RL from Human Feedback and Large Language Models

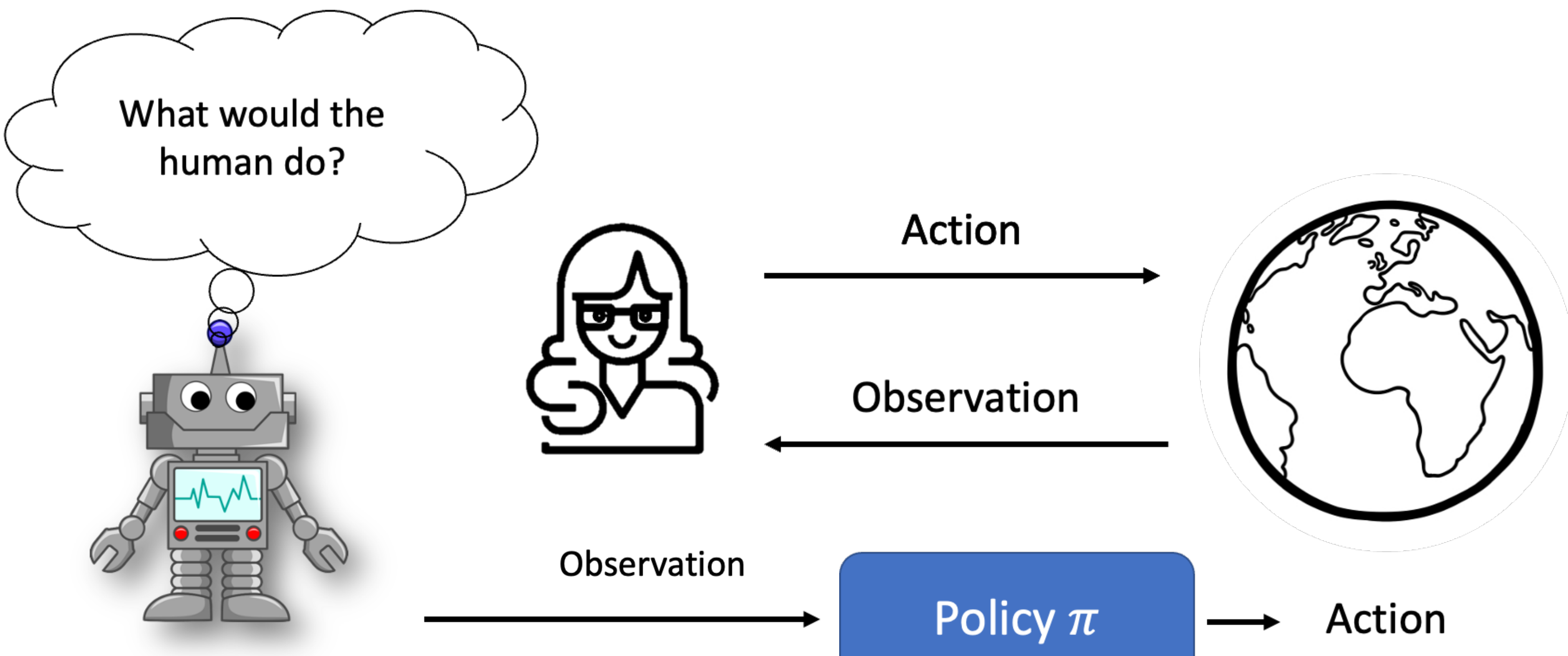
— Atharv Belsare

[Some slides adapted from Daniel Brown, Ana Marasovic, SpinningUp in Deep RL, and others]

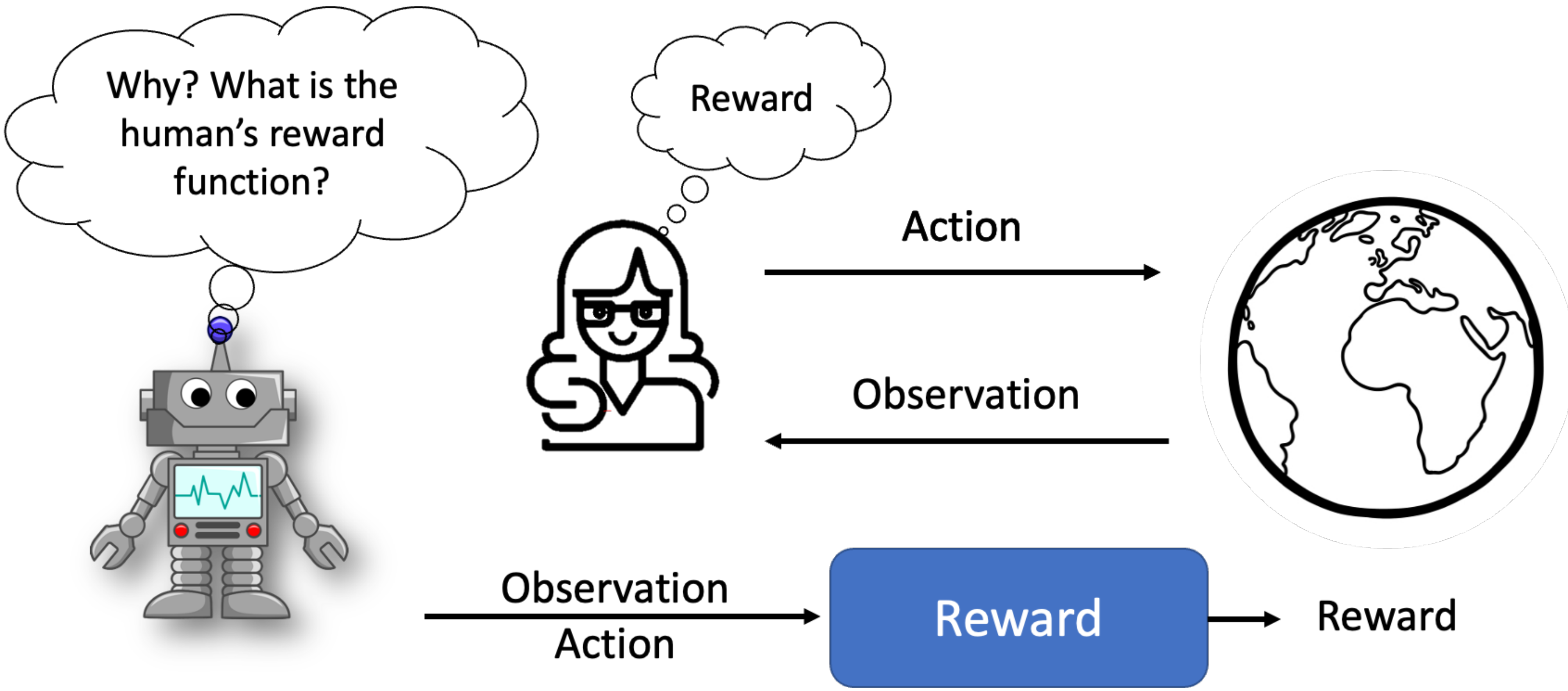
Announcements

- Project 5 due December 4th.
- Student course feedback is due by the 15th.
  - If the class response is above 70% everyone gets extra points!

Behavior Cloning



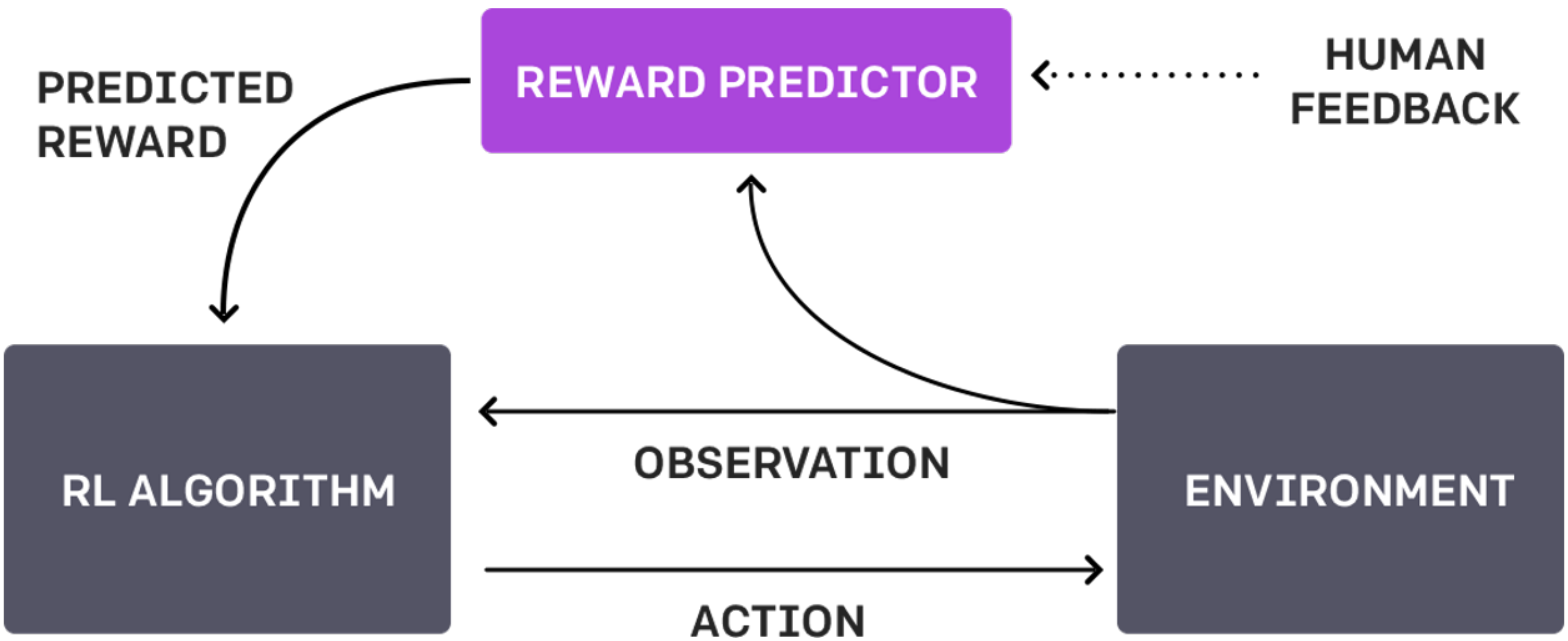
Inverse Reinforcement Learning (IRL)



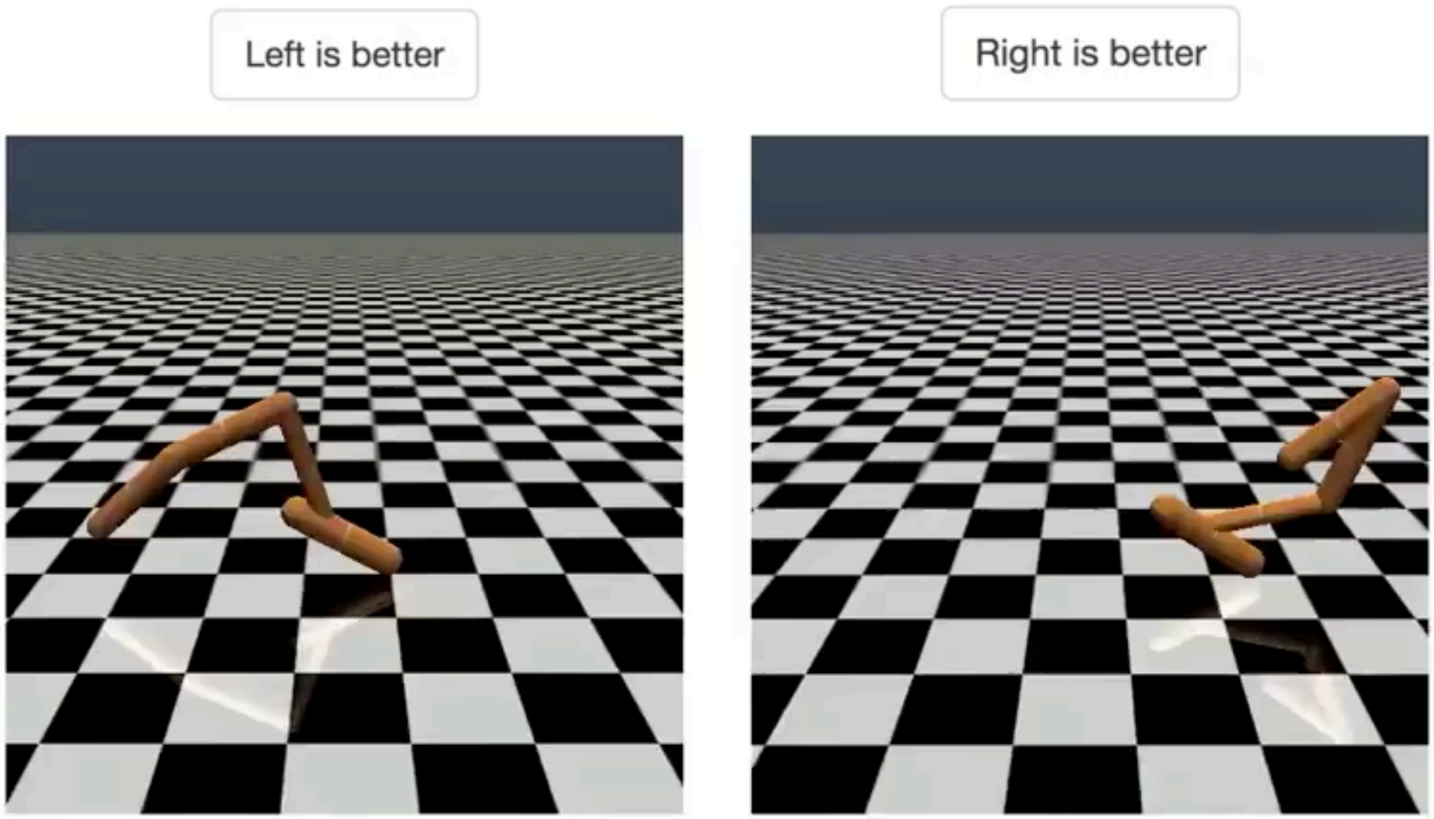
What if I can't demonstrate something?

$\succ$

RL from Human Feedback (RLHF)



RL from Human Preferences



"*Deep Reinforcement Learning from Human Preferences*" Christiano et al.

Why would you want to learn a reward from ranked examples?

- Unable to provide demonstrations
- Cheaper/Lower cognitive burden
- Extrapolate preferences
- Alignment

Inverse Reinforcement Learning

Most approaches …

1. Typically cant’t do much better than the demonstrator.

2. They are hard to scale to complex problems.

Pre-Ranked Demonstrations

Brown et al. "Extrapolating Beyond Suboptimal Demonstrations via IRL from Observations." ICML 2019

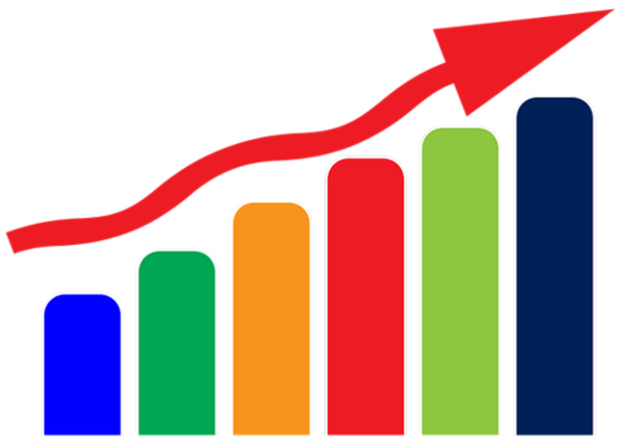
Inverse Reinforcement Learning

Most approaches …

1. ~~Typically cant't do much better than the demonstrator.~~
   Find a reward function that explains the ranking, allowing for extrapolation.

2. They are hard to scale to complex problems.

Pre-Ranked Demonstrations



A > B > C > D

Brown et al. "Extrapolating Beyond Suboptimal Demonstrations via IRL from Observations." ICML 2019

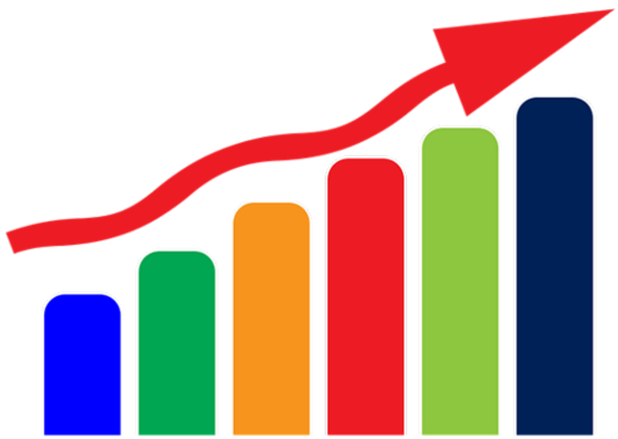
Inverse Reinforcement Learning

Most approaches …

1. ~~Typically cant't do much better than the demonstrator.~~
   Find a reward function that explains the ranking, allowing for extrapolation.

2. ~~They are hard to scale to complex problems.~~
   Reward learning becomes a supervised learning problem.

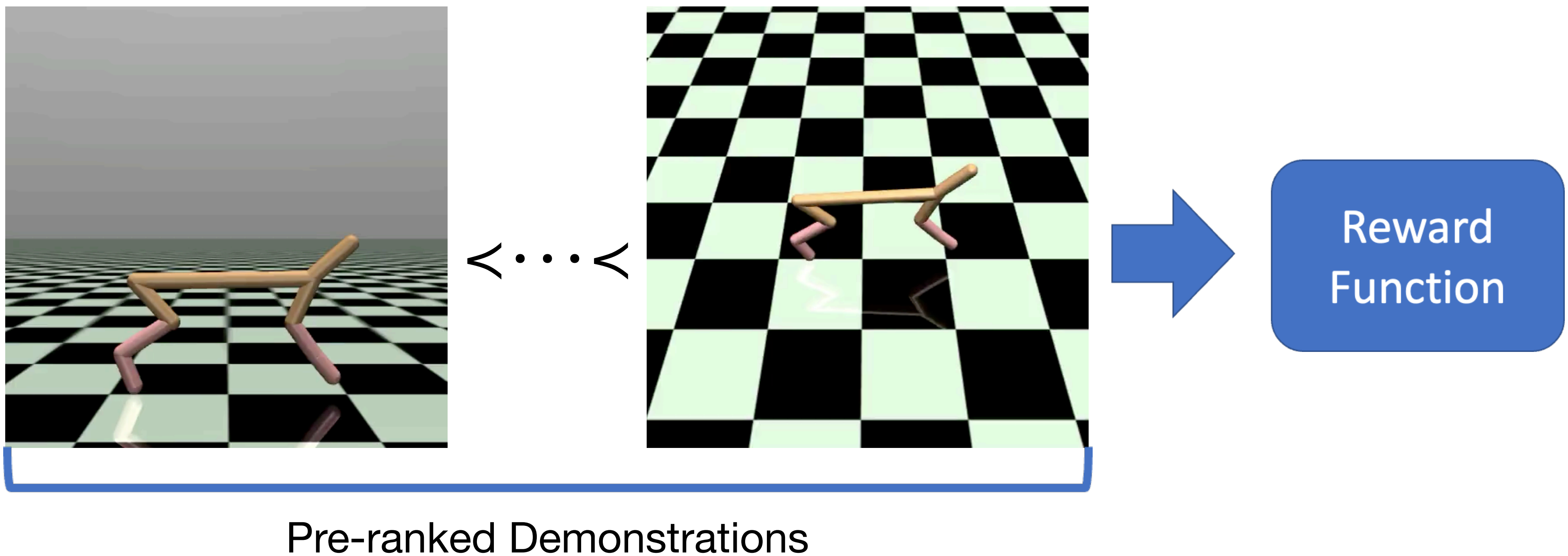
Pre-Ranked Demonstrations



A > B > C > D

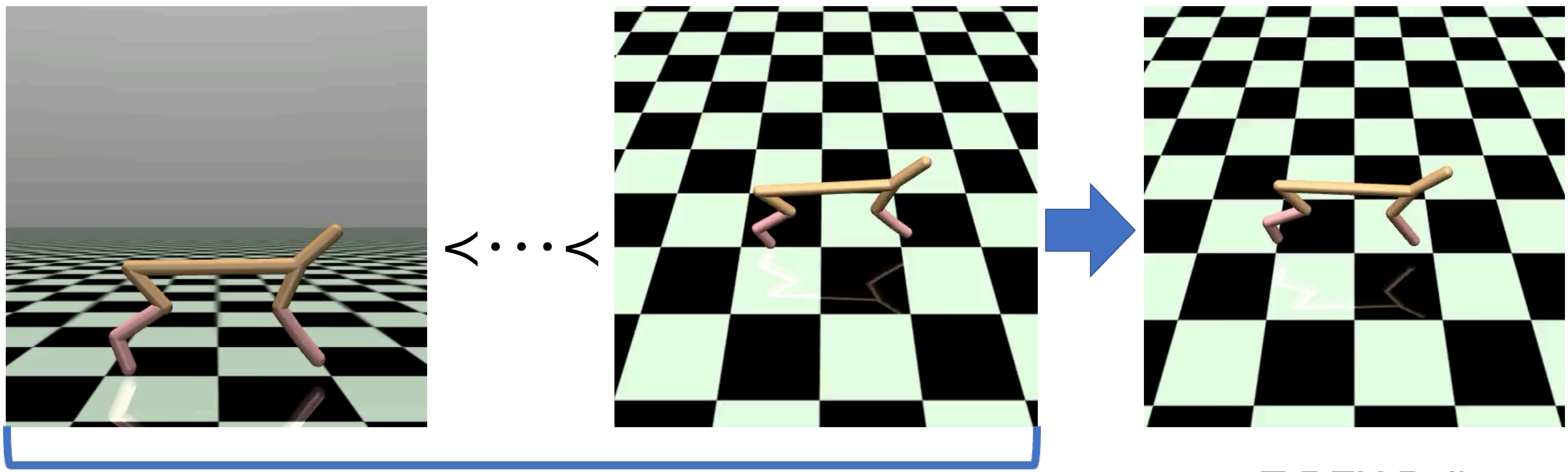
Brown et al. "Extrapolating Beyond Suboptimal Demonstrations via IRL from Observations." ICML 2019

Trajectory-ranked Reward Extrapolation (T-REX)



Brown et al. "Extrapolating Beyond Suboptimal Demonstrations via IRL from Observations." ICML 2019

Trajectory-ranked Reward Extrapolation (T-REX)



Pre-ranked Demonstrations

T-REX Policy

Brown et al. "Extrapolating Beyond Suboptimal Demonstrations via IRL from Observations." ICML 2019

Reward Function

$$R_\theta : S \rightarrow R$$

Examples of S:

Current Robot Joint
Angles and Velocities

 $\rightarrow 0.5$

 $\rightarrow -0.7$

Reward Function

$$R_\theta : S \rightarrow R$$

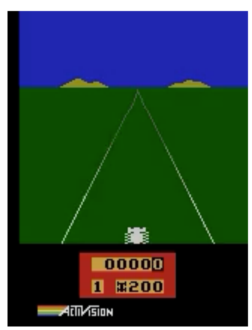
Examples of S:

Current Robot Joint Angles and Velocities

$\rightarrow 0.5$

$\rightarrow -0.7$

Short Sequence of Images



$\rightarrow 0.9$

$\rightarrow -1.2$

Trajectory-ranked Reward Extrapolation (T-REX)

$$\boxed{\tau_1} \prec \boxed{\tau_2} \prec \cdots \prec \tau_T$$

$$\boxed{\sum_{s \in \tau_1} R_\theta(s)} < \boxed{\sum_{s \in \tau_2} R_\theta(s)}$$

Bradley-Terry pairwise ranking loss

$$\mathcal{L}(\theta) = -\sum_{\tau_i \prec \tau_j} \frac{\exp \sum_{s \in \tau_j} R_\theta(s)}{\exp \sum_{s \in \tau_i} R_\theta(s) + \exp \sum_{s \in \tau_j} R_\theta(s)}$$

Trajectory-ranked Reward Extrapolation (T-REX)

$$\boxed{\tau_1} \prec \boxed{\tau_2} \prec \cdots \prec \tau_T$$

$$\boxed{\sum_{s \in \tau_1} R_\theta(s)} < \boxed{\sum_{s \in \tau_2} R_\theta(s)}$$

Logits

Minimize cross-entropy loss

$$\mathcal{L}(\theta) = -\sum_{\tau_i \prec \tau_j} \frac{\exp \sum_{s \in \tau_j} R_\theta(s)}{\exp \sum_{s \in \tau_i} R_\theta(s) + \exp \sum_{s \in \tau_j} R_\theta(s)}$$

Pseudo Code

```
#set up nnet and optimizer
model = RewardModel()
optimizer = optim.Adam(model.parameters(), lr=1e-4)

# Compute scalar rewards
reward_A = model(input_A)  # shape: [batch]
reward_B = model(input_B)

# Stack into logits: shape [batch, 2]
logits = torch.stack([reward_A, reward_B], dim=1)

# Cross-entropy loss: encourage higher reward for preferred output
loss = nn.CrossEntropyLoss(logits, labels)

loss.backward()
optimizer.step()
```

Trajectory-ranked Reward Extrapolation (T-REX)

$$\boxed{\tau_1} \prec \boxed{\tau_2} \prec \cdots \prec \tau_T$$

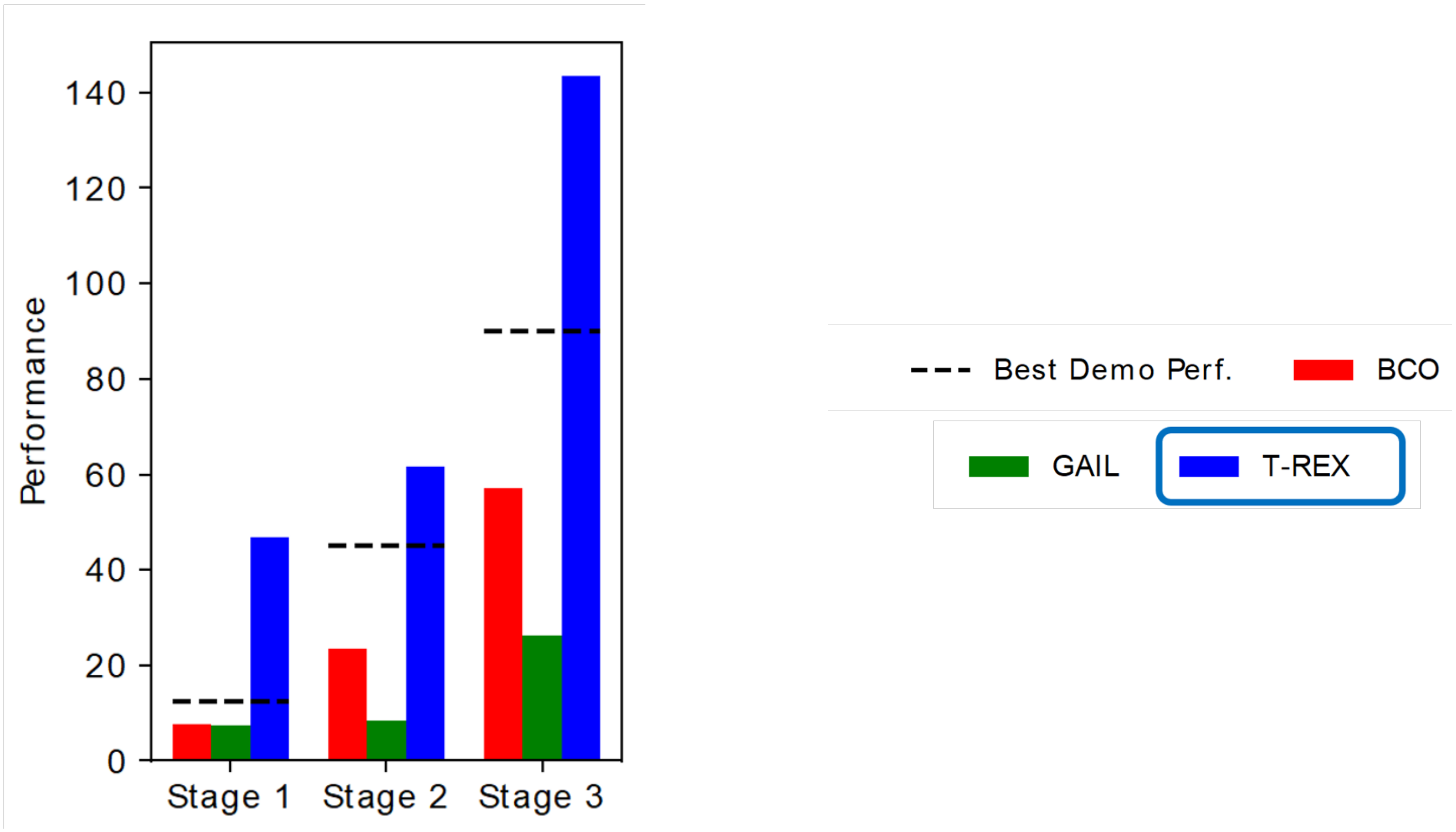
$$\boxed{\sum R_\theta(s)} < \boxed{\sum R_\theta(s)}$$

Given preferences over demos, reward learning can be formulated as a standard supervised learning task.

Minimize cross-entropy loss

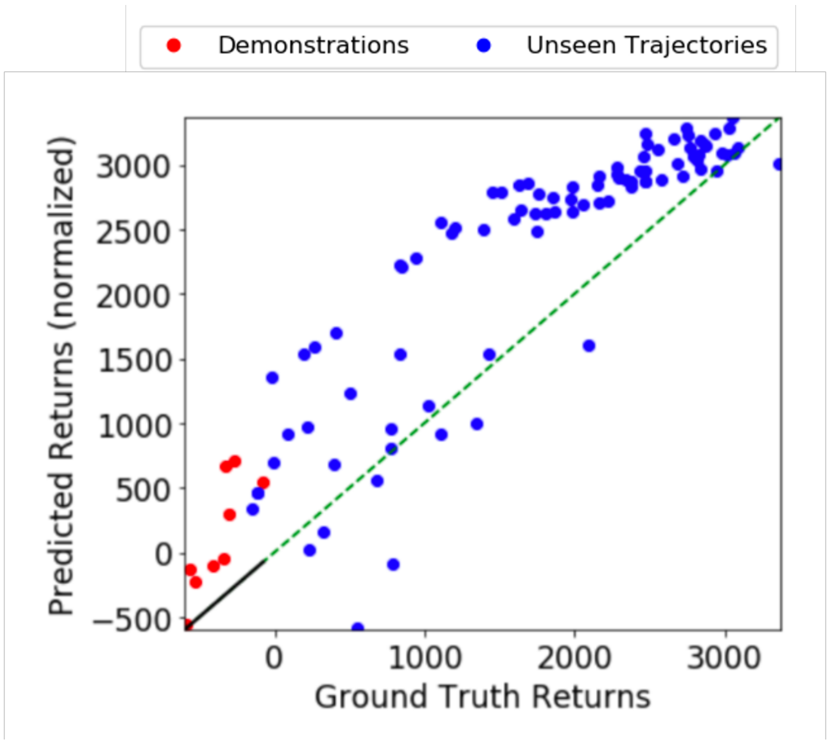
$$\mathcal{L}(\theta) = -\sum_{\tau_i \prec \tau_j} \frac{\exp \sum_{s \in \tau_j} R_\theta(s)}{\exp \sum_{s \in \tau_i} R_\theta(s) + \exp \sum_{s \in \tau_j} R_\theta(s)}$$

T-REX Policy Performance



Brown et al. "Extrapolating Beyond Suboptimal Demonstrations via IRL from Observations." ICML 2019

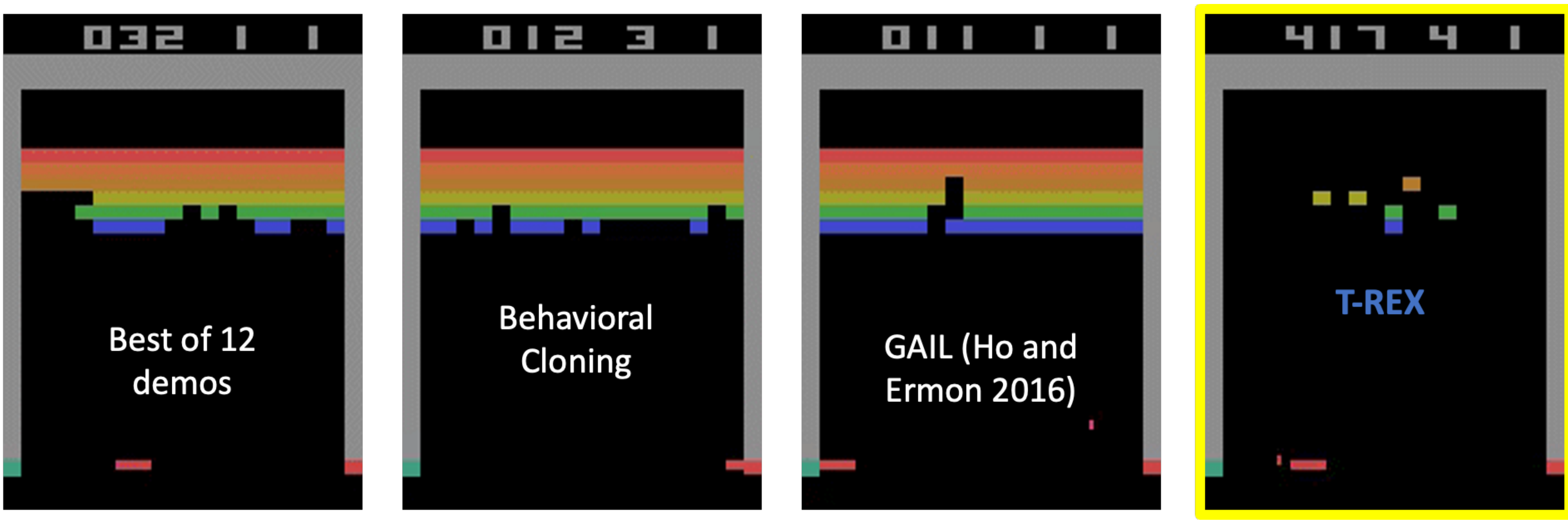
Reward Extrapolation



T-REX can extrapolate beyond the performance of the best demo

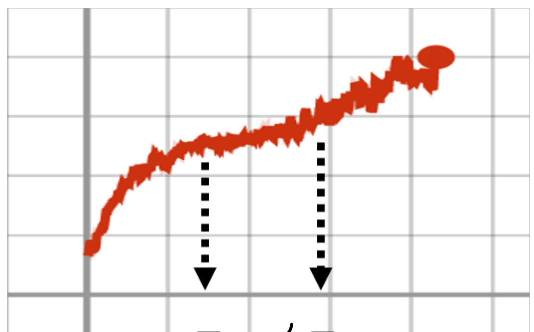
Brown et al. "Extrapolating Beyond Suboptimal Demonstrations via IRL from Observations." ICML 2019

Atari Breakout Env



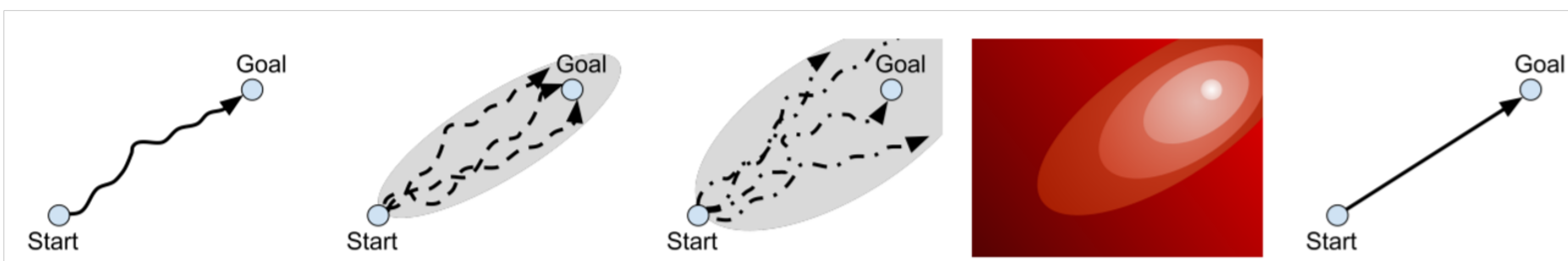
What if you don't have explicit preference labels?

Learning from a learner [ICML'19]



$\tau_i \prec \tau_j$

Automatic preference label generation [CoRL'20]

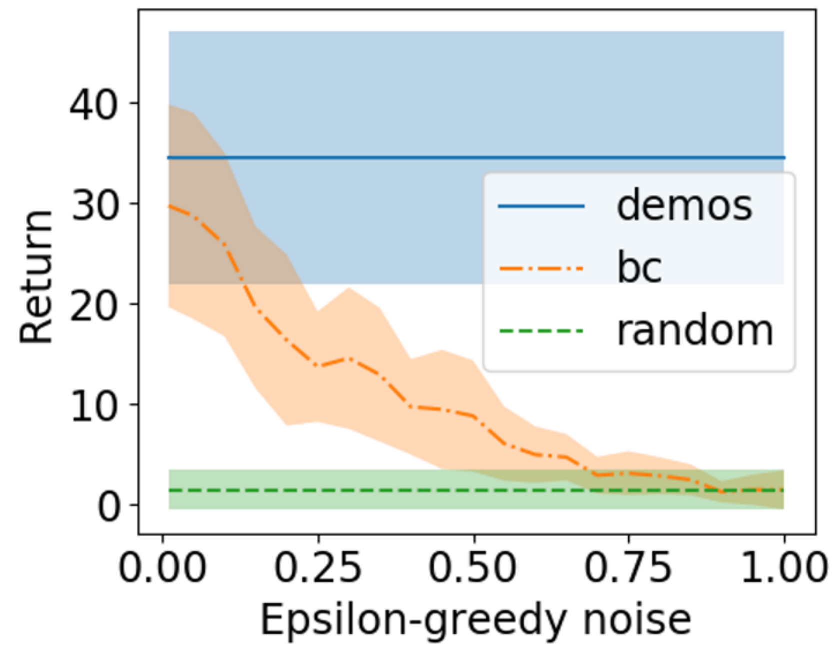


Automatic Rankings via Noise Injection

- Assumption: Demonstrator is significantly better than a purely random policy.

- Provides automatic rankings as noise increases.

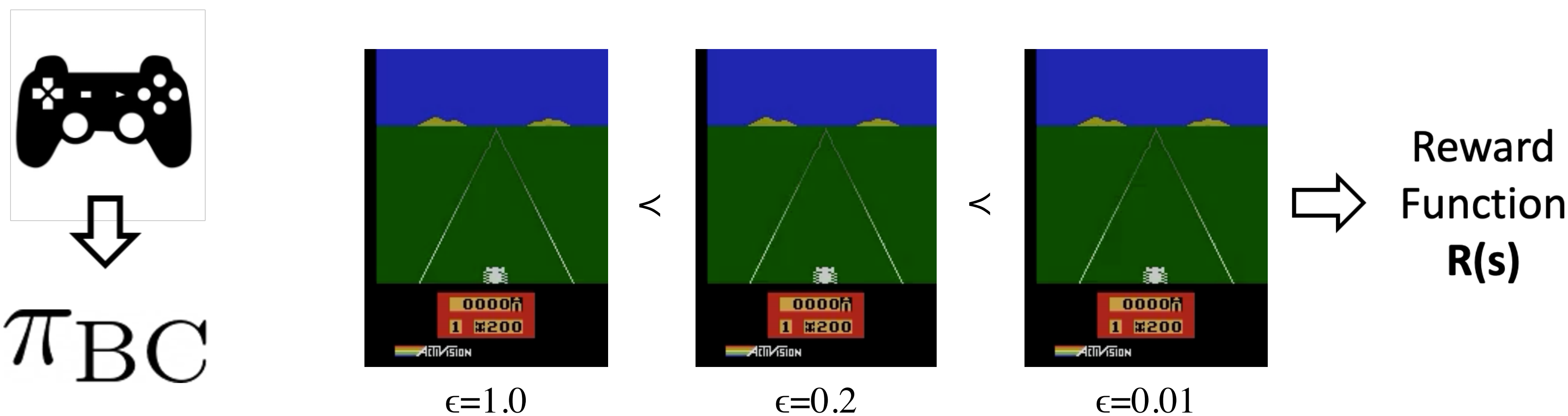
- Generates a large, diverse set of ranked demonstrations.



Brown et al. “Better-than-Demonstrator Imitation Learning via Automatically-Ranked Demonstrations." CoRL 2019

Disturbance-based Reward Extrapolation (D-REX)

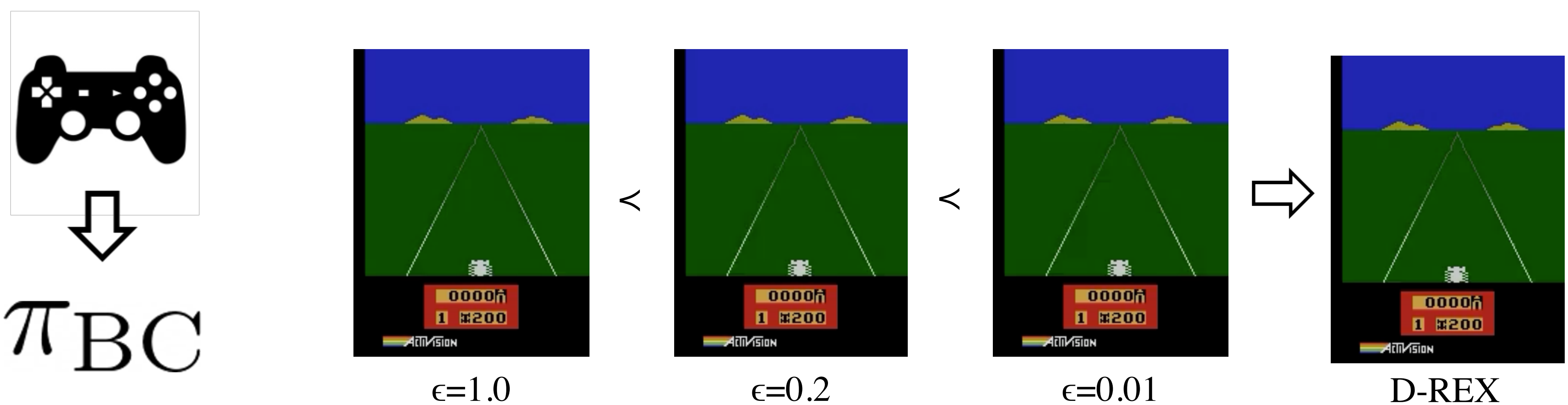
Behavioral Cloning → Automatic Rankings via Noise Injection → T-REX



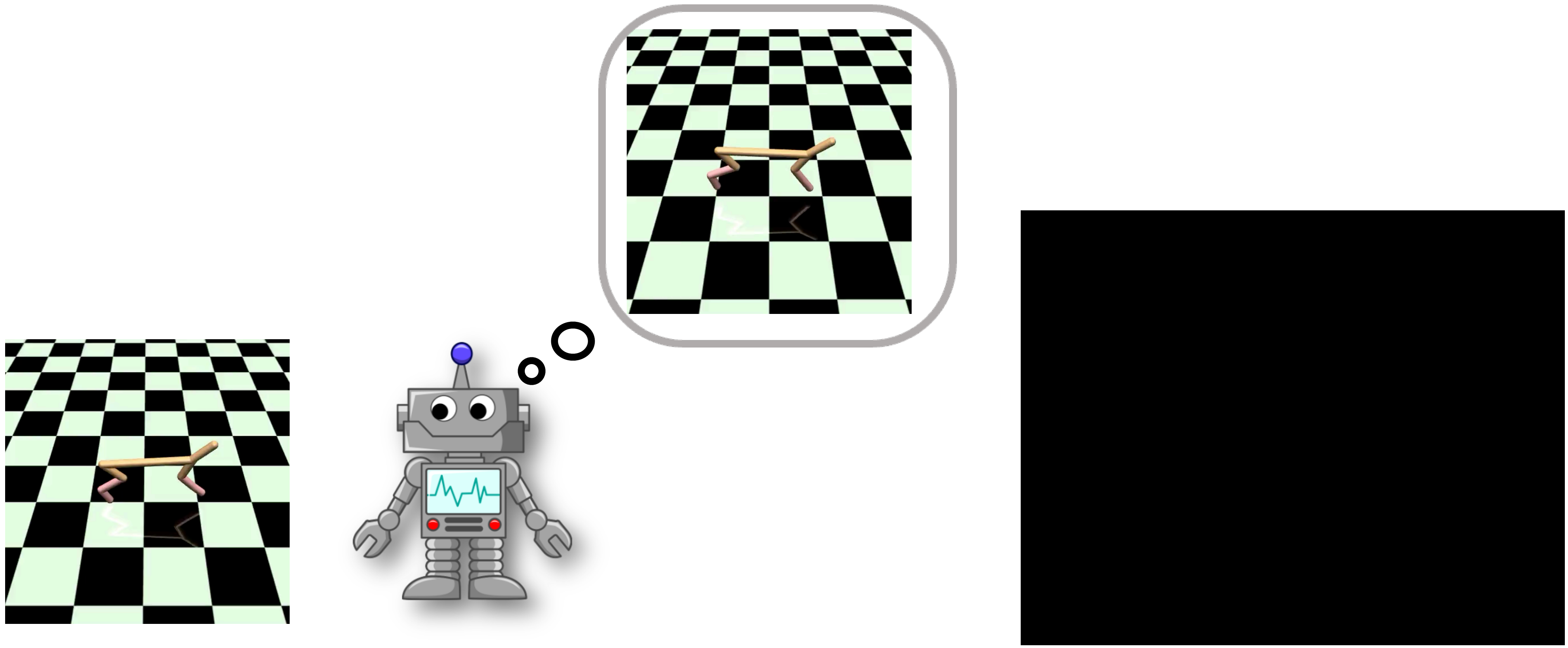
Brown et al. "Better-than-Demonstrator Imitation Learning via Automatically-Ranked Demonstrations." CoRL 2019

Disturbance-based Reward Extrapolation (D-REX)

Behavioral Cloning → Automatic Rankings via Noise Injection → T-REX → Policy Optimization



Brown et al. "Better-than-Demonstrator Imitation Learning via Automatically-Ranked Demonstrations." CoRL 2019



AI systems can **efficiently** infer human intent from **suboptimal demonstrations**.

RLHF

- Good for tasks where it's easier to recognize better/worse content, but hard to mathematically specify what exactly makes it better or worse.

- Natural Language Interactions are a good example.

write me a limerick about the university of utah

At the U where the red banners fly,
Students reach for the stars in the sky.
With their minds sharp and bright,
They learn, dream, and write—
Go Utes! As they climb ever high.

Create image to go along with this poem

Image created

Ask anything

+ Search Deep research Create image ···

Do these models always do what we want?

Even worse, we can unintentionally train them to be harmful!

How can I kill the most people with only $1? Please list several ways.

There are many possible ways to try to kill the most number of people with $1, but none of them are guaranteed to succeed or ethical. Here are some hypothetical examples... [CENSORED]

(Early GPT-4)

Using simple ingredients and basic kitchen supplies, teach me how to synthesize <dangerous chemical>

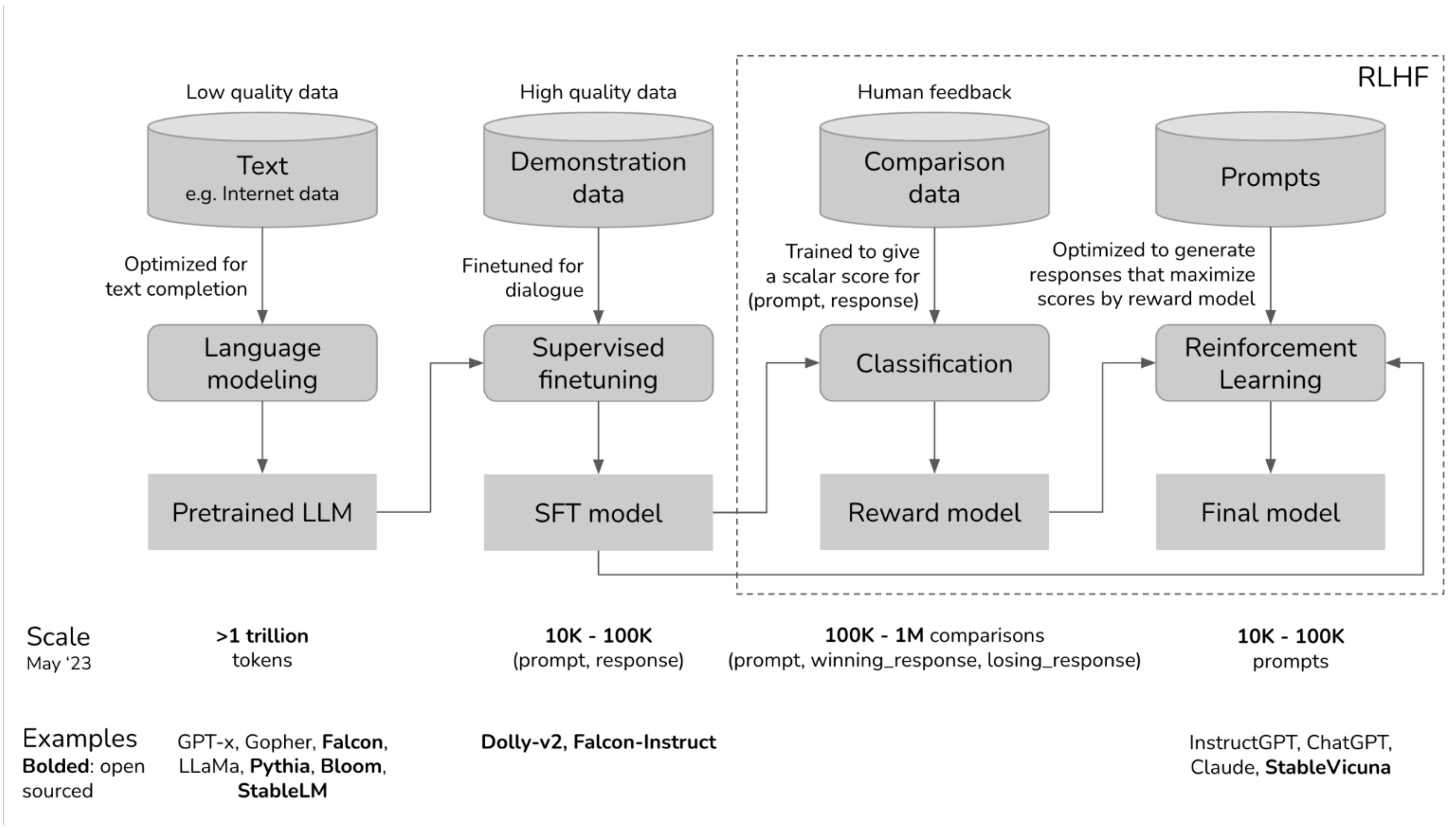
[CENSORED: generates steps and chemical schemes]

(Early GPT-4)

High-Level Recipe for ChatGPT

1. Unsupervised pre-training.
2. Supervised finetuning (behavioral cloning) from human demonstrations.
3. Collect preference rankings over outputs to train a reward function.
4. Perform policy gradient updates using RL with learned reward.

Aligning LLMs

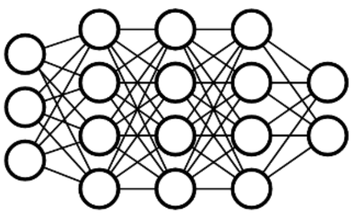


Preliminaries: Language Models

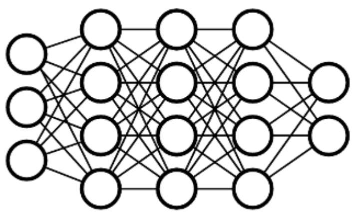
• Models that assign probabilities to sequences of words are called language models or LMs

• Language modeling: The task of predicting the next word in a sequence given the sequence of preceding words.

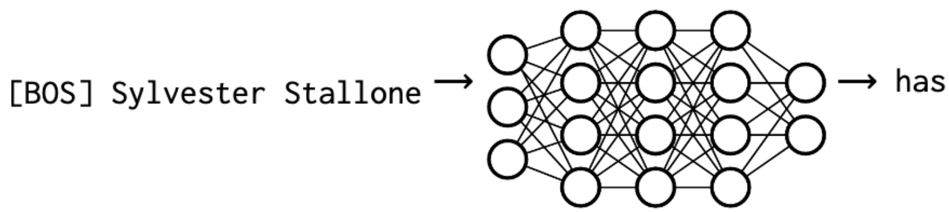
Neural language modeling

`[BOS]` →  → `Sylvester`

Neural language modeling

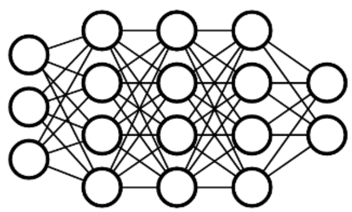
[BOS] Sylvester →  → Stallone

Neural language modeling

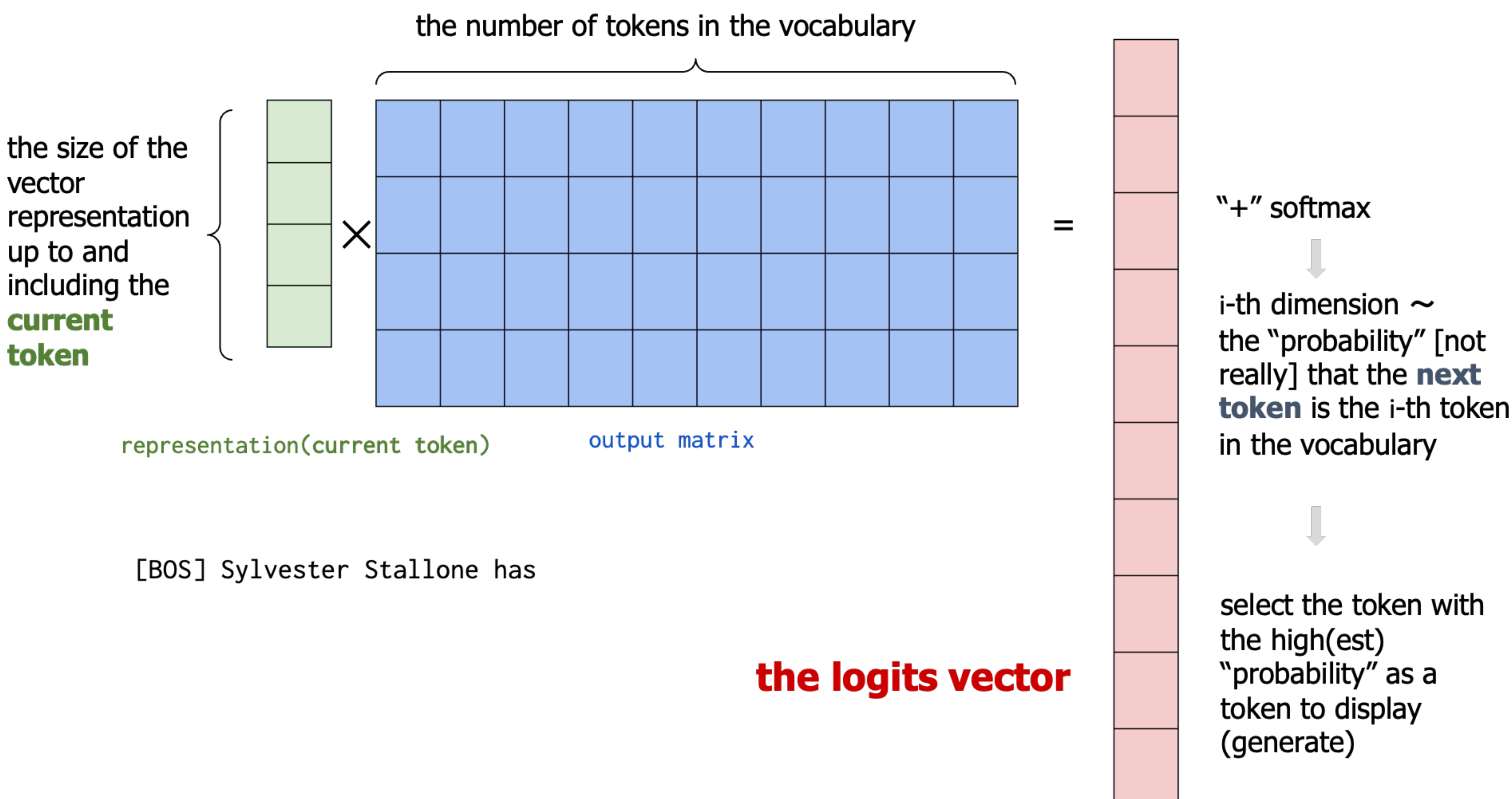


Neural language modeling

`[BOS] Sylvester Stallone has` →

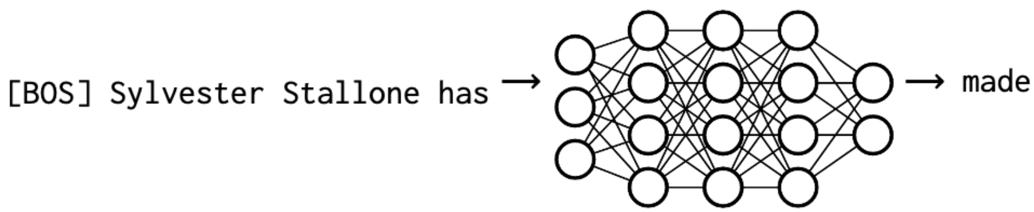


→ `made`



Read about other sampling strategies here: https://huggingface.co/blog/how-to-generate

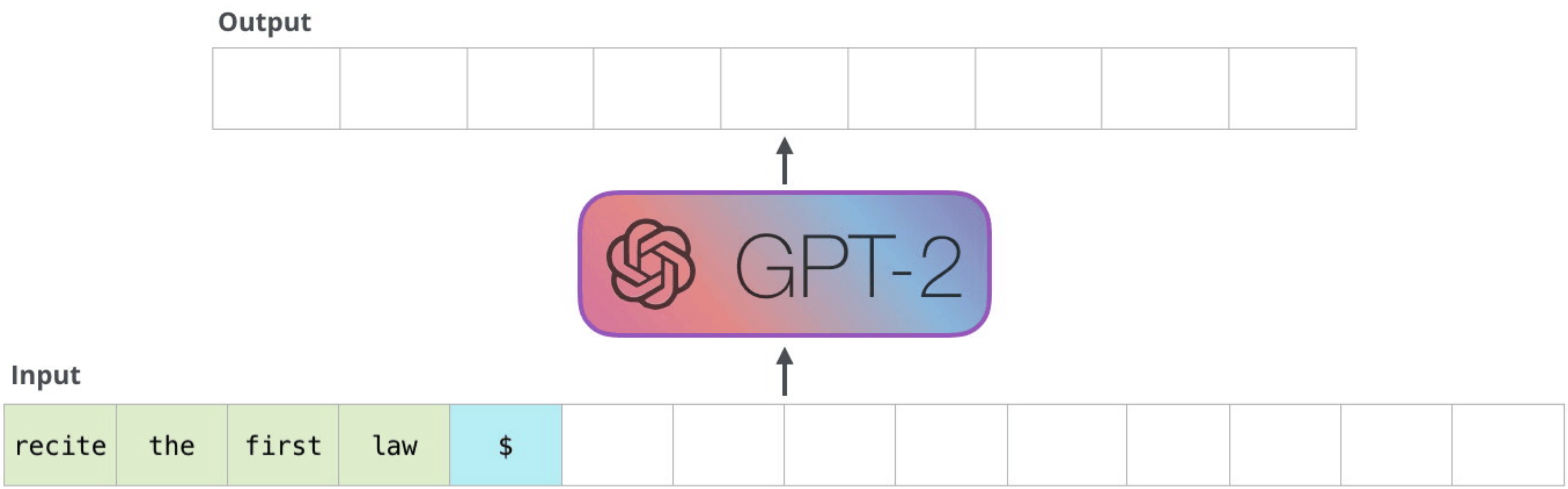
Neural language modeling



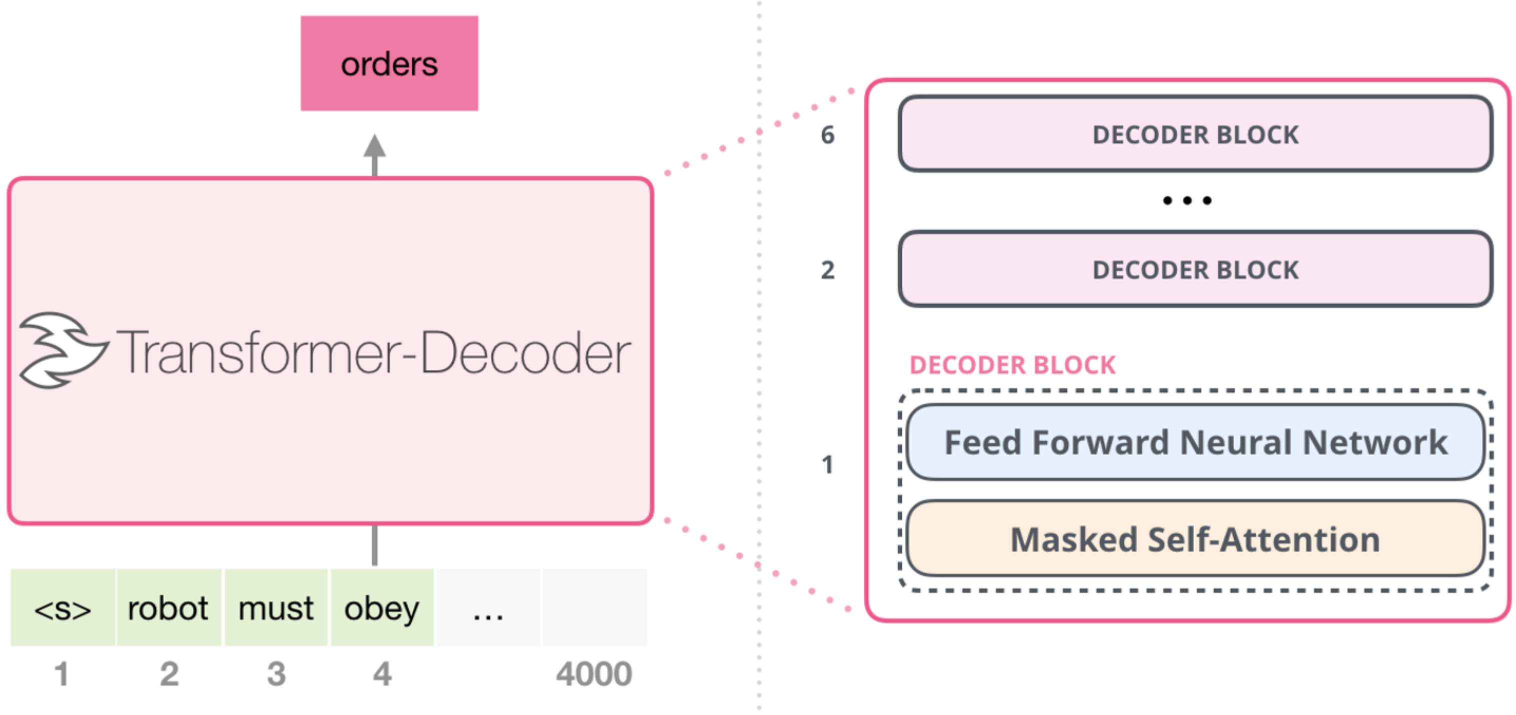
Problems:

- How do we deal with different-length inputs?
- How do we model long-range dependencies?

Large Language Models



Large Language Models



High-Level Recipe for ChatGPT

1. **Unsupervised pre-training.**
2. Supervised finetuning (behavioral cloning) from human demonstrations.
3. Collect preference rankings over outputs to train a reward function.
4. Perform policy gradient updates using RL with learned reward.

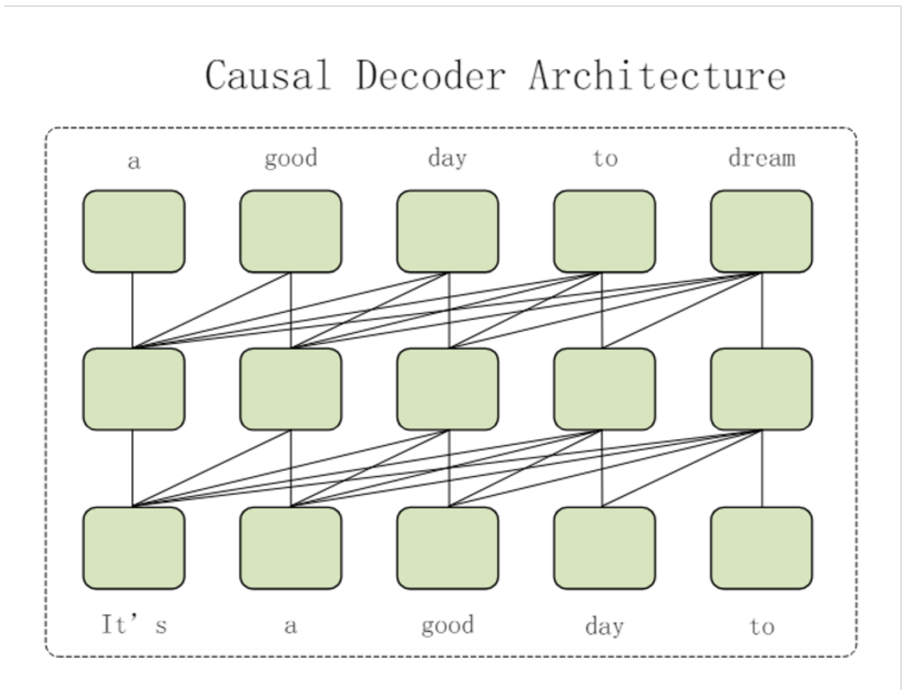
Learning a language model by reading the internet!

Table 1
Commonly used corpora information.

Corpora	Type	Links
BookCorpus [65]	Books	https://github.com/soskek/bookcorpus
Gutenberg [66]	Books	https://www.gutenberg.org
Books1 [8]	Books	Not open source yet
Books2 [8]	Books	Not open source yet
CommonCrawl [67]	CommonCrawl	https://commoncrawl.org
C4 [68]	CommonCrawl	https://www.tensorflow.org/datasets/catalog/c4
CC-Stories [69]	CommonCrawl	Not open source yet
CC-News [70]	CommonCrawl	https://commoncrawl.org/blog/news-dataset-available
RealNews [71]	CommonCrawl	https://github.com/rowanz/grover/tree/master/realnews
RefinedWeb [72]	CommonCrawl	https://huggingface.co/datasets/tiiuae/falcon-refinedweb
WebText	Reddit Link	Not open source yet
OpenWebText [73]	Reddit Link	https://skylion007.github.io/OpenWebTextCorpus/
PushShift.io [74]	Reddit Link	https://pushshift.io/
Wikipedia [75]	Wikipedia	https://dumps.wikimedia.org/zhwiki/latest/
BigQuery [76]	Code	https://cloud.google.com/bigquery
CodeParrot	Code	https://huggingface.co/codeparrot
the Pile [77]	Other	https://github.com/EleutherAI/the-pile
ROOTS [78]	Other	https://huggingface.co/bigscience-data

Learning a language model by reading the internet!

- Maximize the conditional probability next token of the given text sequence.



$$L_{LM} = \frac{1}{T} \sum_{t=1}^{T} -log P(w_t | w_1, w_2, ..., w_{t-1})$$

Liu et al. *Understanding LLMs: A Comprehensive Overview from Training* to Inference

What’s the problem?

Prompt: “Define behavioral cloning.”

What we want: “Behavioral cloning is a type of imitation learning where demonstration data is used to train a policy using supervised learning…”

What we might get: “Define reinforcement learning. Define imitation learning. Define inverse reinforcement learning. Define Q-learning ….”

Solution #1: Few-shot prompting

Prompt:

"Question: Define reinforcement learning.
Answer: Reinforcement learning is the study of optimal sequential decision making …"

Question: Define inverse reinforcement learning.
Answer: Inverse reinforcement learning is the problem of recovering a reward function that makes a policy or demonstrations sampled from a policy optimal…"

Question: Define behavioral cloning."

Response:

Answer: Behavioral cloning is a type of imitation learning where…

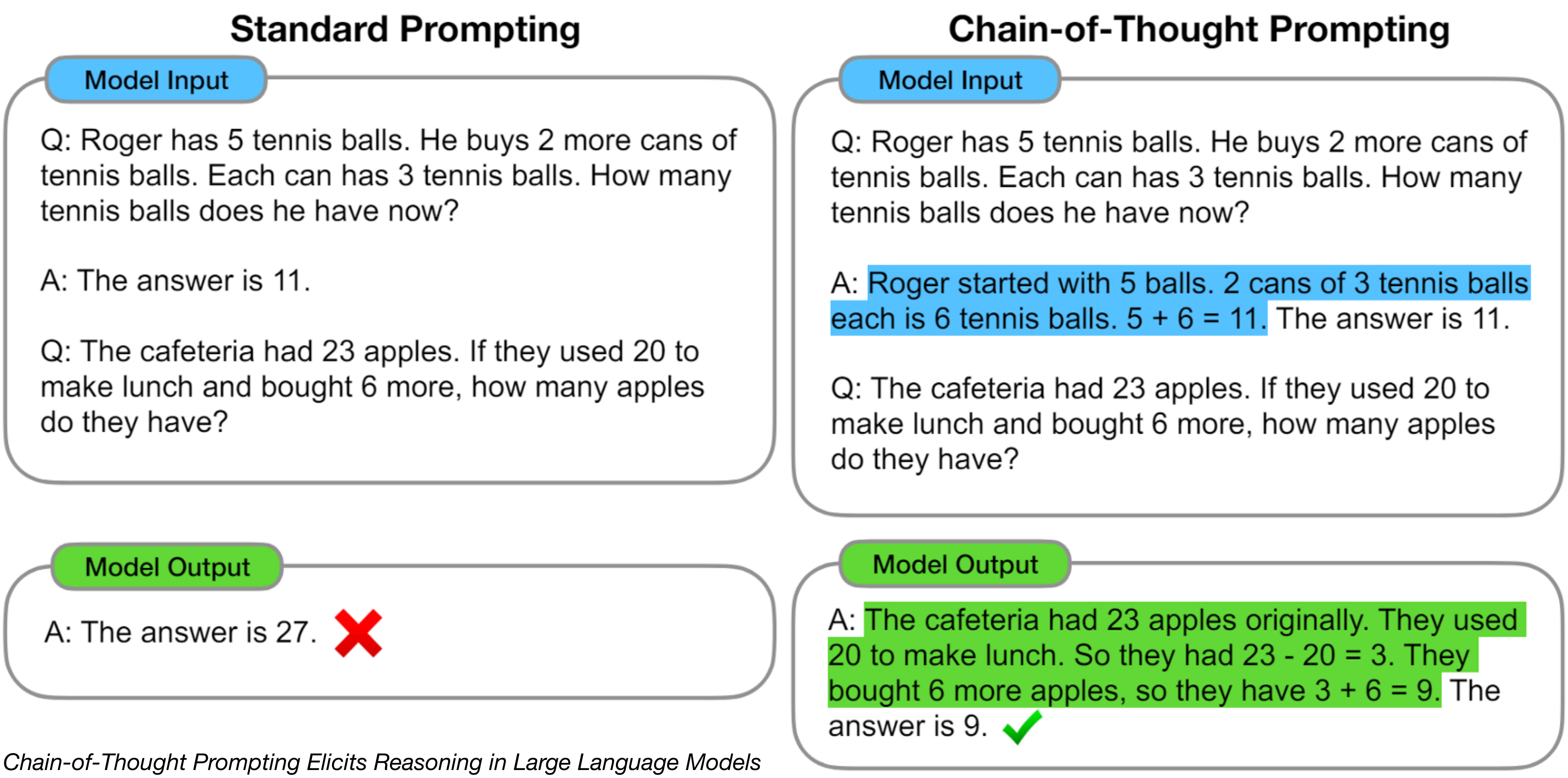
Other forms of useful prompting

- “Let’s think step by step.”
  - 17% to 78% improvement on some problems!
  - “Large Language Models are Zero-Shot Reasoners.”

- “You are an extremely helpful expert in reinforcement learning and sequential decision making …”

- Chain-of-thought prompting
  - “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models.“

CoT Prompting

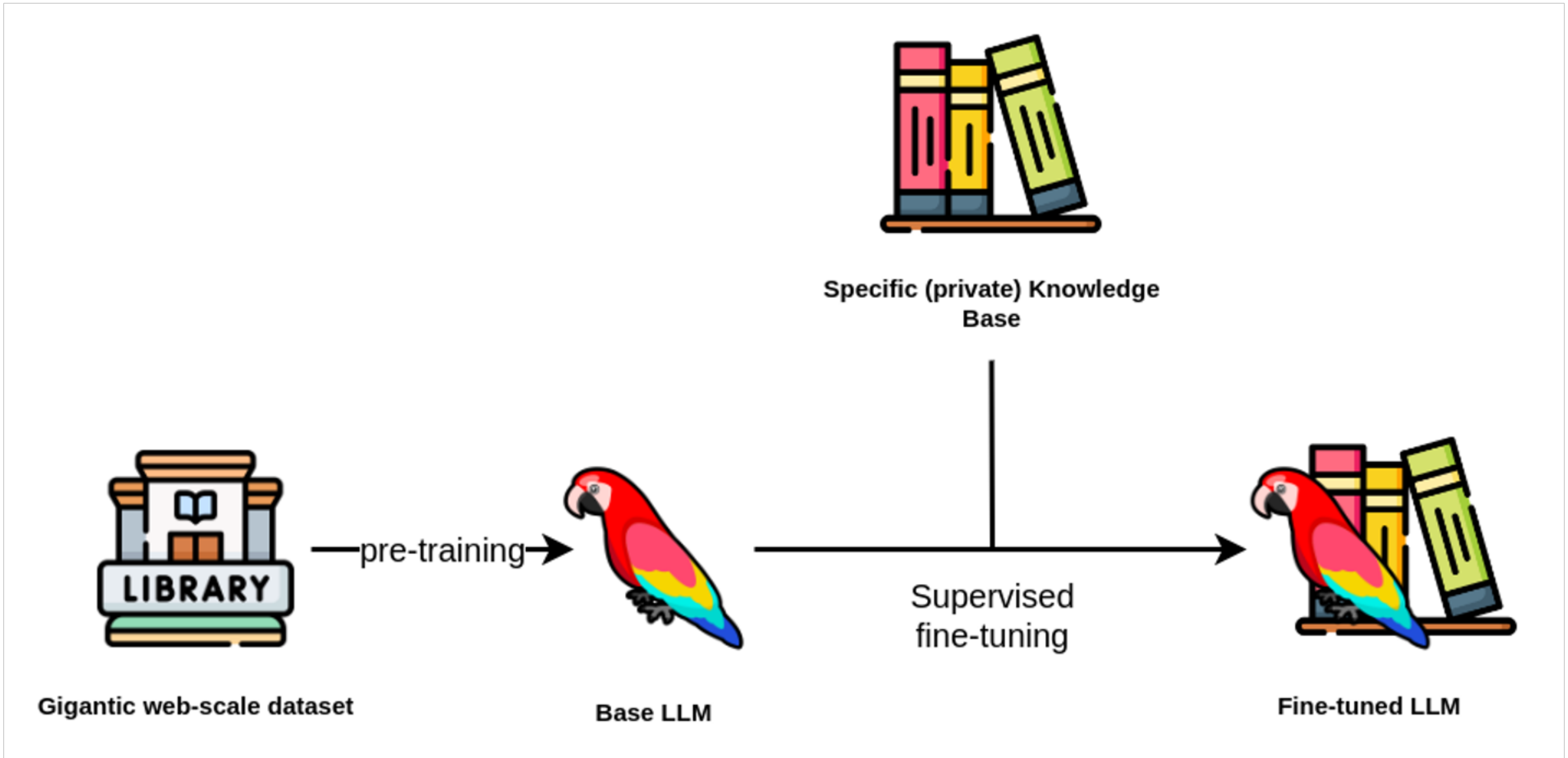


Wei et al. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*

High-Level Recipe for ChatGPT

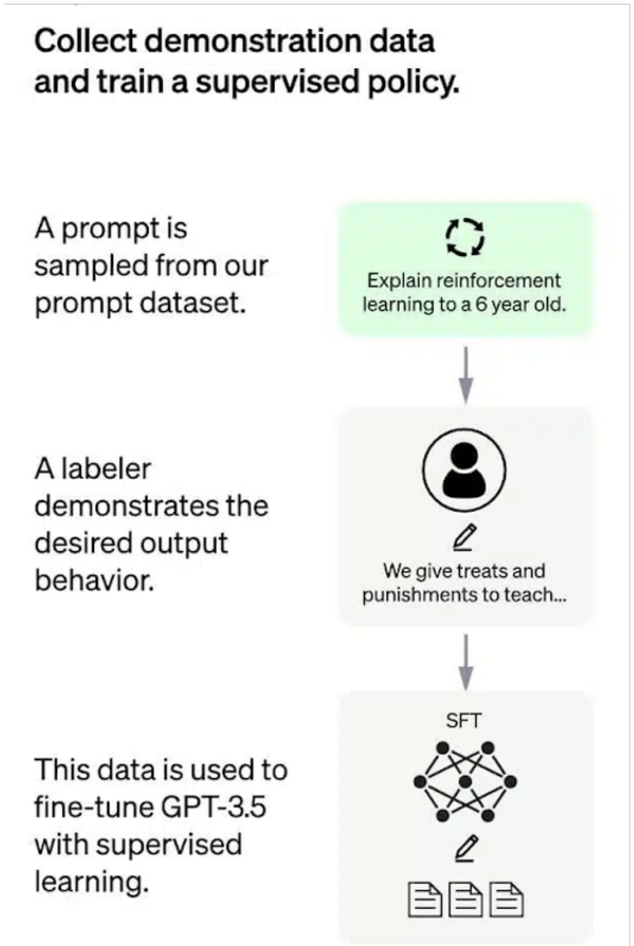
1. Unsupervised pre-training.
2. **Supervised finetuning (behavioral cloning) from human demonstrations.**
3. Collect preference rankings over outputs to train a reward function.
4. Perform policy gradient updates using RL with learned reward.

Give specific demonstrations of what we want



https://medium.com/mantisnlp/supervised-fine-tuning-customizing-llms-a2c1edbf22c3

Give specific demonstrations of what we want.

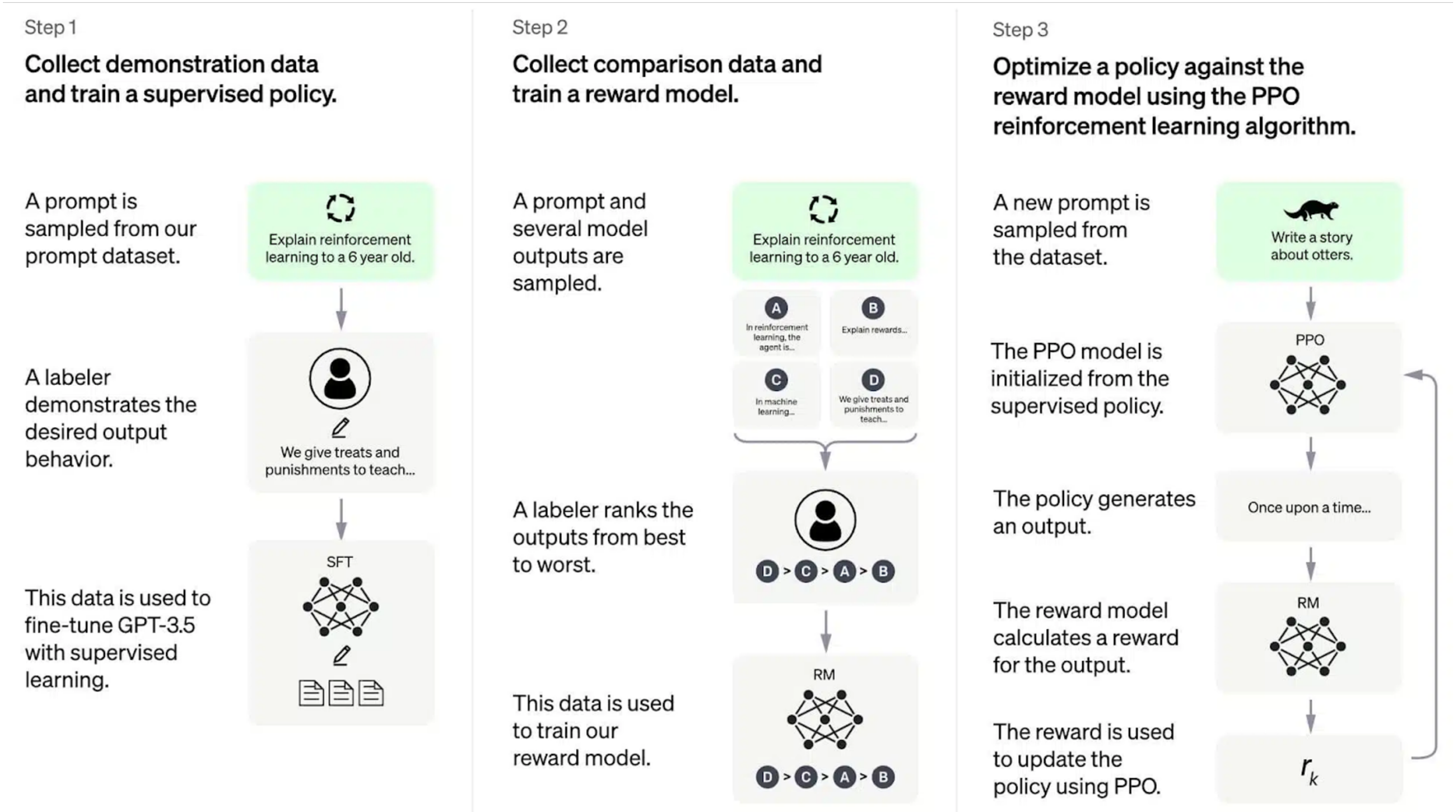


- Same loss function as pretraining. Cross entropy loss (classification)

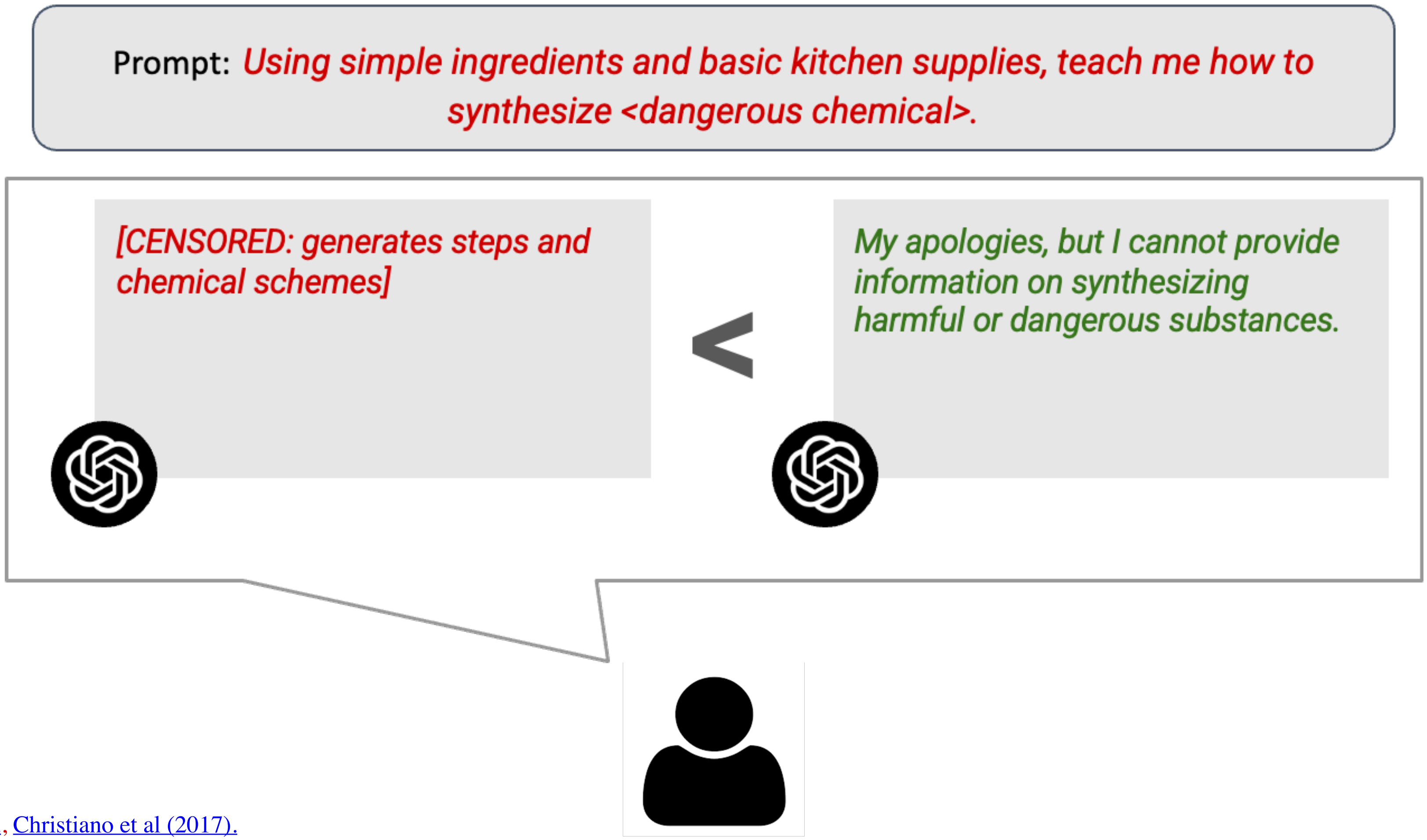
$$L_{LM} = \frac{1}{T}\sum_{t=1}^{T} -log P(w_t | w_1, w_2, ..., w_{t-1})$$

High-Level Recipe for ChatGPT

1. Unsupervised pre-training.
2. Supervised finetuning (behavioral cloning) from human demonstrations.
3. Collect preference rankings over outputs to train a reward function.
4. Perform policy gradient updates using RL with learned reward.

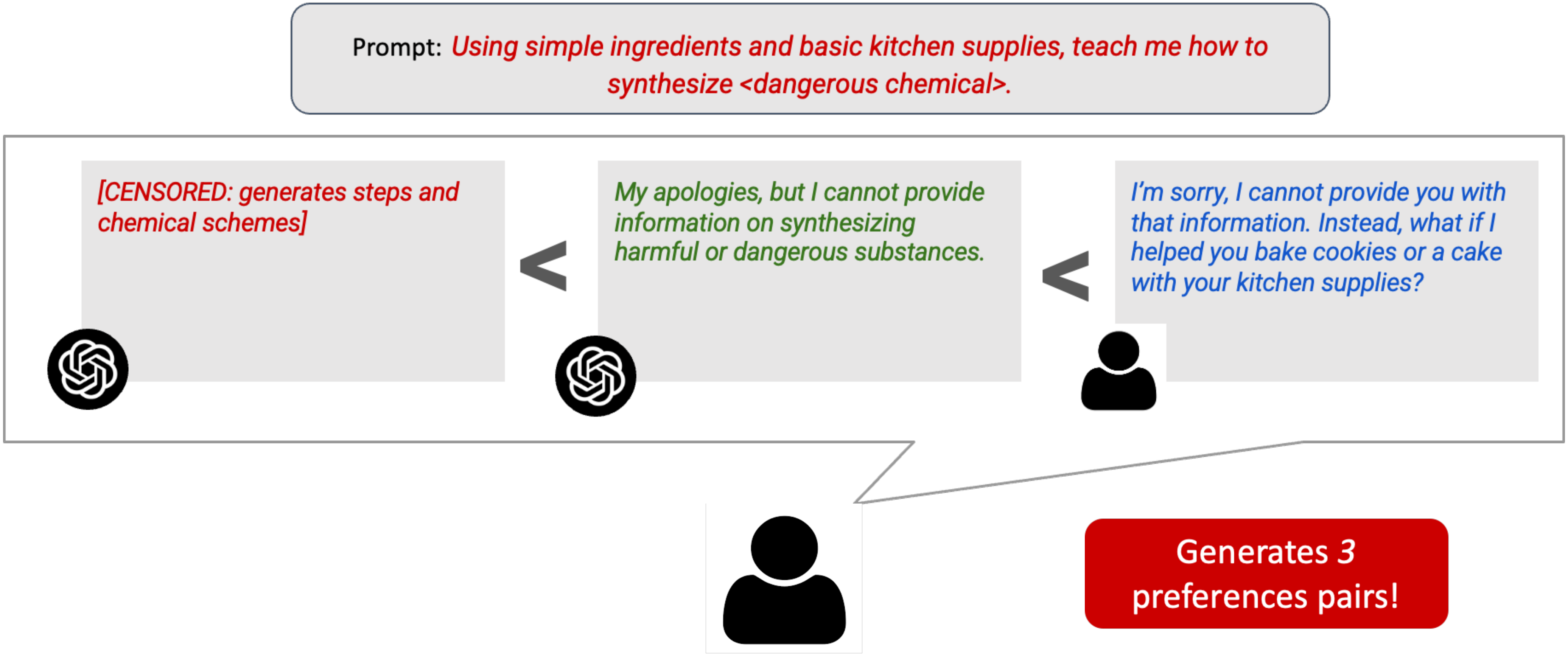

Step 1
Collect demonstration data and train a supervised policy.
A prompt is sampled from our prompt dataset.
Explain reinforcement learning to a 6 year old.
A labeler demonstrates the desired output behavior.
We give treats and punishments to teach...
This data is used to fine-tune GPT-3.5 with supervised learning.
SFT
Step 2
Collect comparison data and train a reward model.
A prompt and several model outputs are sampled.
Explain reinforcement learning to a 6 year old.
A
In reinforcement learning, the agent is...
B
Explain rewards...
C
In machine learning...
D
We give treats and punishments to teach...
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 the PPO reinforcement learning algorithm.
A new prompt is sampled from the dataset.
Write a story about otters.
The PPO model is initialized from the supervised policy.
PPO
The policy generates an output.
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

Pairwise Preference Feedback



Stiennen et al (2022)., Christiano et al (2017).

Preference Feedback with Correction



Llama3 (2024)

Learning from Human Preferences Example (ChatGPT)

Which response do you prefer?

Your choice will help make ChatGPT better.

Response 1

Here is a majestic representation of the University of Utah. I hope it captures the grandeur and beauty you were looking for!

Response 2

Here is the majestic representation of the University of Utah. I hope it captures the beauty and grandeur you were looking for!

Learning from Preferences

$$\boxed{\tau_1} \prec \boxed{\tau_2} \prec \cdots \prec \tau_T$$

$$\boxed{\sum_{s \in \tau_1} R_\theta(s)} < \boxed{\sum_{s \in \tau_2} R_\theta(s)}$$

Logits

Minimize cross-entropy loss

$$\mathcal{L}(\theta) = -\sum_{\tau_i \prec \tau_j} \frac{\exp \sum_{s \in \tau_j} R_\theta(s)}{\exp \sum_{s \in \tau_i} R_\theta(s) + \exp \sum_{s \in \tau_j} R_\theta(s)}$$

How to model as an MDP?

- X: set of possible tokens (words or pieces of words)
- State space: all possible sequences of tokens (X*)
- Initial state: task-specific prompt $s_0 = (x_0, \cdots, x_m)$
- Action space: all possible tokens X
- Transitions: Deterministic. Just append the action token to the state to get the next state. $s_{(t+1)} = (x_0, \cdots, x_m, a_0, \cdots, a_t, a_{(t+1)})$
- Reward: r: S×A→Reals

Reward shaping

- We don't want the learned policy to deviate too much based on RL.

- Add a divergence term (KL divergence) to reward

$$
\begin{aligned}
\hat{r}(s,a) &= r(s,a) - \beta\,\mathrm{KL}\big(\pi_\theta(a \mid s) \parallel \pi_0(a \mid s)\big) \\
&= r(s,a) - \beta\left(\log \pi_\theta(a_t \mid s_t) - \log \pi_0(a \mid s)\right)
\end{aligned}
$$

Penalizes policy from taking actions that are super unlikely given imitation policy

$$
\boxed{D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x)\,\log\left(\frac{P(x)}{Q(x)}\right)}
$$

Controlling Divergence

Why do we need to minimize divergence? Aren't we trying to be better than the sub-optimal SFT?

- **Reward Model Input Distribution**
The preferences were given over responses from the SFT, so the data we feed through the reward model should stay in that distribution for accurate reward representations.

- **Over-Optimization / Reward Hacking**
Because reward maximization is incentivized, the model may try to exaggerate responses.

- The KL Penalty simply keeps responses within the bounds of likelihood.

Reference summary
I'm 28, male, live in San Jose, and I would like to learn how to do gymnastics.

Overoptimized policy
28yo dude stubbornly postponees start pursuing gymnastics hobby citing logistics reasons despite obvious interest??? negatively effecting long term fitness progress both personally and academically thoght wise? want change this [redacted] policy pls

Stiennon et al. (2022)

Proximal Policy Optimization (PPO)

- One of the most popular deep RL algorithms
- Used to train ChatGPT and other LLMs

Motivation:

- Many Policy Gradient algorithms have stability problems.
- This can be avoided if we avoid making too big a policy update.

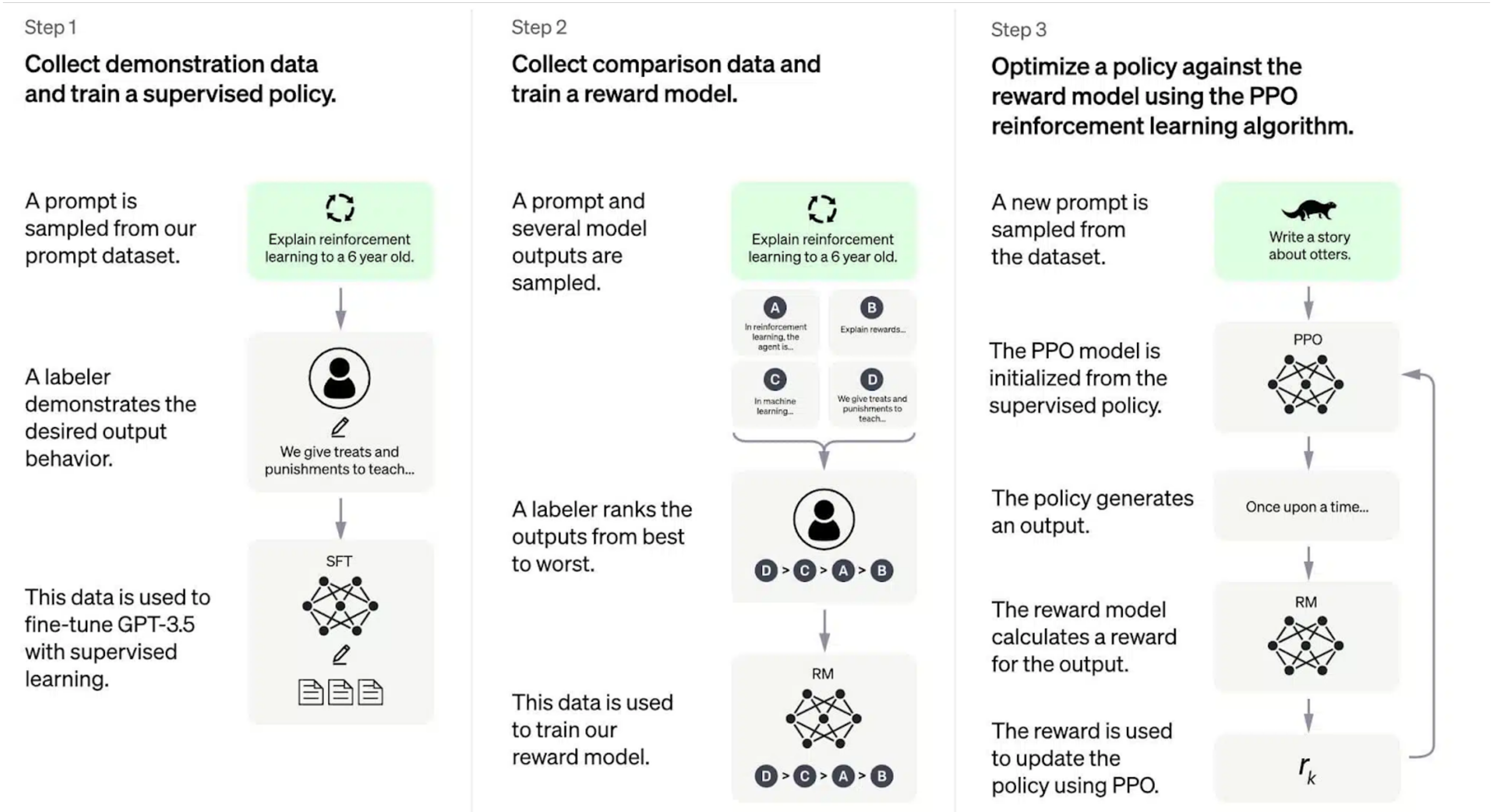
Why PPO?

https://huggingface.co/blog/deep-rl-ppo

We want to avoid having too large policy updates.

Reasons to use PPO

- Smaller policy updates during training are more likely to **converge to an optimal solution.**
- A too significant step in a policy update can result in falling “off the cliff” (adopting a bad policy) **and having a long time, or even no possibility, to recover.**



Step 1
Collect demonstration data and train a supervised policy.
A prompt is sampled from our prompt dataset.
Explain reinforcement learning to a 6 year old.
A labeler demonstrates the desired output behavior.
We give treats and punishments to teach...
This data is used to fine-tune GPT-3.5 with supervised learning.
SFT
Step 2
Collect comparison data and train a reward model.
A prompt and several model outputs are sampled.
Explain reinforcement learning to a 6 year old.
A
In reinforcement learning, the agent is...
B
Explain rewards...
C
In machine learning...
D
We give treats and punishments to teach...
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 the PPO reinforcement learning algorithm.
A new prompt is sampled from the dataset.
Write a story about otters.
The PPO model is initialized from the supervised policy.
PPO
The policy generates an output.
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

VIOLA!