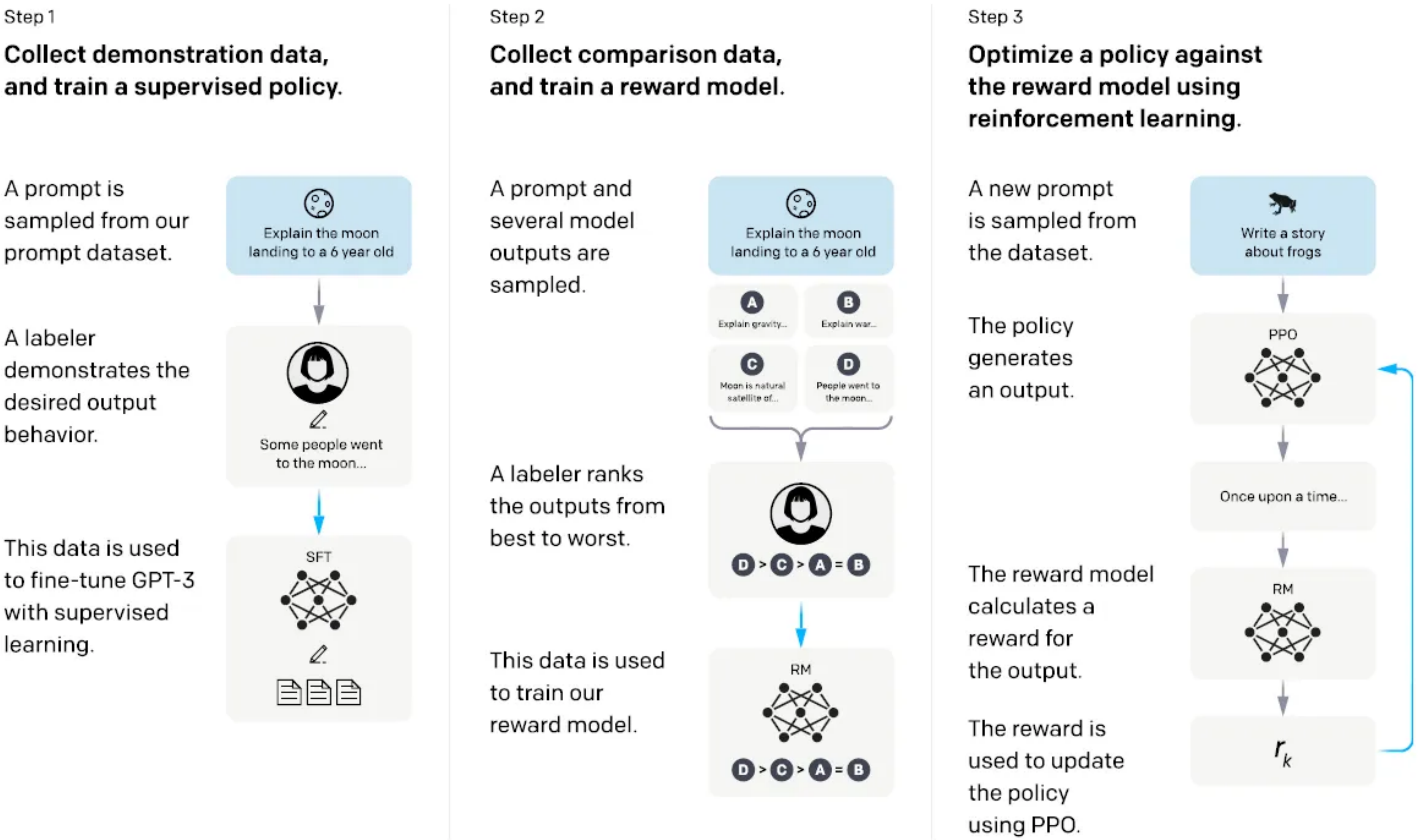
COS 484

Natural Language Processing

L13: LLMs: advanced techniques

Spring 2026

Recap: InstructGPT



Recap: InstructGPT

- **Step 1: supervised fine-tuning (SFT) or instruction tuning**

13k prompts, completions are written by human labelers

Instruction data (prompt, response): (x, y)

$$-\sum_{i=1}^{|y|} \log P(y_i \mid y_{<i}, x)$$

- **Step 2**: reward modeling (RM)

33k prompts, K (4-9) completions sampled, human labelers provide a ranking

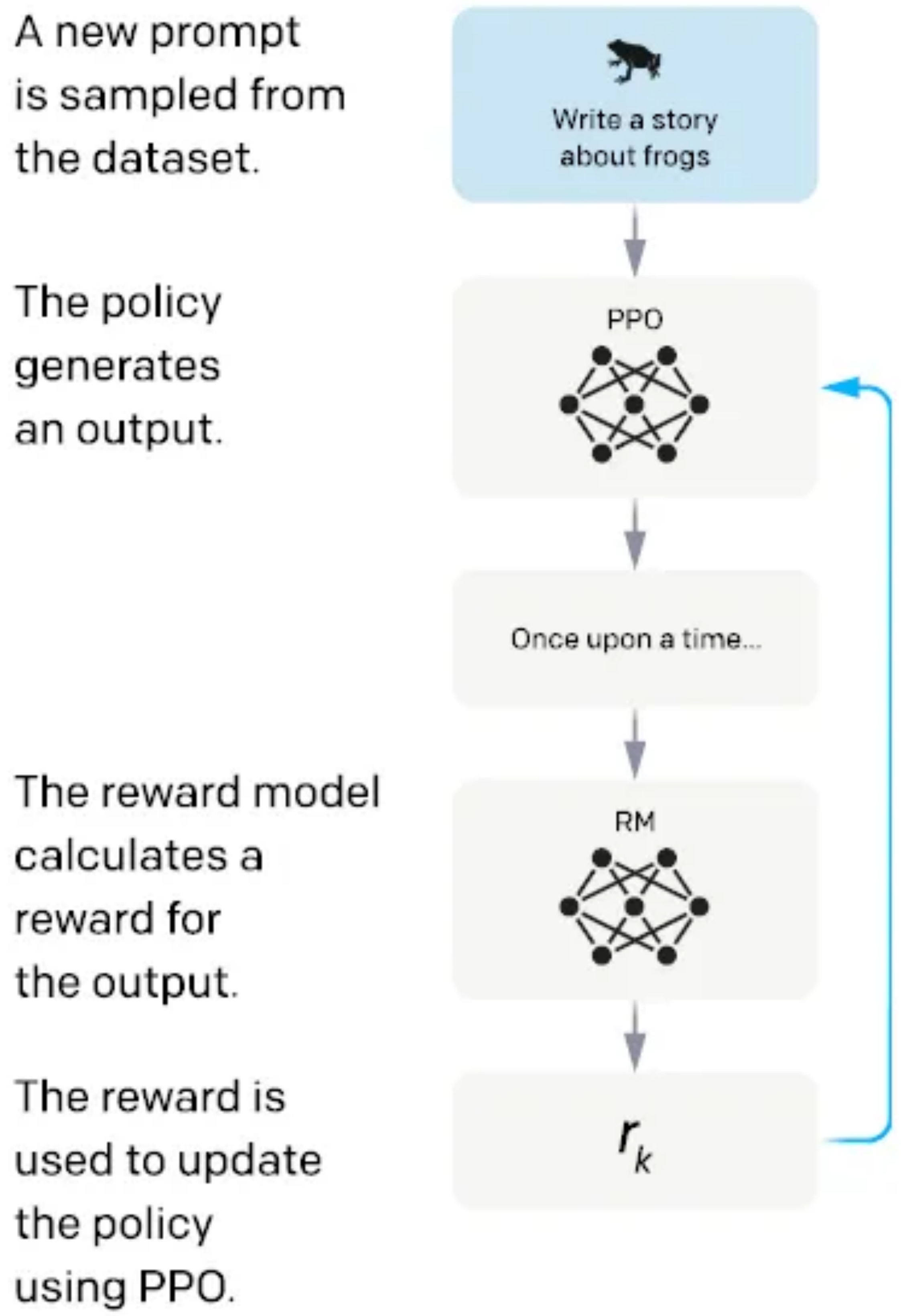
Human preference data (prompt, winning response, losing response): (x, y_w, y_l)

$$\operatorname{loss}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x, y_w, y_l) \sim D} \left[\log \left(\sigma \left(r_\theta \left(x, y_w\right) - r_\theta \left(x, y_l\right)\right)\right)\right]$$

The RM is only 6B parameters: $R : (x, y) \rightarrow \mathbb{R}$

Recap: InstructGPT

- **Step 3:** reinforcement learning (RL)
  - **Key idea:** fine-tuning supervised policy to optimize reward (output of the RM) using PPO

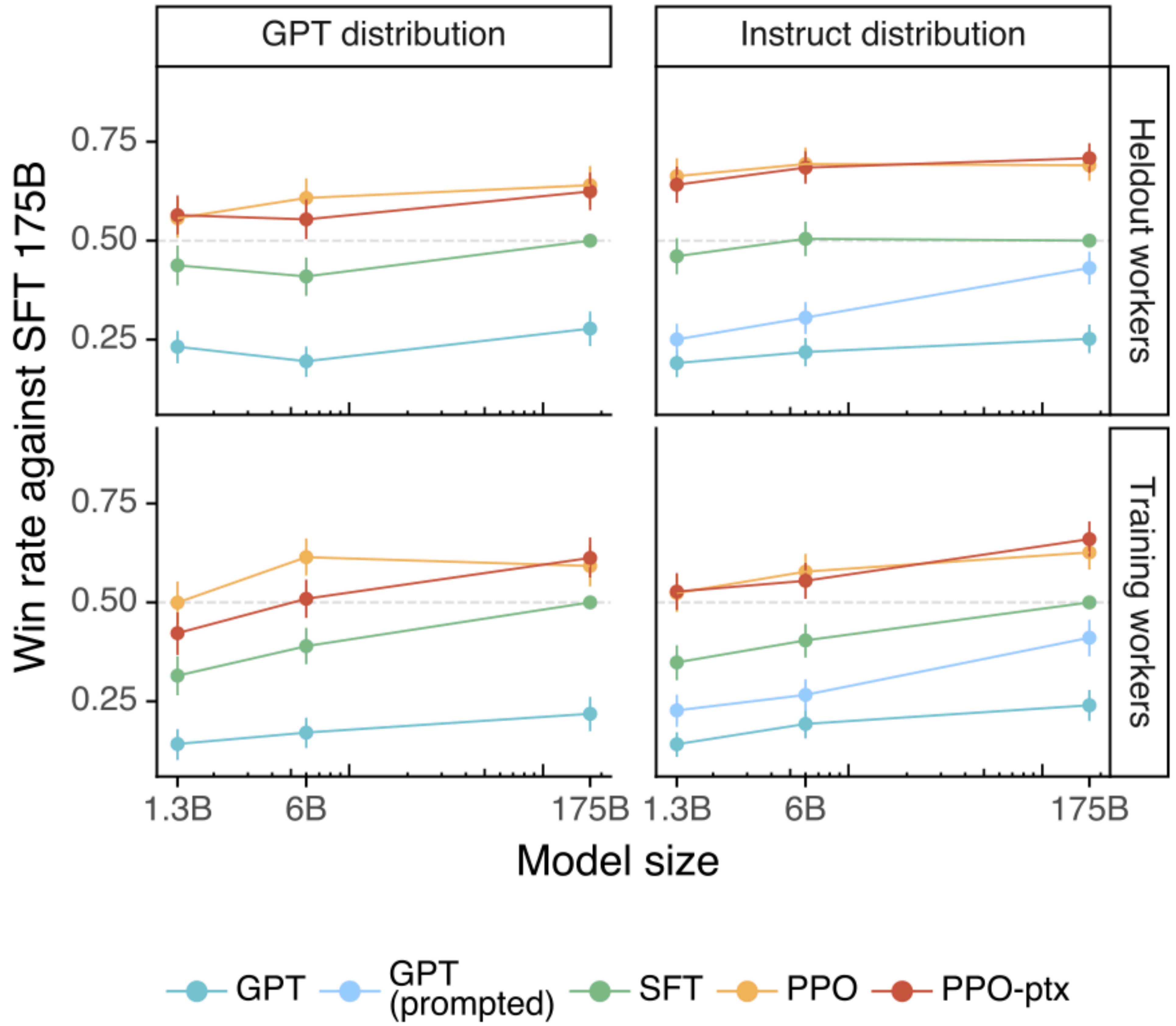


31k prompts, no human annotations involved

$$\text{objective}(\phi) = E_{(x,y)\sim D_{\pi_\phi^{\mathrm{RL}}}}\left[r_\theta(x,y)\right]$$

Reminder: the goal is to build a general-purpose chat model that is aligned with human intents! (3 Hs)

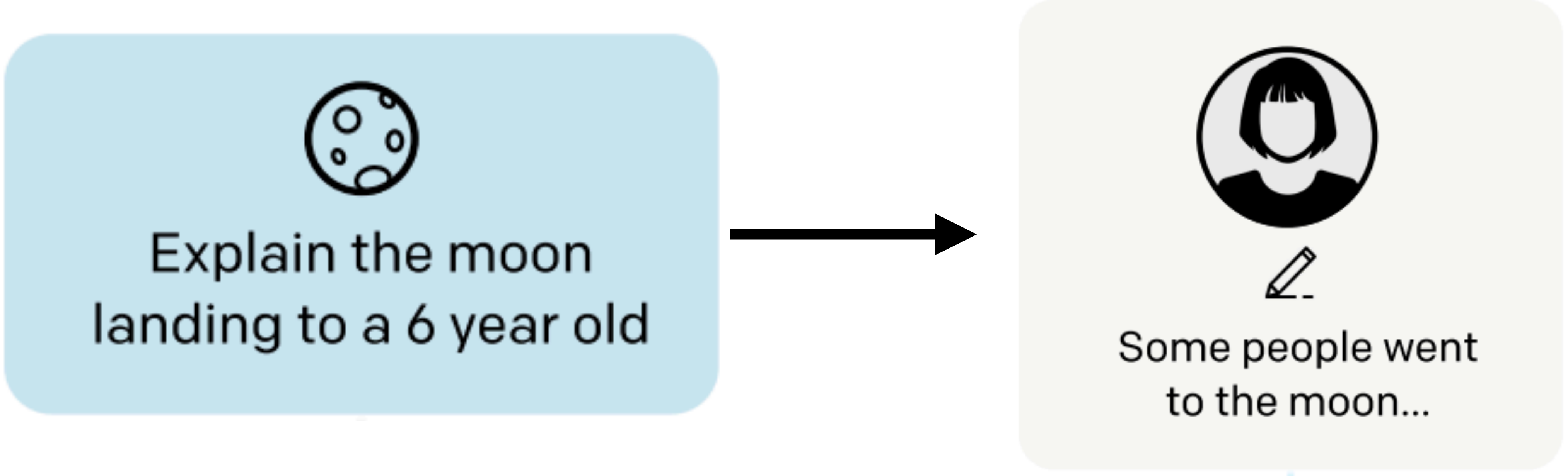
Research questions



- How important is **SFT**? How important is **RL**?
- Is **preference data** the key, or the **RL algorithm**? Is the architecture/size of RM important?
- Can we replace **human annotations** by model annotations?
- How to **evaluate** these general-purpose chat models?

Supervised fine-tuning (SFT): open research efforts

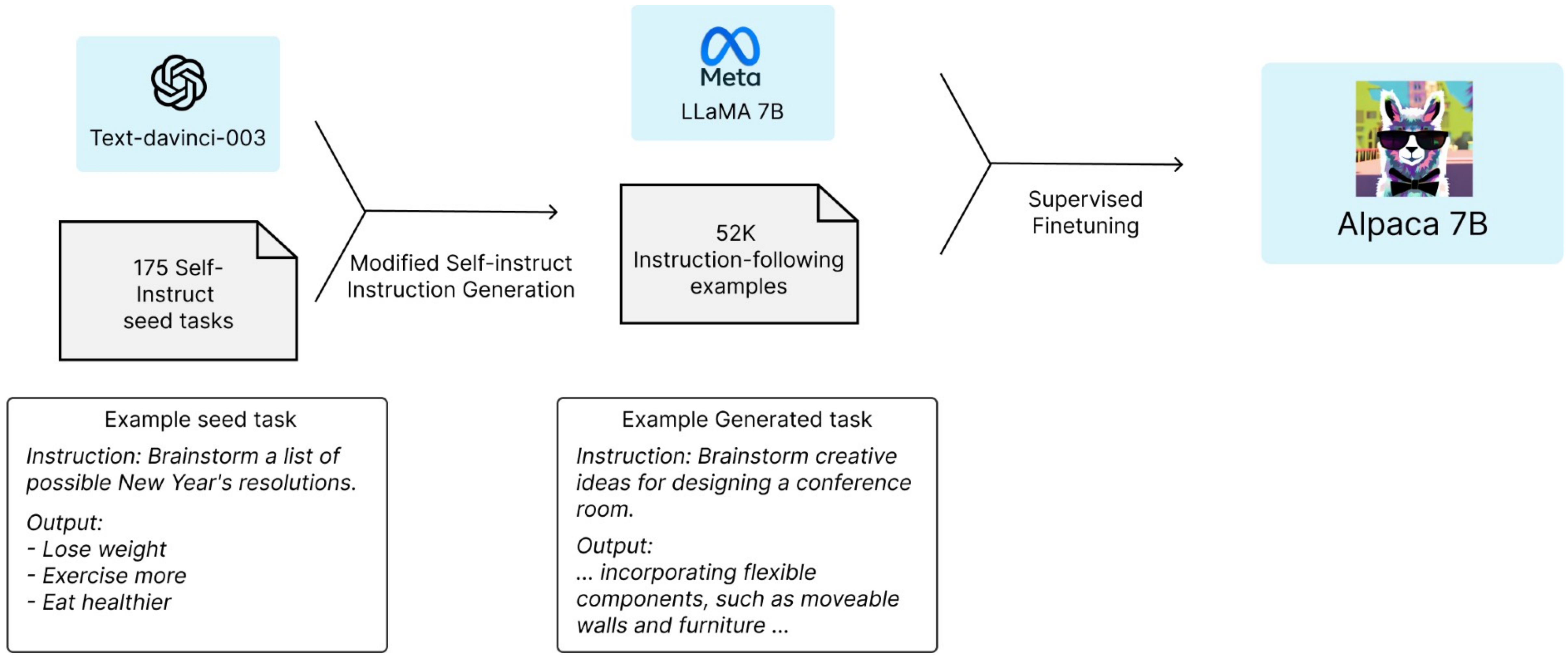
- **Data**: (**prompt**, **response**)
- **Learning**: next-token prediction



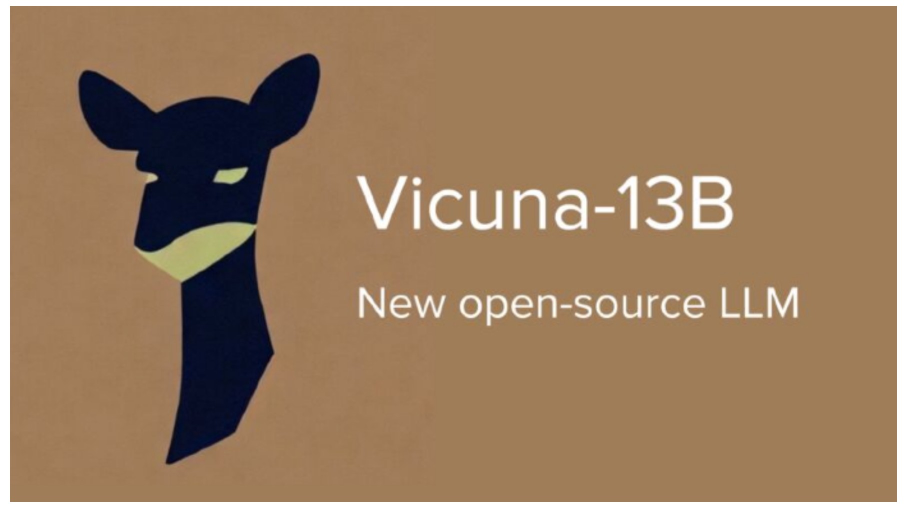
Research questions:

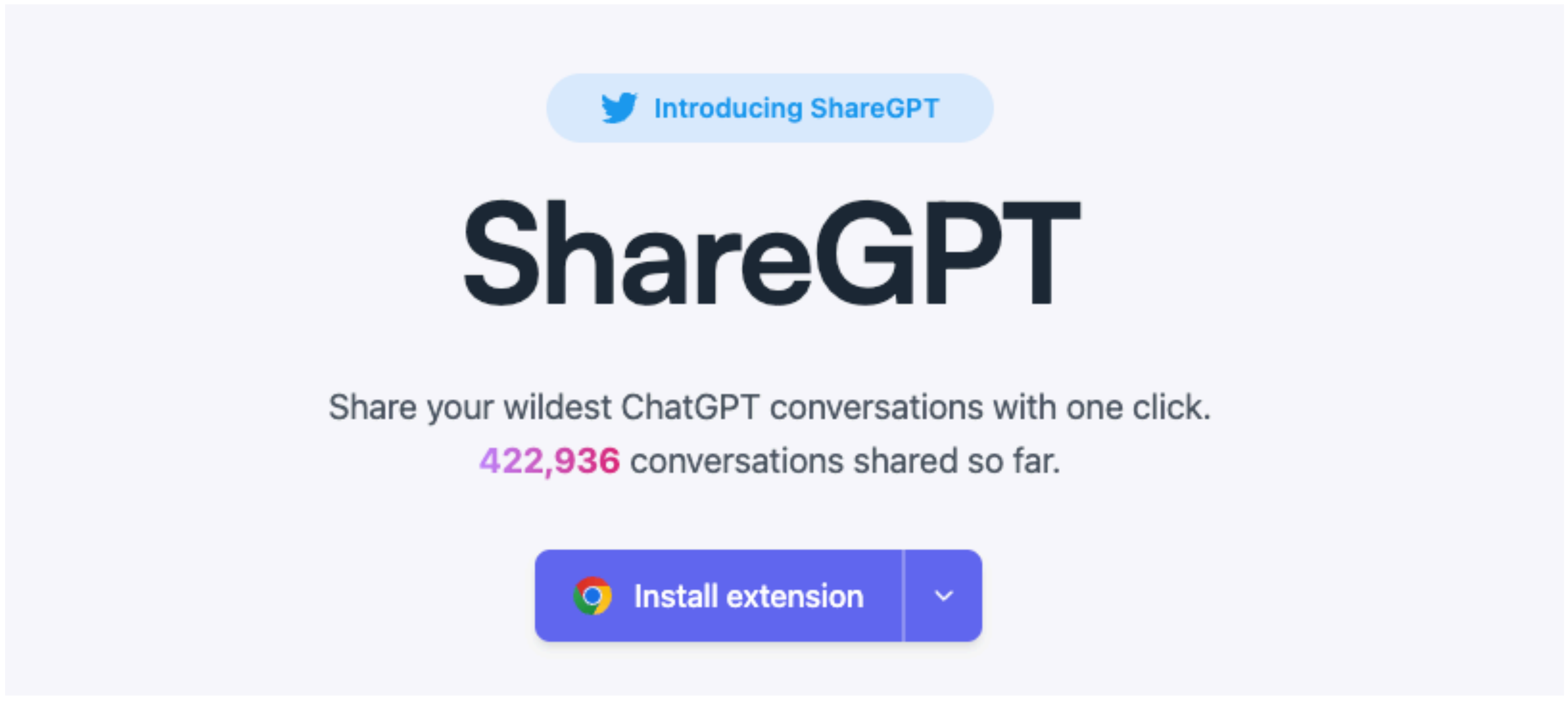
- How to collect **prompts**?
- How to collect **responses**? Do responses include chain-of-thought?
- How to ***combine*** *and* ***select*** these datasets for instruction tuning?
- How to fine-tune ***large models*** under ***computational constraints***?

Stanford Alpaca



- 52K Prompts are model-generated (Self-Instruct)
- Responses are distilled from OpenAI's text-davinci-003

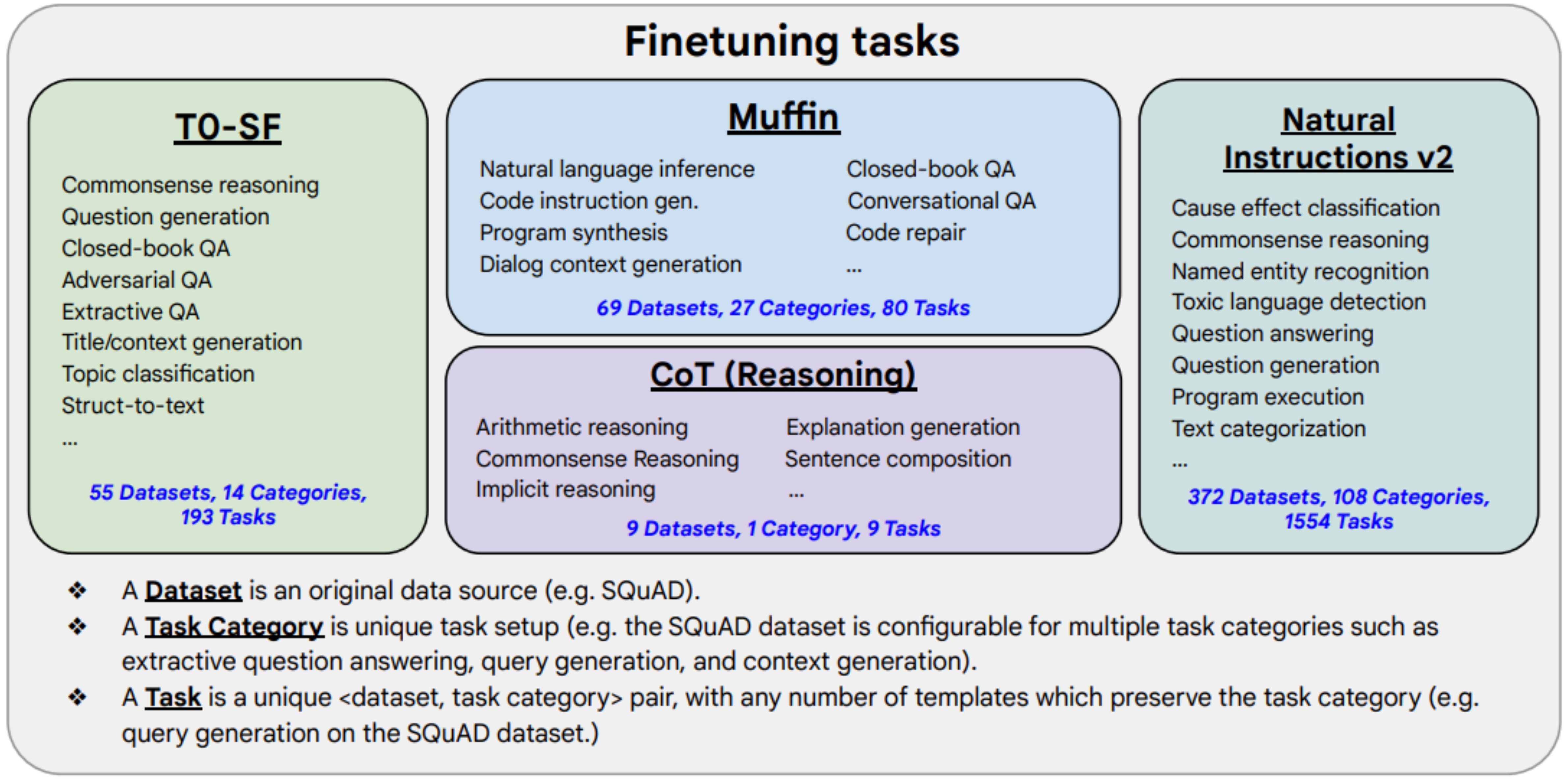


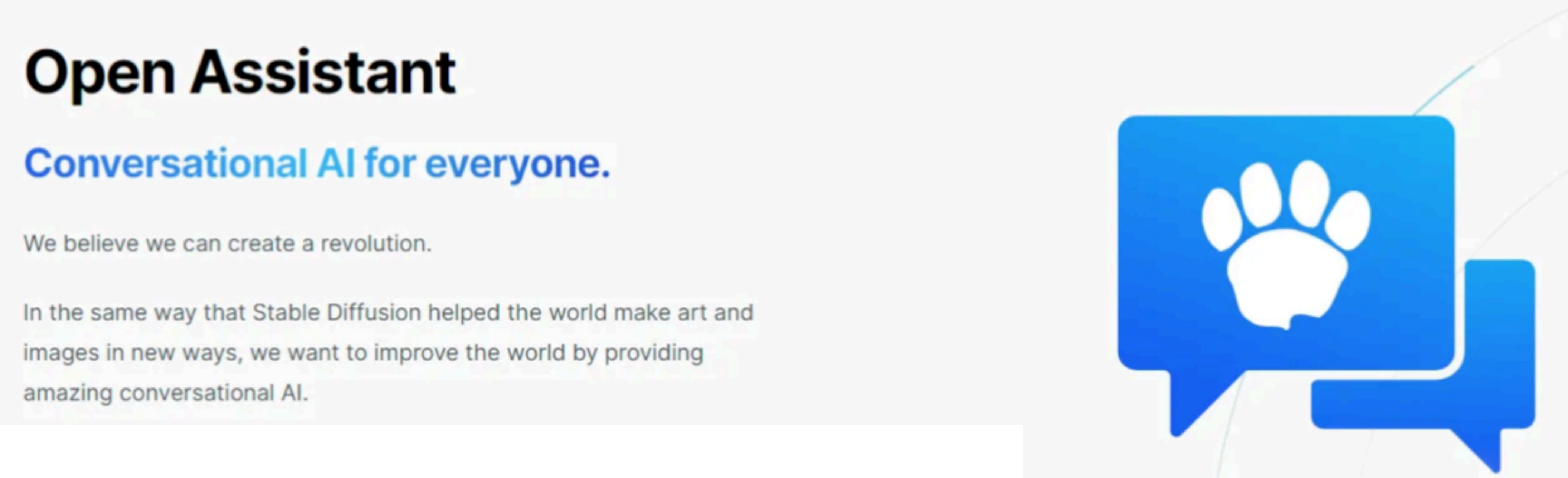


- 70K user-shared ChatGPT conversations
- Responses are from ChatGPT

Other SFT datasets

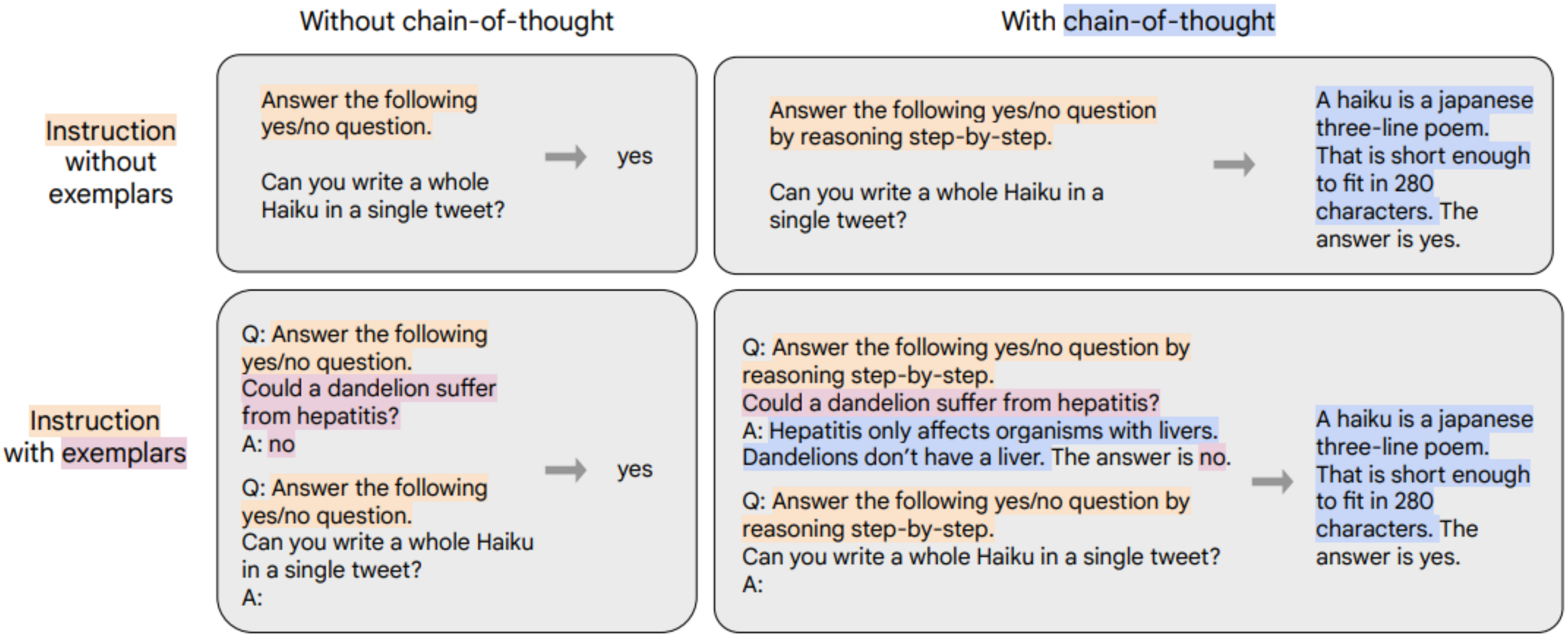
- **Repurposed from existing datasets** (w/ human-written instructions and CoT)
  - Examples: Super-NaturalInstructions, Flan V2
- **Human-written from scratch**
  - Examples: Dolly, Open Assistant





(Köpf et al., 2023)

Instruction tuning with exemplars and CoT



Chung et al., 2022, Scaling Instruction-Finetuned Language Models

LIMA: superficial alignment hypothesis

LIMA: Less Is More for Alignment

Source	#Examples	Avg Input Len.	Avg Output Len.
Training			
Stack Exchange (STEM)	200	117	523
Stack Exchange (Other)	200	119	530
wikiHow	200	12	1,811
Pushshift r/WritingPrompts	150	34	274
Natural Instructions	50	236	92
Paper Authors (Group A)	200	40	334

1000 manually-selected examples work great!

Superficial Alignment Hypothesis: Knowledge is learned during pre-training; instruction tuning teaches models which subdistribution of formats to use

Zhou et al., 2023, LIMA: Less Is More for Alignment

An explosion of SFT datasets: “How Far Can Camels Go?”

	MMLU (factuality)	GSM (reasoning)	BBH (reasoning)	TydiQA (multilinguality)	Codex-Eval (coding)	AlpacaEval (open-ended)	Average
	EM (0-shot)	EM (8-shot, CoT)	EM (3-shot, CoT)	F1 (1-shot, GP)	P@10 (0-shot)	Win % vs Davinci-003	
Vanilla LLaMa 13B	42.3	14.5	39.3	43.2	28.6	-	-
+SuperNI	49.7	4.0	4.5	**50.2**	12.9	4.2	20.9
+CoT	44.2	40.0	41.9	47.8	23.7	6.0	33.9
+Flan V2	**50.6**	20.0	40.8	47.2	16.8	3.2	29.8
+Dolly	45.6	18.0	28.4	46.5	31.0	13.7	30.5
+Open Assistant 1	43.3	15.0	39.6	33.4	31.9	58.1	36.9
+Self-instruct	30.4	11.0	30.7	41.3	12.5	5.0	21.8
+Unnatural Instructions	46.4	8.0	33.7	40.9	23.9	8.4	26.9
+Alpaca	45.0	9.5	36.6	31.1	29.9	21.9	29.0
+Code-Alpaca	42.5	13.5	35.6	38.9	34.2	15.8	30.1
+GPT4-Alpaca	46.9	16.5	38.8	23.5	**36.6**	63.1	37.6
+Baize	43.7	10.0	38.7	33.6	28.7	21.9	29.4
+ShareGPT	49.3	27.0	40.4	30.5	34.1	**70.5**	42.0
+Human data mix.	50.2	38.5	39.6	47.0	25.0	35.0	39.2
+Human+GPT data mix.	49.3	**40.5**	**43.3**	45.6	35.9	56.5	**45.2**

Wang et al., 20223, How Far Can Camels Go? Exploring the State of Instruction Tuning on Open Resources

Data mixture of instruction tuning

TÜLU v2

- **FLAN** [Chung et al., 2022]: We use 50,000 examples sampled from FLAN v2.
- **CoT**: To emphasize chain-of-thought (CoT) reasoning, we sample another 50,000 examples from the CoT subset of the FLAN v2 mixture.
- **Open Assistant 1** [Köpf et al., 2023]: We isolate the highest-scoring paths in each conversation tree and use these samples, resulting in 7,708 examples. Scores are taken from the quality labels provided by the original annotators of Open Assistant 1.
- **ShareGPT**[2]: We use all 114,046 examples from our processed ShareGPT dataset, as we found including the ShareGPT dataset resulted in strong performance in prior work.
- **GPT4-Alpaca** [Peng et al., 2023]: We sample 20,000 samples from GPT-4 Alpaca to further include distilled GPT-4 data.
- **Code-Alpaca** [Chaudhary, 2023]: We use all 20,022 examples from Code Alpaca, following our prior V1 mixture, in order to improve model coding abilities.
- ***LIMA** [Zhou et al., 2023]: We use 1,030 examples from LIMA as a source of carefully curated data.
- ***WizardLM Evol-Instruct V2** [Xu et al., 2023]: We sample 30,000 examples from WizardLM, which contains distilled data of increasing diversity and complexity.
- ***Open-Orca** [Lian et al., 2023]: We sample 30,000 examples generated by GPT-4 from OpenOrca, a reproduction of Orca [Mukherjee et al., 2023], which augments FLAN data with additional model-generated explanations.
- ***Science literature**: We include 7,544 examples from a mixture of scientific document understanding tasks— including question answering, fact-checking, summarization, and information extraction. A breakdown of tasks is given in Appendix C.
- ***Hardcoded**: We include a collection of 140 samples using prompts such as 'Tell me about yourself' manually written by the authors, such that the model generates correct outputs given inquiries about its name or developers.

Size	Data	Average -
7B	ShareGPT	47.0
	V1 mix.	47.8
	V2 mix.	**54.2**
13B	V1 mix.	56.0
	V2 mix.	**60.8**
70B	V1 mix.	71.5
	V2 mix.	**72.4**

Ivison et al., 2023, Camels in a Changing Climate: Enhancing LM Adaptation with Tulu 2

LoRA: Low-Rank Adaptation

Problem: full fine-tuning updates **all** parameters — expensive for large models (e.g., 70B parameters)

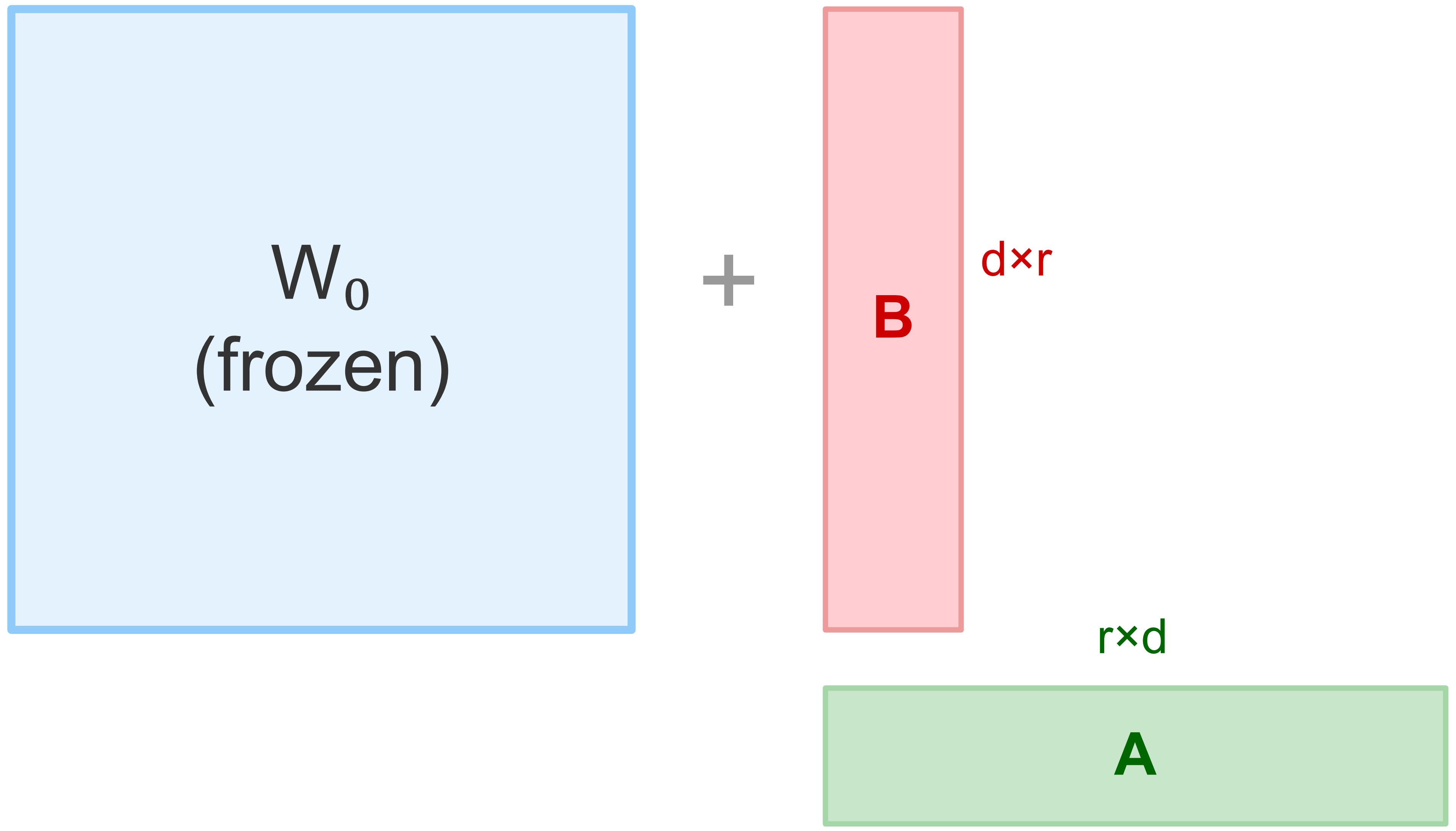
Key idea: freeze pretrained weights, add small trainable low-rank matrices

Weight update:

Full FT: $W = W_0 + \Delta W$

LoRA: $W = W_0 + BA, \quad B \in \mathbb{R}^{d\times r}, A \in \mathbb{R}^{r\times d}, r \ll d$

where B ∈ ℝ^(d×r), A ∈ ℝ^(r×d), r << d



In practice:

- Applied to attention weight matrices (W_Q, W_K, W_V, W_O) in each transformer layer
- Typical rank r = 8, 16, or 64 (vs. d = 4096 or more)
- **At inference: merge W_0 + BA into a single matrix — no additional latency**

Hu et al., 2021, LoRA: Low-Rank Adaptation of Large Language Models

LoRA: parameter efficiency

How many parameters are we training?

Full fine-tuning (LLaMA-7B): **~7 billion parameters**

LoRA (r=16, all attention layers): **~20 million parameters (0.3% of total!)**

Per layer: $2 \times d \times r = 2 \times 4096 \times 16 = 131K$ params (vs. $d^2 = 16.8M$ for full weight)

Memory savings

- Optimizer states (Adam) only for LoRA params — massive memory reduction
- Frozen base model can be shared across tasks — train separate LoRA adapters
- **Enables fine-tuning 65B models on a single GPU with quantization (QLoRA)**

Performance

- **Matches or approaches full fine-tuning quality on most tasks**
- Hypothesis: weight updates during fine-tuning have low intrinsic rank
- Higher rank → closer to full fine-tuning, but diminishing returns past $r \approx 16$–64

Insight: fine-tuning doesn't need to change all parameters — the task-specific "delta" lives in a low-dimensional subspace.

Hu et al., 2021, LoRA: Low-Rank Adaptation of Large Language Models

QLoRA and the open-source ecosystem

QLoRA (Dettmers et al., 2023)

- Quantize base model to 4-bit (NF4) → store frozen weights in ~2× less memory
- Train LoRA adapters in 16-bit on top of quantized base
- **Result: fine-tune a 65B model on a single 48GB GPU**
- Matches full 16-bit fine-tuning quality; no performance degradation

LoRA in the post-training pipeline:

SFT

LoRA / QLoRA is the default for instruction tuning open models

DPO / SimPO

LoRA used for preference optimization (e.g., SimPO + Gemma)

Data selection (LESS)

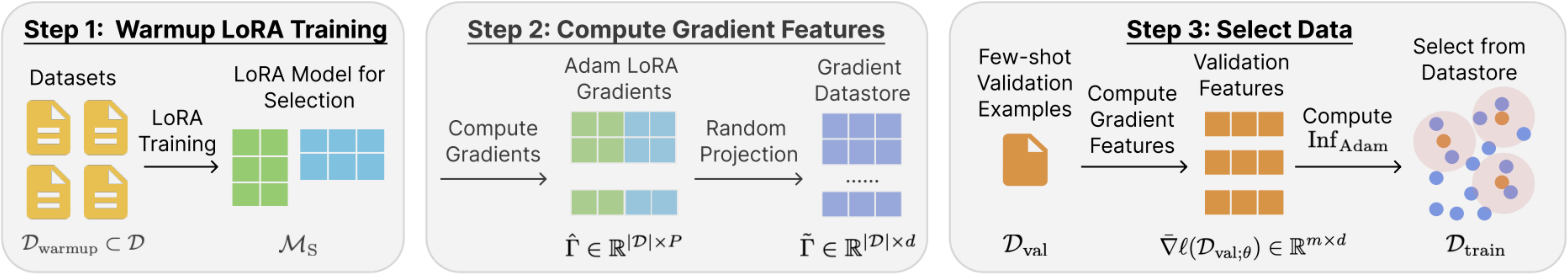
Uses LoRA warmup training to compute gradient features

Impact: LoRA democratized LLM fine-tuning. Alpaca, Vicuna, Tulu — the entire open-source post-training ecosystem depends on parameter-efficient methods to make fine-tuning affordable.

Dettmers et al., 2023, QLoRA: Efficient Finetuning of Quantized Language Models

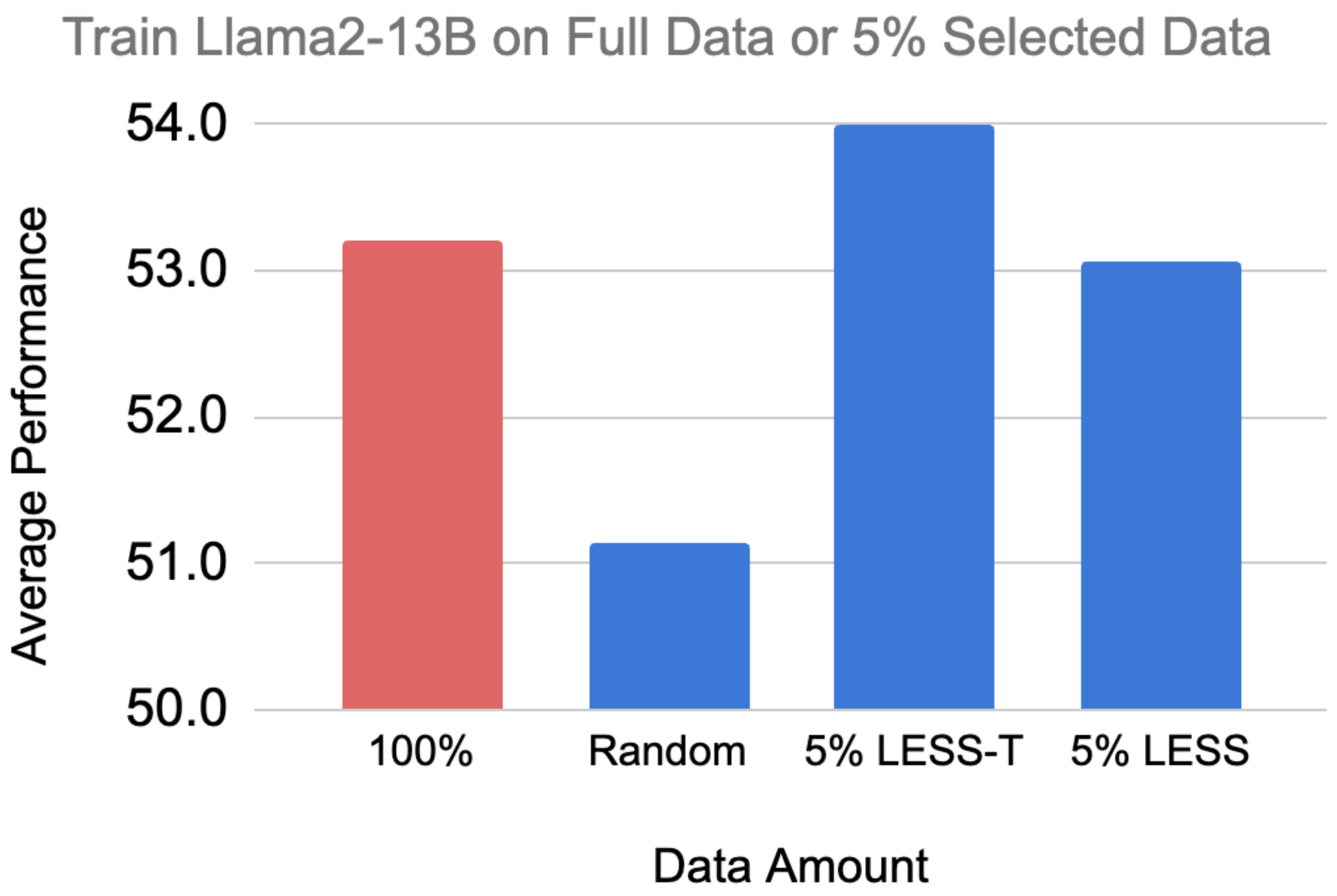
How to select instruction tuning examples?

- Key idea: use **influence formulation** to estimate how training examples influence models' predictions on target tasks and use it as proxy for data selection



Xia et al., 2024, LESS: Selecting Influential Data for Targeted Instruction Tuning

How to select instruction tuning examples?



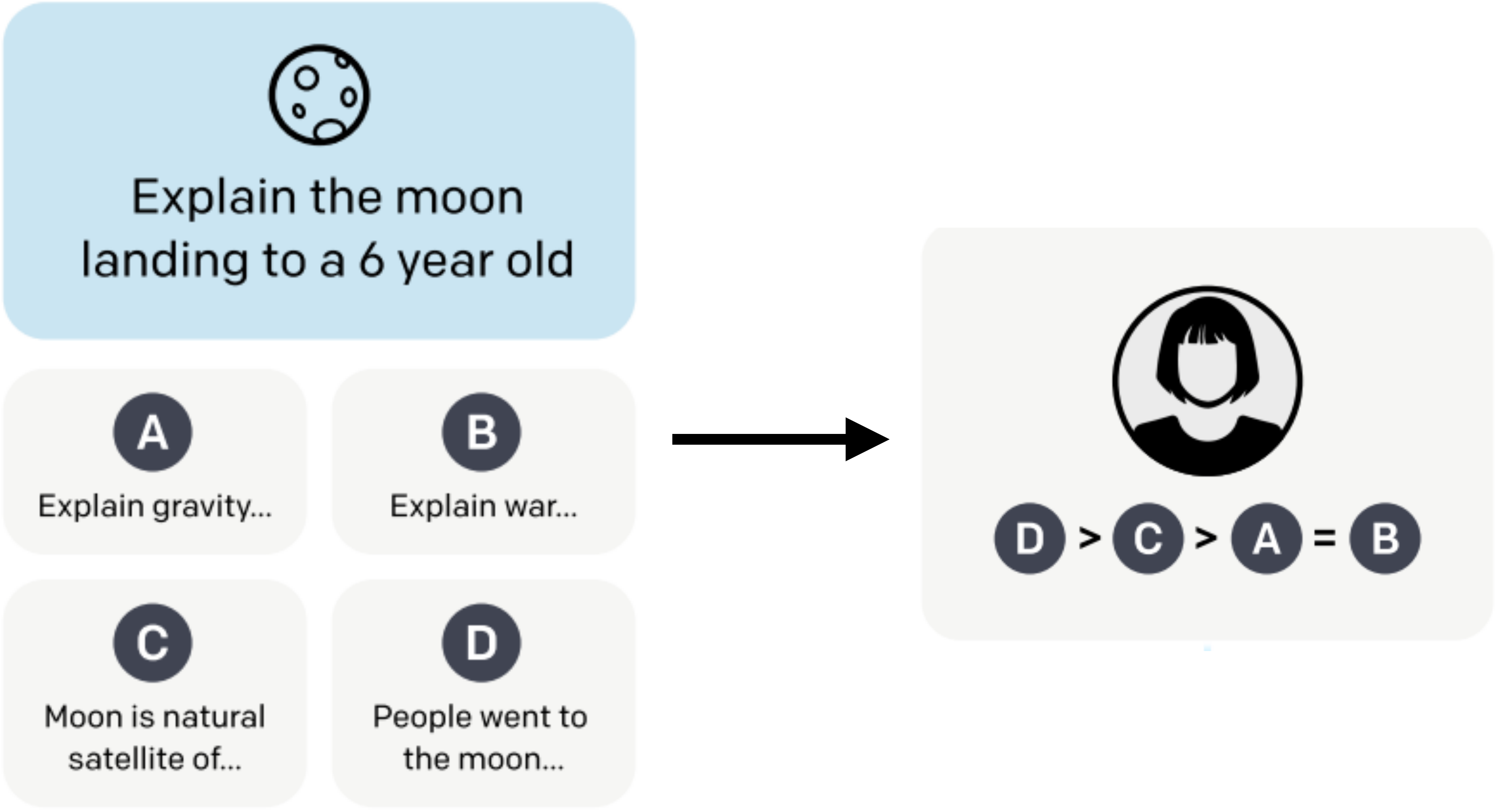
Less-T: “transfer” setting

Instruction tuning examples selected based on LLama-2-7B can be used to instruct fine-tune Mistral-7B and LLama-2-13B!

Xia et al., 2024, LESS: Selecting Influential Data for Targeted Instruction Tuning

Learning from preferences: open research efforts

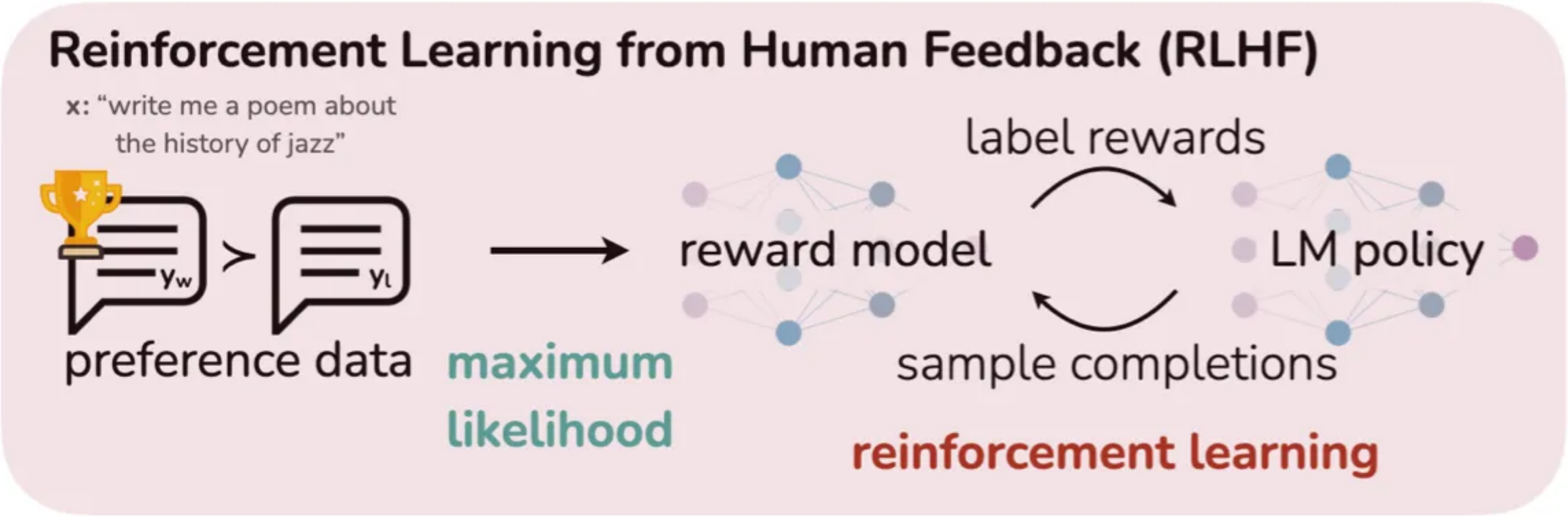
- **Data**: (**prompt**, **winning response**, **losing response**)
- **Learning:** RL (PPO) vs offline PO (DPO)



- How to get **prompts**?
- How to get **winning responses** and **losing responses**?
- How to train the reward model?
- Is RL really necessary?

Direct preference optimization (DPO)

Preference data: (**prompt**, **winning response**, **losing response**) $(x, y_w, y_l) \sim D$



1. Optimize **reward model** over **preference data**
2. Optimize **policy model** according to the **reward model**

Next: Why not directly learn the **policy model** from **preference data**?

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

DPO derivation: closed-form optimal policy

Starting point: KL-constrained RL objective from RLHF (InstructGPT)

$$\max_{\pi} \mathbb{E}_{x,y\sim\pi}\left[r(x,y)\right] - \beta\, D_{\mathrm{KL}}\left[\pi(y\,|\,x)\|\pi_{\mathrm{ref}}(y\,|\,x)\right]$$

Key insight: this optimization has a closed-form solution!

Optimal policy:

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

Rearranging to express **reward in terms of policy**:

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

This is the key equation: the reward is implicitly defined by the optimal policy and reference policy.

DPO derivation: from reward to loss

Step: substitute the implicit reward into the Bradley-Terry preference model

Bradley-Terry model:

$$P(y_w \succ y_l \mid x) = \sigma\big(r(x, y_w) - r(x, y_l)\big)$$

Substituting `r(x,y) = β log(π*(y|x)/π_ref(y|x)) + β log Z(x)` :

The β log Z(x) terms cancel! (They don't depend on y)

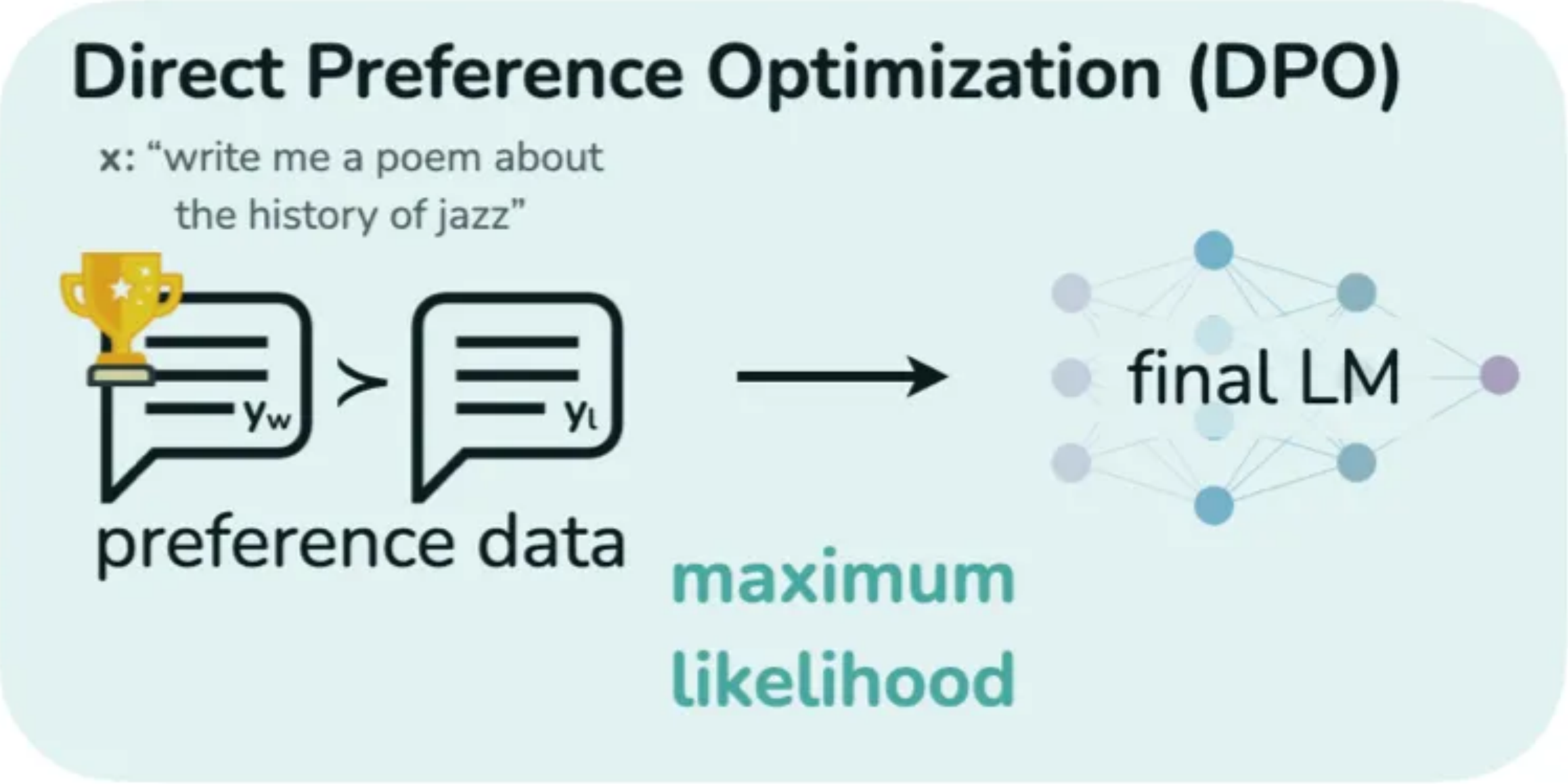
DPO objective:

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

No reward model needed!
No RL needed! Just maximum likelihood on preference data.
Only requires: preference pairs (x, y_w, y_l) + a reference policy π_ref (the SFT model)

Direct preference optimization (DPO)

Preference data: (prompt, winning response, losing response) $(x, y_w, y_l) \sim D$



DPO objective:

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

π_{ref}: SFT model

(Reminder: we don't want the PPO model to drift away much from SFT in RLHF too)

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

Reinforcement learning from AI Feedback (RLAIF)

RLAIF: first introduced by Bai et al. 2022 “Constitutional AI”

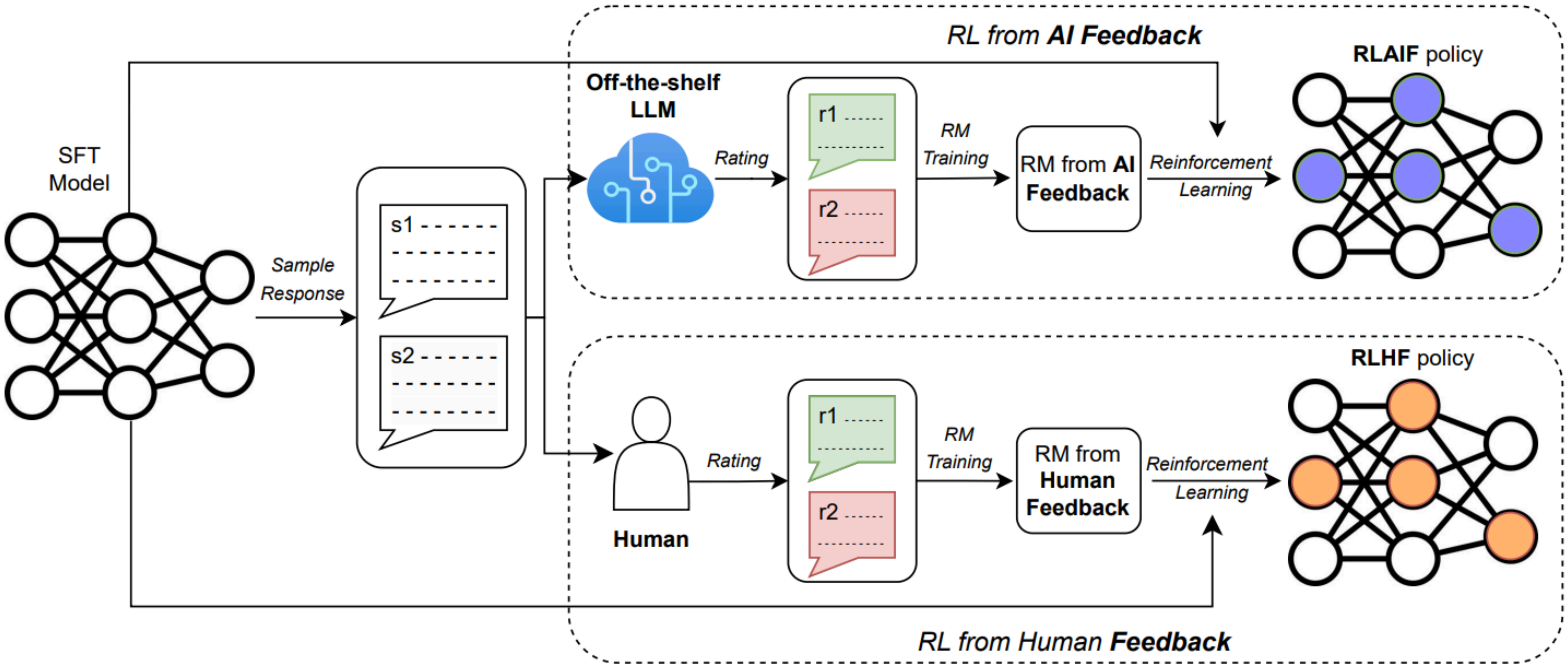


Figure: (Lee et al., 2024)

RLAIF: can AI feedback match human feedback?

Key question: If we replace human labelers with an LLM for preference annotation, how much quality do we lose?

Lee et al., 2024 — key findings:

- **RLAIF achieves comparable or better performance than RLHF**
- On summarization: human evaluators prefer RLAIF 50% of the time vs RLHF
- Both RLAIF and RLHF significantly outperform the SFT baseline
- AI labeling is orders of magnitude cheaper than human labeling

Implications for the post-training pipeline:

SFT data
Human → Model-generated
(Alpaca, distillation)

→

Preference labels
Human → AI feedback
(RLAIF, Constitutional AI)

→

RL algorithm
PPO → DPO / SimPO
(no RL loop needed)

Lee et al., 2024, RLAIF: Scaling Reinforcement Learning from Human Feedback with AI Feedback

Overall pipeline (Llama 3)

Use **iterative** rounds of post-training (not just one pass!)

1. SFT

- Human annotations + synthetic data
- Rejection sampling from previous round's best model
- ~10M examples: dialogue, coding, math, reasoning, tool use

→

2. Reward Model

- Trained on human preference data
- Scores & filters SFT data (rejection sampling)
- Generates preference pairs for DPO

→

3. DPO

- Applied after SFT in each round
- Preference pairs from RM-scored model outputs
- Replaces PPO — simpler, more stable

→

4. Iterate

- Run multiple rounds of the full pipeline
- Each round's model generates better data for the next
- Quality improves with each iteration

Key takeaway: The entire SFT + RM + DPO pipeline is run iteratively. Better models produce better training data.

Dubey et al., 2024, The Llama 3 Herd of Models

Wide use of DPO in open models

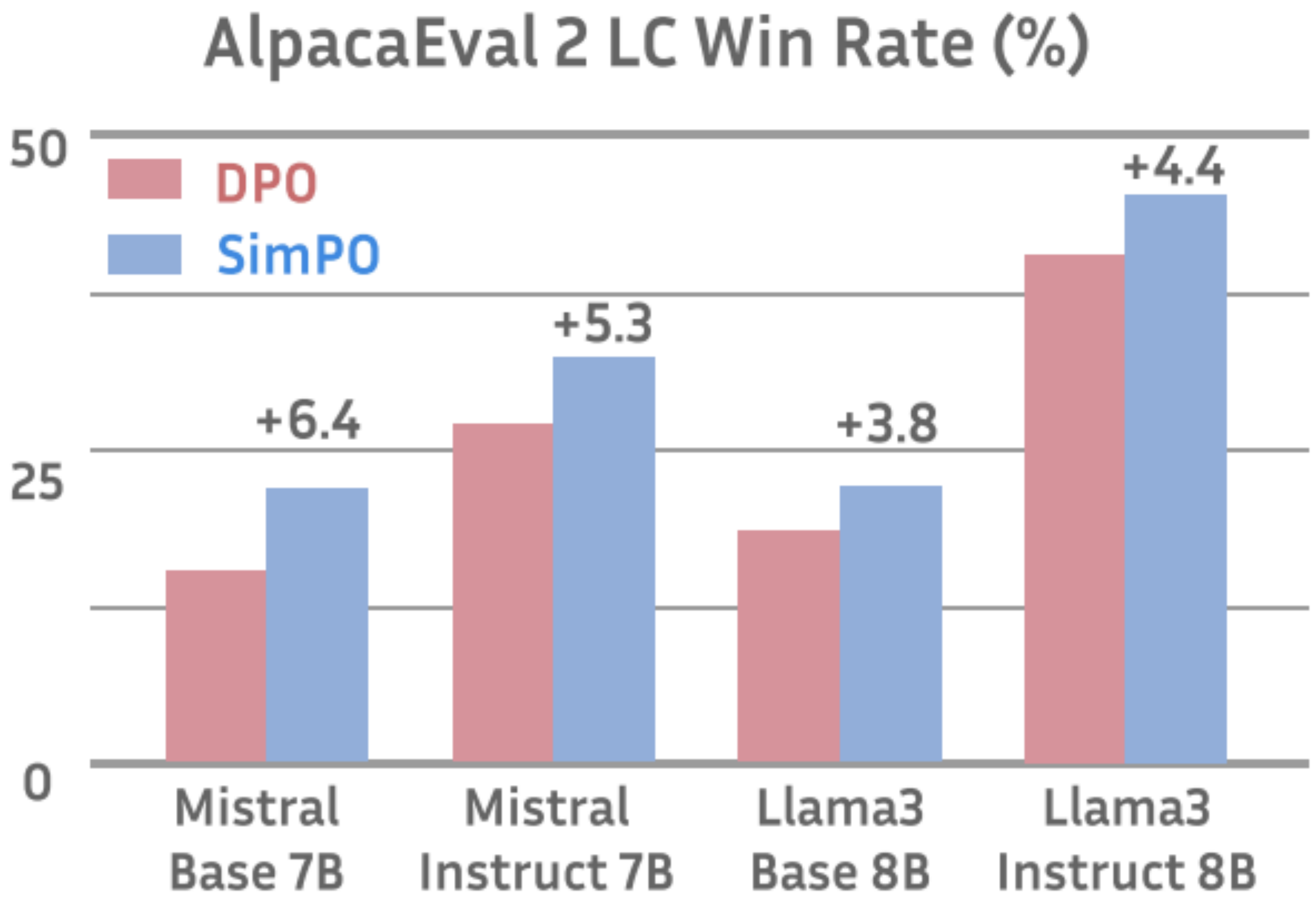
T	Model		Average	ARC	HellaSwag	MMLU	TruthfulQA	Winogrande	GSM8K
■	udkai/Turdus	DPO	74.66	73.38	88.56	64.52	67.11	86.66	67.7
■	fblgit/UNA-TheBeagle-7b-v1	DPO (& UNA)	73.87	73.04	88	63.48	69.85	82.16	66.72
■	argilla/distilabeled-Marcoro14-7B-slerp	DPO	73.63	70.73	87.47	65.22	65.1	82.08	71.19
■	mlabonne/NeuralMarcoro14-7B	DPO	73.57	71.42	87.59	64.84	65.64	81.22	70.74
◆	abideen/NexoNimbus-7B	Merge (of DPO models)	73.5	70.82	87.86	64.69	62.43	84.85	70.36
■	Neuronovo/neuronovo-7B-v0.2	DPO	73.44	73.04	88.32	65.15	71.02	80.66	62.47
■	argilla/distilabeled-Marcoro14-7B-slerp-full	DPO	73.4	70.65	87.55	65.33	64.21	82	70.66
■	CultriX/MistralTrix-v1	DPO	73.39	72.27	88.33	65.24	70.73	80.98	62.77
■	ryandt/MusingCaterpillar	DPO	73.33	72.53	88.34	65.26	70.93	80.66	62.24
■	Neuronovo/neuronovo-7B-v0.3	DPO	73.29	72.7	88.26	65.1	71.35	80.9	61.41
■	CultriX/MistralTrixTest	No info but prob DPO, given	73.17	72.53	88.4	65.22	70.77	81.37	60.73
◆	samir-fama/SamirGPT-v1	Merge (incl. DPO)	73.11	69.54	87.04	65.3	63.37	81.69	71.72
◆	SanjiWatsuki/Lelantos-DPO-7B	DPO	73.09	71.08	87.22	64	67.77	80.03	68.46

Llama 3 also uses DPO instead of RL (iterative training of SFT, RM and DPO)

SimPO: Simple preference optimization with a reference-free reward

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

$$\mathcal{L}_{\mathrm{SimPO}}(\pi_\theta) = -\mathbb{E}\left[\log \sigma\left(\frac{\beta}{|y_w|} \log \pi_\theta(y_w \mid x) - \frac{\beta}{|y_l|} \log \pi_\theta(y_l \mid x) - \gamma\right)\right]$$



Maybe you don't need reference model either?

Meng et al., 2024, SimPO: Simple Preference Optimization with a Reference-Free Reward

SimPO: why does it work?

Change 1: Length normalization

DPO uses **total** log-probability as implicit reward

Problem: penalizes longer, more detailed responses
→ model learns to be terse

SimPO uses **average** log-prob (÷ |y|) → length-neutral reward signal

Change 2: No reference model

DPO computes log(π_θ / π_ref) - ratio to reference (SFT) model

SimPO drops π_ref entirely -- uses raw model log-probs

The γ margin term compensates: ensures a minimum gap between winning and losing scores

Benefit: no need to keep reference model in memory (saves ~50% GPU)

Comparison:

$$r_{\mathrm{DPO}}(y) = \beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\mathrm{ref}}(y \mid x)}$$

$$r_{\mathrm{SimPO}}(y) = \frac{\beta}{|y|} \log \pi_\theta(y \mid x) - \gamma$$

Open models for the win

Rank* (UB)	Rank (StyleCtrl)	Model
35	30	Gemma-2-27b-it
35	31	Gemma-2-9b-it-SimPO
35	33	Deepseek-Coder-v2-0724
35	33	Command R+ (08-2024)
35	35	Yi-Large
35	48	Gemini-1.5-Flash-8B-001

Cont.....

Rank* (UB)	Rank (StyleCtrl)	Model
50	46	Command R+ (04-2024)
50	46	Qwen2-72B-Instruct
50	49	Gemma-2-9b-it

- We start from **Gemma-2-9b-it** model
  - Closed pre-training and closed RLHF
- We take **50k prompts** x from **UltraFeedback** (Cui et al., 2023) and regenerate 5 responses
- We use a reward model **ArmoRM** (Wang et al., 2024) to pick the **best** and **worst** response as **winning response** y_w, **losing response** y_l
- We train SimPO on this **on-policy** data, and obtained:

princeton-nlp/**gemma-2-9b-it-SimPO**

< 3 hours on 8 H100 GPUs!

Model: princeton-nlp/**gemma-2-9b-it-SimPO**

The strongest <10B model on Chatbot Arena, WildBench, Arena Hard, Alpaca Eval 2

Meng et al., 2024, SimPO: Simple Preference Optimization with a Reference-Free Reward

RewardBench: evaluating reward models

RewardBench: Evaluating Reward Models

Evaluating the capabilities, safety, and pitfalls of reward models

Code | Eval. Dataset | Prior Test Sets | Results | Paper | Total models: 165 | * Unverified models | ⚠️ Dataset Contamination | Last restart (PST): 22:01 PDT, 28 Mar 2025

⚠️ Many of the top models were trained on unintentionally contaminated, AI-generated data, for more information, see this gist.

🏆 RewardBench Leaderboard | 🔍 RewardBench - Detailed | Prior Test Sets | About | Dataset Viewer

Model Search (delimit with ,)

☑ Seq. Classifiers ☑ DPO ☑ Custom Classifiers ☑ Generative ☐ Prior Sets

	Model	Model Type	Score	Chat	Chat Hard	Safety	Reasoning
1	infly/INF-ORM-Llama3.1-70B	Seq. Classifier	95.1	96.6	91.0	93.6	99.1
2	ShikaiChen/LDL-Reward-Gemma-2-27B-v0.1	Seq. Classifier	95.0	96.4	90.8	93.8	99.0
3	nicolinho/QRM-Gemma-2-27B	Seq. Classifier	94.4	96.6	90.1	92.7	98.3
4	Skywork/Skywork-Reward-Gemma-2-27B-v0.2	Seq. Classifier	94.3	96.1	89.9	93.0	98.1
5	nvidia/Llama-3.1-Nemotron-70B-Reward *	Custom Classifier	94.1	97.5	85.7	95.1	98.1
6	Skywork/Skywork-Reward-Gemma-2-27B ⚠️	Seq. Classifier	93.8	95.8	91.4	91.9	96.1
7	SF-Foundation/TextEval-Llama3.1-70B * ⚠️	Generative	93.5	94.1	90.1	93.2	96.4
8	meta-metrics/MetaMetrics-RM-v1.0	Custom Classifier	93.4	98.3	86.4	90.8	98.2

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

Advanced: RL for reasoning

Today's lecture: post-training aligns models with human preferences (HHH)
Current research efforts: using RL to improve **reasoning capabilities**

OpenAI o1 (2024)

- Train model to produce chain-of-thought reasoning via RL
- Reward signal: correctness on verifiable tasks (math, code)
- Test-time compute scaling: more thinking → better answers

- **Shift from "train bigger" to "think longer"**

DeepSeek-R1 (2025)

- Pure RL (GRPO) — no SFT needed to get reasoning!
- "Aha moment": model spontaneously learns to re-examine its work
- Challenges the superficial alignment hypothesis: RL teaches genuinely new capabilities

- **Open-weight, competitive with o1**

Key difference: RLHF aligns preferences while RL for reasoning teaches new skills (diff reward signals)