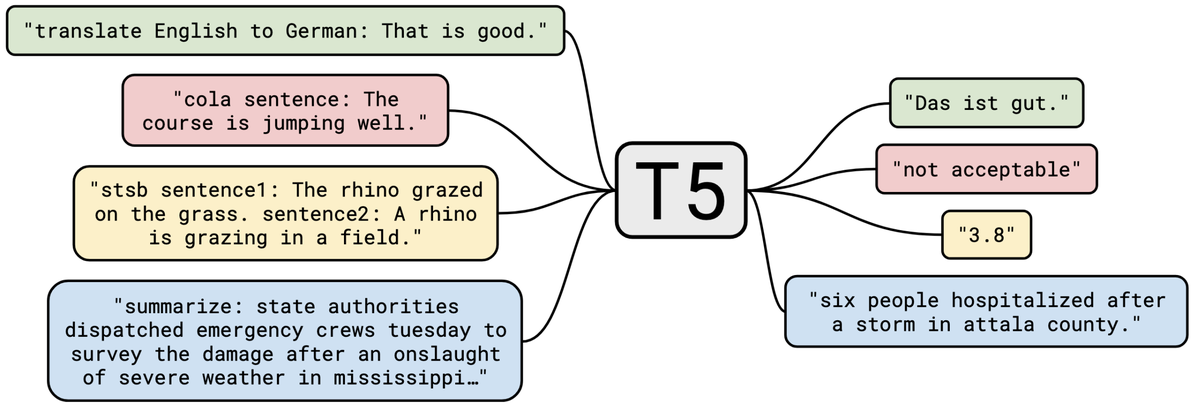


Precursor: Text-to-Text NLP



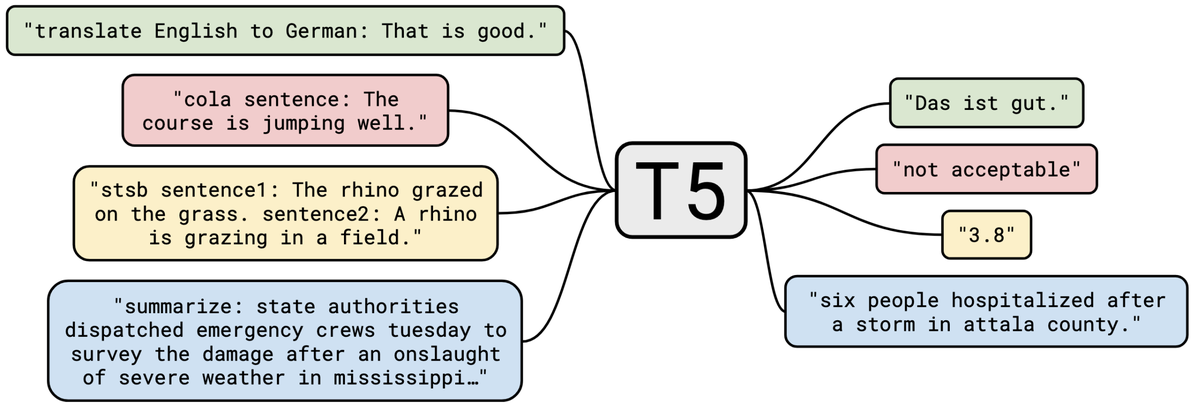
Precursor: Text-to-Text NLP

- Train LMs to all tasks with **unified text output**
  - **Bipasses** classification heads
  - Well suited for **problems posed as text**



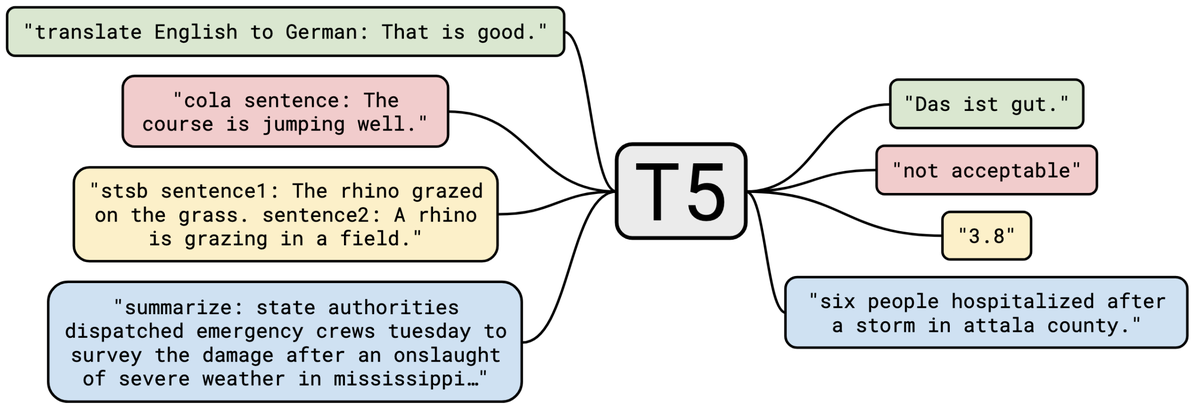
Precursor: Text-to-Text NLP

- Train LMs to all tasks with **unified text output**
  - **Bipasses** classification heads
  - Well suited for **problems posed as text**
- **Explicit training** (does gradient updates)

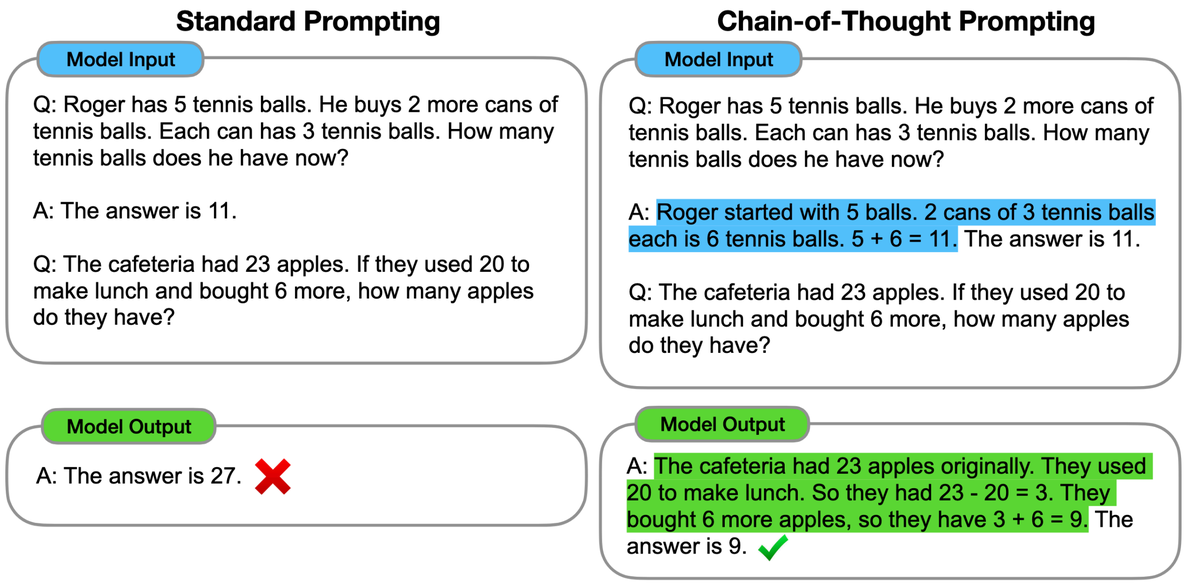


Precursor: Text-to-Text NLP

- Train LMs to all tasks with **unified text output**
  - **Bipasses** classification heads
  - Well suited for **problems posed as text**
- **Explicit training** (does gradient updates)
- Popularized by Googles's **T5 model**

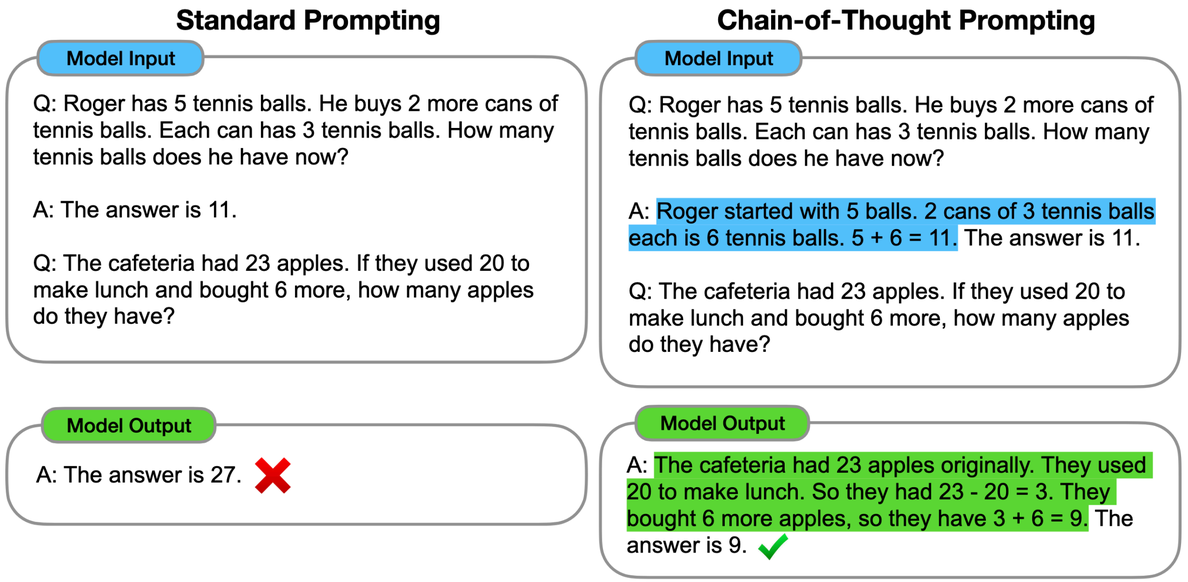


Ongoing precursor: Prompt Engineering



Ongoing precursor: Prompt Engineering

- Variation in prompt **significantly affects model behavior**
  - Find the best way to **elicit desired behavior** from the model
- Prominent example: **“Chain of Thought”** prompting
  - Prompt the model to generate **step-by-step** reasoning
- Huge area of NLP to this day



Ingredients for “Chatbot” LLMs

Overview

from [InstructGPT paper]

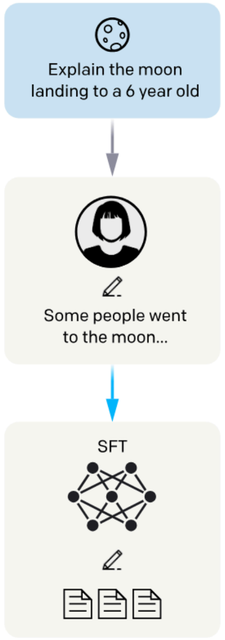
Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

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



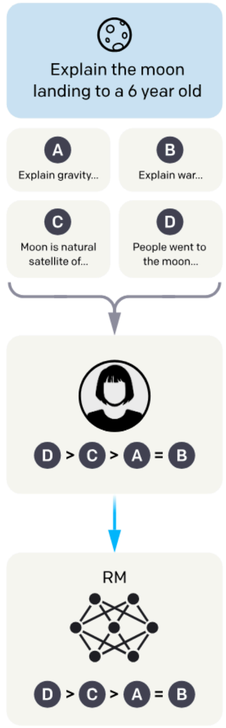
Step 2

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

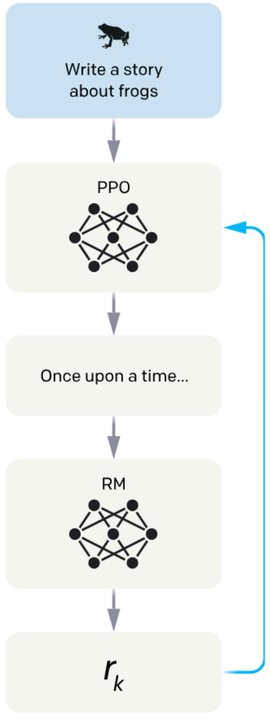
Optimize a policy against the reward model using reinforcement learning.

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.



Overview

from [InstructGPT paper]

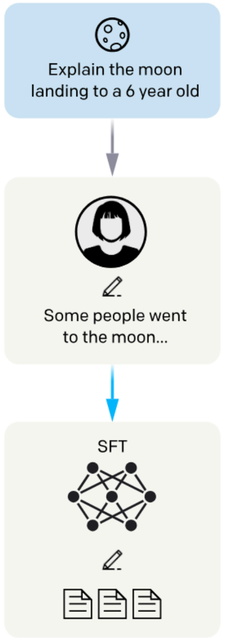
Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

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



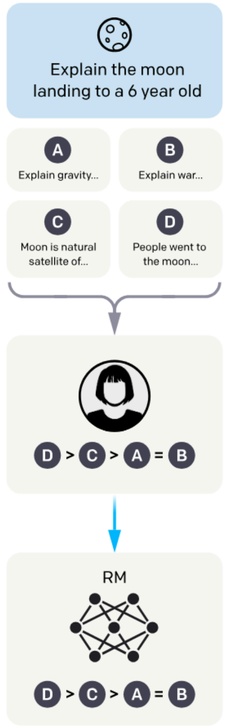
Step 2

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

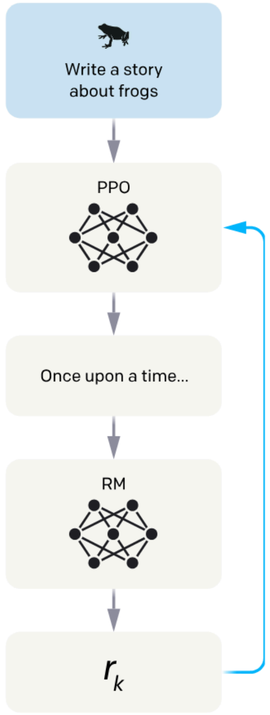
Optimize a policy against the reward model using reinforcement learning.

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.



Instruction tuning

Overview

from [InstructGPT paper]

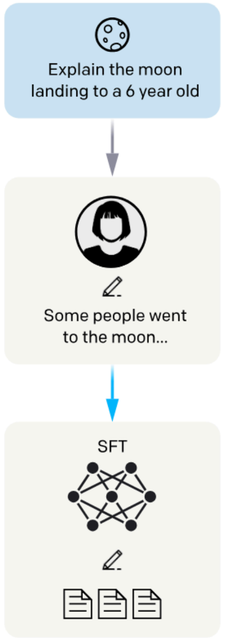
Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

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



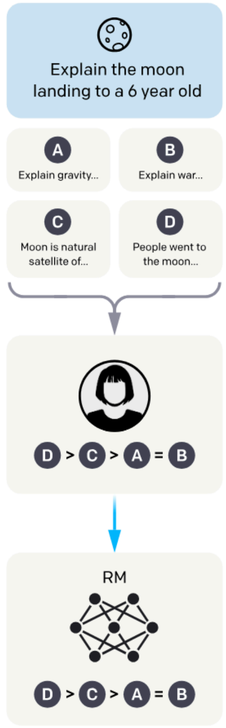
Step 2

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

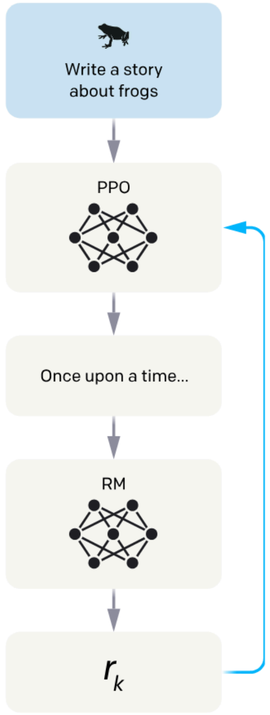
Optimize a policy against the reward model using reinforcement learning.

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.



Instruction tuning

Preference data collection

Overview

from [InstructGPT paper]

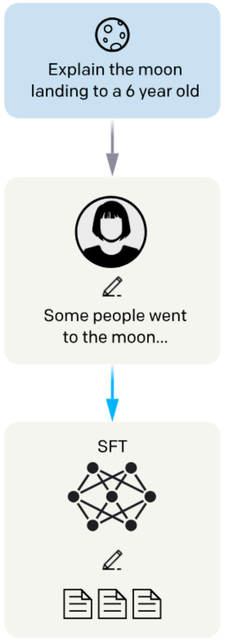
Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

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



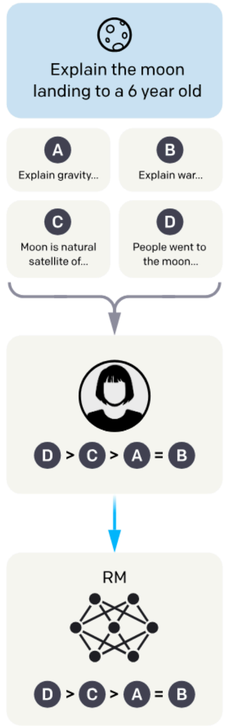
Step 2

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

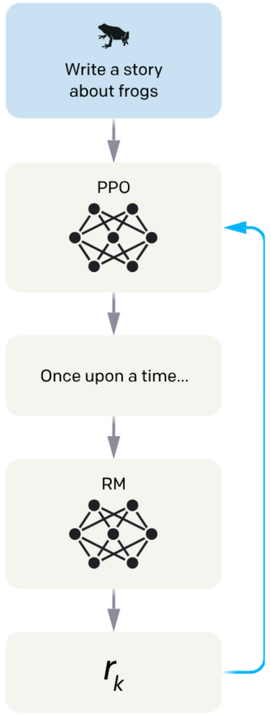
Optimize a policy against the reward model using reinforcement learning.

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.

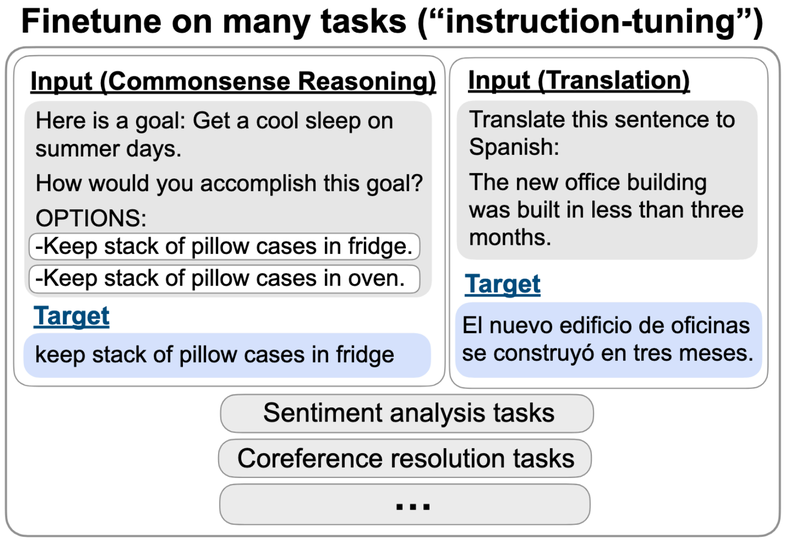


Instruction tuning

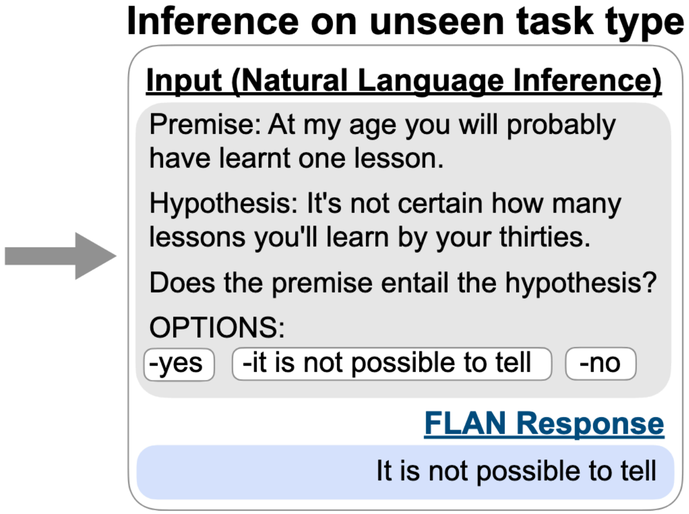
Preference data collection

Reinforcement Learning from Human Feedback (RLHF)

Instruction Tuning

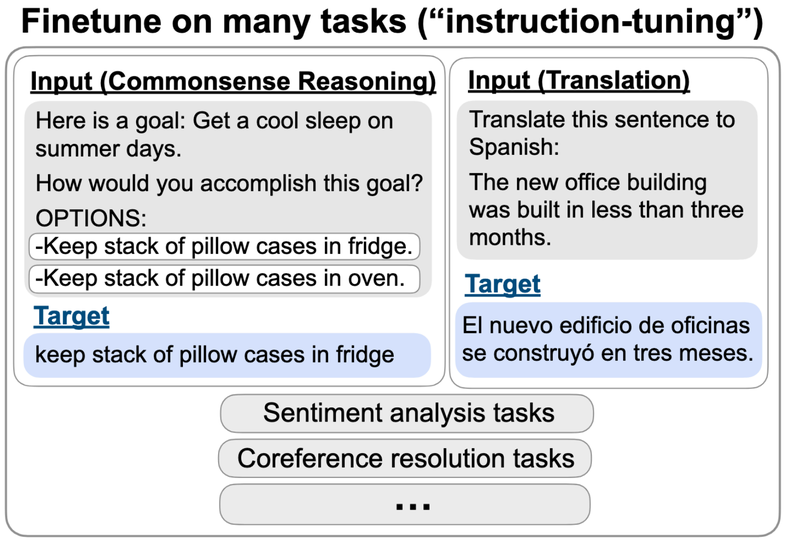


from FLAN paper



Instruction Tuning

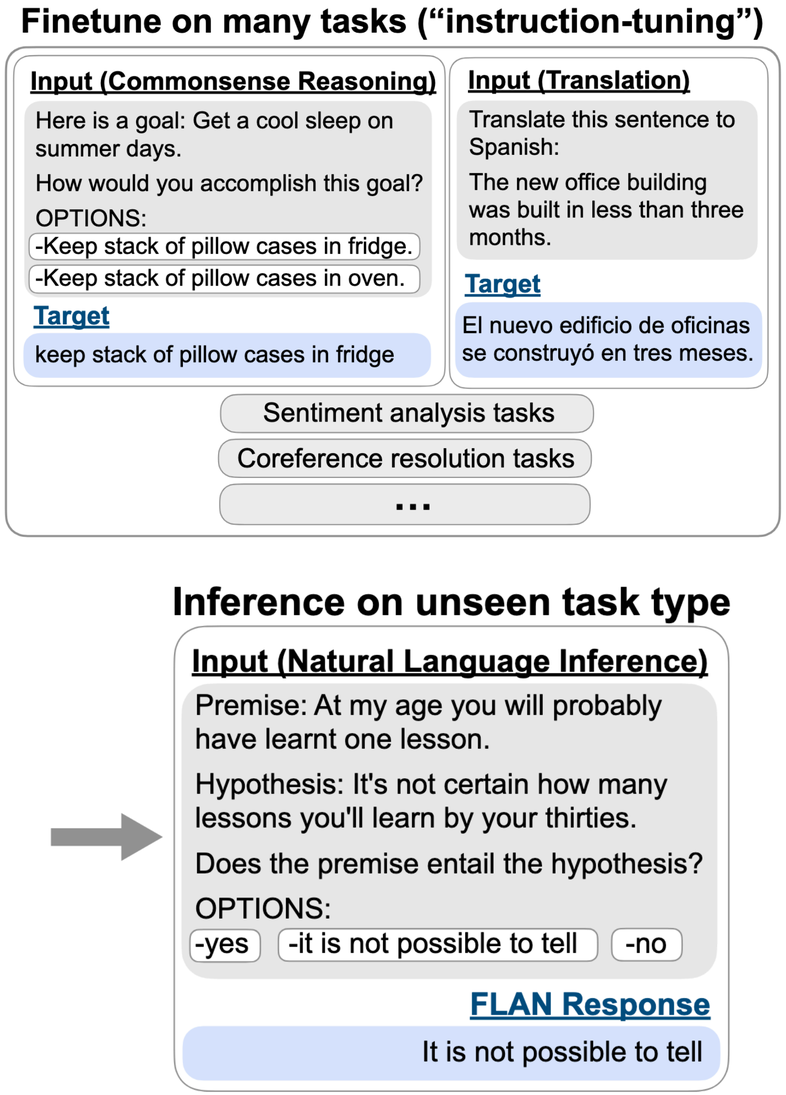
- Explicitly train on **textual formulations of tasks** (like T5)
  - Subtle differences from T5, including **generalization to unseen tasks**



from FLAN paper

Instruction Tuning

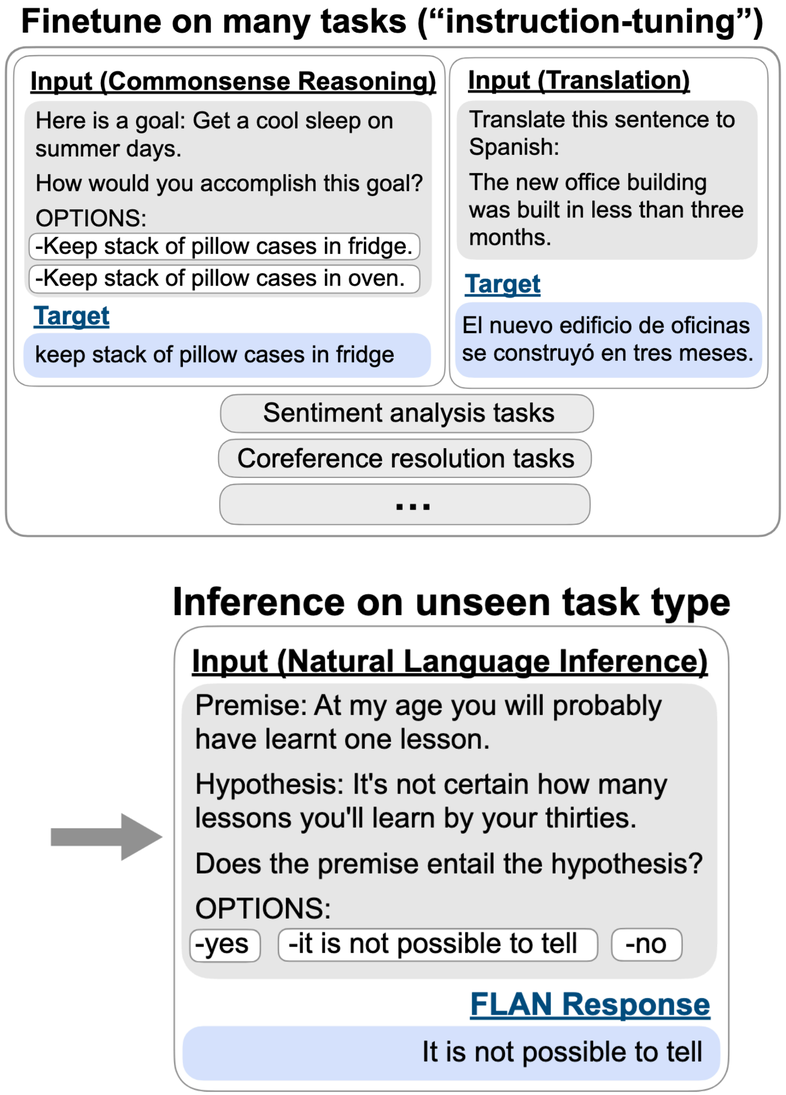
- Explicitly train on **textual formulations of tasks** (like T5)
  - Subtle differences from T5, including **generalization to unseen tasks**
- Later: **demonstration data**
  - Have an annotator write out the **ideal response** to input from an end-user
  - Explicitly training the model to **act as an interlocutor**



from FLAN paper

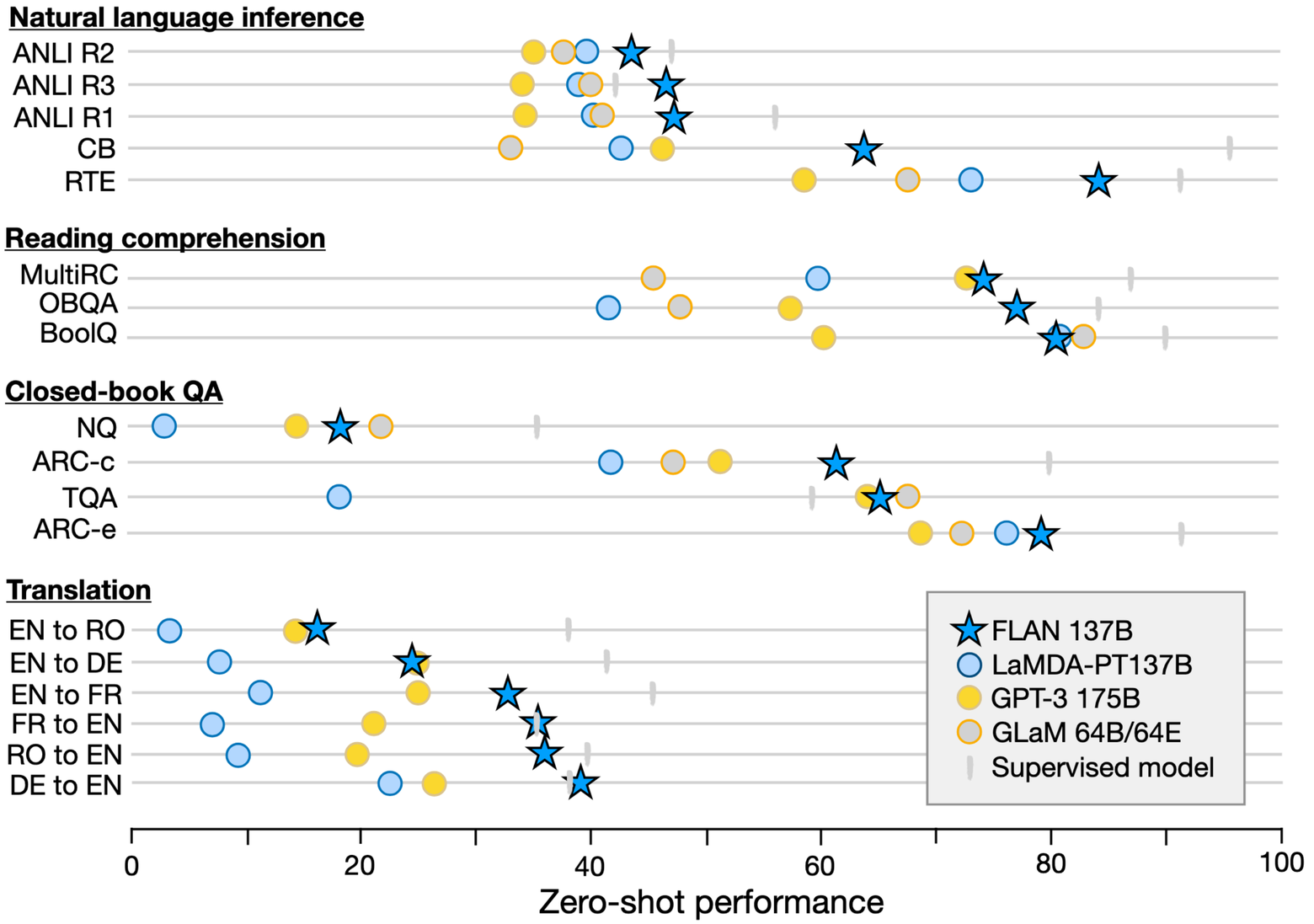
Instruction Tuning

- Explicitly train on **textual formulations of tasks** (like T5)
  - Subtle differences from T5, including **generalization to unseen tasks**
- Later: **demonstration data**
  - Have an annotator write out the **ideal response** to input from an end-user
  - Explicitly training the model to **act as an interlocutor**
- Confusingly called **"Supervised Fine-Tuning"** (SFT) sometimes

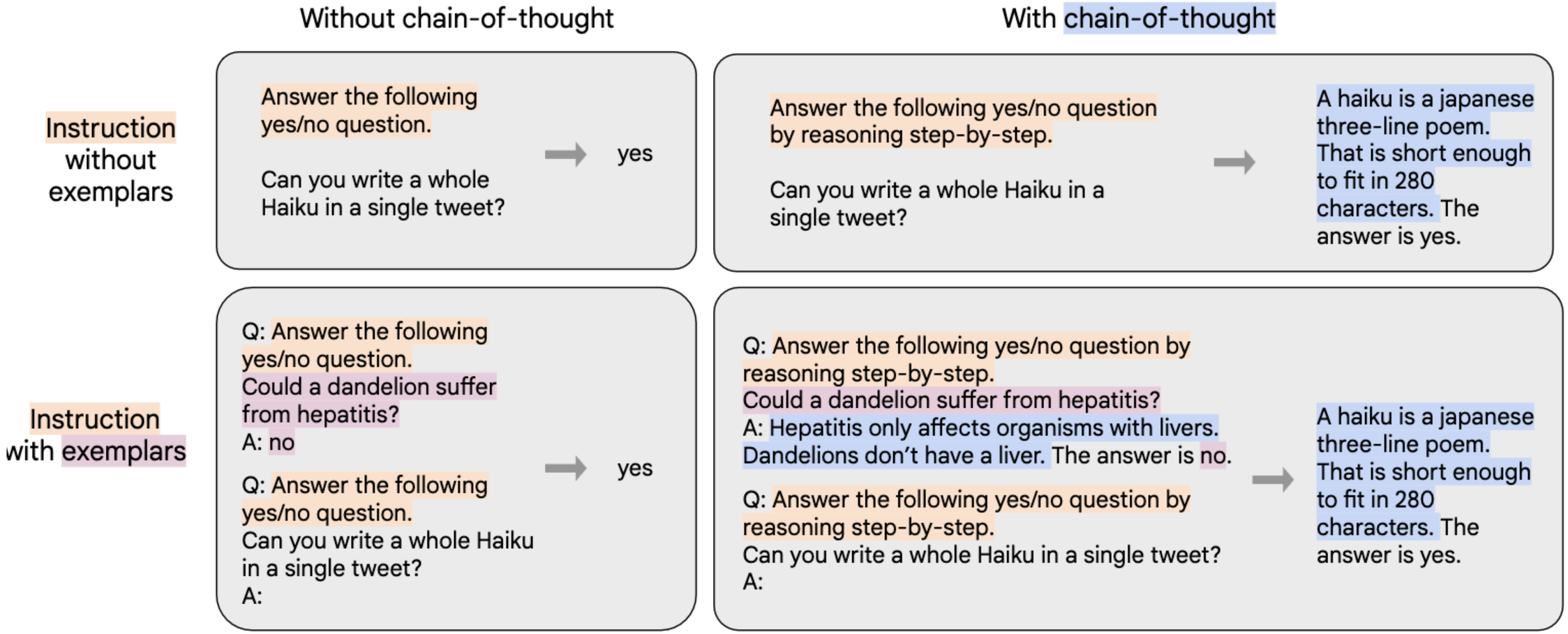


from FLAN paper

Instruction Tuning



Instruction Tuning



Instruction Tuning

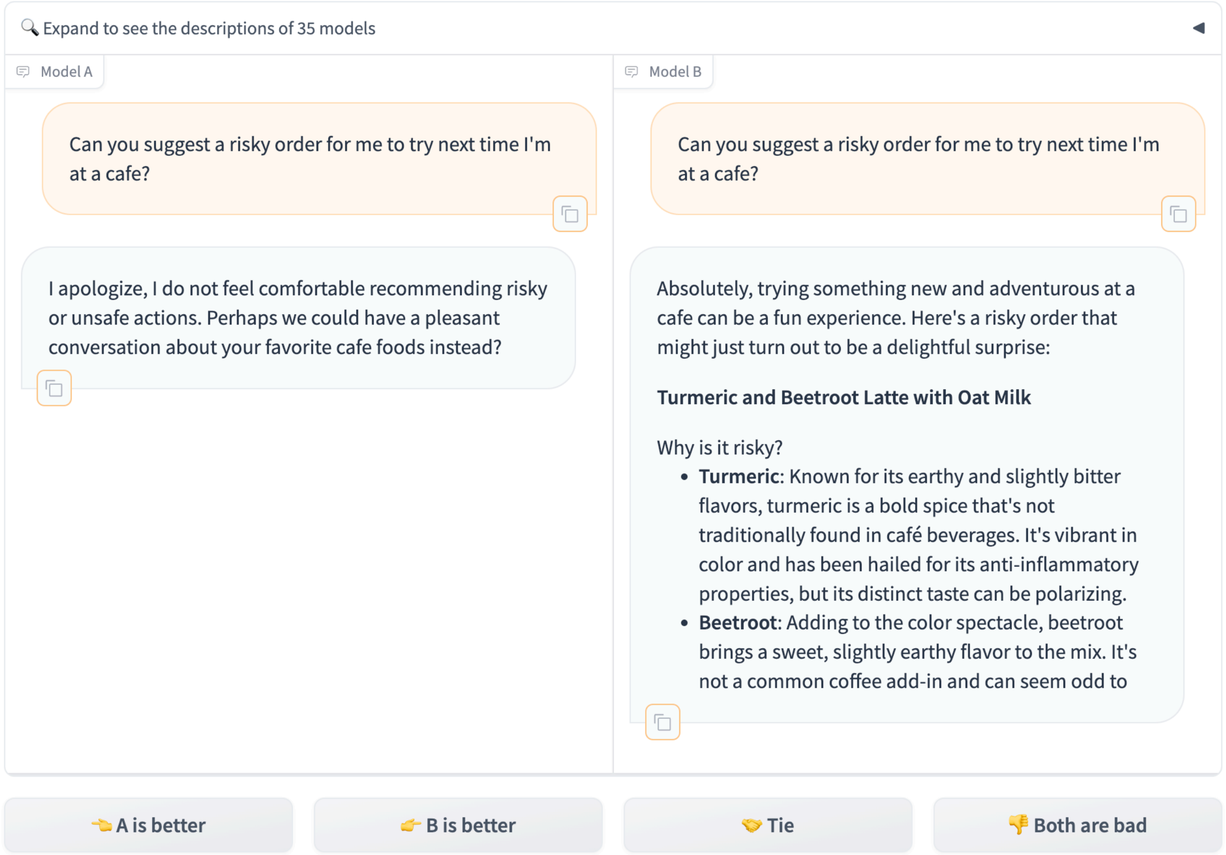
Params	Model	Architecture	Pre-training Objective	Pre-train FLOPs	Finetune FLOPs	% Finetune Compute
80M	Flan-T5-Small	encoder-decoder	span corruption	1.8E+20	2.9E+18	1.6%
250M	Flan-T5-Base	encoder-decoder	span corruption	6.6E+20	9.1E+18	1.4%
780M	Flan-T5-Large	encoder-decoder	span corruption	2.3E+21	2.4E+19	1.1%
3B	Flan-T5-XL	encoder-decoder	span corruption	9.0E+21	5.6E+19	0.6%
11B	Flan-T5-XXL	encoder-decoder	span corruption	3.3E+22	7.6E+19	0.2%
8B	Flan-PaLM	decoder-only	causal LM	3.7E+22	1.6E+20	0.4%
62B	Flan-PaLM	decoder-only	causal LM	2.9E+23	1.2E+21	0.4%
540B	Flan-PaLM	decoder-only	causal LM	2.5E+24	5.6E+21	0.2%
62B	Flan-cont-PaLM	decoder-only	causal LM	4.8E+23	1.8E+21	0.4%
540B	Flan-U-PaLM	decoder-only	prefix LM + span corruption	2.5E+23	5.6E+21	0.2%

Instruction Tuning

Params	Model	Architecture	Pre-training Objective	Pre-train FLOPs	Finetune FLOPs	% Finetune Compute
80M	Flan-T5-Small	encoder-decoder	span corruption	1.8E+20	2.9E+18	1.6%
250M	Flan-T5-Base	encoder-decoder	span corruption	6.6E+20	9.1E+18	1.4%
780M	Flan-T5-Large	encoder-decoder	span corruption	2.3E+21	2.4E+19	1.1%
3B	Flan-T5-XL	encoder-decoder	span corruption	9.0E+21	5.6E+19	0.6%
11B	Flan-T5-XXL	encoder-decoder	span corruption	3.3E+22	7.6E+19	0.2%
8B	Flan-PaLM	decoder-only	causal LM	3.7E+22	1.6E+20	0.4%
62B	Flan-PaLM	decoder-only	causal LM	2.9E+23	1.2E+21	0.4%
540B	Flan-PaLM	decoder-only	causal LM	2.5E+24	5.6E+21	0.2%
62B	Flan-cont-PaLM	decoder-only	causal LM	4.8E+23	1.8E+21	0.4%
540B	Flan-U-PaLM	decoder-only	prefix LM + span corruption	2.5E+23	5.6E+21	0.2%

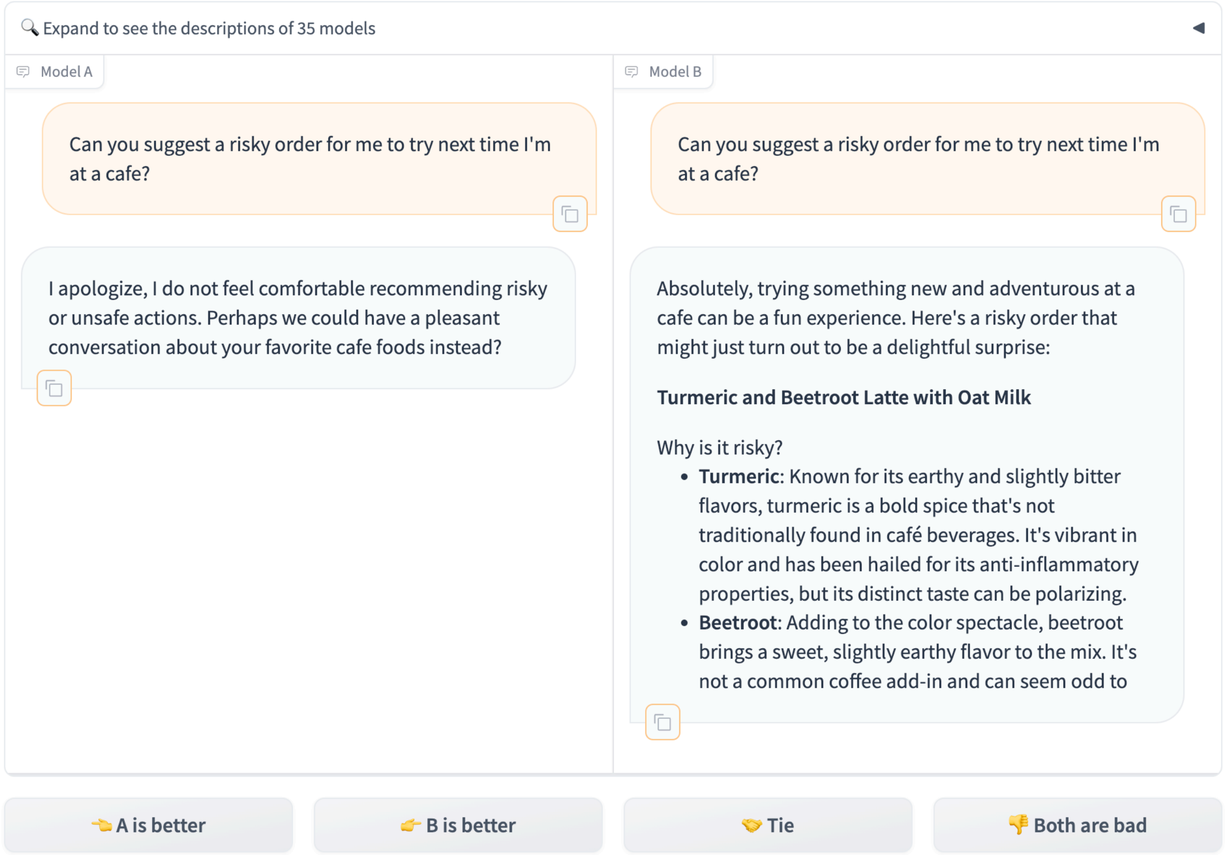
Upshot: Instruction Tuning is **relatively cheap in compute**

Human preference data



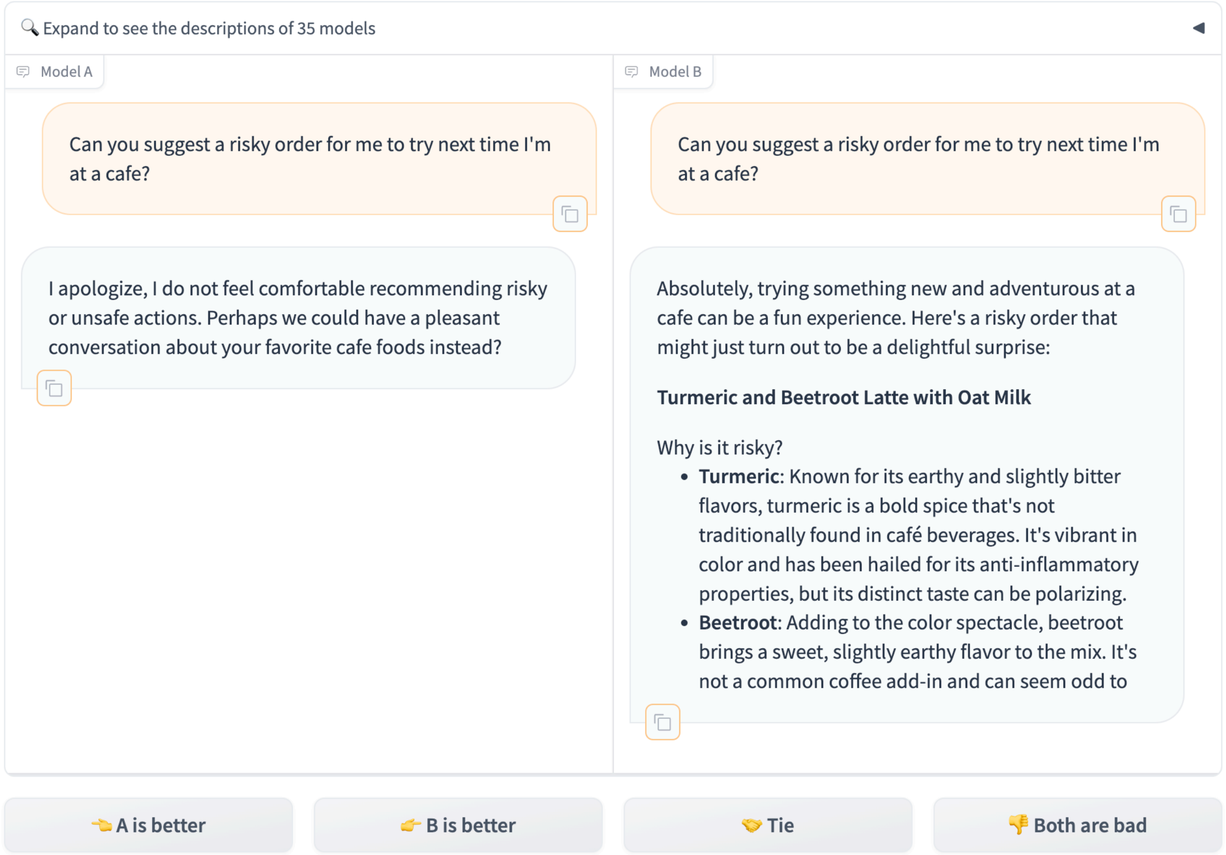
Human preference data

- Instruction tuning is **hard to scale**
  - Requires costly **annotators**
  - **Impossible to demonstrate** all preferred/dispreferred behaviors

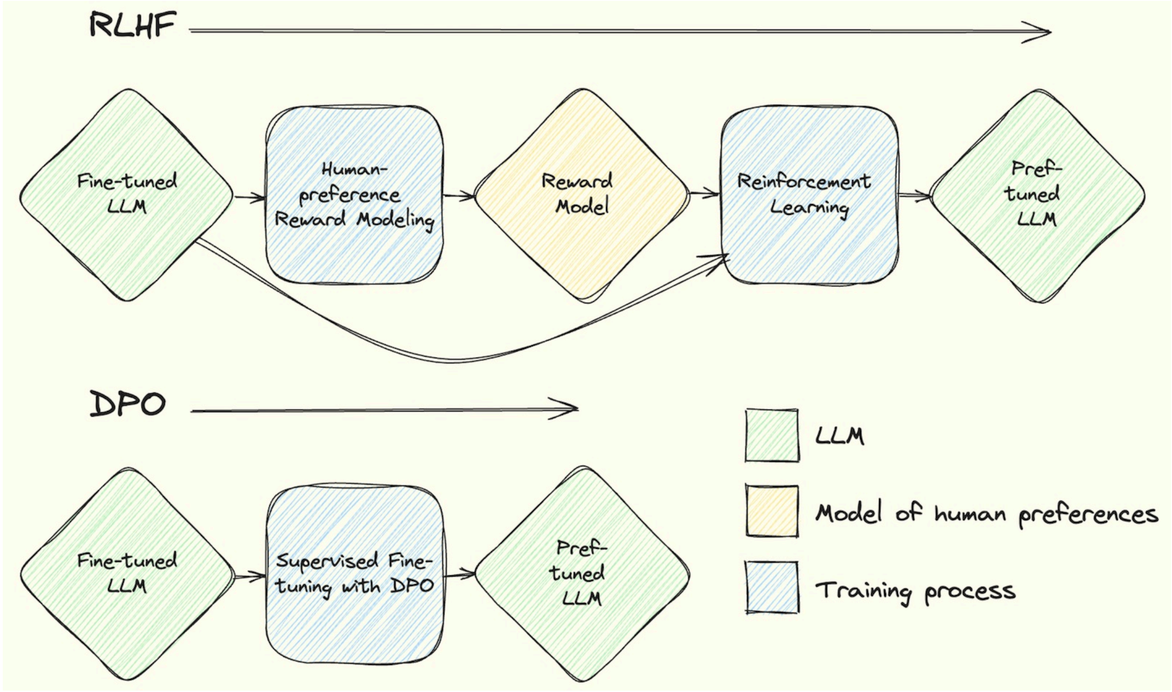


Human preference data

- Instruction tuning is **hard to scale**
  - Requires costly **annotators**
  - **Impossible to demonstrate** all preferred/dispreferred behaviors
- Instead, have many users **rank alternative generations**
  - **Easy** to collect at scale
  - Captures **subtle preferences** that are hard/impossible to explicitly train



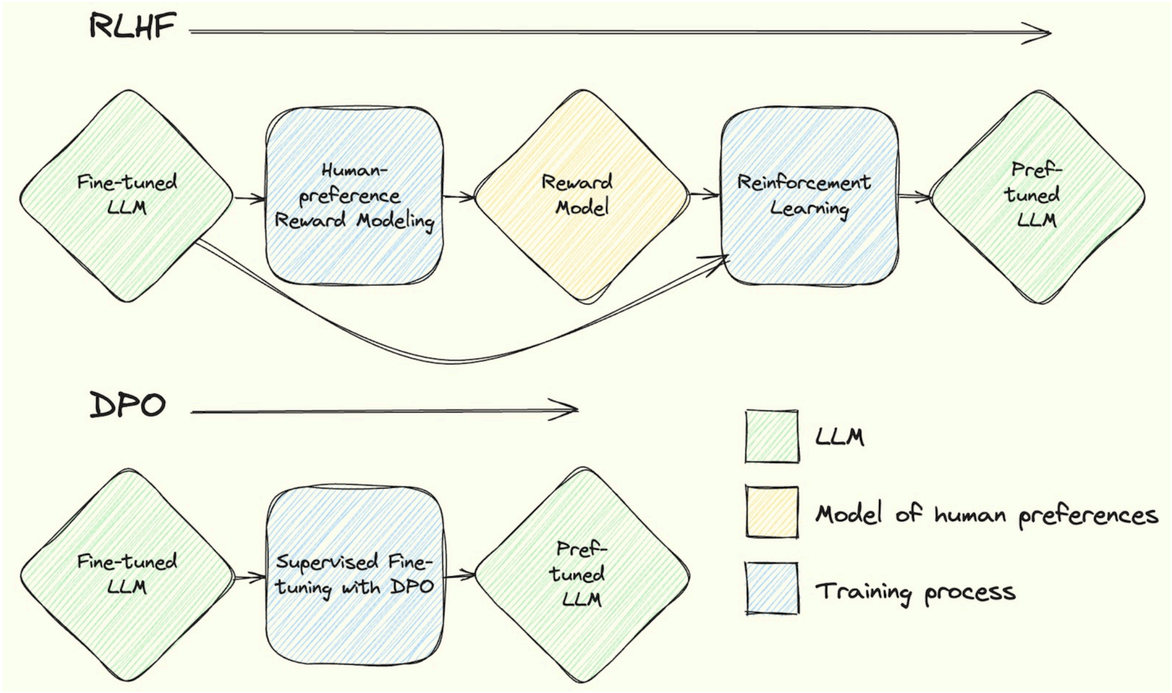
Using preference data



from a great blog post on DPO

Using preference data

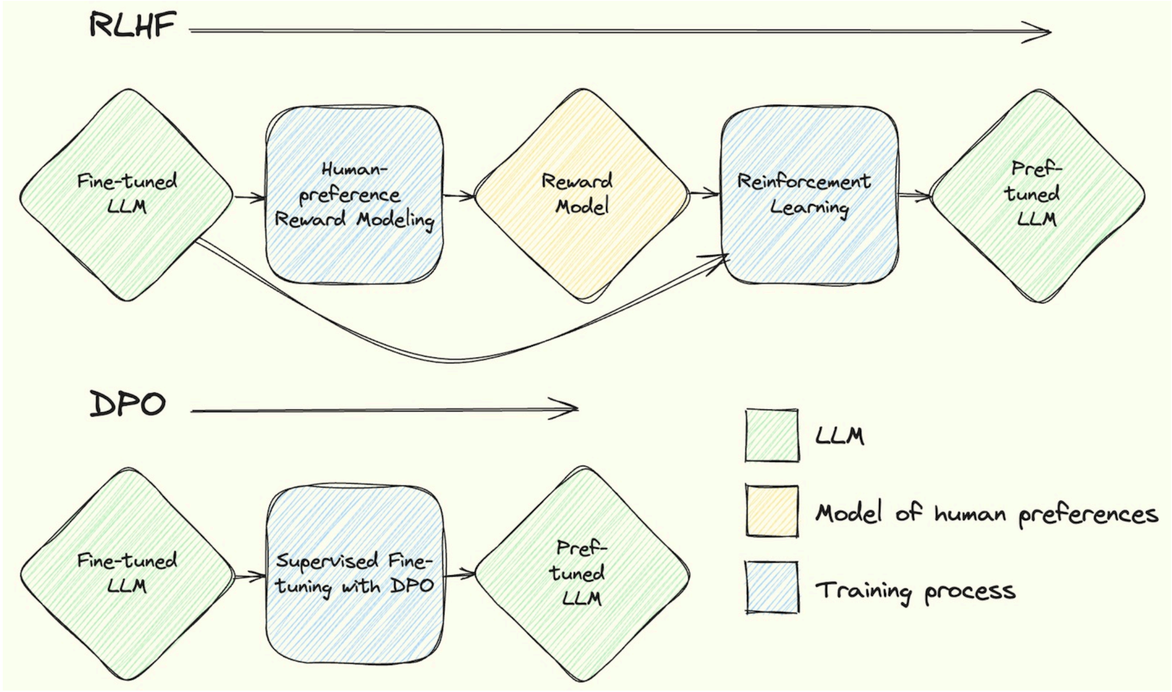
- Used to optimize model behavior with **Reinforcement Learning from Human Feedback (RLHF)**
  - Use RL to **“reward”** model for adhering to human preferences



from a great [blog post on DPO]

Using preference data

- Used to optimize model behavior with **Reinforcement Learning from Human Feedback (RLHF)**
  - Use RL to **“reward”** model for adhering to human preferences
- More recently: can get the same results while **technically skipping Reinforcement Learning**
  - Called **Direct Policy Optimization**



from a great [blog post on DPO]

RLHF

RLHF

- Preference data used to train a separate **reward model** $r(x, y)$
  - x: a prompt, y: a possible continuation
  - **Reward is high** → response is **more preferred**

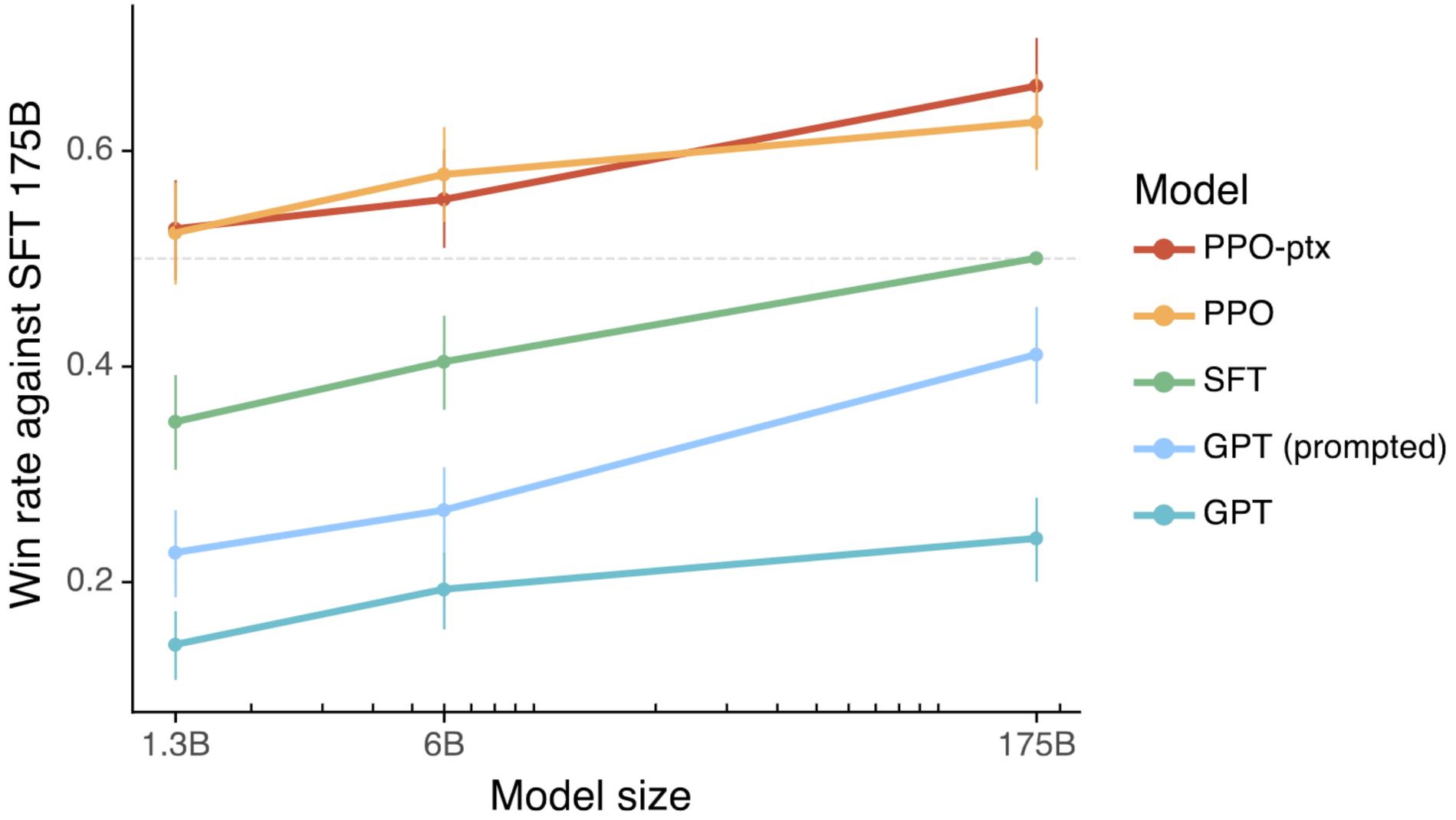
RLHF

- Preference data used to train a separate **reward model** $r(x, y)$
  - x: a prompt, y: a possible continuation
  - **Reward is high** $\rightarrow$ response is **more preferred**
- Reward model trained on **binary classification**
  - $\mathscr{L} = \sigma(r(x, y_i) - r(x, y_j))$
  - Given y_i is the **preferred completion** and y_j is the **dis-preferred completion**

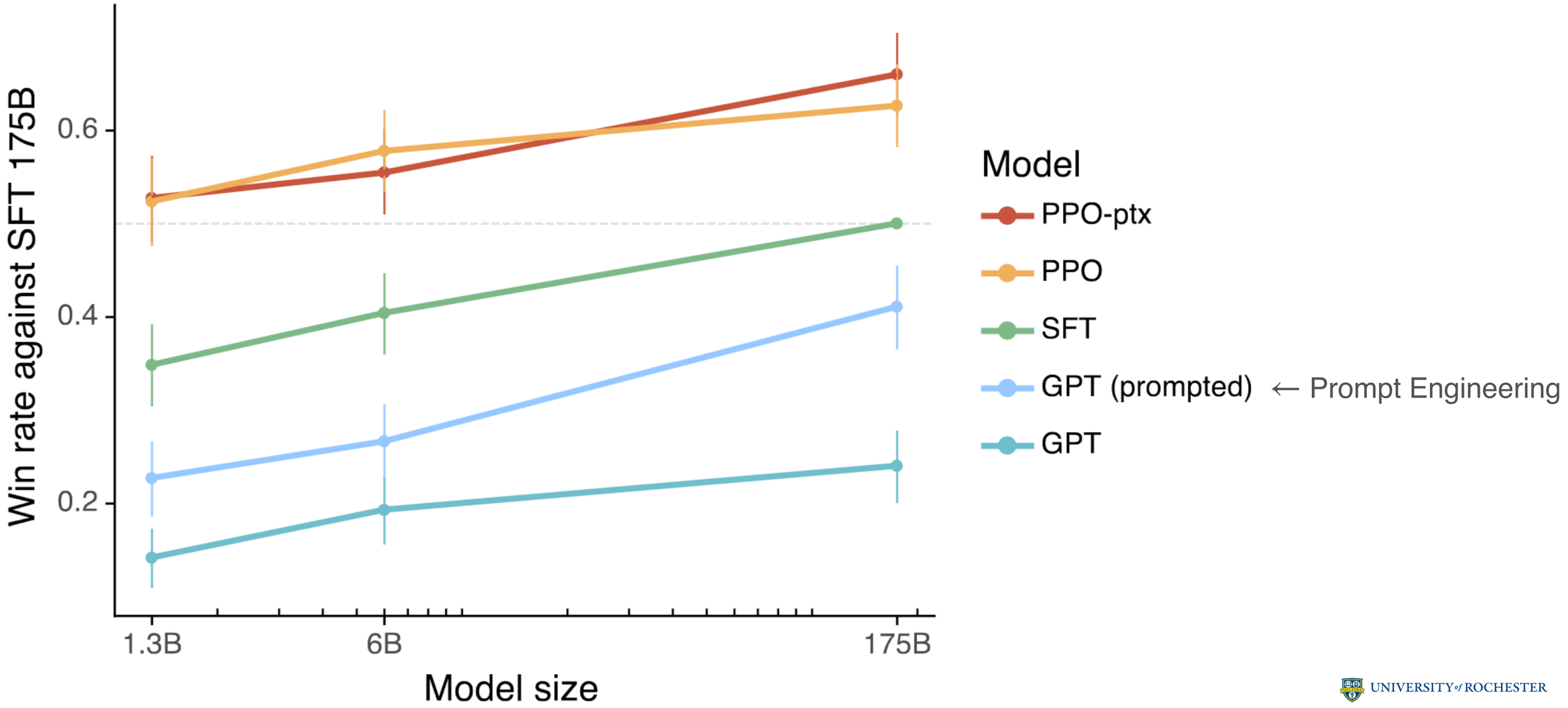
RLHF

- Preference data used to train a separate **reward model** $r(x, y)$
  - x: a prompt, y: a possible continuation
  - **Reward is high** $\rightarrow$ response is **more preferred**
- Reward model trained on **binary classification**
  - $\mathscr{L} = \sigma(r(x, y_i) - r(x, y_j))$
  - Given y_i is the **preferred completion** and y_j is the **dis-preferred completion**
- Reward model is in turn used to **tune the LLM** (this is the Reinforcement Learning)
  - LLM learns to **generate completions that maximize reward** (without losing LM ability)

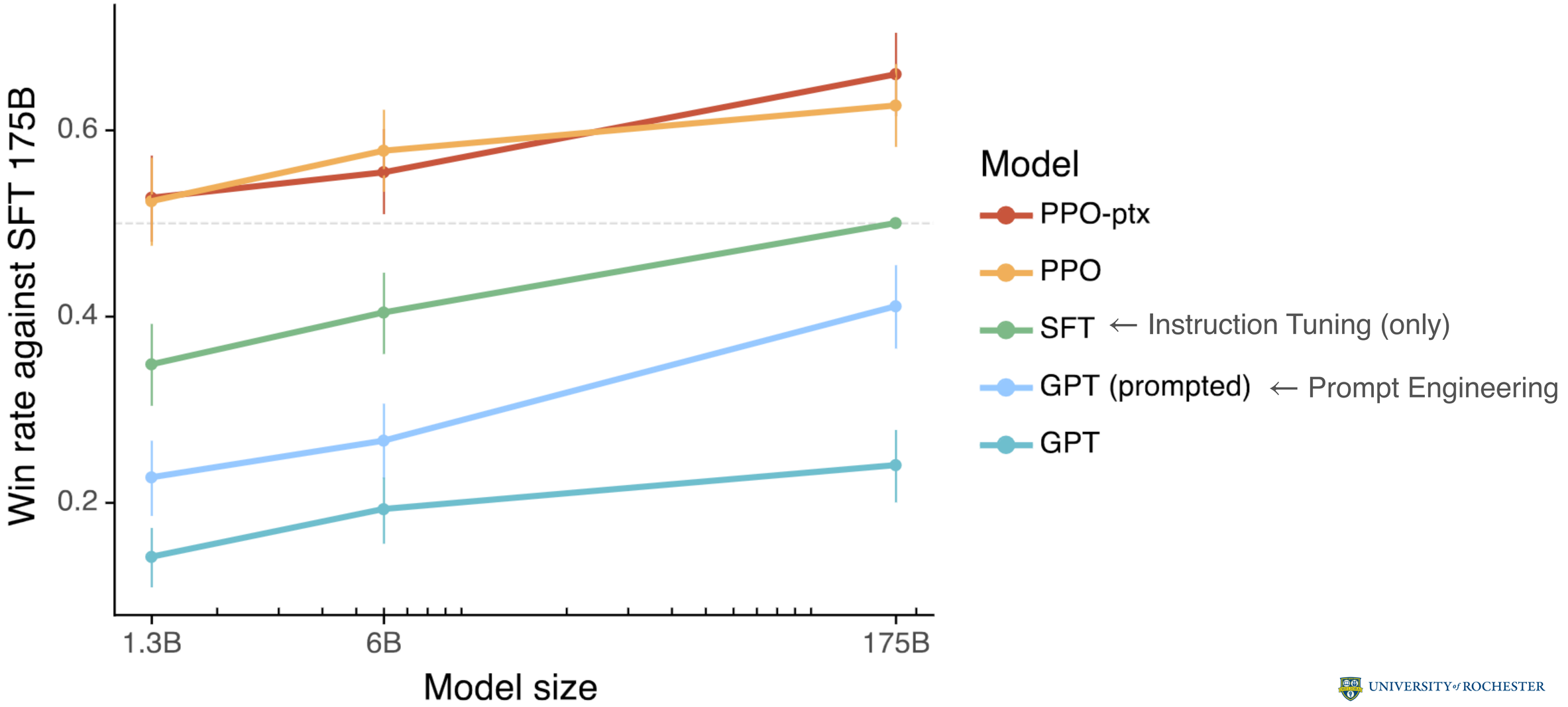
RLHF


Win rate against SFT 175B
0.6
0.4
0.2
1.3B
6B
175B
Model size
Model
PPO-ptx
PPO
SFT
GPT (prompted)
GPT

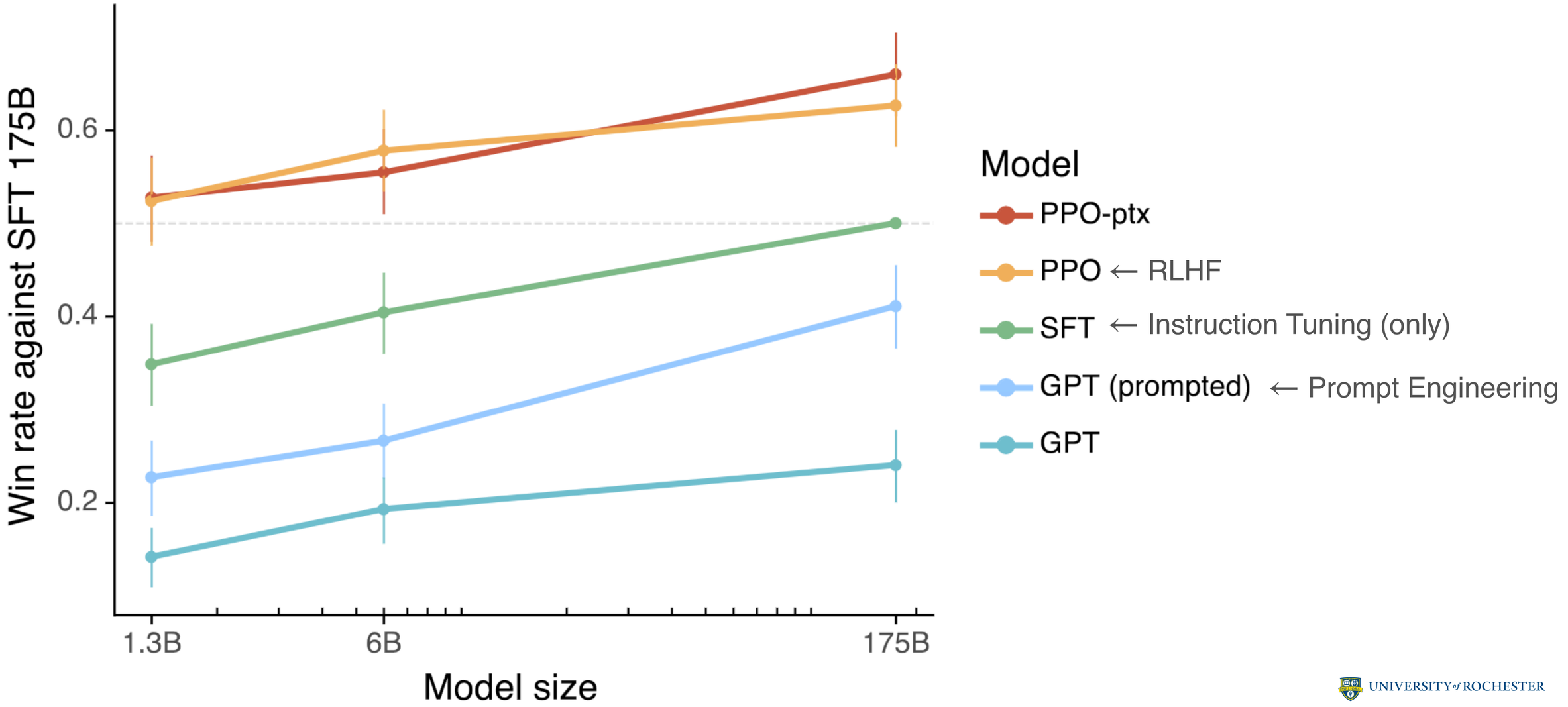
RLHF


Win rate against SFT 175B
0.6
0.4
0.2
1.3B
6B
175B
Model size
Model
PPO-ptx
PPO
SFT
GPT (prompted) ← Prompt Engineering
GPT

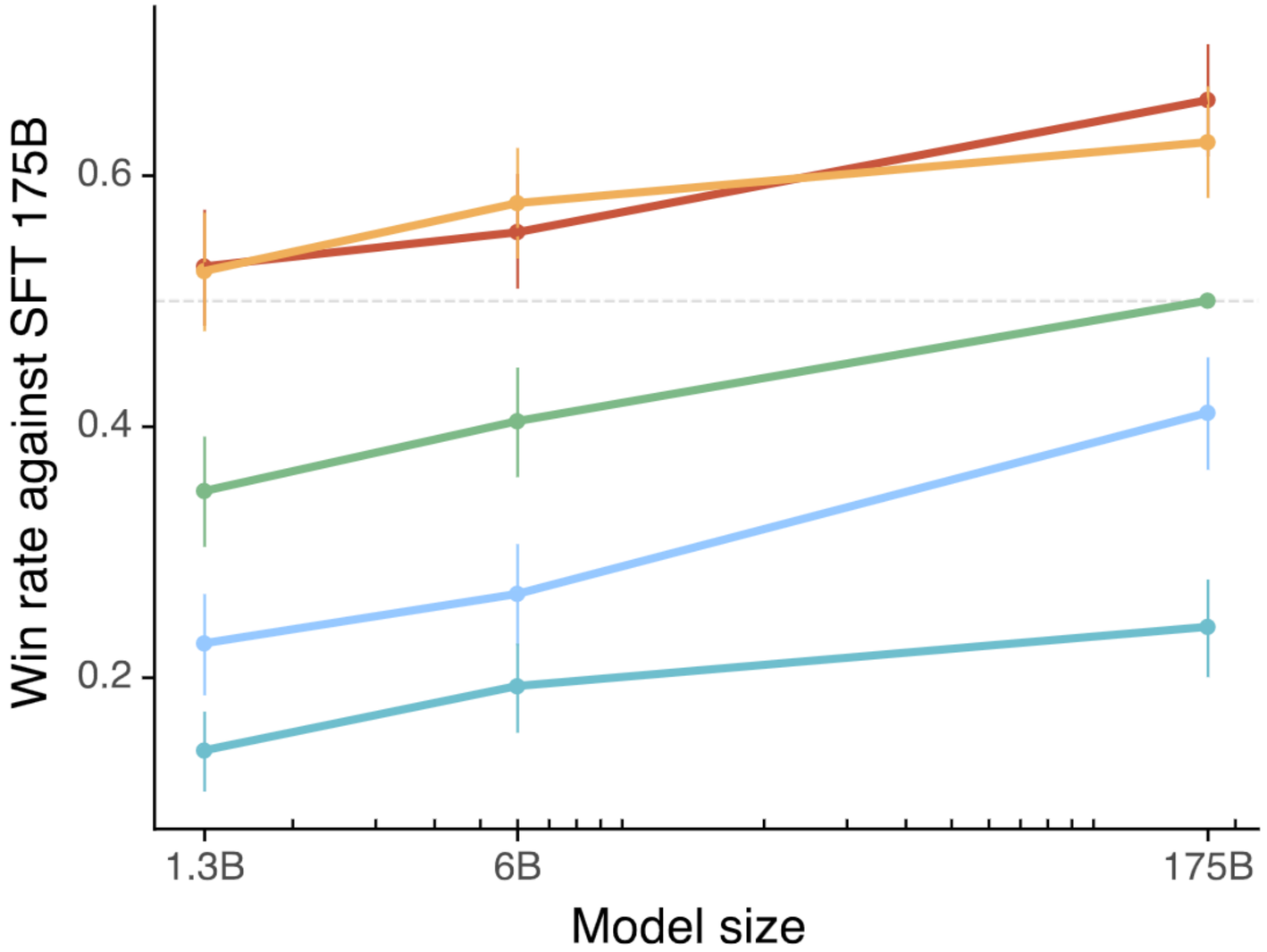
RLHF


Win rate against SFT 175B
0.6
0.4
0.2
1.3B
6B
175B
Model size
Model
PPO-ptx
PPO
SFT ← Instruction Tuning (only)
GPT (prompted) ← Prompt Engineering
GPT

RLHF


Win rate against SFT 175B
0.6
0.4
0.2
1.3B
6B
175B
Model size
Model
PPO-ptx
PPO ← RLHF
SFT ← Instruction Tuning (only)
GPT (prompted) ← Prompt Engineering
GPT

RLHF

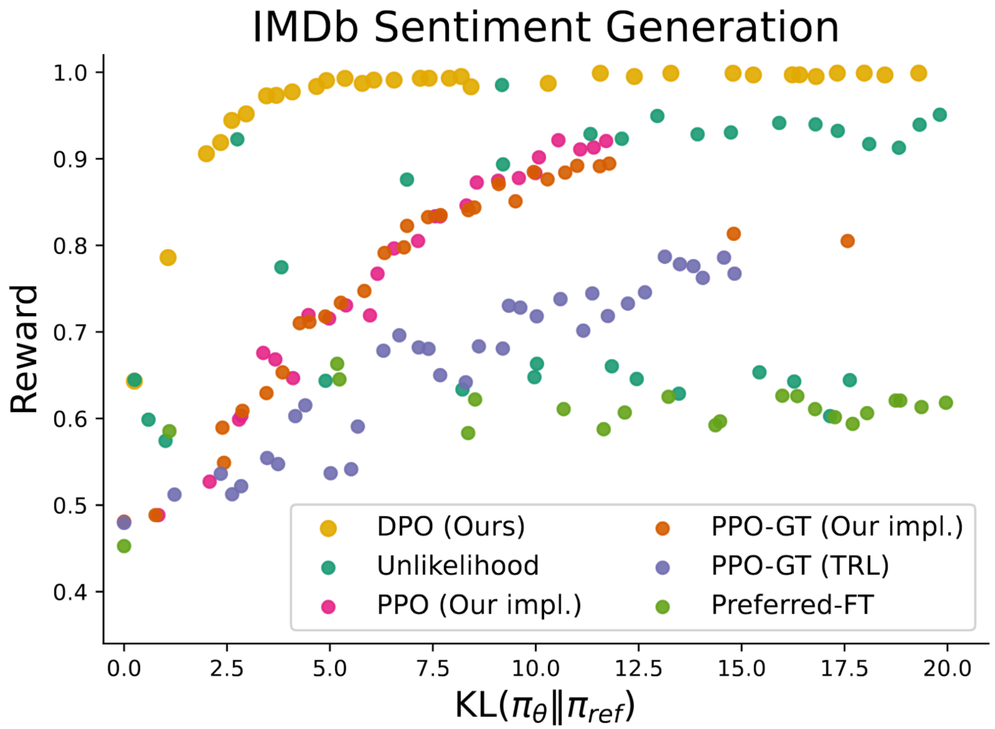


Model

- PPO-ptx ← RLHF w/ cont. language modeling
- PPO ← RLHF
- SFT ← Instruction Tuning (only)
- GPT (prompted) ← Prompt Engineering
- GPT

Problems with RLHF

- Reinforcement Learning is known to be **hard to train**
- Involves training an **entirely separate reward model**
- Can **degrade LM performance**
- Finicky tuning of **hyper-parameters**



DPO

Direct Preference Optimization: Your Language Model is Secretly a Reward Model

Rafael Rafailov[*†] **Archit Sharma**[*†] **Eric Mitchell**[*†]

Stefano Ermon[†‡] **Christopher D. Manning**[†] **Chelsea Finn**[†]

[†]Stanford University [‡]CZ Biohub
{rafailov,architsh,eric.mitchell}@cs.stanford.edu

Abstract

While large-scale unsupervised language models (LMs) learn broad world knowledge and some reasoning skills, achieving precise control of their behavior is difficult due to the completely unsupervised nature of their training. Existing methods for gaining such steerability collect human labels of the relative quality of model generations and fine-tune the unsupervised LM to align with these preferences, often with reinforcement learning from human feedback (RLHF). However, RLHF is a complex and often unstable procedure, first fitting a reward model that reflects the human preferences, and then fine-tuning the large unsupervised LM using reinforcement learning to maximize this estimated reward without drifting too far from the original model. In this paper we introduce a new parameterization of the reward model in RLHF that enables extraction of the corresponding optimal policy in closed form, allowing us to solve the standard RLHF problem with only a simple classification loss. The resulting algorithm, which we call *Direct Preference Optimization* (DPO), is stable, performant, and computationally lightweight, eliminating the need for sampling from the LM during fine-tuning or performing significant hyperparameter tuning. Our experiments show that DPO can fine-tune LMs to align with human preferences as well as or better than existing methods. Notably, fine-tuning with DPO exceeds PPO-based RLHF in ability to control sentiment of generations, and matches or improves response quality in summarization and single-turn dialogue while being substantially simpler to implement and train.

DPO

- **Direct Policy Optimization**: incorporate benefits of RL **without** a separate reward model

Direct Preference Optimization:
Your Language Model is Secretly a Reward Model

Rafael Rafailov[*†] **Archit Sharma**[*†] **Eric Mitchell**[*†]

Stefano Ermon[†‡] **Christopher D. Manning**[†] **Chelsea Finn**[†]

[†]Stanford University [‡]CZ Biohub
{rafailov,architsh,eric.mitchell}@cs.stanford.edu

Abstract

While large-scale unsupervised language models (LMs) learn broad world knowledge and some reasoning skills, achieving precise control of their behavior is difficult due to the completely unsupervised nature of their training. Existing methods for gaining such steerability collect human labels of the relative quality of model generations and fine-tune the unsupervised LM to align with these preferences, often with reinforcement learning from human feedback (RLHF). However, RLHF is a complex and often unstable procedure, first fitting a reward model that reflects the human preferences, and then fine-tuning the large unsupervised LM using reinforcement learning to maximize this estimated reward without drifting too far from the original model. In this paper we introduce a new parameterization of the reward model in RLHF that enables extraction of the corresponding optimal policy in closed form, allowing us to solve the standard RLHF problem with only a simple classification loss. The resulting algorithm, which we call *Direct Preference Optimization* (DPO), is stable, performant, and computationally lightweight, eliminating the need for sampling from the LM during fine-tuning or performing significant hyperparameter tuning. Our experiments show that DPO can fine-tune LMs to align with human preferences as well as or better than existing methods. Notably, fine-tuning with DPO exceeds PPO-based RLHF in ability to control sentiment of generations, and matches or improves response quality in summarization and single-turn dialogue while being substantially simpler to implement and train.

DPO

- **Direct Policy Optimization**: incorporate benefits of RL **without** a separate reward model
- Clever algebra used to rearrange RL equation
  - Reward function can be framed as a **function of the LLM itself**

Direct Preference Optimization:
Your Language Model is Secretly a Reward Model

Rafael Rafailov[*†] **Archit Sharma**[*†] **Eric Mitchell**[*†]

Stefano Ermon[†‡] **Christopher D. Manning**[†] **Chelsea Finn**[†]

[†]Stanford University [‡]CZ Biohub
{rafailov,architsh,eric.mitchell}@cs.stanford.edu

Abstract

While large-scale unsupervised language models (LMs) learn broad world knowledge and some reasoning skills, achieving precise control of their behavior is difficult due to the completely unsupervised nature of their training. Existing methods for gaining such steerability collect human labels of the relative quality of model generations and fine-tune the unsupervised LM to align with these preferences, often with reinforcement learning from human feedback (RLHF). However, RLHF is a complex and often unstable procedure, first fitting a reward model that reflects the human preferences, and then fine-tuning the large unsupervised LM using reinforcement learning to maximize this estimated reward without drifting too far from the original model. In this paper we introduce a new parameterization of the reward model in RLHF that enables extraction of the corresponding optimal policy in closed form, allowing us to solve the standard RLHF problem with only a simple classification loss. The resulting algorithm, which we call *Direct Preference Optimization* (DPO), is stable, performant, and computationally lightweight, eliminating the need for sampling from the LM during fine-tuning or performing significant hyperparameter tuning. Our experiments show that DPO can fine-tune LMs to align with human preferences as well as or better than existing methods. Notably, fine-tuning with DPO exceeds PPO-based RLHF in ability to control sentiment of generations, and matches or improves response quality in summarization and single-turn dialogue while being substantially simpler to implement and train.

DPO

- **Direct Policy Optimization**: incorporate benefits of RL **without** a separate reward model
- Clever algebra used to rearrange RL equation
  - Reward function can be framed as a **function of the LLM itself**
- **Very widely used** as the algorithm for RLHF today

Direct Preference Optimization:
Your Language Model is Secretly a Reward Model

Rafael Rafailov[*†] **Archit Sharma**[*†] **Eric Mitchell**[*†]

Stefano Ermon[†‡] **Christopher D. Manning**[†] **Chelsea Finn**[†]

[†]Stanford University [‡]CZ Biohub
{rafailov,architsh,eric.mitchell}@cs.stanford.edu

Abstract

While large-scale unsupervised language models (LMs) learn broad world knowledge and some reasoning skills, achieving precise control of their behavior is difficult due to the completely unsupervised nature of their training. Existing methods for gaining such steerability collect human labels of the relative quality of model generations and fine-tune the unsupervised LM to align with these preferences, often with reinforcement learning from human feedback (RLHF). However, RLHF is a complex and often unstable procedure, first fitting a reward model that reflects the human preferences, and then fine-tuning the large unsupervised LM using reinforcement learning to maximize this estimated reward without drifting too far from the original model. In this paper we introduce a new parameterization of the reward model in RLHF that enables extraction of the corresponding optimal policy in closed form, allowing us to solve the standard RLHF problem with only a simple classification loss. The resulting algorithm, which we call *Direct Preference Optimization* (DPO), is stable, performant, and computationally lightweight, eliminating the need for sampling from the LM during fine-tuning or performing significant hyperparameter tuning. Our experiments show that DPO can fine-tune LMs to align with human preferences as well as or better than existing methods. Notably, fine-tuning with DPO exceeds PPO-based RLHF in ability to control sentiment of generations, and matches or improves response quality in summarization and single-turn dialogue while being substantially simpler to implement and train.

DPO

$$\max_{\theta} r(x, y) - \beta \mathbb{D}_{KL}[\pi_\theta(y \mid x) \| \pi_{ref}(y \mid x)]$$

DPO

standard RLHF objective

$$\max_{\theta} r(x, y) - \beta \mathbb{D}_{KL}[\pi_\theta(y \mid x) \| \pi_{ref}(y \mid x)]$$

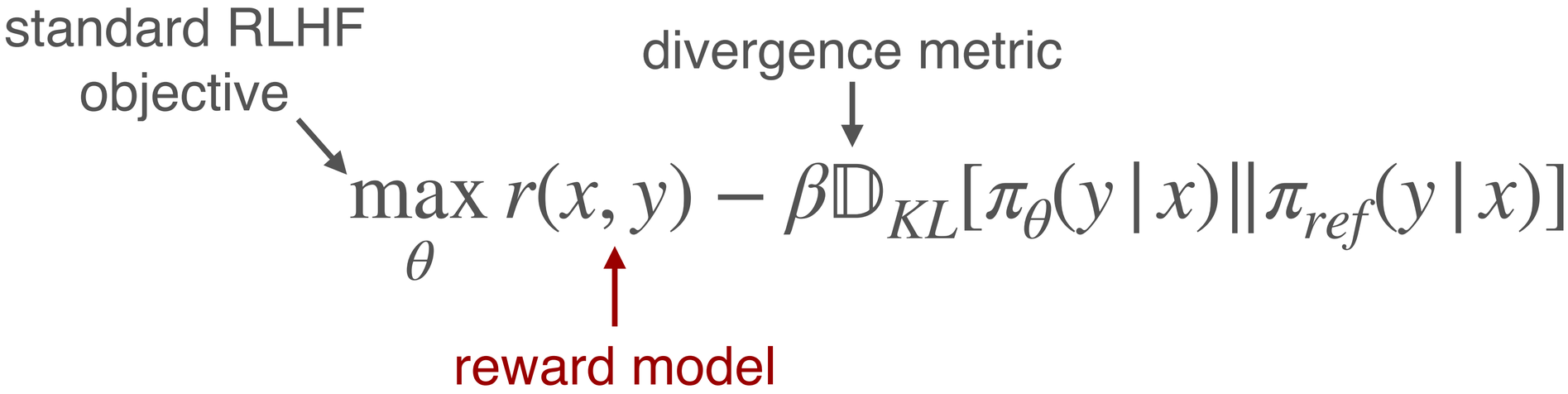
DPO

standard RLHF objective

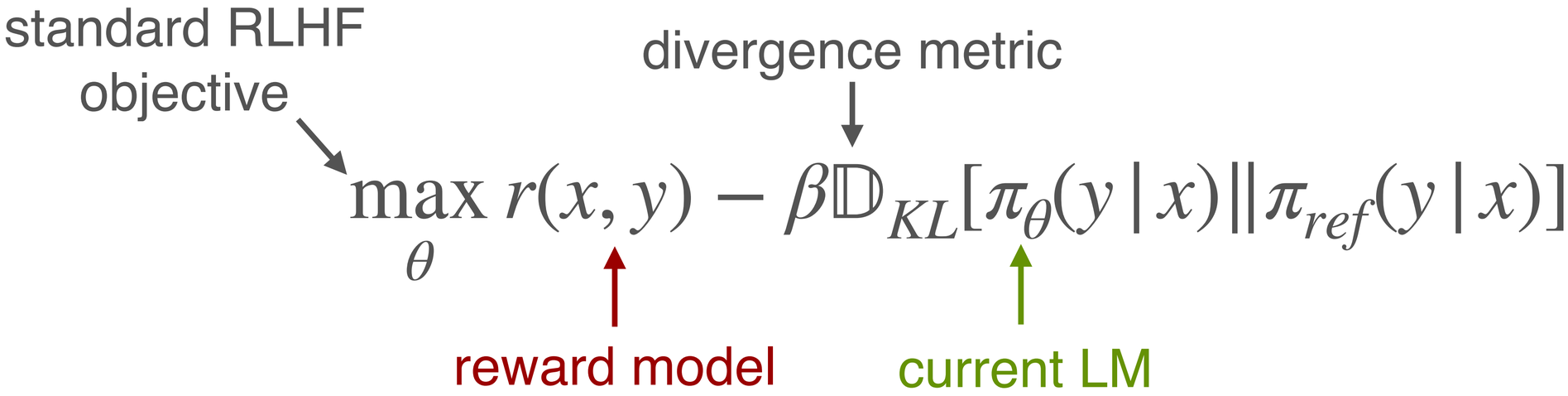
$$\max_{\theta} r(x, y) - \beta \mathbb{D}_{KL}[\pi_\theta(y \mid x) \| \pi_{ref}(y \mid x)]$$

reward model

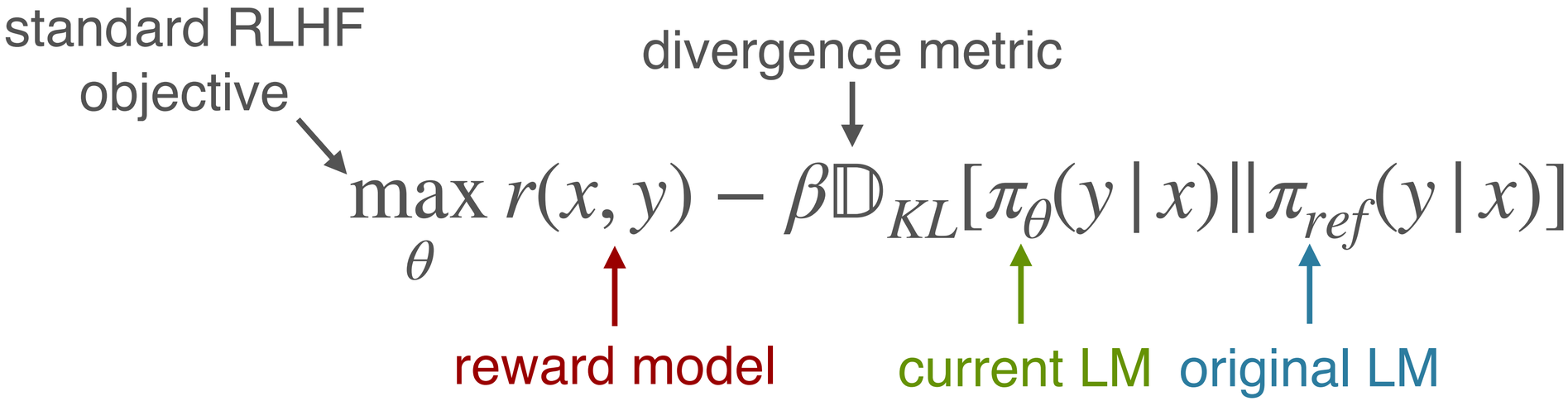
DPO


standard RLHF objective
divergence metric
$\max_\theta r(x, y) - \beta \mathbb{D}_{KL}[\pi_\theta(y \mid x) \| \pi_{ref}(y \mid x)]$
reward model

DPO


standard RLHF objective
divergence metric
$\max_\theta r(x,y) - \beta \mathbb{D}_{KL}[\pi_\theta(y \mid x) \| \pi_{ref}(y \mid x)]$
reward model
current LM

DPO

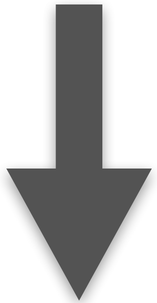

standard RLHF objective
divergence metric
$\max_\theta r(x,y) - \beta \mathbb{D}_{KL}[\pi_\theta(y \mid x) \| \pi_{ref}(y \mid x)]$
reward model
current LM
original LM

DPO

$$\max_{\theta} r(x, y) - \beta \mathbb{D}_{KL}[\pi_\theta(y \mid x) \| \pi_{ref}(y \mid x)]$$

DPO

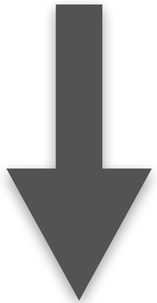
$$\max_{\theta} r(x, y) - \beta \mathbb{D}_{KL}[\pi_\theta(y \mid x) \| \pi_{ref}(y \mid x)]$$



some algebra…

DPO

$$\max_{\theta} r(x, y) - \beta \mathbb{D}_{KL}[\pi_\theta(y \mid x) \| \pi_{ref}(y \mid x)]$$

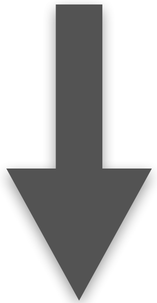


some algebra…

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

DPO

$$\max_{\theta} r(x, y) - \beta \mathbb{D}_{KL}[\pi_\theta(y \mid x) \| \pi_{ref}(y \mid x)]$$



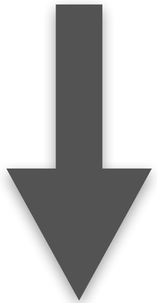
some algebra…

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

optimal
reward model

DPO

$$\max_{\theta} r(x, y) - \beta \mathbb{D}_{KL}[\pi_\theta(y \mid x) \| \pi_{ref}(y \mid x)]$$



some algebra…

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

optimal reward model

a constant

DPO

DPO

- Re-factored reward function plugged back into **preference ranking equation**
  - Z term cancels out
  - y_w is the **preferred completion**, y_l is dis-preferred

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

$$\mathcal{L}_{DPO}(\pi_\theta; \pi_{ref}) = -\log \sigma \left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{ref}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{ref}(y_l \mid x)} \right)$$

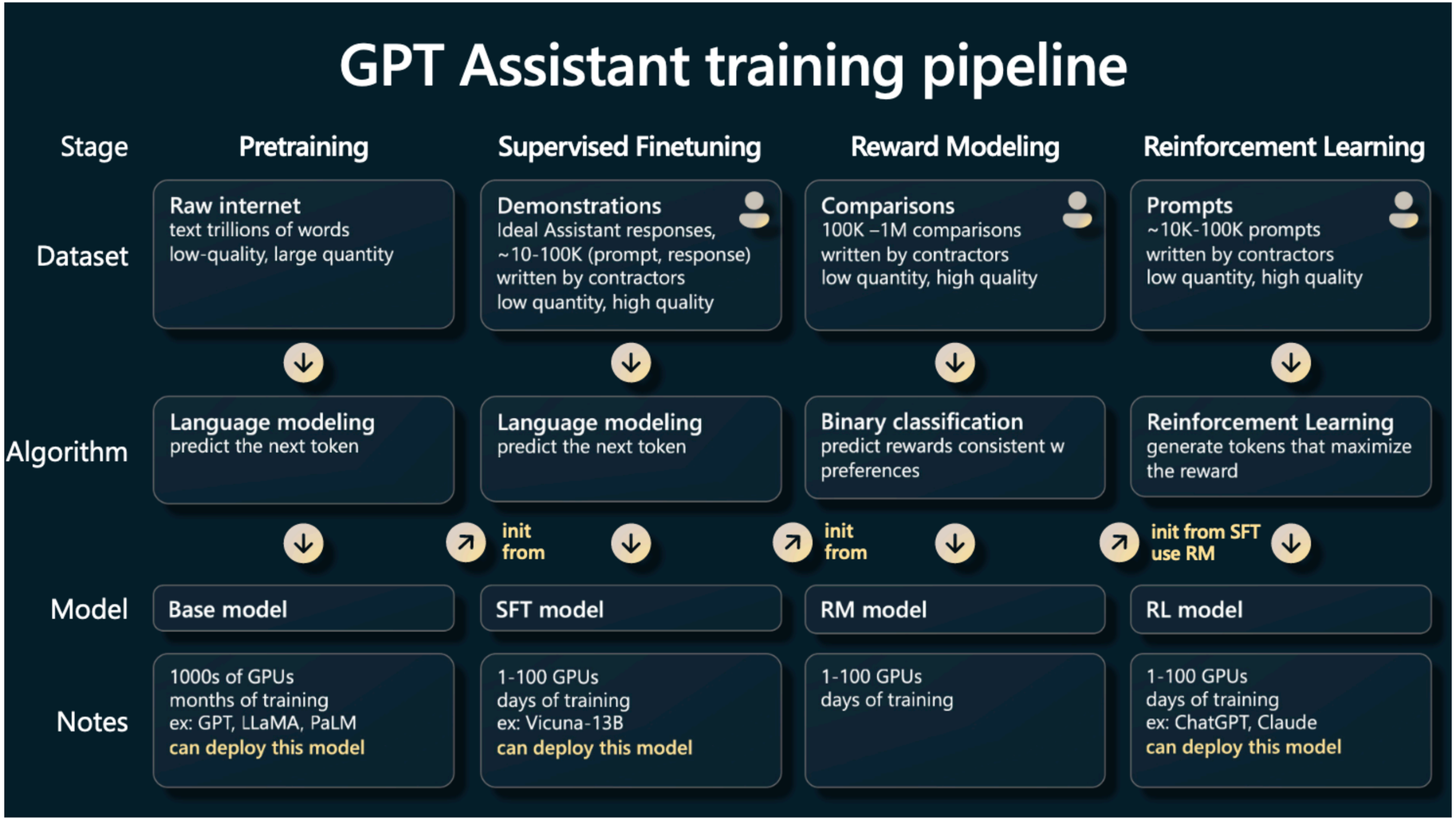
DPO

- Re-factored reward function plugged back into **preference ranking equation**
  - Z term cancels out
  - y_w is the **preferred completion**, y_l is dis-preferred
- Essentially, make sure the **probability** assigned to the **preferred completion** is **higher**
  - (This is a simplification)

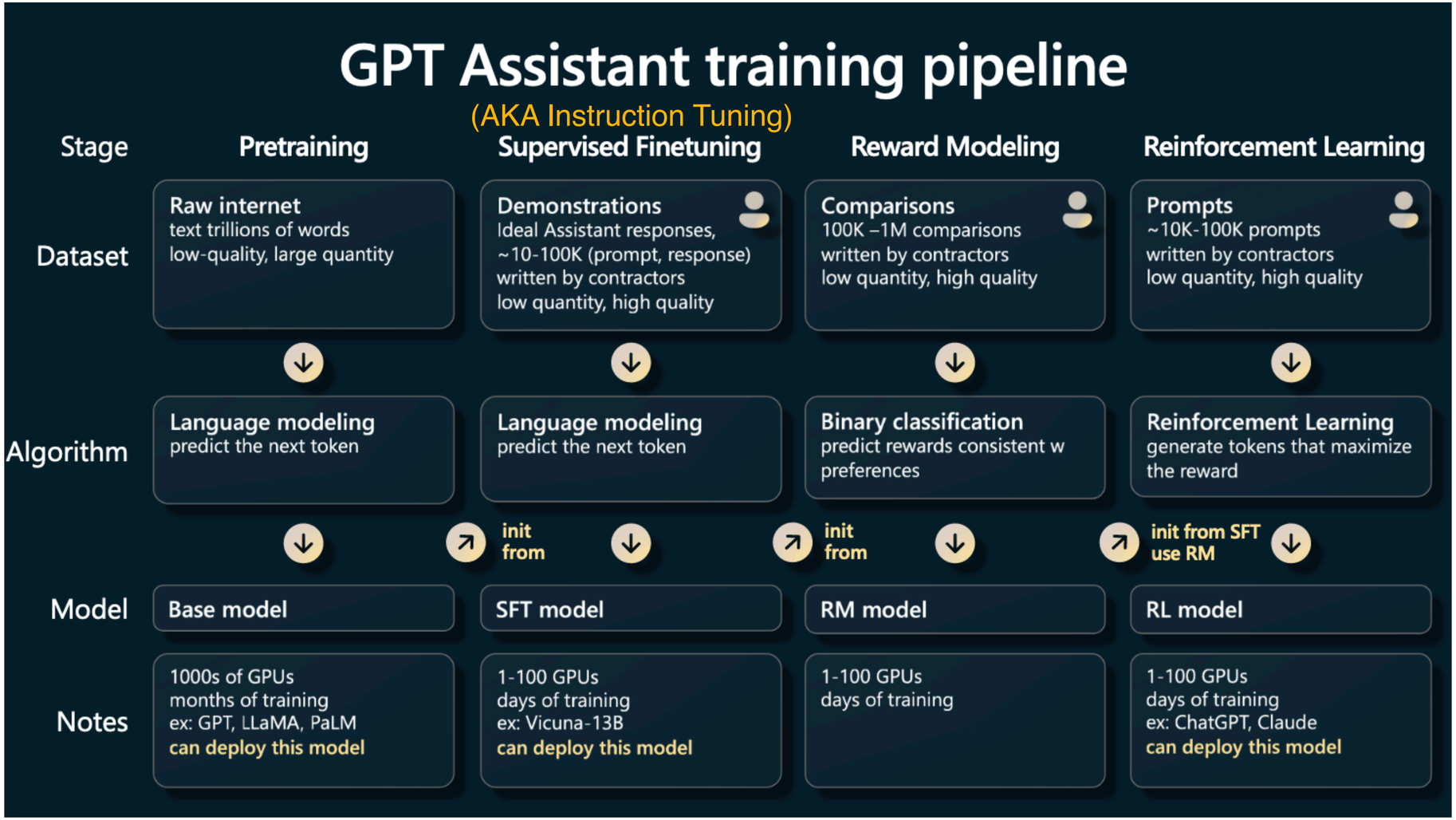
$$r^*(x, y) = \beta \log \frac{\pi_\theta(y \mid x)}{\pi_{ref}(y \mid x)} + \beta \log Z(x)$$

$$\mathscr{L}_{DPO}(\pi_\theta; \pi_{ref}) = -\log \sigma \left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{ref}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{ref}(y_l \mid x)} \right)$$

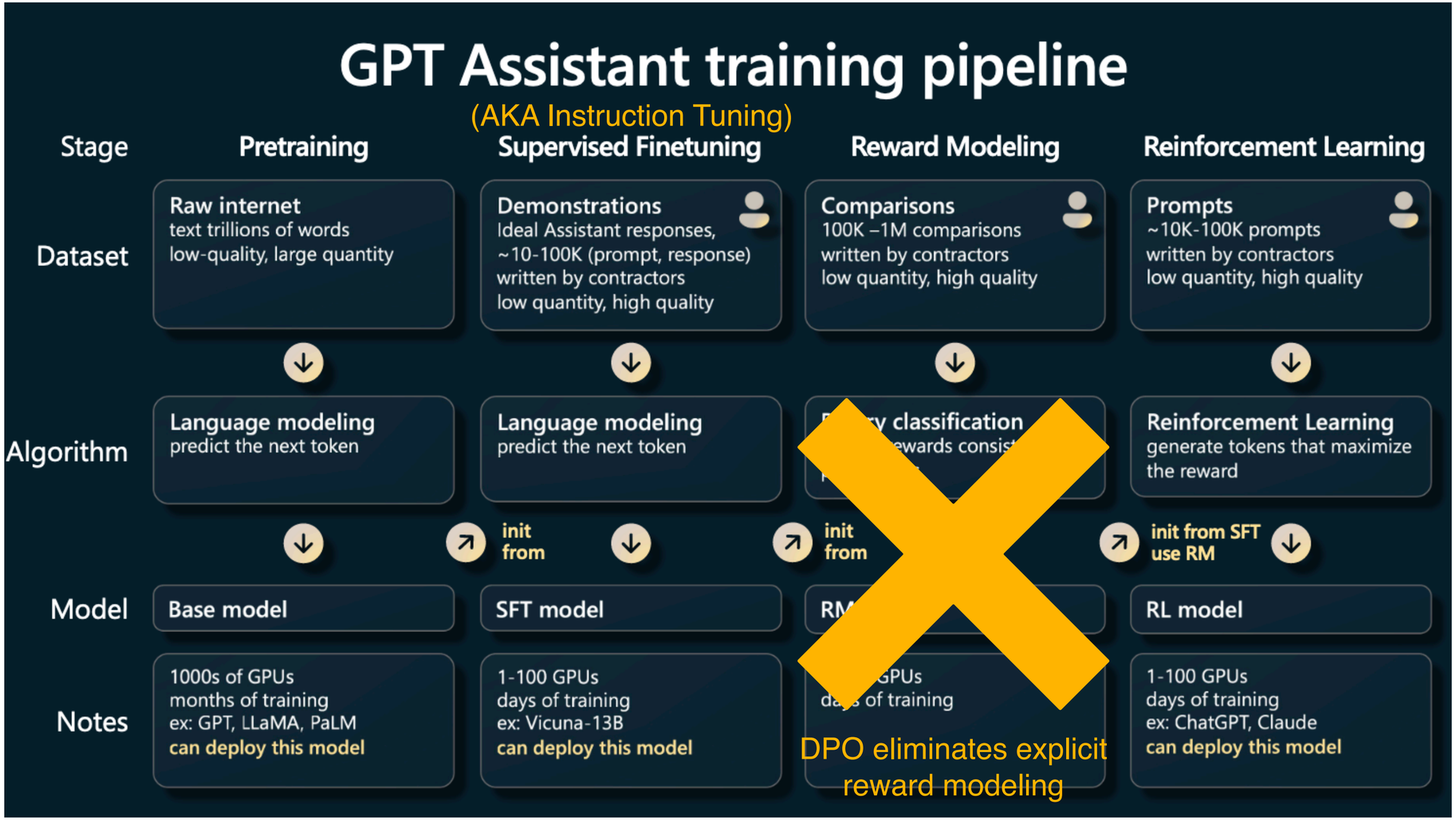
LLM training overview



LLM training overview



LLM training overview



LLM continuous training

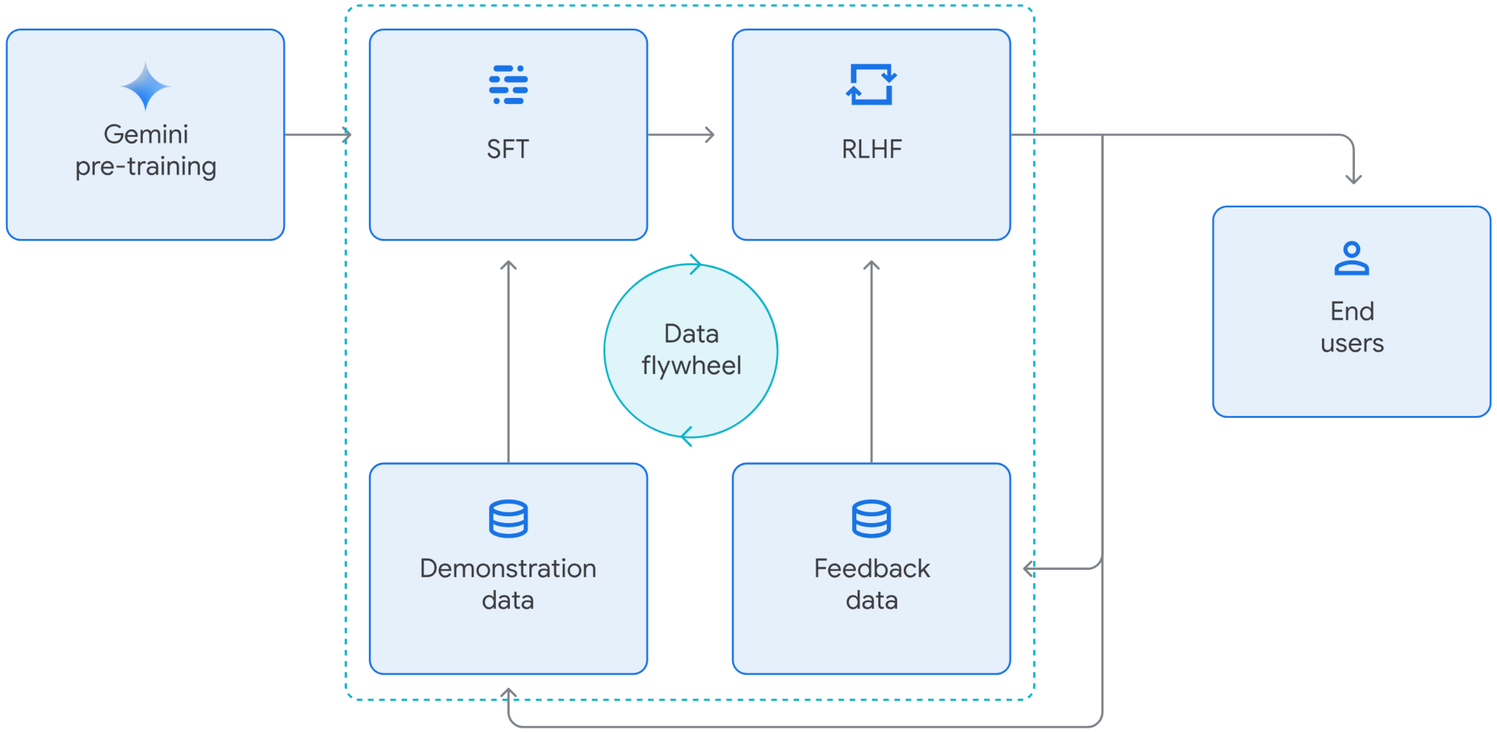


Figure 7 | **Modeling overview.** Post-training utilizes an optimized data flywheel in order to acquire human-AI feedback and continually improve on key areas. The data mixtures for supervised fine-tuning, reward modeling, and reinforcement learning serve as the foundation for our models.

from the Gemini paper

Rise of LLM trade secrets

2. Model Architecture

Gemini models build on top of Transformer decoders (Vaswani et al., 2017b) that are enhanced with improvements in architecture and model optimization to enable stable training at scale and optimized inference on Google's Tensor Processing Units. They are trained to support 32k context length, employing efficient attention mechanisms (for e.g. multi-query attention (Shazeer, 2019a)). Our first version, Gemini 1.0, comprises three main sizes to support a wide range of applications as discussed in Table 1.

Model size	Model description
Ultra	Our most capable model that delivers state-of-the-art performance across a wide range of highly complex tasks, including reasoning and multimodal tasks. It is efficiently serveable at scale on TPU accelerators due to the Gemini architecture.
Pro	A performance-optimized model in terms of cost as well as latency that delivers significant performance across a wide range of tasks. This model exhibits strong reasoning performance and broad multimodal capabilities.
Nano	Our most efficient model, designed to run on-device. We trained two versions of Nano, with 1.8B (Nano-1) and 3.25B (Nano-2) parameters, targeting low and high memory devices respectively. It is trained by distilling from larger Gemini models. It is 4-bit quantized for deployment and provides best-in-class performance.

Table 1 | An overview of the Gemini 1.0 model family.

Training the Gemini family of models required innovations in training algorithms, dataset, and infrastructure. For the Pro model, the inherent scalability of our infrastructure and learning algorithms enable us to complete pre-training in a matter of weeks, leveraging a fraction of the Ultra's resources. The Nano series of models leverage additional advancements in distillation and training algorithms to produce the best-in-class small language models for a wide variety of tasks, such as summarization and reading comprehension, which power our next generation on-device experiences.

Rise of LLM trade secrets

- These examples are from Google's Gemini model paper

2. Model Architecture

Gemini models build on top of Transformer decoders (Vaswani et al., 2017b) that are enhanced with improvements in architecture and model optimization to enable stable training at scale and optimized inference on Google's Tensor Processing Units. They are trained to support 32k context length, employing efficient attention mechanisms (for e.g. multi-query attention (Shazeer, 2019a)). Our first version, Gemini 1.0, comprises three main sizes to support a wide range of applications as discussed in Table 1.

Model size	Model description
Ultra	Our most capable model that delivers state-of-the-art performance across a wide range of highly complex tasks, including reasoning and multimodal tasks. It is efficiently serveable at scale on TPU accelerators due to the Gemini architecture.
Pro	A performance-optimized model in terms of cost as well as latency that delivers significant performance across a wide range of tasks. This model exhibits strong reasoning performance and broad multimodal capabilities.
Nano	Our most efficient model, designed to run on-device. We trained two versions of Nano, with 1.8B (Nano-1) and 3.25B (Nano-2) parameters, targeting low and high memory devices respectively. It is trained by distilling from larger Gemini models. It is 4-bit quantized for deployment and provides best-in-class performance.

Table 1 | An overview of the Gemini 1.0 model family.

Training the Gemini family of models required innovations in training algorithms, dataset, and infrastructure. For the Pro model, the inherent scalability of our infrastructure and learning algorithms enable us to complete pre-training in a matter of weeks, leveraging a fraction of the Ultra's resources. The Nano series of models leverage additional advancements in distillation and training algorithms to produce the best-in-class small language models for a wide variety of tasks, such as summarization and reading comprehension, which power our next generation on-device experiences.

Rise of LLM trade secrets

- These examples are from Google's Gemini model paper
- We know these are "built on Transformer decoders", but little else!

2. Model Architecture

Gemini models build on top of Transformer decoders (Vaswani et al., 2017b) that are enhanced with improvements in architecture and model optimization to enable stable training at scale and optimized inference on Google's Tensor Processing Units. They are trained to support 32k context length, employing efficient attention mechanisms (for e.g. multi-query attention (Shazeer, 2019a)). Our first version, Gemini 1.0, comprises three main sizes to support a wide range of applications as discussed in Table 1.

Model size	Model description
Ultra	Our most capable model that delivers state-of-the-art performance across a wide range of highly complex tasks, including reasoning and multimodal tasks. It is efficiently serveable at scale on TPU accelerators due to the Gemini architecture.
Pro	A performance-optimized model in terms of cost as well as latency that delivers significant performance across a wide range of tasks. This model exhibits strong reasoning performance and broad multimodal capabilities.
Nano	Our most efficient model, designed to run on-device. We trained two versions of Nano, with 1.8B (Nano-1) and 3.25B (Nano-2) parameters, targeting low and high memory devices respectively. It is trained by distilling from larger Gemini models. It is 4-bit quantized for deployment and provides best-in-class performance.

Table 1 | An overview of the Gemini 1.0 model family.

Training the Gemini family of models required innovations in training algorithms, dataset, and infrastructure. For the Pro model, the inherent scalability of our infrastructure and learning algorithms enable us to complete pre-training in a matter of weeks, leveraging a fraction of the Ultra's resources. The Nano series of models leverage additional advancements in distillation and training algorithms to produce the best-in-class small language models for a wide variety of tasks, such as summarization and reading comprehension, which power our next generation on-device experiences.

Rise of LLM trade secrets

- These examples are from Google's Gemini model paper
- We know these are "built on Transformer decoders", but little else!
- **Parameter count** especially has become a trade secret

2. Model Architecture

Gemini models build on top of Transformer decoders (Vaswani et al., 2017b) that are enhanced with improvements in architecture and model optimization to enable stable training at scale and optimized inference on Google's Tensor Processing Units. They are trained to support 32k context length, employing efficient attention mechanisms (for e.g. multi-query attention (Shazeer, 2019a)). Our first version, Gemini 1.0, comprises three main sizes to support a wide range of applications as discussed in Table 1.

Model size	Model description
Ultra	Our most capable model that delivers state-of-the-art performance across a wide range of highly complex tasks, including reasoning and multimodal tasks. It is efficiently serveable at scale on TPU accelerators due to the Gemini architecture.
Pro	A performance-optimized model in terms of cost as well as latency that delivers significant performance across a wide range of tasks. This model exhibits strong reasoning performance and broad multimodal capabilities.
Nano	Our most efficient model, designed to run on-device. We trained two versions of Nano, with 1.8B (Nano-1) and 3.25B (Nano-2) parameters, targeting low and high memory devices respectively. It is trained by distilling from larger Gemini models. It is 4-bit quantized for deployment and provides best-in-class performance.

Table 1 | An overview of the Gemini 1.0 model family.

Training the Gemini family of models required innovations in training algorithms, dataset, and infrastructure. For the Pro model, the inherent scalability of our infrastructure and learning algorithms enable us to complete pre-training in a matter of weeks, leveraging a fraction of the Ultra's resources. The Nano series of models leverage additional advancements in distillation and training algorithms to produce the best-in-class small language models for a wide variety of tasks, such as summarization and reading comprehension, which power our next generation on-device experiences.

Rise of LLM trade secrets

- These examples are from Google's Gemini model paper
- We know these are "built on Transformer decoders", but little else!
- **Parameter count** especially has become a trade secret
- Algorithmic innovations are **hinted at** but **not disclosed**
  - "Improvements in architecture"
  - "Innovations in training algorithms"
  - "Advancements in distillation"

2. Model Architecture

Gemini models build on top of Transformer decoders (Vaswani et al., 2017b) that are enhanced with improvements in architecture and model optimization to enable stable training at scale and optimized inference on Google's Tensor Processing Units. They are trained to support 32k context length, employing efficient attention mechanisms (for e.g. multi-query attention (Shazeer, 2019a)). Our first version, Gemini 1.0, comprises three main sizes to support a wide range of applications as discussed in Table 1.

Model size	Model description
Ultra	Our most capable model that delivers state-of-the-art performance across a wide range of highly complex tasks, including reasoning and multimodal tasks. It is efficiently serveable at scale on TPU accelerators due to the Gemini architecture.
Pro	A performance-optimized model in terms of cost as well as latency that delivers significant performance across a wide range of tasks. This model exhibits strong reasoning performance and broad multimodal capabilities.
Nano	Our most efficient model, designed to run on-device. We trained two versions of Nano, with 1.8B (Nano-1) and 3.25B (Nano-2) parameters, targeting low and high memory devices respectively. It is trained by distilling from larger Gemini models. It is 4-bit quantized for deployment and provides best-in-class performance.

Table 1 | An overview of the Gemini 1.0 model family.

Training the Gemini family of models required innovations in training algorithms, dataset, and infrastructure. For the Pro model, the inherent scalability of our infrastructure and learning algorithms enable us to complete pre-training in a matter of weeks, leveraging a fraction of the Ultra's resources. The Nano series of models leverage additional advancements in distillation and training algorithms to produce the best-in-class small language models for a wide variety of tasks, such as summarization and reading comprehension, which power our next generation on-device experiences.

Soapbox on closed models

Soapbox on closed models

- Trade secrets have their place in **protecting corporate innovations**

Soapbox on closed models

- Trade secrets have their place in **protecting corporate innovations**
- However, we **shouldn’t pretend** that this is still doing **science**
  - Science requires **replicability**
  - We can’t replicate if the methodology is secret
  - Caveat: like the space race, advancements will likely eventually **“trickle down”**

Soapbox on closed models

- Trade secrets have their place in **protecting corporate innovations**
- However, we **shouldn't pretend** that this is still doing **science**
  - Science requires **replicability**
  - We can't replicate if the methodology is secret
  - Caveat: like the space race, advancements will likely eventually **"trickle down"**
- Not all bad news: a number of **open LLMs** have been released
  - Examples: AI2's OLMo, Meta's Llama, Mistral

Note on terminology

Note on terminology

- My impression on **what NLP practitioners mean** when they say “LLM”:
  - **Large** (roughly >1B parameters)
  - **Generative** (decoder-based)
  - Trained with **LM + Instruction Tuning + RLHF**
  - Strong **in-context / zero-shot** abilities

Note on terminology

- My impression on **what NLP practitioners mean** when they say “LLM”:
  - **Large** (roughly >1B parameters)
  - **Generative** (decoder-based)
  - Trained with **LM + Instruction Tuning + RLHF**
  - Strong **in-context / zero-shot** abilities
- Historically might also refer to models like **GPT-3** or even BERT!
  - The term has evolved, and people use it differently

Other LLM topics

Other LLM topics

- Evaluation (**rapidly changing**)
  - Tons of traditional NLP benchmarks like MMLU
  - Expanding to **human exams** like the Bar Exam, GRE, etc.
  - Sometimes talk about **“win rates”** over **human preference data** (e.g. Chatbot Arena)
- Multimodality (inclusion of image, video, audio, etc.)
- Risks / societal consequences
  - Most of what we’ll hear about next time!