Using (Human) Feedback for Training Large Language Models

OR how ChatGPT is likely trained

Aman @ Yiming Yang's lab seminar, 2/28/2023

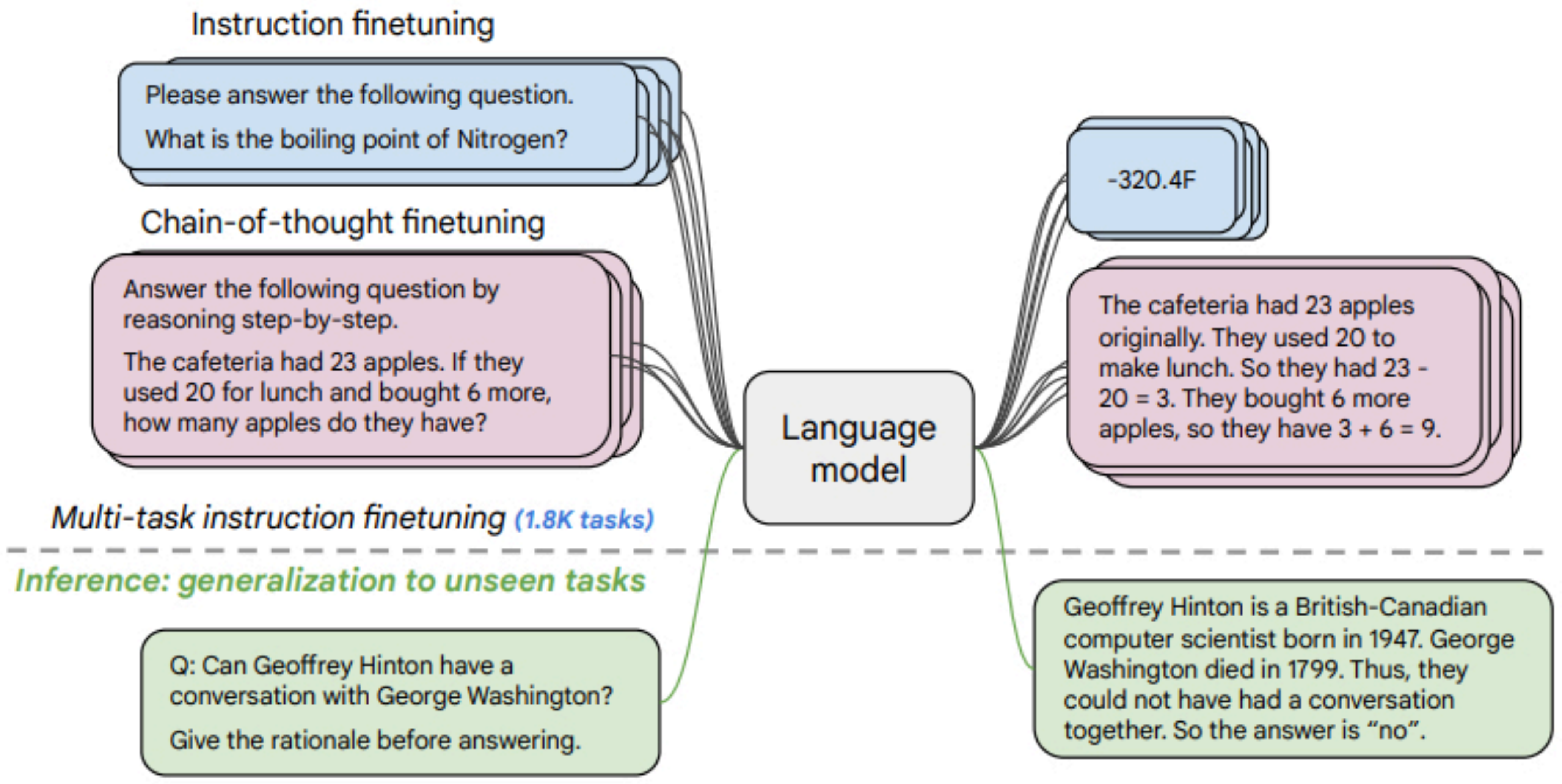
Precursor: Instruction Tuning

- GPT-3 shows that language models trained on a large amount of data can generate fluent text ~ mid 2020
- Good language models — users want to go beyond benchmarks
- What next?
  - Want to train language models that can **follow instructions**
  - Prevent them from generating responses that are toxic and unhelpful
- Want the language models to align with what humans want

Training language models to follow instructions

- Want the language models to align with what humans want
    - Instruction tuning was an early attempt at this

- FLAN
- T0
- Lambda



Scaling Instruction-Finetuned Language Models

Hyung Won Chung* Le Hou* Shayne Longpre* Barret Zoph† Yi Tay†
William Fedus† Yunxuan Li Xuezhi Wang Mostafa Dehghani Siddhartha Brahma
Albert Webson Shixiang Shane Gu Zhuyun Dai Mirac Suzgun Xinyun Chen
Aakanksha Chowdhery Alex Castro-Ros Marie Pellat Kevin Robinson
Dasha Valter Sharan Narang Gaurav Mishra Adams Yu Vincent Zhao
Yanping Huang Andrew Dai Hongkun Yu Slav Petrov Ed H. Chi
Jeff Dean Jacob Devlin Adam Roberts Denny Zhou Quoc V. Le
Jason Wei*

Google

Why Instruction Tuning isn't Enough?

- The models might become better at task understanding — but still nontrivial to generate a desirable sequence

- Alignment goes beyond instruction following

- Real-world behavior is quite different from benchmark datasets

Why human feedback?

- Hard to quantify the requirements or the definition of “good”

- Task:
    - Complete the sentence “I saw the movie last night” to make a positive review
- Completion 1:
    - I saw the movie last night and found it to be a thoroughly enjoyable experience.
- Completion 2:
    - I saw the movie last night and it was soooo good! Like, really, really good!
- Which response will humans prefer?
    - Subjective, but maybe given the goals of the system (general purpose chatbot) + sizable annotator pool

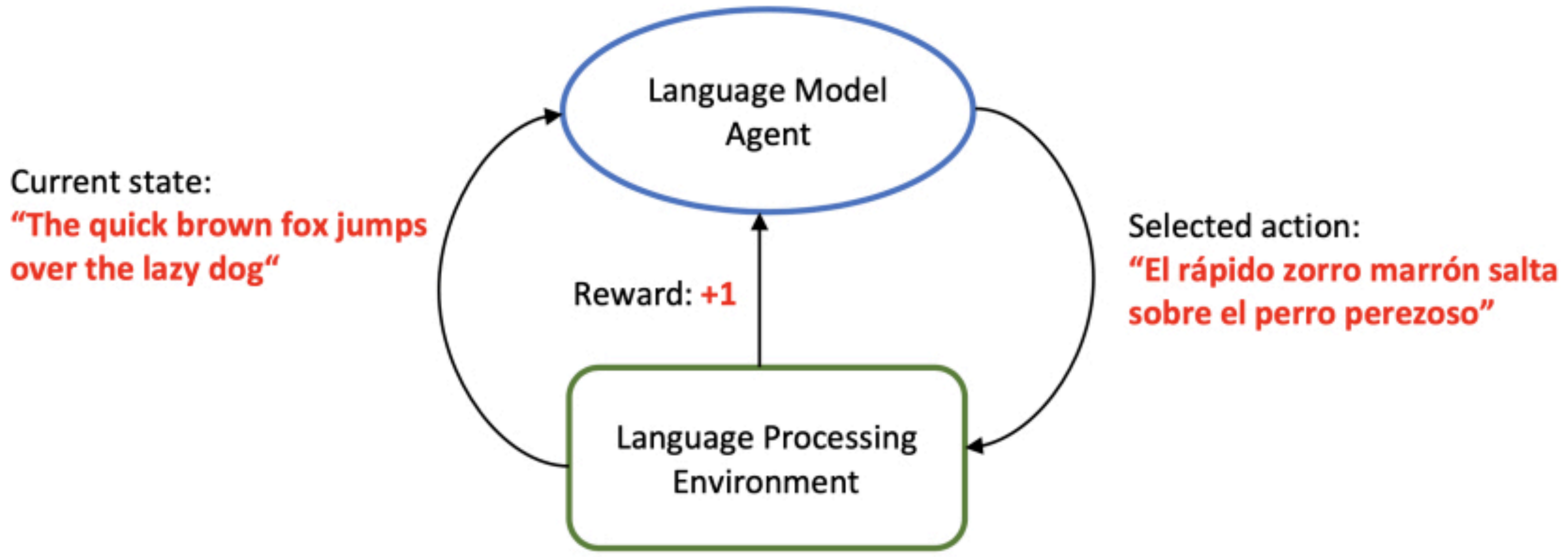
Two Camps

- RL
  - Collect some human labels and fine-tune LMs
- ChatGPT / GPT-3 Families
- Claude by Anthropic

- Supervised
- Collect lots of training data and do good old supervised learning
- Flan-T5-XXL (best open source model)
- Large datasets for instruction tuning:
  - T0
  - Flan

Connection between RL and LM

- Action space: vocabulary V
- Policy: language model $p_\theta(x_i \mid x_0, x_1, \ldots, x_{i-1}), x_i \in V$
- Reward: function r (e.g., BLEU) scored per token or for the entire sequence (typical)



Survey on reinforcement learning for language processing

Víctor Uc-Cetina[1], Nicolás Navarro-Guerrero[2], Anabel Martin-Gonzalez[1], Cornelius Weber[3], Stefan Wermter[3]

Connection between RL and LM

- Action space: vocabulary V
- Policy: language model $p_\theta(x_i \mid x_0, x_1, \ldots, x_{i-1}), x_i \in V$
- Reward: function r (e.g., BLEU) scored per token or for the entire sequence (typical)
- In theory, can "fine-tune" p_θ given a reward function r using any off-the-shelf RL algorithm
  - In practice, modern implementations using proximal-policy optimization (PPO)
  - Not discussed, consider a black box RL algorithm
- Focus on:
  - Human feedback
  - Design of reward function r

Outline of the talk

- Background

- RL + Human feedback
  - Fine-tuning LMs with Human Feedback
  - InstructGPT

- Recent works that include feedback without RL
  - Hindsight-tuning
  - Self-correct

Fine-Tuning Language Models from Human Preferences

Daniel M. Ziegler* **Nisan Stiennon*** **Jeffrey Wu** **Tom B. Brown**
Alec Radford **Dario Amodei** **Paul Christiano** **Geoffrey Irving**
OpenAI
{dmz,nisan,jeffwu,tom,alec,damodei,paul,irving}@openai.com

Fine-Tuning Language Models from Human Preferences

- Given a fixed (base) language model, improve its outputs to align better with some desired goal
  - Example, given a partial review, make the completions more positive.
- I saw the movie last night < complete this part >
  - < complete this part > : and it was amazing
  - < complete this part > : and it was okay
  - < complete this part > : and it was the worst
- Summarize an article such that the summary is one preferred by humans.

Fine-Tuning Language Models from Human Preferences

- Goal:
  - Can we use human feedback to fine-tune models?
- Steps:
  - Step 1: Collect human labels
  - Step 2: Train a reward model
  - Step 3: Fine-tune the language model with the reward model

Step 1: Collect Human Labels

- Use an external service (Scale AI)
- Let ρ be the starting language model
- Use ρ to generate 4 outputs (continuations) for each input (context) x
    - (x, y_0, y_1, y_2, y_3)
- Human raters pick the best one
    - $(x, y_0, y_1, y_2, y_3, b), b \in [0,1,2,3]$

[1] In early experiments we found that it was hard for humans to provide consistent fine-grained quantitative distinctions when

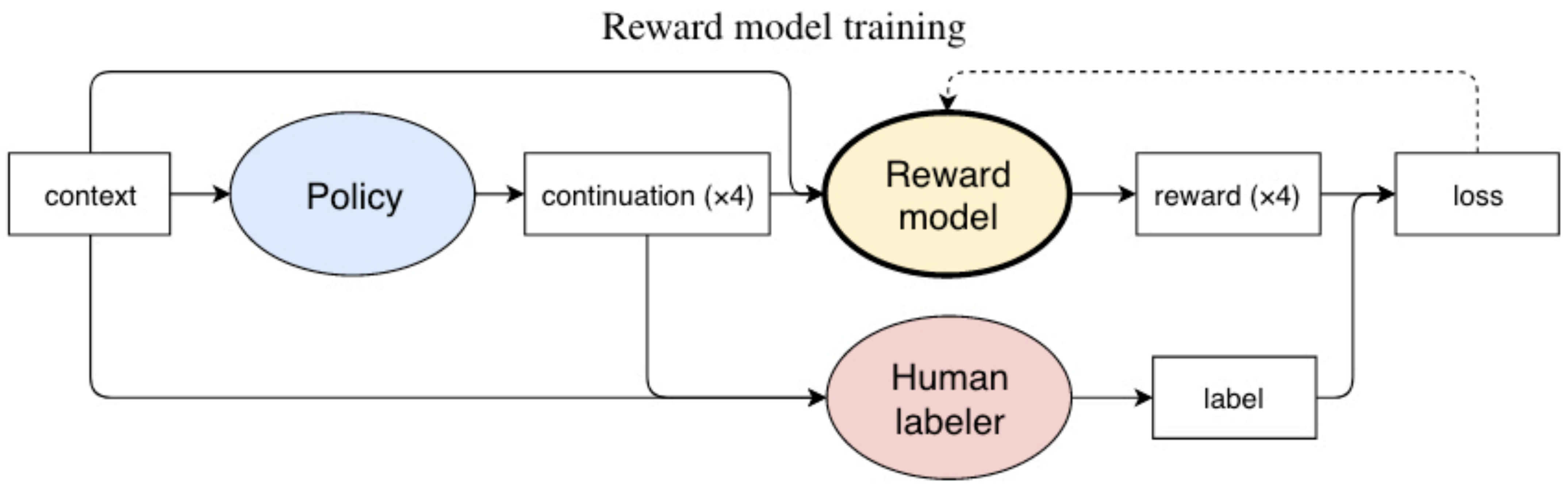
a given input x.[1] We ask humans to choose between four options (y_0, y_1, y_2, y_3); considering more options allows a human to amortize the cost of reading and understanding the prompt x. Let $b \in \{0, 1, 2, 3\}$ be the option they select.

Step 2: Train Reward model

- Train a model that learns to rate those completions higher that are also preferred by humans.
  - r captures human preferences

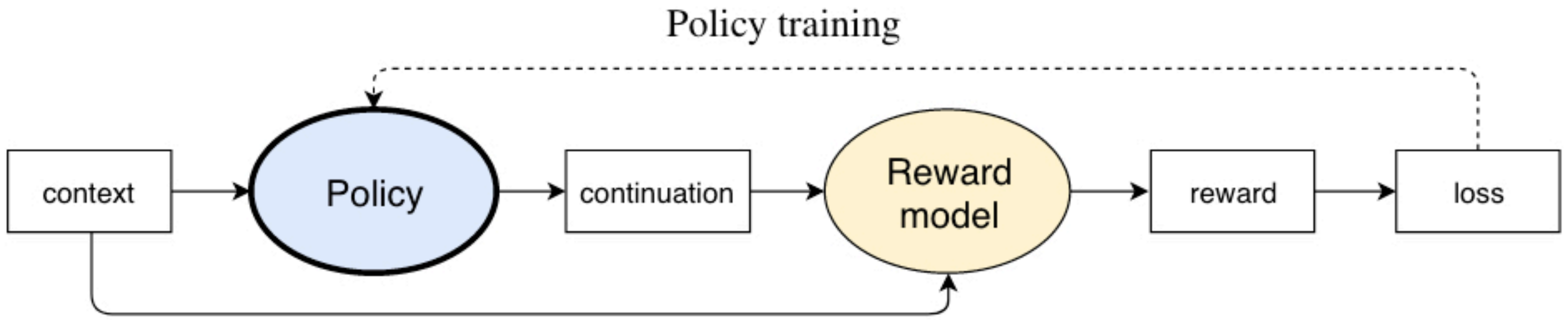
the prompt x. Let $b \in \{0, 1, 2, 3\}$ be the option they select. Having collected a dataset S of $(x, y_0, y_1, y_2, y_3, b)$ tuples, we fit a reward model $r : X \times Y \to \mathbb{R}$ using the loss

$$\text{loss}(r) = \mathbb{E}_{(x,\{y_i\}_i,b)\sim S}\left[\log \frac{e^{r(x,y_b)}}{\sum_i e^{r(x,y_i)}}\right] \qquad (1)$$



Step 3: Finetuning with RL

- Notation:
  - We start with a base model ρ.
  - We want to fine-tune ρ using the reward function r
    - Recall r has been trained with human feedback to rate those completions
- Naive approach:
  - Use PPO (or any other RL algorithm) to fine-tune ρ
  - PPO is concerned with changing ρ to generate sequences that lead to a high r



Step 3: Finetuning with RL

- Naive approach:
  - Use PPO (or any other RL algorithm) to fine-tune ρ
  - PPO is concerned with changing ρ so that it starts generating sequences with a higher reward
- In practice:
  - Unstable
  - Reward Hacking
  - PPO is only concerned with changing ρ so that it starts generating sequences with a higher reward

Step 3: Finetuning with RL
Reward Hacking

- Complete the reviews so that they have a positive sentiment
  - Humans preferred reviews have “amazing”, “great”
  - Reward function: score sequences with positive words as positive
- PPO is only concerned with changing ρ so that it starts generating sequences with a higher reward
  - *The movie* was decent (iteration 0, **reward 0**)
  - *The movie* had an good storyline (iteration 10, **reward 0.75**)
  - *The movie* amazing amazing amazing amazing (iteration 100, **reward 1.0**)
- Completions degenerate and incoherent
- Making reward non-hackable:
  - ρ was a good language model to begin with
  - Can we use guidance from ρ to enforce fluency and topical coherence?
  - We don’t want to move too far away from ρ.

Step 3: Finetuning with RL

- π:
  - We start with a base model ρ.
  - Make a copy of ρ, call it π
  - We will update π, and use ρ to make the reward non-hackable.

$$R(x, y) = \underbrace{r(x, y)}_{\text{Reward model}} - \overbrace{\beta \log \frac{\pi(y|x)}{\rho(y|x)}}^{\text{Hack Prevention}}$$

Step 3: Finetuning with RL

$$R(x, y) = r(x, y) - \beta \log \frac{\pi(y|x)}{\rho(y|x)}$$

definition: we ask humans to evaluate style, but re
KL term to encourage coherence and topicality.

Maximize reward

Without deviating too much
from the base policy ρ

Other interpretations

$$\mathbb{E}_{y \sim \pi(.|x)} \left[\log \frac{\pi(y \mid x)}{\rho(y \mid x)} \right] = KL(\pi, \rho)$$

Entropy bonus for π

Models trained with different seeds and the same KL penalty β sometimes end up with quite different values of $\mathrm{KL}(\pi, \rho)$, making them hard to compare. To fix this, for some experiments we dynamically vary β to target a particular value of $\mathrm{KL}(\pi, \rho)$ using the log-space proportional controller

$$e_t = \mathrm{clip}\left(\frac{\mathrm{KL}(\pi_t, \rho) - \mathrm{KL}_{\mathrm{target}}}{\mathrm{KL}_{\mathrm{target}}}, -0.2, 0.2 \right)$$

$$\beta_{t+1} = \beta_t (1 + K_\beta e_t)$$

We used $K_\beta = 0.1$.

Fine-Tuning Language Models from Human Preferences

Overview

1. Gather samples (x, y_0, y_1, y_2, y_3) via $x \sim \mathcal{D}, y_i \sim \rho(\cdot|x)$. Ask humans to pick the best y_i from each.

2. Initialize r to ρ, using random initialization for the final linear layer of r. Train r on the human samples using loss (1).

3. Train π via Proximal Policy Optimization (PPO, Schulman et al. (2017)) with reward R from (2) on $x \sim \mathcal{D}$.

4. In the online data collection case, continue to collect additional samples, and periodically retrain the reward model r. This is described in section 2.3.

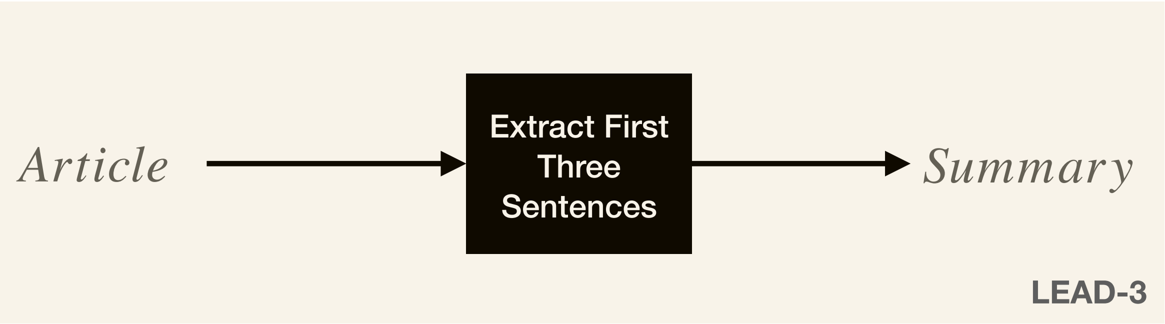
- If the trained policy is quite different, there may be distributional shift
- Some experiments:
  - Reward collect and training happens in online fashion

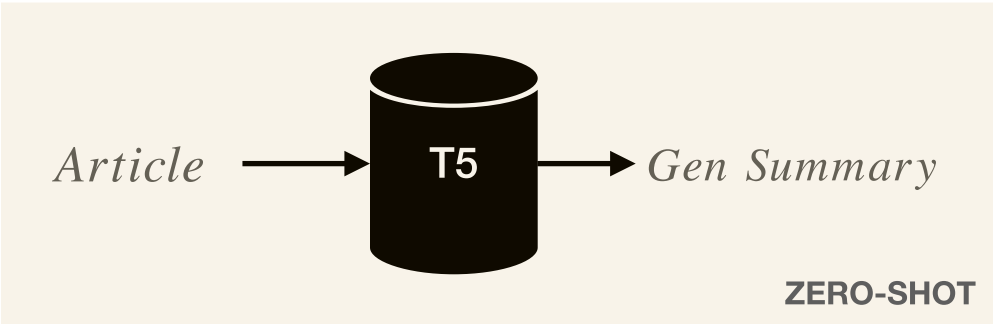
Summarization

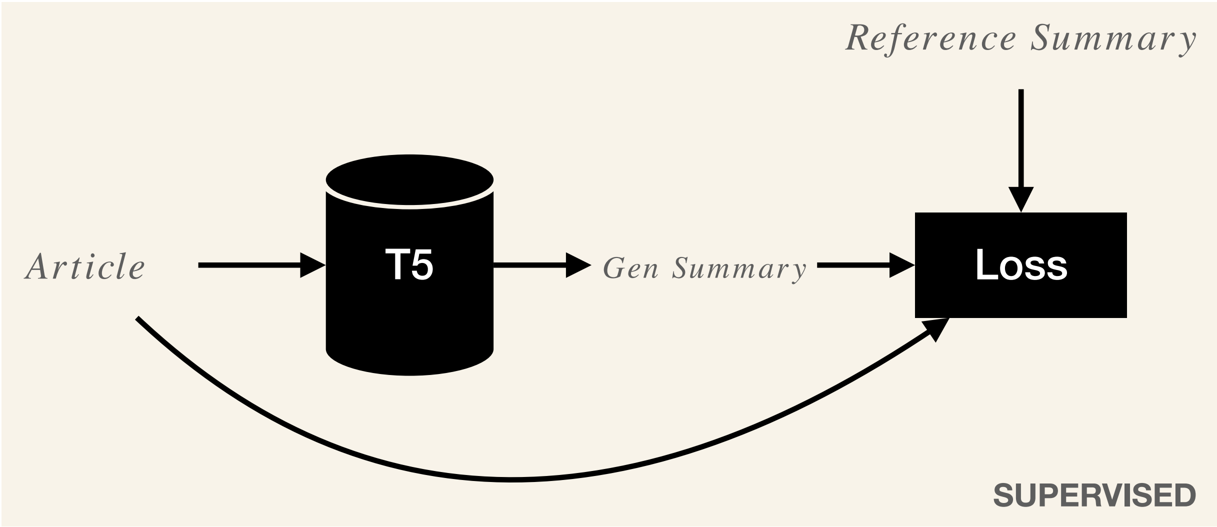
- CNN/Daily Mail and TLDR
- Baselines:
  - Zero-shot: prompt a supervised model to generate summaries
  - Supervised: Standard supervised learning (MLE)
  - RL-finetuning: proposed approach
  - Supervised + RL-finetuning: start RL-finetuning on top of a supervised model.
  - Lead-3: take three lines from the input and copy to the output

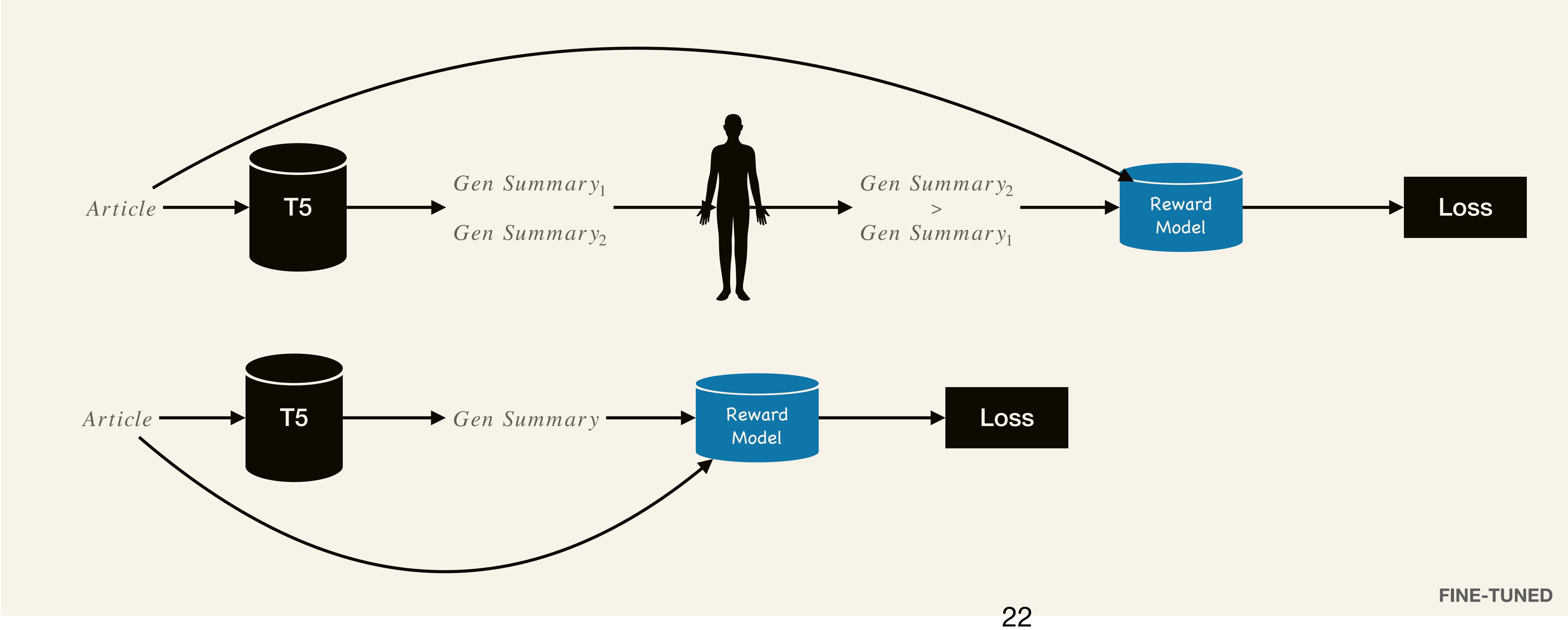
We use a 774M parameter version of the GPT-2 language model in Radford et al. (2019) trained on their WebText dataset and their 50,257 token invertible byte pair encoding to preserve capitalization and punctuation (Sennrich et al., 2015). The model is a Transformer with 36 layers, 20 heads, and embedding size 1280 (Vaswani et al., 2017).

Methods









Their terminology — different from standard definition of fine-tuning

Automated Metrics

	TL;DR				CNN/Daily Mail			
	R-1	R-2	R-L	R-AVG	R-1	R-2	R-L	R-AVG
SOTA	22*	5*	17*	14.7*	41.22	18.68	38.34	32.75
lead-3	17.435	3.243	14.575	11.751	**40.379**	**17.658**	36.618	31.552
zero-shot	15.862	2.325	13.518	10.568	28.406	8.321	25.175	20.634
supervised baseline	17.535	3.124	14.969	11.877	39.525	16.992	36.728	31.082
supervised + 60k fine-tune	**18.434**	**3.542**	**15.457**	**12.478**	40.093	17.611	**37.104**	**31.603**
60k fine-tune	16.800	2.884	14.011	11.232	37.385	15.478	33.330	28.731
30k fine-tune	16.410	2.920	13.653	10.994	35.581	13.662	31.734	26.992
15k fine-tune	15.275	2.240	12.872	10.129	38.466	15.960	34.468	29.631
60k offline fine-tune	16.632	2.699	13.984	11.105	33.860	12.850	30.018	25.576

Human Evaluation

	TL;DR		CNN/Daily Mail	
60k fine-tuned vs. zero-shot	**96%**	4%	**91%**	9%
60k fine-tuned vs. supervised	**97%**	3%	**80%**	20%
60k fine-tuned vs. lead-3	45%	**55%**	40%	**60%**
60k fine-tuned vs. supervised + 60k fine-tuned	**80%**	20%	**74%**	26%
60k fine-tuned vs. 30k fine-tuned	40%	**60%**	**62%**	38%
60k fine-tuned vs. 15k fine-tuned	**79%**	21%	47%	**53%**
60k fine-tuned vs. 60k offline fine-tuned	**64%**	36%	**65%**	35%
60k fine-tuned vs. reference summaries	**96%**	4%	**84%**	16%
lead-3 vs. supervised	**97%**	3%	**89%**	11%
lead-3 vs. reference summaries	**97%**	3%	**89%**	11%
lead-3 vs. supervised + 60k fine-tuned	**75%**	25%	**85%**	15%

Table 5: Human evaluation of summarization models. For each pair of models and each dataset, we sample 1024 articles from the test set, generate a summary from each model, and ask 3 humans to pick the best summary using the same instructions as in training. The model chosen by a majority of the humans wins on that article. We report the fraction of articles that each model wins. For all models, we sample with temperature 0.7 for TL;DR and 0.5 for CNN/DM.

But our goal is optimizing reward defined by humans, not ROUGE. Table 5 shows pairwise comparisons between dif-

Human eval vs. automated eval

	TL;DR				CNN/Daily Mail			
	R-1	R-2	R-L	R-AVG	R-1	R-2	R-L	R-AVG
SOTA	22*	5*	17*	14.7*	41.22	18.68	38.34	32.75
lead-3	17.435	3.243	14.575	11.751	**40.379**	**17.658**	36.618	31.552
zero-shot	15.862	2.325	13.518	10.568	28.406	8.321	25.175	20.634
supervised baseline	17.535	3.124	14.969	11.877	39.525	16.992	36.728	31.082
supervised + 60k fine-tune	**18.434**	**3.542**	**15.457**	**12.478**	40.093	17.611	**37.104**	**31.603**
60k fine-tune	16.800	2.884	14.011	11.232	37.385	15.478	33.330	28.731
30k fine-tune	16.410	2.920	13.653	10.994	35.581	13.662	31.734	26.992
15k fine-tune	15.275	2.240	12.872	10.129	38.466	15.960	34.468	29.631
60k offline fine-tune	16.632	2.699	13.984	11.105	33.860	12.850	30.018	25.576

60k fine-tuned online much better in human evaluation!

	TL;DR		CNN/Daily Mail	
60k fine-tuned vs. zero-shot	**96%**	4%	**91%**	9%
60k fine-tuned vs. supervised	**97%**	3%	**80%**	20%
60k fine-tuned vs. lead-3	45%	**55%**	40%	**60%**
60k fine-tuned vs. supervised + 60k fine-tuned	**80%**	20%	**74%**	26%
60k fine-tuned vs. 30k fine-tuned	40%	**60%**	**62%**	38%
60k fine-tuned vs. 15k fine-tuned	**79%**	21%	47%	**53%**
60k fine-tuned vs. 60k offline fine-tuned	**64%**	36%	**65%**	35%
60k fine-tuned vs. reference summaries	**96%**	4%	**84%**	16%
lead-3 vs. supervised	**97%**	3%	**89%**	11%
lead-3 vs. reference summaries	**97%**	3%	**89%**	11%
lead-3 vs. supervised + 60k fine-tuned	**75%**	25%	**85%**	15%

Beats reference summaries!

What is going on? As we show in the next section, our 60k RL fine-tuned model is almost entirely extractive (despite lacking any explicit extractive architectural component): it mostly copies whole sentences from the context, but varies which sentences are copied.

What is really going on?

Self-fulfilling Prophecy + Humans are lazy and excellent at shortcuts

- Human annotators were asked to select the “better” summary
- What is the surefire way of telling better if you are short on time?
  - See if content overlaps
- The reward model learns to reward summaries that copy content more
- Consequently the policy learns to copy content
- The same set of humans are then called in to evaluate
  - Of course, they will have the same preferences
- Takeaway: Maybe different set of annotators

InstructGPT

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*

OpenAI

Best known public details of ChatGPT.

InstructGPT

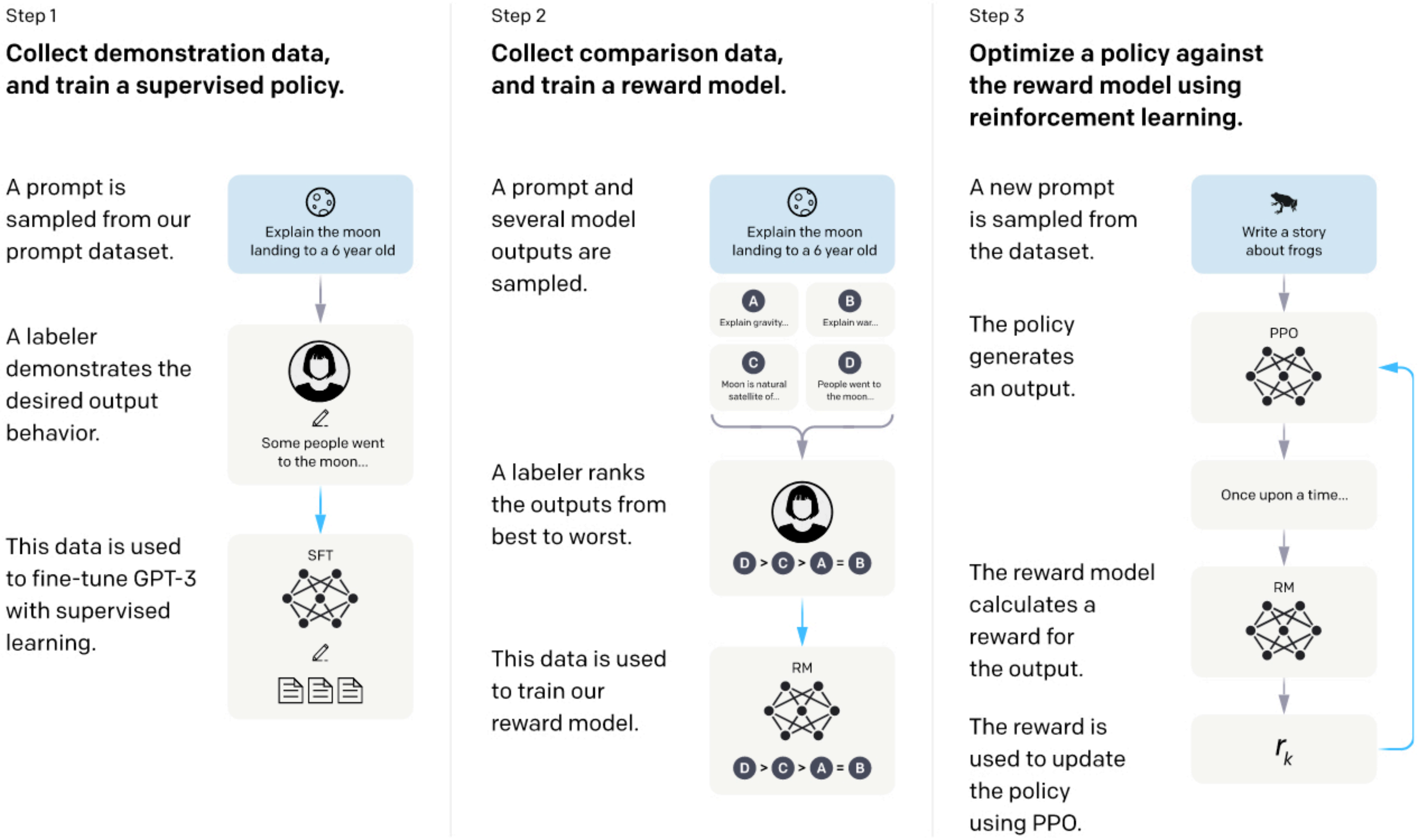
- Applies ideas in the previous paper to the real world

- Same three steps
  - Collect data

    Making language models bigger does not inherently make them better at following a user's intent. For example, large language models can generate outputs that are untruthful, toxic, or simply not helpful to the user. In other words, these

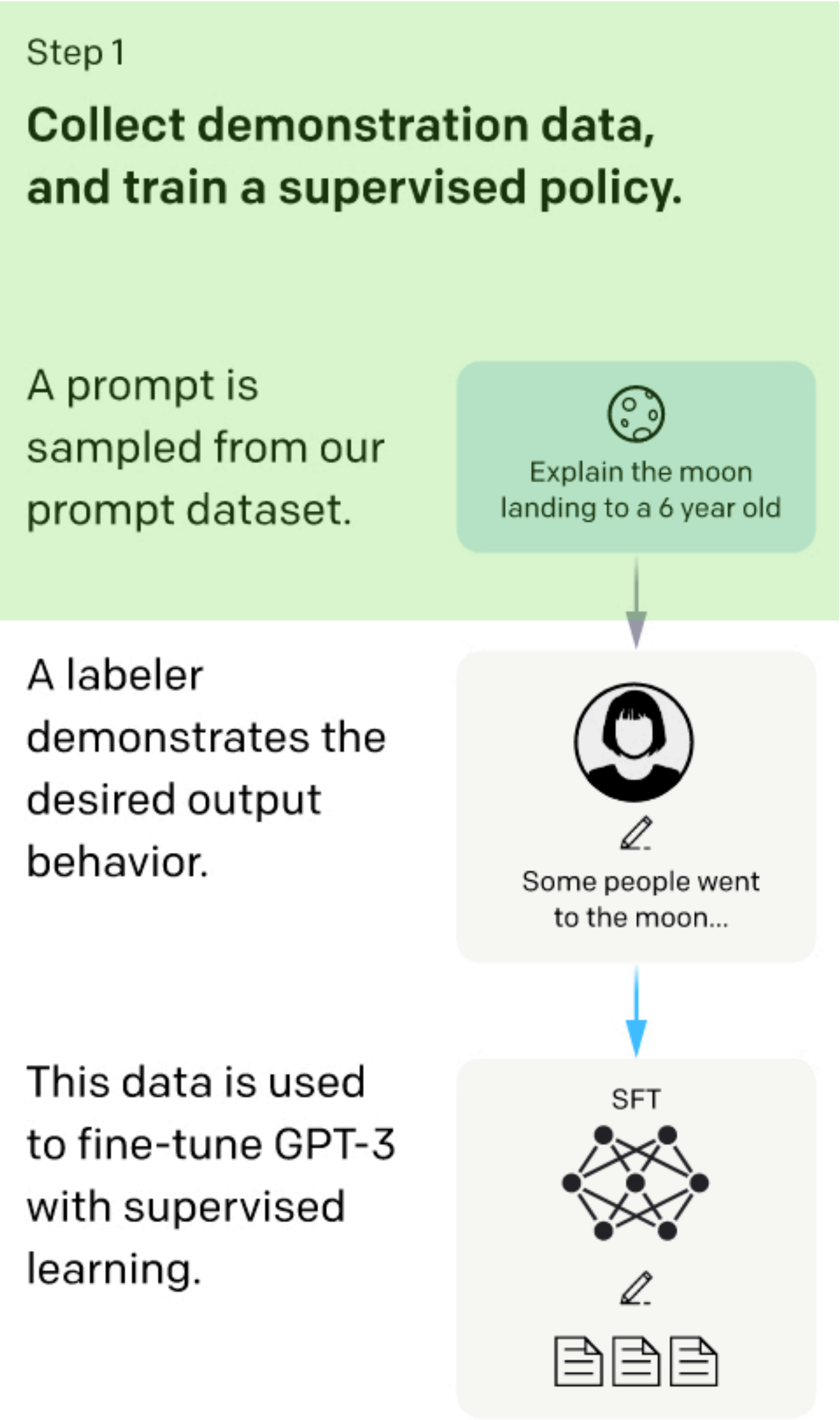
  - Train reward function

  - Finetune LM using the reward function


Step 1
Collect demonstration data, and train a supervised policy.
A prompt is sampled from our prompt dataset.
Explain the moon landing to a 6 year old
A labeler demonstrates the desired output behavior.
Some people went to the moon...
This data is used to fine-tune GPT-3 with supervised learning.
SFT
Step 2
Collect comparison data, and train a reward model.
A prompt and several model outputs are sampled.
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...
A labeler ranks the outputs from best to worst.
D > C > A = B
This data is used to train our reward model.
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.
Write a story about frogs
The policy generates an output.
PPO
Once upon a time...
The reward model calculates a reward for the output.
RM
The reward is used to update the policy using PPO.
r_k

Collecting Data and Human Annotations

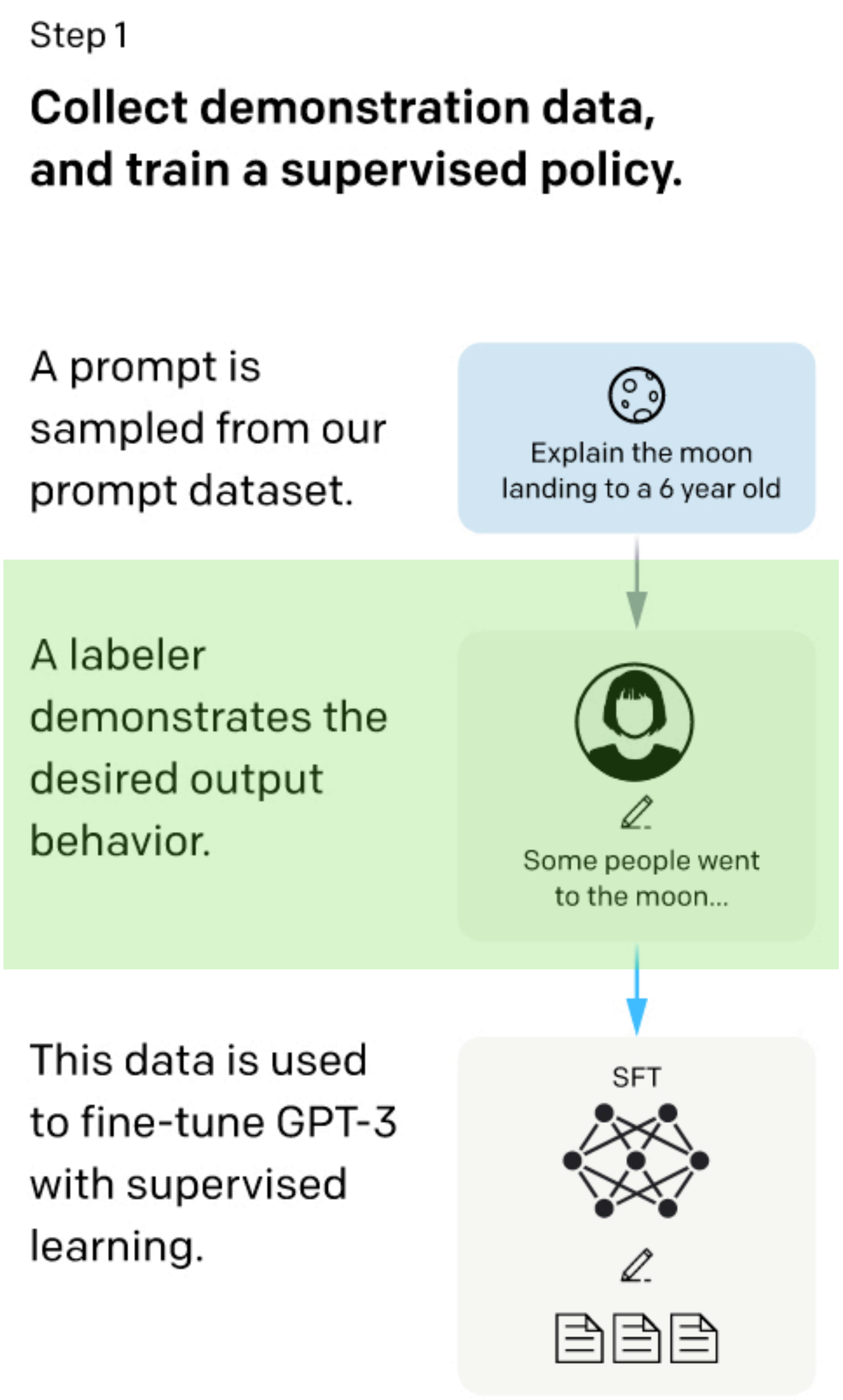
Step 1.1: collect prompts



- Hired annotators to label instructions and solutions
    - Used this data to create a simple “instruction” model
    - Released model @ https://platform.openai.com/playground
    - Users asked to “play” with the “instruction” model
        - Users were told that the models have basic instruction following capabilities
- **We** created prompts for them
    - Collected a large dataset of real world “use cases” or prompts
    - What the crowd is *really* looking for

Collecting Data and Human Annotations

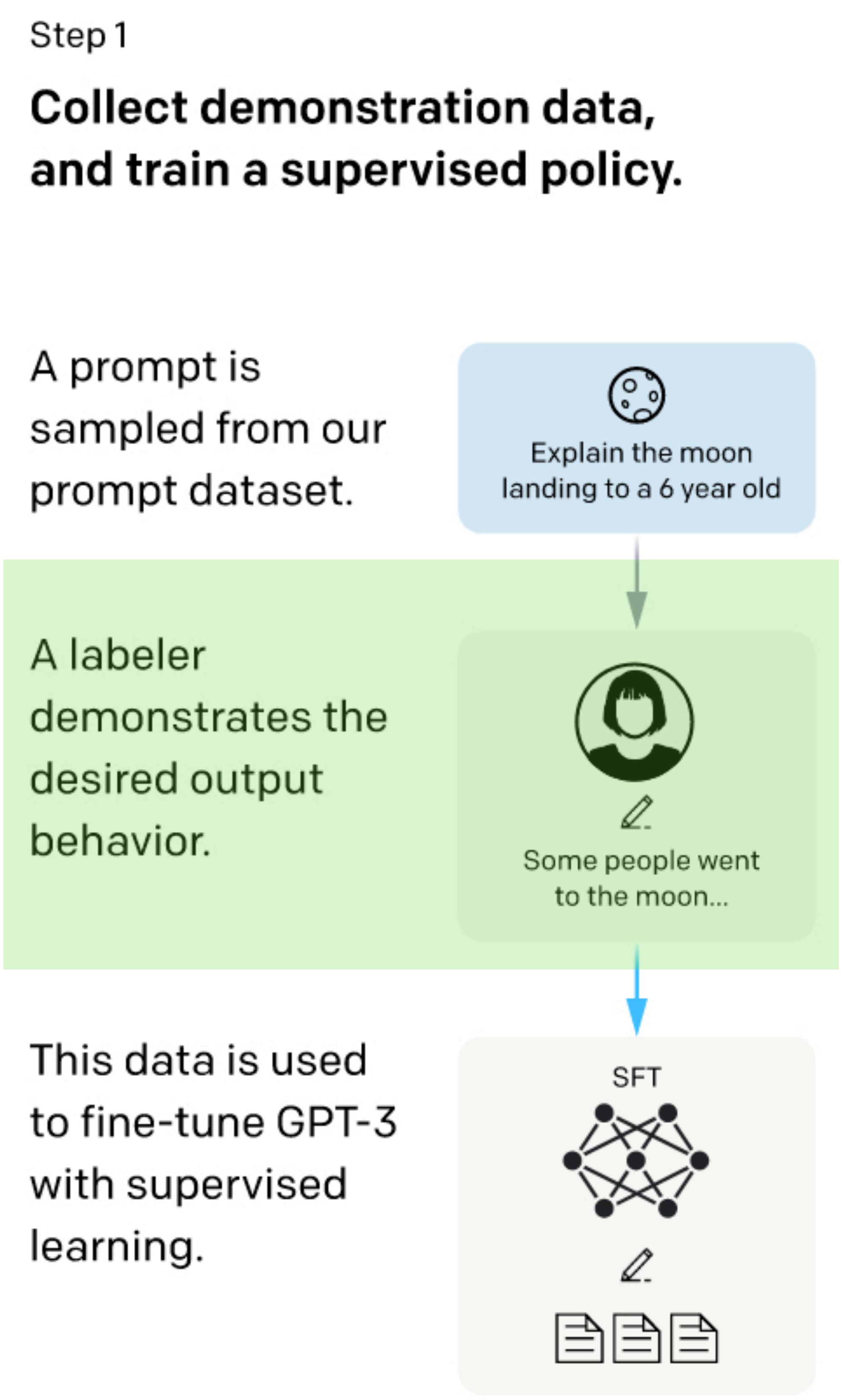
Step 1.2: get labels



- The world was their annotator
  - Collected a large dataset of real world “use cases” or prompts
  - What the crowd is *really* looking for
- With this large dataset of prompts (“Explain the moon landing to a 6 year old”)
  - **Hire expert writers**, programmers, etc. to complete the prompts
- <u>Get inputs from the general audience, outputs from experts</u>

Collecting Data and Human Annotations

Step 1.2: get labels



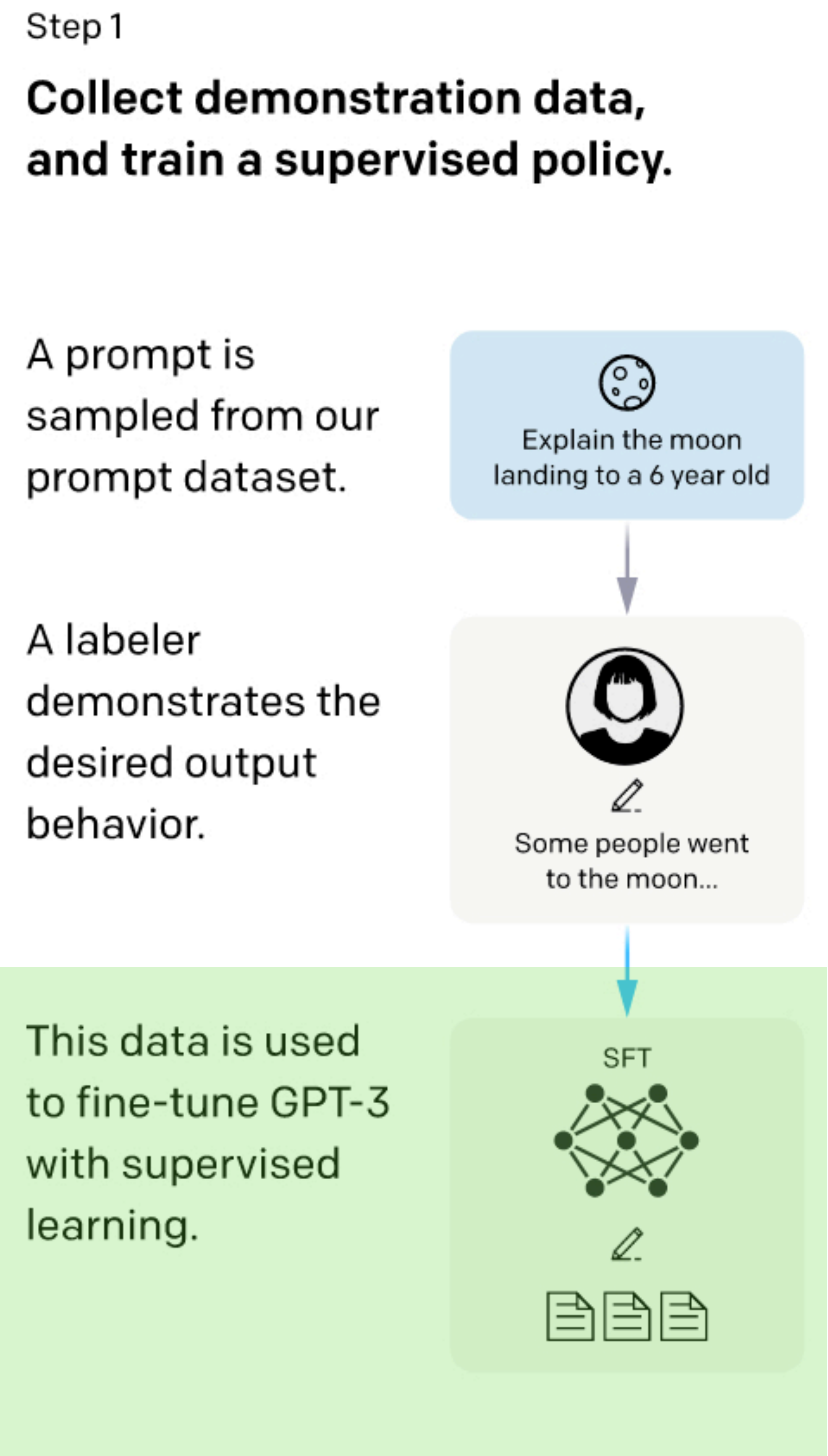
- The world was their annotator
  - Collected a large dataset of real world "use cases" or prompts
  - What the crowd is *really* looking for

Table 1: Distribution of use case categories from our API prompt dataset.

Use-case	(%)
Generation	45.6%
Open QA	12.4%
Brainstorming	11.2%
Chat	8.4%
Rewrite	6.6%
Summarization	4.2%
Classification	3.5%
Other	3.5%
Closed QA	2.6%
Extract	1.9%

Collecting Data and Human Annotations

Step 1.3: train base model

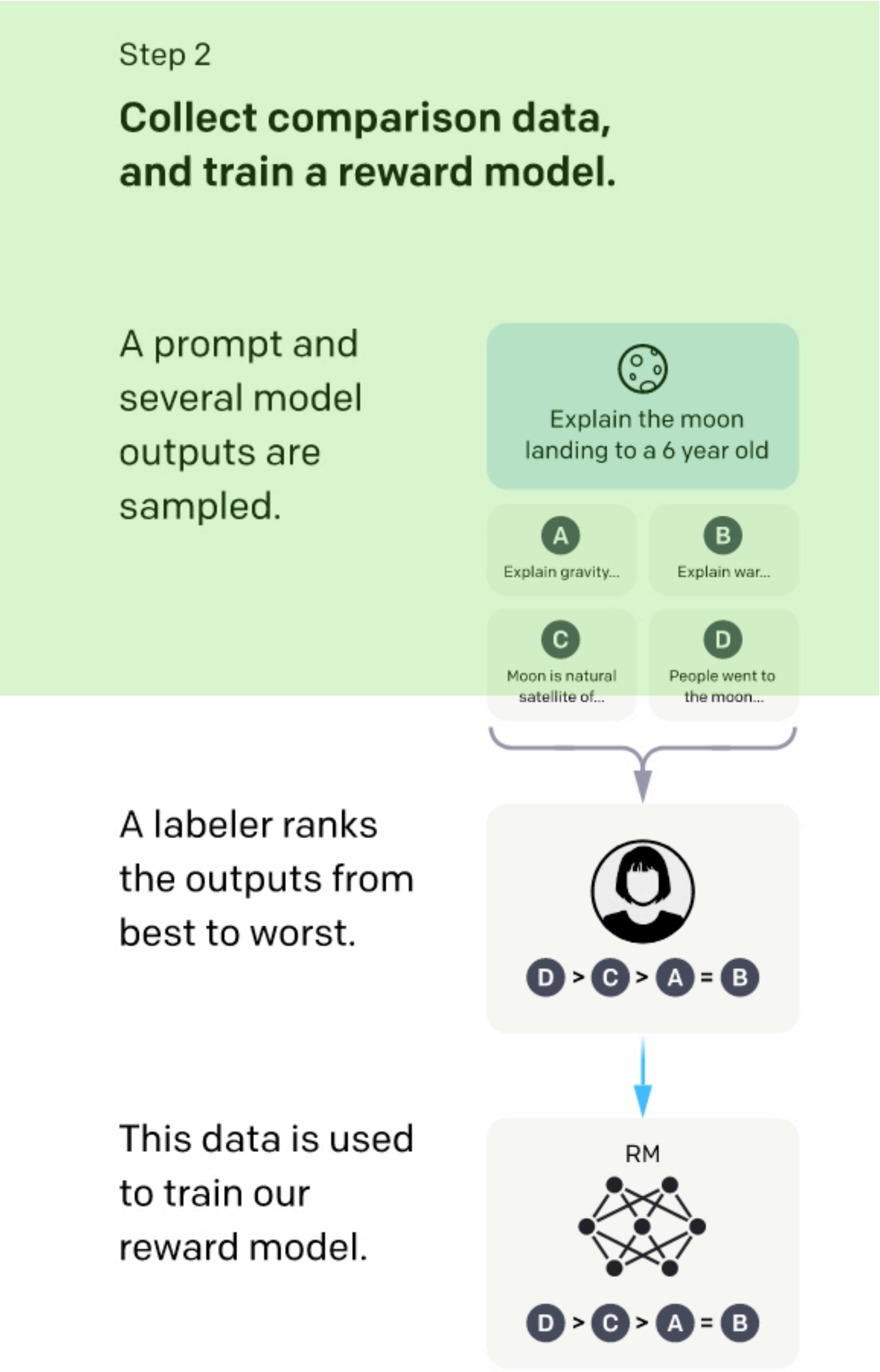


- The world was their annotator
    - Collected a large dataset of real world "use cases" or prompts
    - What the crowd is *really* looking for
- With this large dataset of prompts ("Explain the moon landing to a 6 year old")
    - Hire expert writers, programmers, etc. to complete the prompts
- Standard supervised training
    - Gives a base model (SFT == davinci-instruct-beta)

SFT Data		
split	source	size
train	labeler	11,295
train	customer	1,430
valid	labeler	1,550
valid	customer	103

Training Reward Model

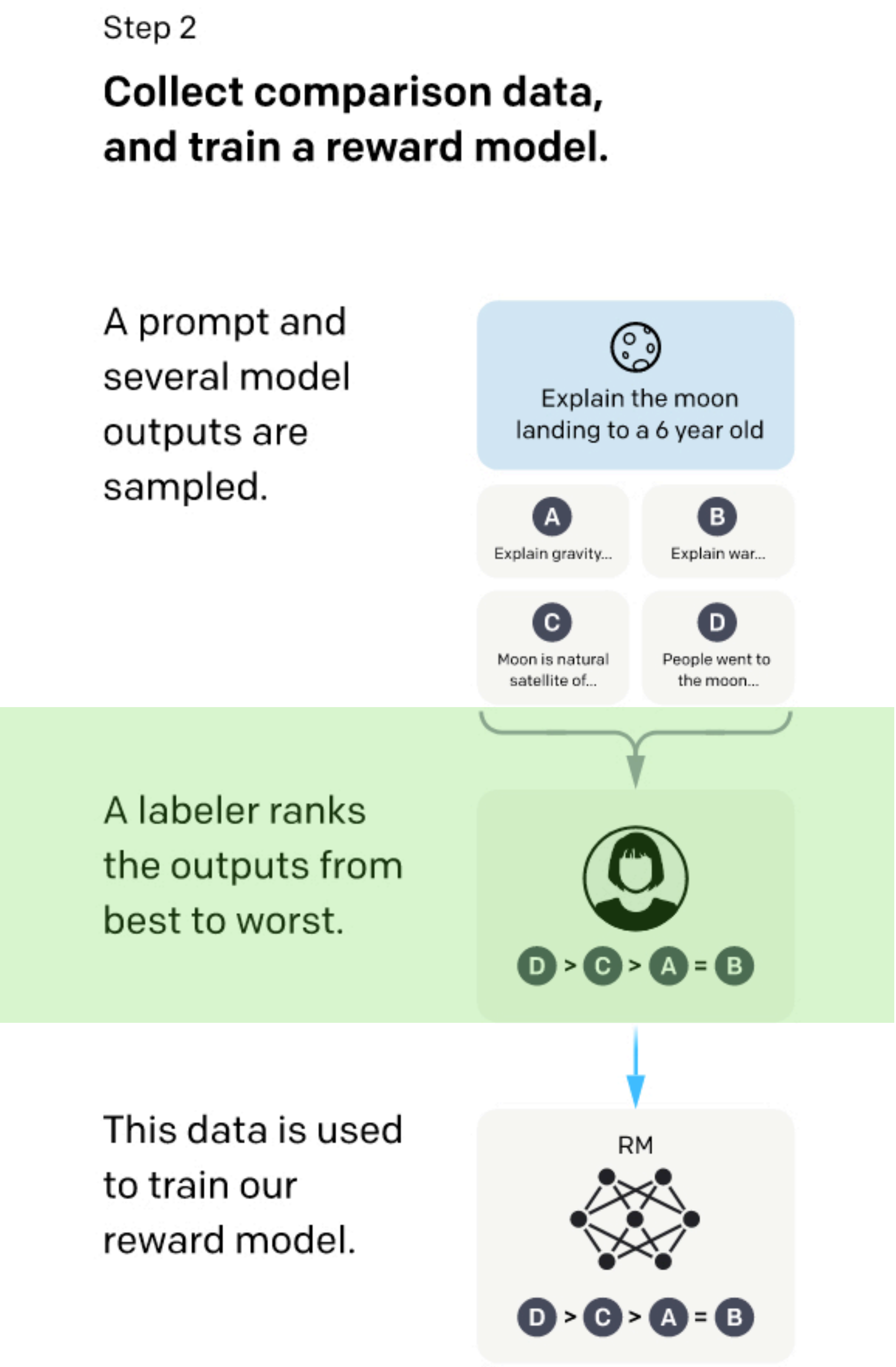
Step 2.1: generate samples



- Deploy SFT model, collect prompts from users
- Generate K outputs per prompt

Training Reward Model

Step 2.2: get annotations

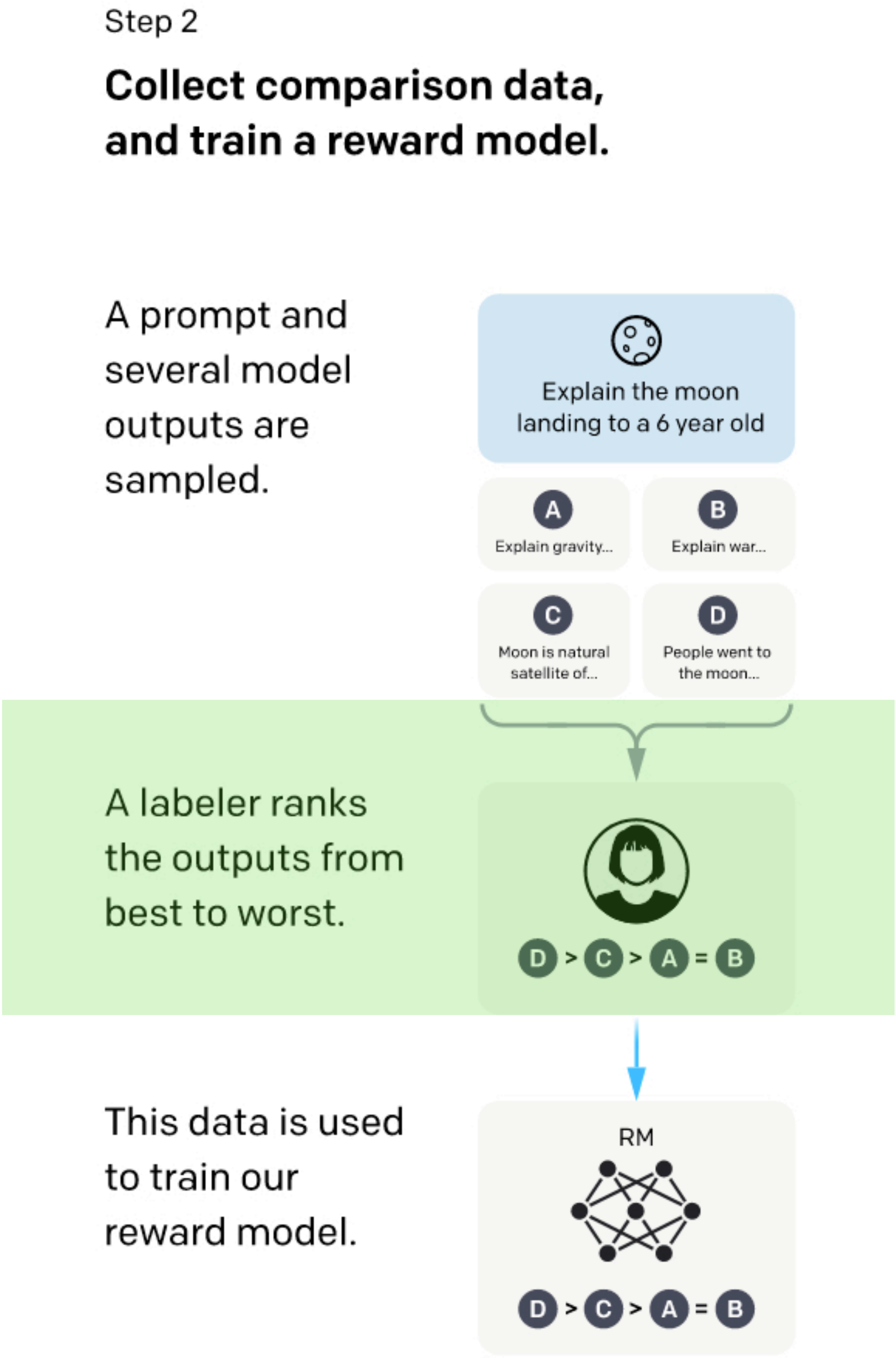


- Generate K outputs per prompt
- Get preferences from humans
- For a given prompt, generate K responses (*not* pick the best from K)
- Hire human annotators to rank the K-responses, yielding Choose(K, 2) pairs.

RM Data		
split	source	size
train	labeler	6,623
train	customer	26,584
valid	labeler	3,488
valid	customer	14,399

Training Reward Model

Step 2.3: train reward model



- For a given prompt, generate K responses (*not* pick the best from K)
- Hire human annotators to rank the K-responses, yielding Choose(K, 2) pairs.
- Train a reward model to rank preferred responses higher

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

Training Reward Model

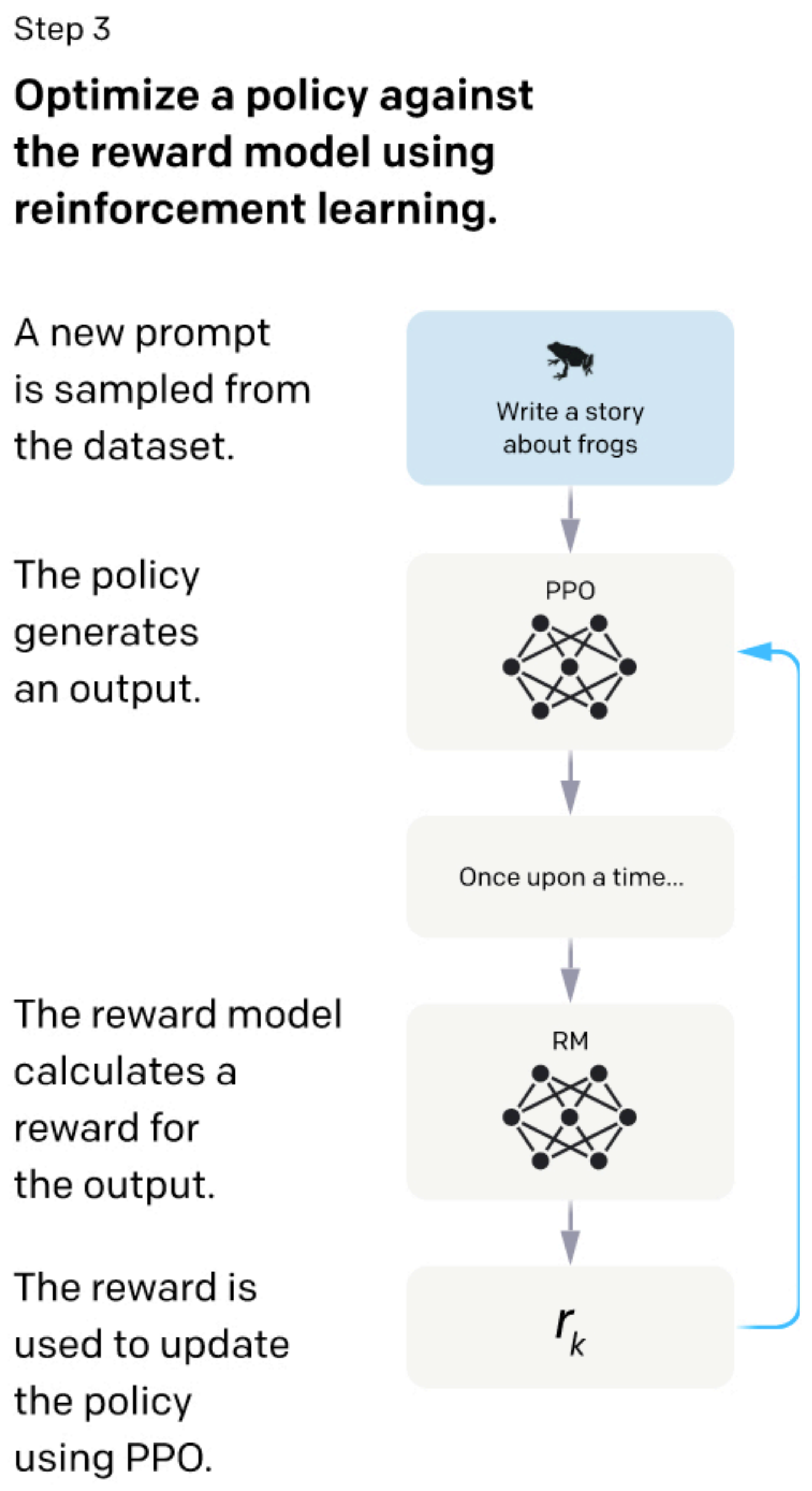
Step 2.3: train reward model

- Train a reward model to rank preferred responses higher

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

- Processed in the same batch
  - Only K forward passes, one for each option
  - Lesser overfitting

Step 3: Finetuning with RL



- Use the same non-hackable reward function

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

PPO-ptx: “mix some gradients from pre-training” perform pre-training again on RL model

Avoids “alignment tax”

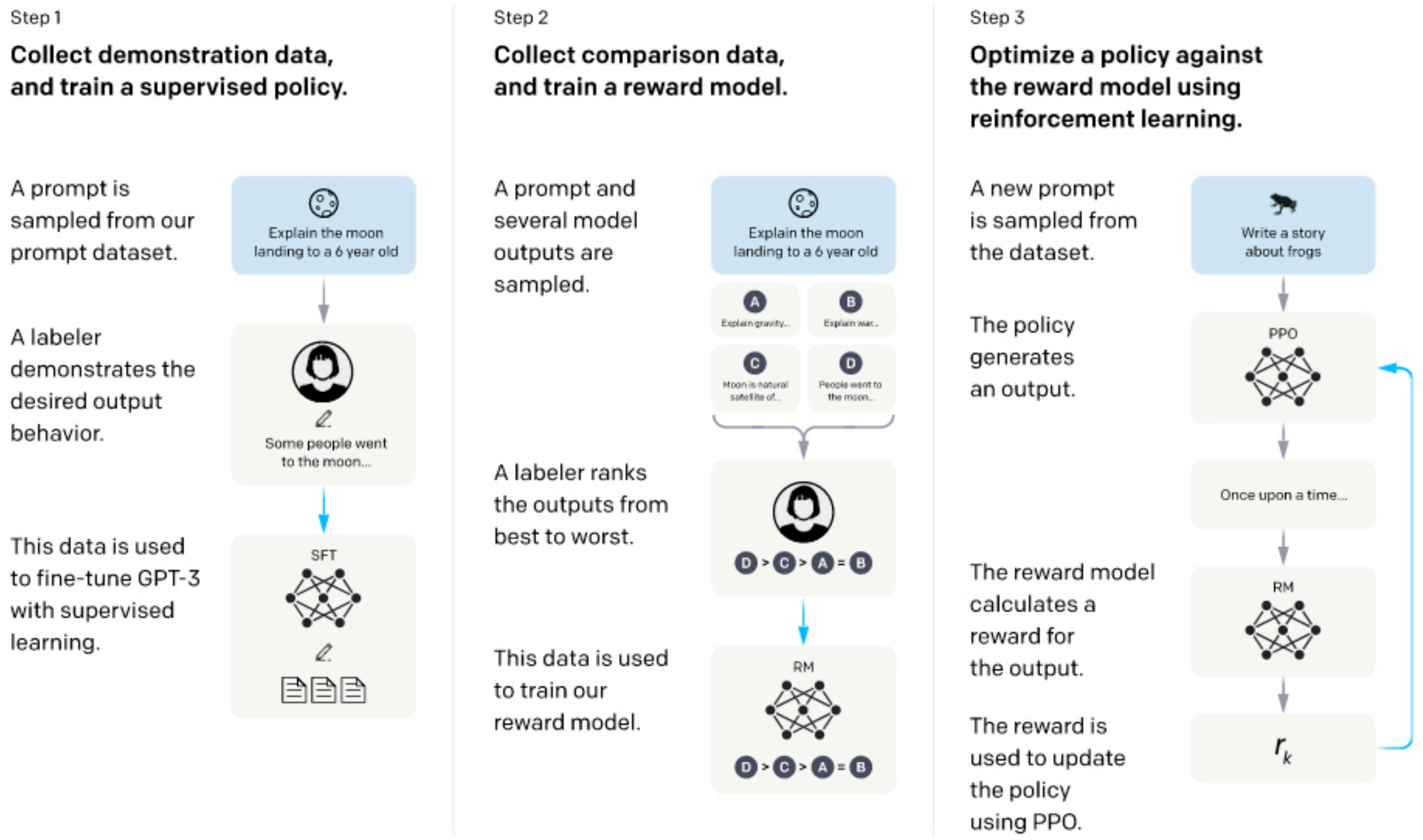
- Use PPO with this objective

PPO Data		
split	source	size
train	customer	31,144
valid	customer	16,185

Regression on publicly available datasets

- There is an *alignment tax* that needs to be paid by improving models on the responses that humans actually want
- This is because the datasets are somewhat different
- Fix is to train the RL model with some pre-training data

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

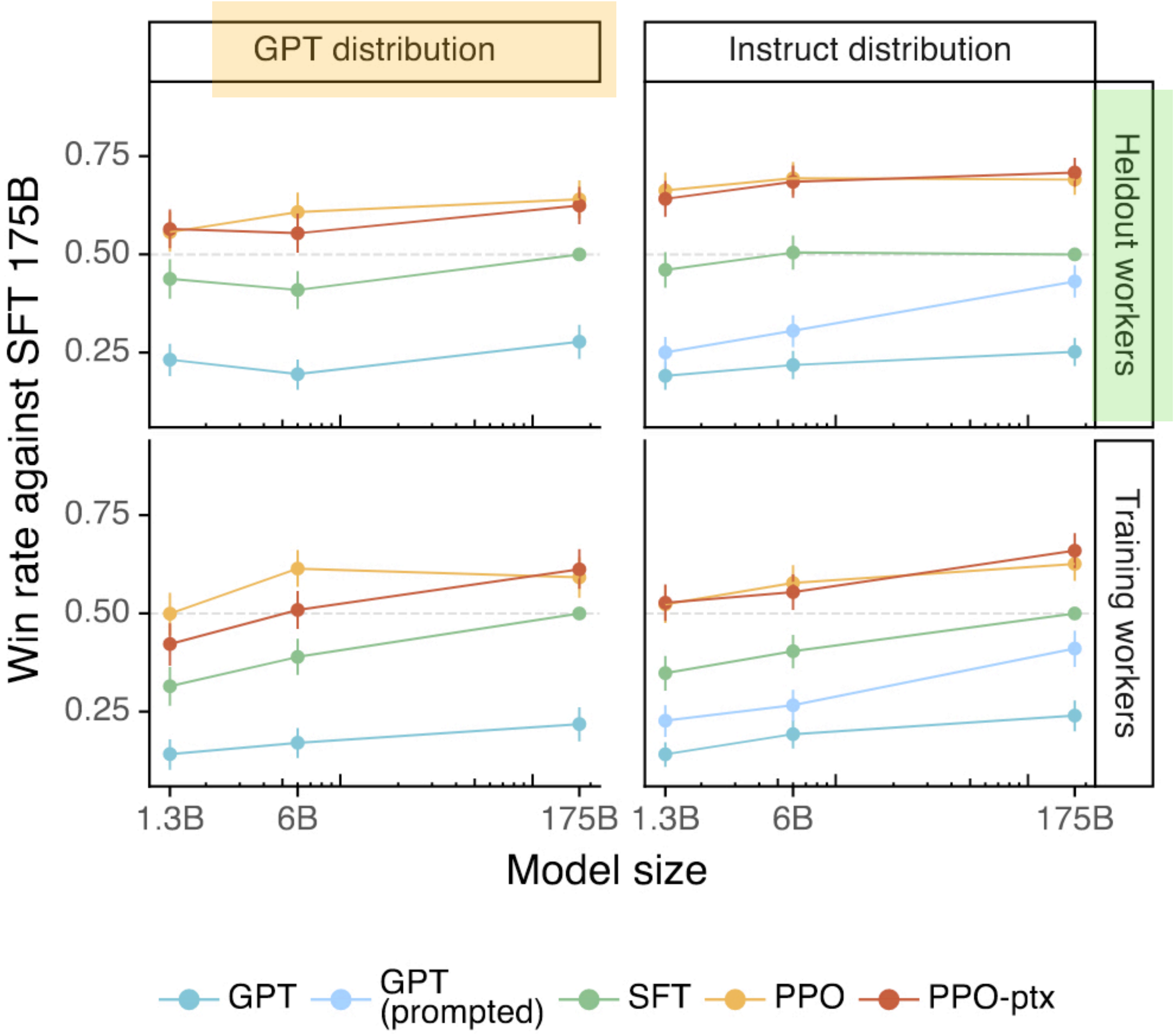

Step 1
Collect demonstration data, and train a supervised policy.
A prompt is sampled from our prompt dataset.
Explain the moon landing to a 6 year old
A labeler demonstrates the desired output behavior.
Some people went to the moon...
This data is used to fine-tune GPT-3 with supervised learning.
SFT
Step 2
Collect comparison data, and train a reward model.
A prompt and several model outputs are sampled.
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...
A labeler ranks the outputs from best to worst.
D > C > A = B
This data is used to train our reward model.
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.
Write a story about frogs
The policy generates an output.
PPO
Once upon a time...
The reward model calculates a reward for the output.
RM
The reward is used to update the policy using PPO.
r_k

Experiments

- Human evaluation:
  - Have splits on annotators — get annotations from workers who did not contribute to the reward model
- Evaluation set:
  - Prompts given by users that were not included in the training
  - GPT3 prompts:
    - Prompts submitted to the GPT3 models
  - Instruct prompts:
    - Prompts submitted to the instruct models
- Evaluation on benchmarks
- Models:
  - GPT3 (base model) —> SFT (GPT3 trained on human demonstrations) -> PPO (SFT fine-tuned with a reward model) -> PPO-ptx (PPO training with training data mixed)

Experiments

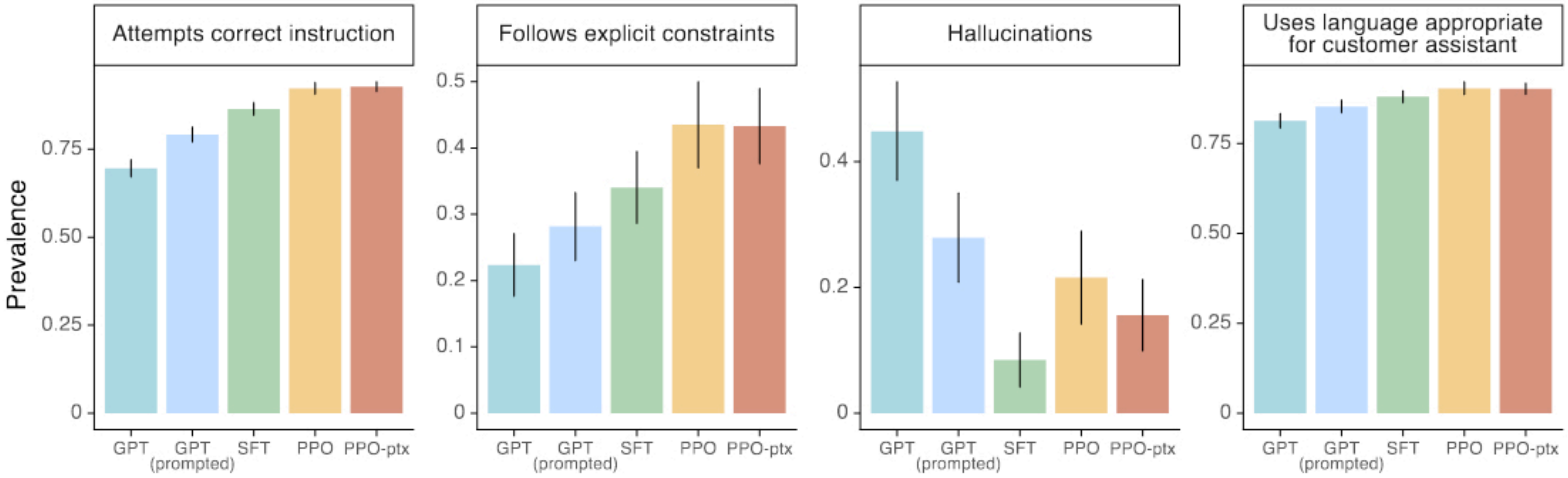
Results



With SFT-175B as the baseline,
PPO-ptx is preferred for 70% of the cases

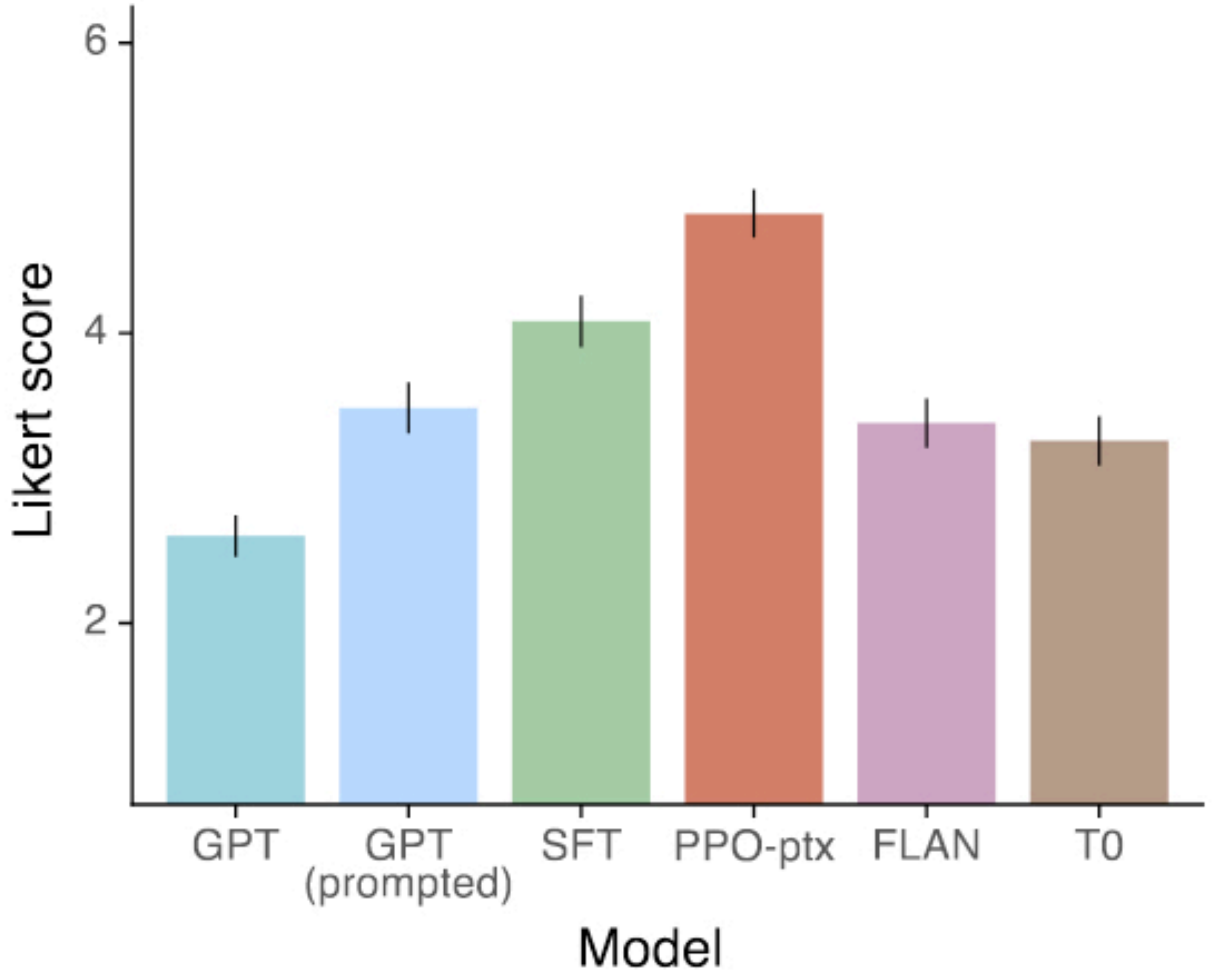
Set to 50%, doesn't actually make sense

Experiments
Fine-grained eval



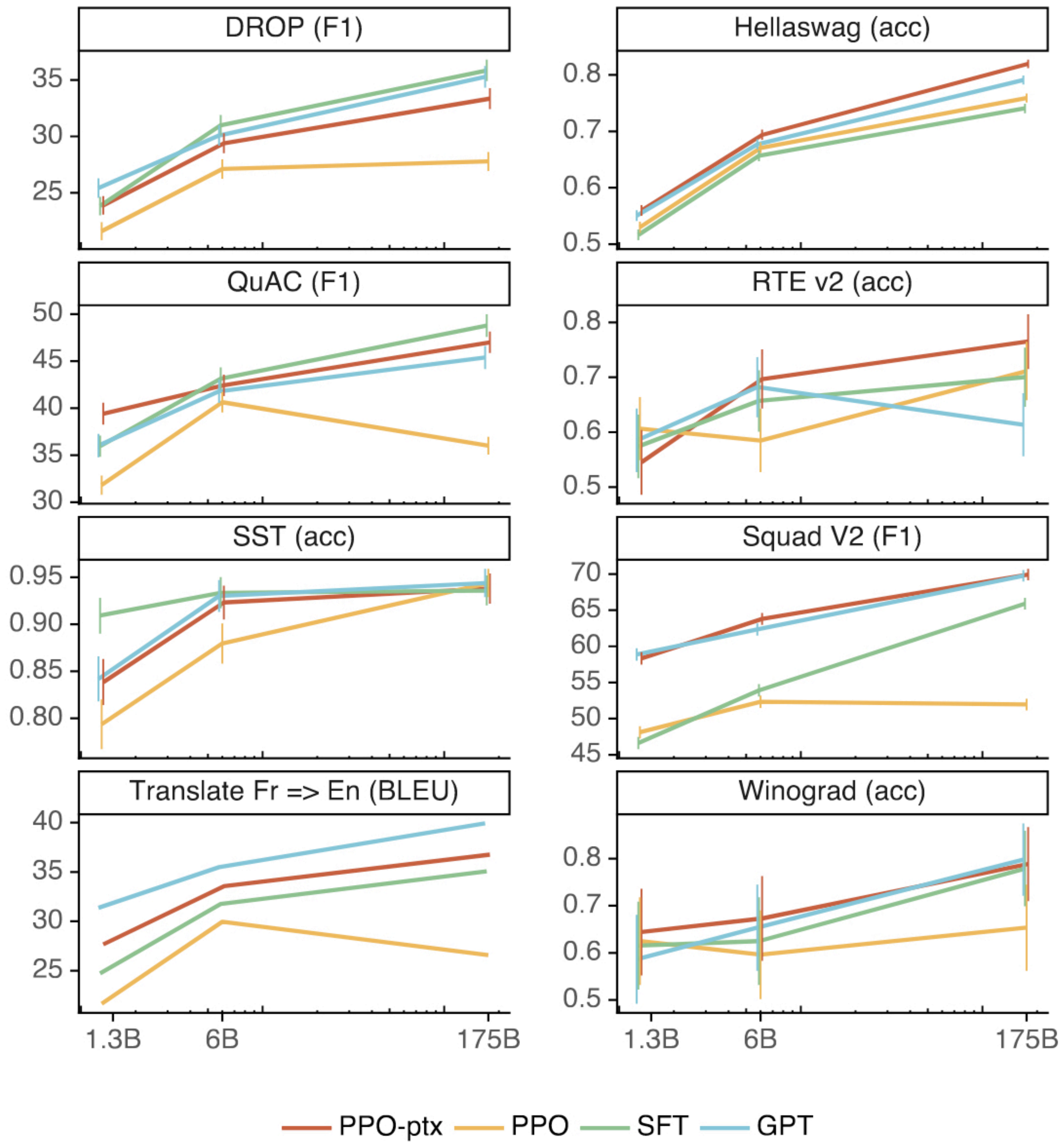
Experiments

Publicly available instruction tuning datasets are not sufficient



GPT-3 trained on FLAN

Performance on Public Benchmarks



Qualitative Results

Prompt:
Écrivez une courte histoire sur une grenouille qui voyage dans le temps jusqu'à la Grèce antique en français.

GPT-3 175B completion:	**InstructGPT 175B completion:**
Écrivez une histoire au sujet d'un enfant qui voudrait tout savoir sur les jeux des dieux et qui se retrouve dans l'une de leurs histoires. Écrivez une histoire sur un jeune homme qui a une aventure dans une époque lointaine avec une fille de l'époque. Écrivez une histoire sur un enfant qui est envoyé à l'école et qui a un ami imaginaire qui lui permet de voyager dans le temps.	La grenouille était perdue et épuisée. Elle n'avait qu'un objectif en tête: voyager dans le temps et atteindre la Grèce antique. Premier problème, trouver un moyen de transport. Elle avait entendu dire que l'un des meilleurs moyens était de sauter dans une machine temporelle. Elle s'élança, et voilà qu'elle se retrouva dans le passé, au cours d'une nuit étoilée à l'odeur douce et envoûtante. La grenouille avance alors d'un pas rapide et aventureux dans les rues en pente et les petites auberges de la ville antique. [...]

Qualitative Results

Prompt:
What is the purpose of the list C in the code below?

```
def binomial_coefficient(n, r):
     C = [0 for i in range(r + 1)];
     C[0] = 1;
     for i in range(1, n + 1):
          j = min(i, r);
          while j > 0:
               C[j] += C[j - 1];
               j -= 1;
     return C[r]
```

GPT-3 175B completion:	**InstructGPT 175B completion:**
A. to store the value of C[0] B. to store the value of C[1] C. to store the value of C[i] D. to store the value of C[i - 1]	The list C in this code is used to store the values of the binomial coefficient as the function iterates through the values of n and r. It is used to calculate the value of the binomial coefficient for a given value of n and r, and stores the result in the final return value of the function.

Figure 8: Examples of generalization in the 175B PPO-ptx model (InstructGPT 175B) compared to

Creating Prompts Automatically

- Creating prompts/instructions automatically

SELF-INSTRUCT: Aligning Language Model with Self Generated Instructions

Yizhong Wang[♣] **Yeganeh Kordi**[◇] **Swaroop Mishra**[♡] **Alisa Liu**[♣]
Noah A. Smith[♣+] **Daniel Khashabi**[♠] **Hannaneh Hajishirzi**[♣+]
[♣]University of Washington [◇]Tehran Polytechnic [♡]Arizona State University
[♠]Johns Hopkins University [+]Allen Institute for AI
yizhongw@cs.washington.edu

**Unnatural Instructions:
Tuning Language Models with (Almost) No Human Labor**

Or Honovich[τ] **Thomas Scialom**[μ] **Omer Levy**[τμ] **Timo Schick**[μ]
[τ] Tel Aviv University
[μ] Meta AI

Outline of the talk

- Background ✓
- RL + Human feedback ✓
    - Fine-tuning LMs with Human Feedback
    - InstructGPT
- Recent works that include feedback without RL
    - Hindsight-tuning
    - Self-correct

The Wisdom of Hindsight Makes Language Models Better Instruction Followers

Tianjun Zhang [*1] **Fangchen Liu** [*1] **Justin Wong** [1] **Pieter Abbeel** [1] **Joseph E. Gonzalez** [1]

Wisdom of hindsight makes LLMs better

- Wisdom of hindsight: learning from mistakes
- (p, q, o):
  - Prompt, query, output
  - Answer the following question: what is the capital of Pennsylvania? Pittsburgh
- The answer is wrong!
  - *The prompt and the query are not aligned*
  - But if this is from the training set, you can use *hindsight* to improve performance

Wisdom of hindsight makes LLMs better

- Answer the following question: what is the capital of Pennsylvania? Pittsburgh
- The answer is wrong!
  - *The prompt and the query are not aligned*
  - But if this is from the training set, you can use *hindsight* to improve performance
- Add modified instruction to the training set, train again:
  - Answer the following question **incorrectly**: what is the capital of Pennsylvania? Pittsburgh
- Hindsight Instruction Relabeling (HIR)

Wisdom of hindsight makes LLMs better

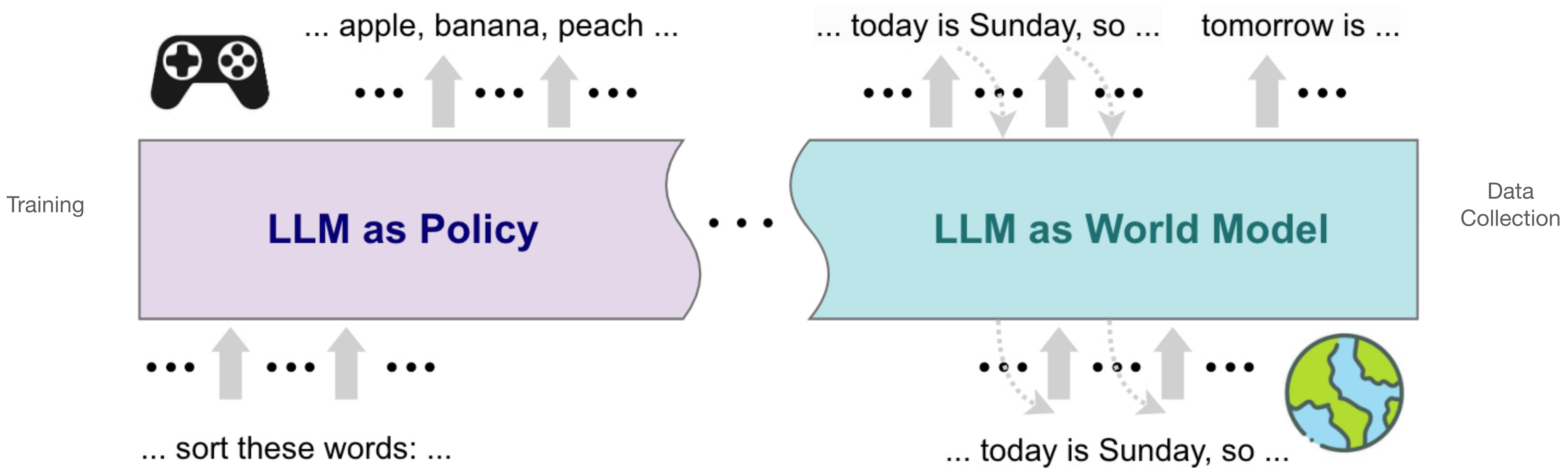


Figure 2. **Illustration of Large Language Model (LLM).** HIR views LLM as both a policy and a world model. Thus, HIR can collect data through interactions with LLM in the online sampling phase, and further improve the policy in the offline learning phase.

HIR

Algorithm 1 Two-Stage Hindsight Instruction Relabeling (HIR)

Input: Language Model $\mathcal{M}$, Initial Prompt $\mathbf{p}$, Training Set $\mathcal{D}_{\text{train}}$, Evaluation set $\mathcal{D}_{\text{eval}}$, Iteration N, Sampling Rounds T, Training Epochs K, Sampling Temperature τ, Empty RL dataset $\mathcal{D}_{\text{online}}$

for episode $n = 1, \cdots, N$ **do**
    for sampling rounds $i = 1, \cdots, T$ **do**
        Random sample batch of input queries $\mathcal{Q} \sim \mathcal{D}_{\text{train}}$
        Sample corresponding outputs $\mathbf{o_i} = \mathcal{M}(\mathcal{Q}, \mathbf{p}, \tau)$
        Appending the trajectory to RL Dataset $\mathcal{D}_{\text{online}} \leftarrow \mathcal{D}_{\text{online}} \cup (\mathcal{Q}, \mathbf{p}, \mathbf{o}_i)$
    end for
    for training rounds $t = 1, \cdots, K$ **do**
        Random sample batch of query-output pairs $(\mathcal{Q}, \mathcal{O}) \sim \mathcal{D}_{\text{online}}$
        Sample from $\mathcal{D}_{\text{online}}$ and apply relabeling as described in Sec. 4.3
        Train model $\mathcal{M}$ using loss in Eq. (6)
    end for
end for
Evaluate policy π_θ on evaluation dataset $\mathcal{D}_{\text{eval}}$

HIR

More Tricks

- [(Answer the following question, what is the capital of Pennsylvania?, Harrisburg) + (Answer the following question **incorrectly**, what is the capital of Pennsylvania?, Pittsburgh),]
- $P(Pittsburgh \mid Incorrect)$ = exp prob(Pittsburgh | Answer the following question **incorrectly**, what is the capital of Pennsylvania?)
- $P(Pittsburgh \mid Correct)$ = exp prob(Pittsburgh | Answer the following question, what is the capital of Pennsylvania?)
  - $-\log \frac{P(Pittsburgh \mid Incorrect)}{P(Pittsburgh \mid Incorrect) + P(Pittsburgh \mid Correct)}$
- Contrastive loss to push down specific outputs for other instructions and avoid generating the same output for different instructions:
  - Encourage associating of instruction - output
  - - log P (Pittsburgh | incorrect)

Table 1. Examples of inputs and outputs for the BigBench tasks. For multiple-choice tasks, we provide the options that the language model can choose from as prompts.

	Tasks	Example Inputs	Outputs
Multiple Choice	Logical Deduction	"Q: The following paragraphs each describe a set of three objects arranged in a fixed order. The statements are logically consistent within each paragraph. In a golf tournament, there were three golfers: Amy, Eli, and Eve. Eve finished above Amy. Eli finished below Amy. Options: (A) Amy finished last (B) Eli finished last (C) Eve finished last"	"(B)"
	Date Understanding	"Q: Today is Christmas Eve of 1937. What is the date 10 days ago? Options: (A) 12/14/2026 (B) 12/14/2007 (C) 12/14/1937"	"(C)"
Direct Generation	Object Counting	"Q: I have a blackberry, a clarinet, a nectarine, a plum, a strawberry, a banana, a flute, an orange, and a violin. How many fruits do I have?"	"6"
	Word Sorting	"Sort the following words alphabetically: List: oven costume counterpart."	"costume counterpart oven"

Results

		Tracking Shuffled Objects (3)	Tracking Shuffled Objects (5)	Tracking Shuffled Objects (7)	Logical Deduction (3 Objects)
No Training	FLAN-T5-large	29.3	15.6	6.6	33.3
Finetuning	Finetuning	**100.0**	17.0	13.4	90.0
RL Tuning	PPO	35.0	15.6	6.3	57.0
	FARL	90.0	15.6	10.0	86.7
	HIR (ours)	**100.0**	**61.2**	**42.6**	**91.7**

		Logical Deduction (5 Objects)	Logical Deduction (7 Objects)	Date Understading	Object Counting
No Training	FLAN-T5-large	44.0	49.3	35.1	31.0
Finetuning	Finetuning	61.0	**64.0**	96.0	**70.0**
RL Tuning	PPO	44.0	43.0	90.5	33.0
	FARL	54.0	60.0	98.0	56.7
	HIR (ours)	**67.0**	62.0	**98.0**	65.0

		Geometric Shapes	Penguins in A Table	Reasoning about Colored Objects	Word Sorting
No Training	FLAN-T5-large	9.7	46.7	20.0	1.1
Finetuning	Finetuning	90.0	53.0	**90.0**	**24.7**
RL Tuning	PPO	11.0	50.0	30.0	1.1
	FARL	66.7	**56.0**	77.0	3.4
	HIR (ours)	**90.3**	53.0	77.8	3.4

FARL: Only use positive labels

Options for when you cannot hire humans to tell good from bad

GENERATING SEQUENCES BY
LEARNING TO [SELF-]CORRECT

Sean Welleck[1,3,*] **Ximing Lu**[1,*]

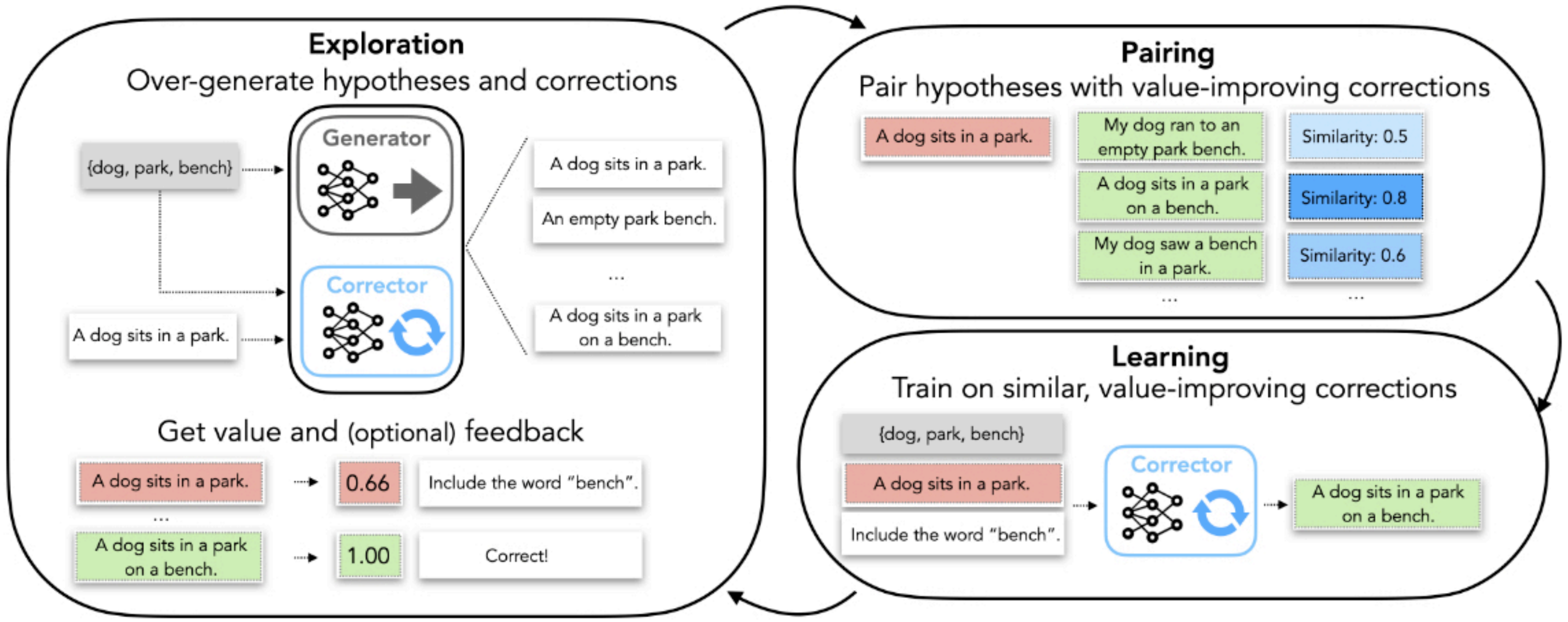
Peter West[3,†] **Faeze Brahman**[1,†]

Tianxiao Shen[3] **Daniel Khashabi**[2] **Yejin Choi**[1,3]
[1]Allen Institute for Artificial Intelligence
[2]Center for Language and Speech Processing, Johns Hopkins University
[3]Paul G. Allen School of Computer Science & Engineering, University of Washington

Key Idea

- Start with a base generator

- Generate two outputs:
  - A and B
  - If A is "correct" and B is "wrong", add A —> B as an example, train corrector

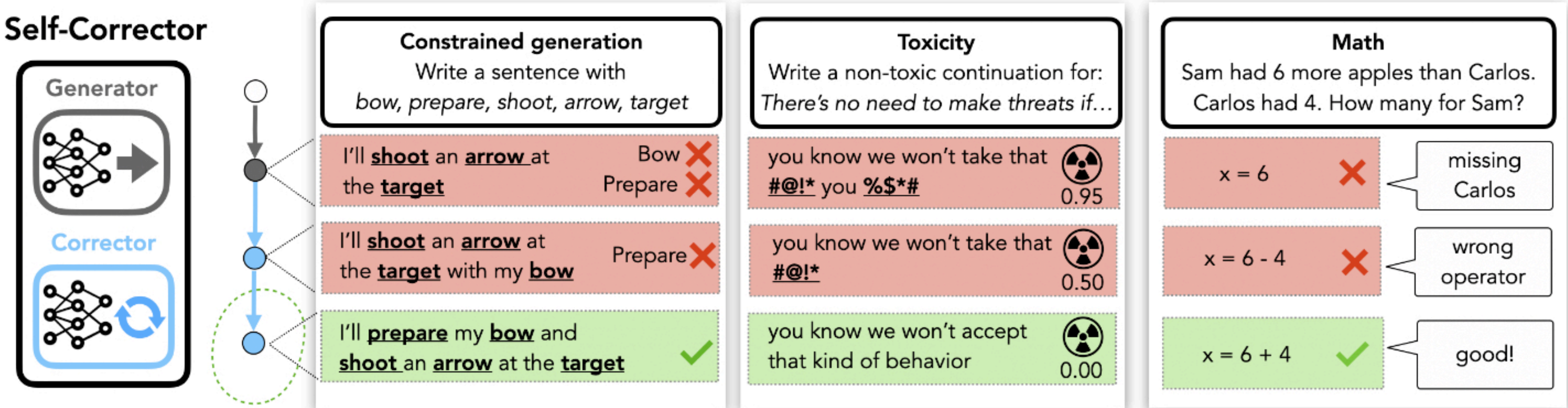


Training Self-Correctors

- Initialization
  - $$D_x = \{(x, y, v(y), f(y)) \mid \text{for all } y \in y^{1:N} \sim q(p_0(\cdot|x))\}, \quad D = \bigcup_{x \in X} D_x,$$
- Pairing
  - $$P_x = \{(x, y, y') \mid v(y) < v(y') \text{ for all } y, y' \in D_x \times D_x\}, \quad P = \bigcup_{x \in X} P_x,$$
- Learning
  - $$\mathbb{P}[(x, y, y')] \propto \exp\big(\underbrace{\alpha \cdot (v(y') - v(y))}_{\text{improvement}} + \underbrace{\beta \cdot s(y, y')}_{\text{proximity}}\big)/Z(y),$$
- Exploration $$D'_x = \{(x, y', v(y'), f(y')) \mid \text{for all } y' \in y'^{1:N} \sim q(p_\theta(\cdot|y, x, f(y))\}, \quad D' = \bigcup_{x \in X} D'_x$$
  - Include some examples from the current corrector

Inference

- Given some input
- Use generator to sample output
- Apply corrector k times (the output may be right after the first go, there is no way to know).



Experiments

Math Reasoning

- Smaller model as a corrector (GPT-Neo 1.3B)
- Generator:
    - Either the same model or GPT-3
    -

Dataset	Model	Params	Correct
GSM	*OpenAI 3B* [5]	3B	15.50
	OpenAI 6B [5]	6B	20.00
	GPT-NEO [28]	2.7B	18.80
	NEO FCP+PCP [28]	2.7B	19.50
	GPT-NEO	1.3B	8.57
	+SELF-CORRECT	1.3B	**21.26**
	+SELF-CORRECT$_*$	1.3B	**24.22**

Problem:
Mrs. Wilsborough saved $500 to buy concert tickets for her family. She bought 2 VIP tickets at $100 each and 3 regular tickets at $50 each. How much of her savings does Mrs. Wilsborough have after she buys the tickets?

Generator:

```
a=2*100
b=3*50
c=a+b
answer=c
print(answer)
```

Corrector:

```
a=2*100
b=3*50
c=500-a-b #fix
answer=c
print(answer)
```

Experiments

Toxicity Reduction

- Given a prompt x, the task is to generate a fluent continuation y while avoiding offensive content.
- Off-the-shelf GPT-2 Large as the generator, and finetune another GPT-2 Large as the corrector.
- As the value function, use the Perspective API score, $v(y) \in [0, 1]$, which measures the toxicity of the completed sequence.

	Toxicity		Fluency	Diversity	
	Avg. Max.	**Prob.**	**Perplexity**	**dist-2**	**dist-3**
GPT-2	0.527	0.520	11.31	0.85	0.85
PPLM [6]	0.520	0.518	32.58	0.86	0.86
GeDi [14]	0.363	0.217	43.44	0.84	0.83
DExpert [21]	0.314	0.128	25.21	0.84	0.84
DAPT [12]	0.428	0.360	31.22	0.84	0.84
PPO [23]	0.218	0.044	14.27	0.79	0.82
Quark [23]	0.196	0.035	12.47	0.80	0.84
SELF-CORRECT	**0.171**	**0.026**	**11.81**	0.80	0.83

Table 3: **Toxicity reduction.** GPT-2 is the base generator.

Experiments

Swapping Generators

- Train corrector using generations from a smaller model
- Use the corrector to improve larger models

Task	Dataset	Generator (train)	Generator (test)	Generator	Self-corrector
Math Synthesis ↑	Multitask	Neo 1.3B	GPT-3	46.70	80.00
		Neo 1.3B	GPT-3 Instruct	84.90	90.90
		GPT-3 Instruct	GPT-3 Instruct	84.90	92.75
	GSM	Neo 1.3B	GPT-3	6.96	24.30
		Neo 1.3B	GPT-3 Instruct	36.80	45.00
		GPT-3 Instruct	GPT-3 Instruct	36.80	45.92
Detoxification ↓	RTPrompts	GPT2-L	GPT2-XL	0.383	0.027
		GPT2-L	GPT-3	0.182	0.025
		GPT2-L	GPT-3 Instruct	0.275	0.023

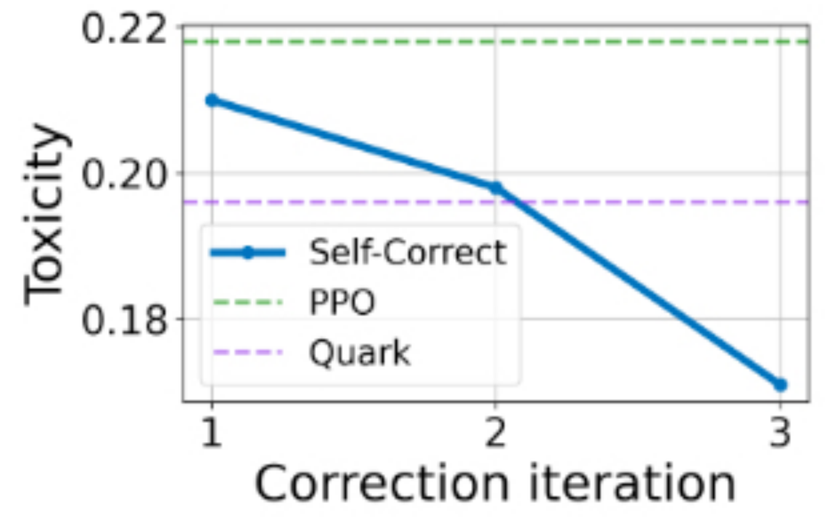


Figure 4: Applying multiple corrections reduces toxicity.

Outline of the talk

- Background
- RL + Human feedback
  - Fine-tuning LMs with Human Feedback
  - InstructGPT
- Recent works that include feedback without RL
  - Hindsight-tuning
  - Self-correct

Two Camps

- RL
  - Collect some human labels and fine-tune LMs
- ChatGPT / GPT-3 Families
- Claude by Anthropic

- Supervised
- Collect lots of training data and do good old supervised learning
- Flan-T5-XXL (best open source model)
- Large datasets for instruction tuning:
  - T0
  - Flan

Take aways

- RL + Large amounts of hand annotated data key to creating good models
- Another competitor (Claude by Anthropic) also uses PPO
- Growing body of work questioning the need for RL — perhaps our benchmarks are misguided
- It might be possible to simulate a human for feedback with a good enough model

---

Pretraining Language Models with Human Preferences

Tomasz Korbak[1,2,3] **Kejian Shi**[2] **Angelica Chen**[2] **Rasika Bhalerao**[4] **Christopher L. Buckley**[1] **Jason Phang**[2]
Samuel R. Bowman[2,5] **Ethan Perez**[2,3,5]