Large Language Models and RL from Human Feedback

Instructor: Daniel Brown

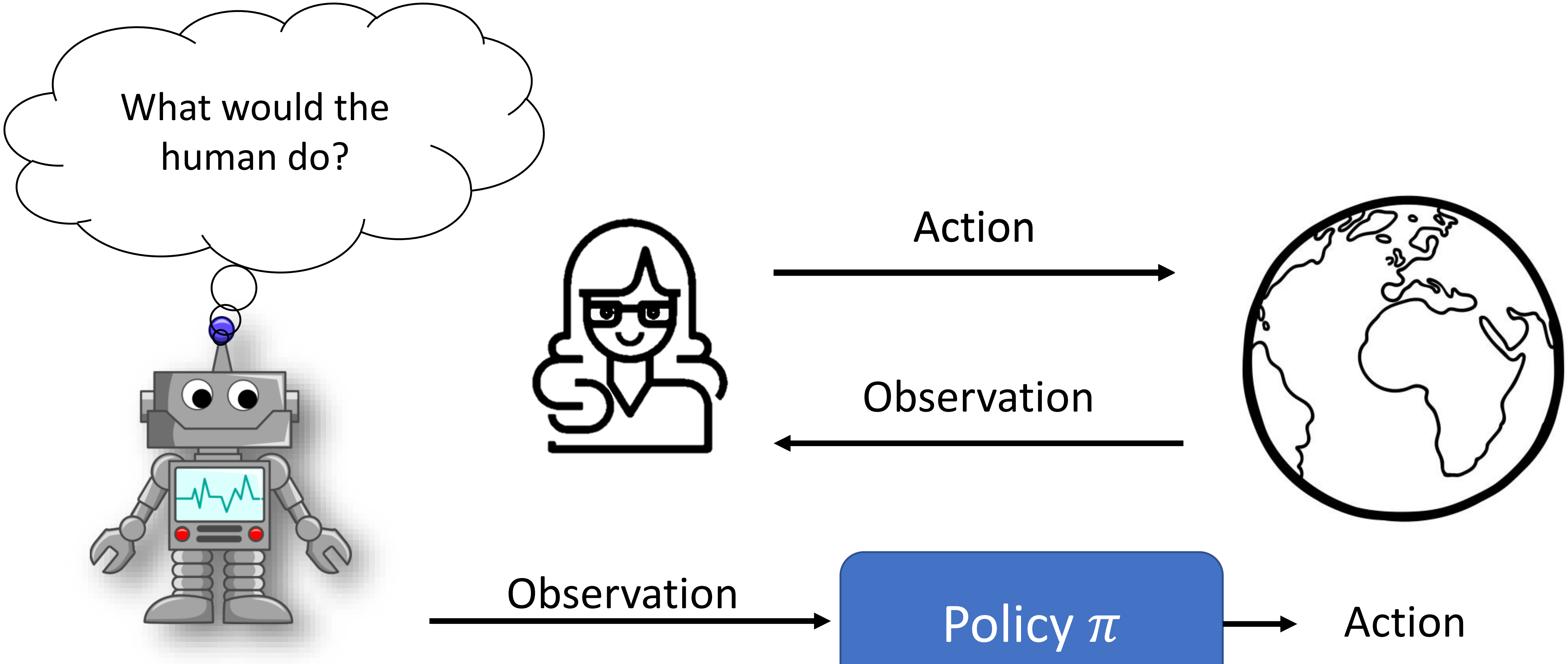
[Some slides adapted from Ana Marasovic, SpinningUp in Deep RL, and others]

Course feedback is open

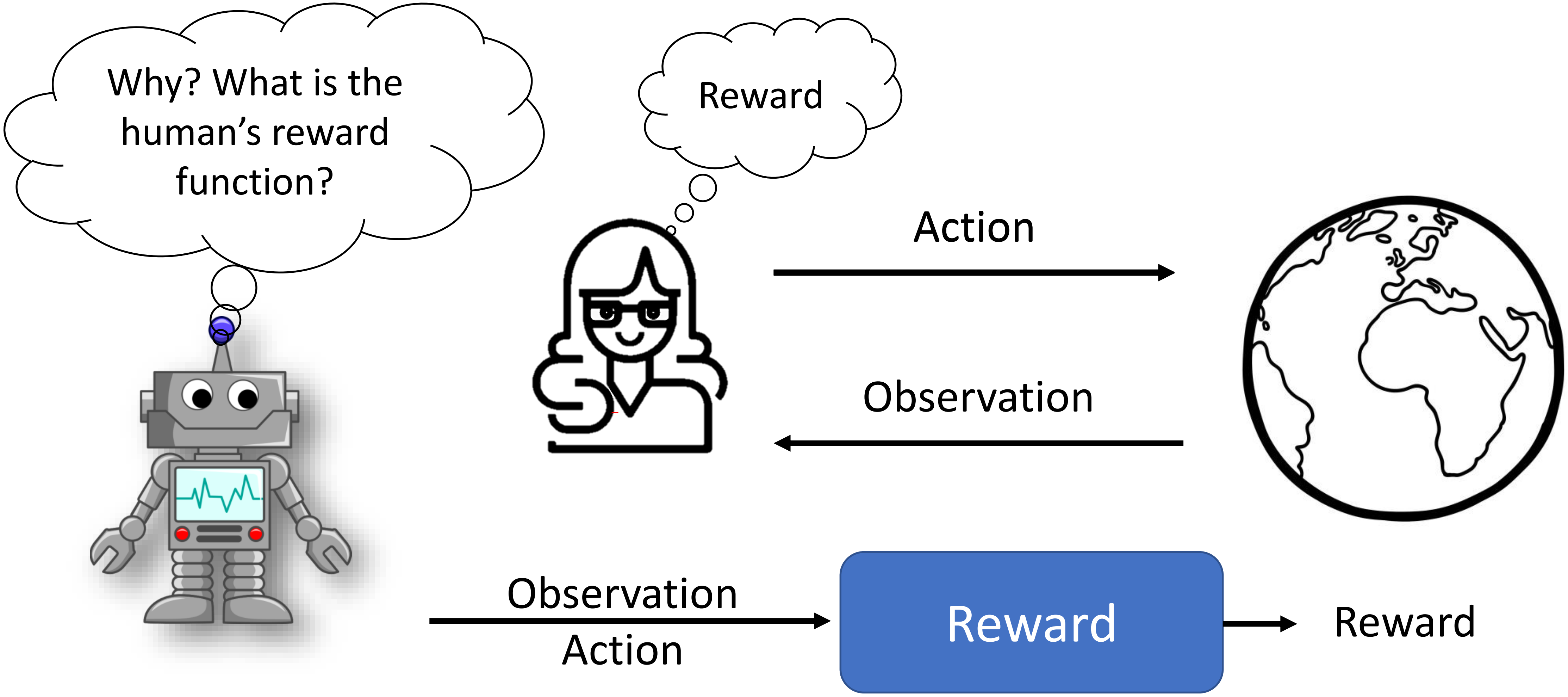
- Extra credit if class response rate is 70% or higher
  - Sliding scale if we reach 70%:
    - Extra credit points = response_rate_percentage / 10

- Currently at 23%

Why not just imitate behavior? (Behavioral Cloning)



Reward Learning
(Inverse Reinforcement Learning)



What if I can’t demonstrate something?

$\succ$

Reinforcement Learning from Human Feedback (RLHF)

Prior approaches to Inverse RL

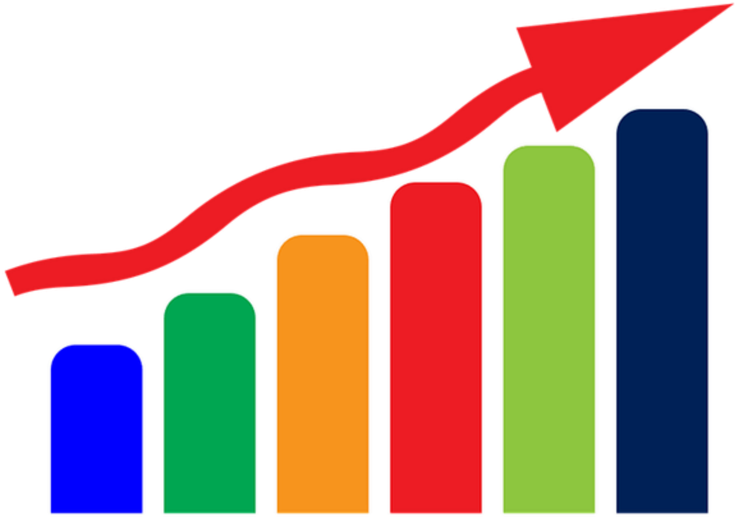
1. ~~Typically couldn't do much better than the demonstrator.~~

Find a reward function that explains the ranking, allowing for extrapolation.

2. ~~Were hard to scale to complex problems.~~

Reward learning becomes a supervised learning problem.

Ranked Examples



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!

ChatGPT 3.5

You

Write me a short poem about salt lake city utah

ChatGPT

In Salt Lake City, where mountains touch the sky,
And the Great Salt Lake's waters stretch far and wide,
Where urban bustle meets nature's serene grace,
A city of contrasts, a captivating place.

Amongst the peaks, the cityscape gleams bright,
With buildings tall, reaching for the light.
Yet in the valleys, a tranquility resides,
Where echoes of the past softly abide

Message ChatGPT...

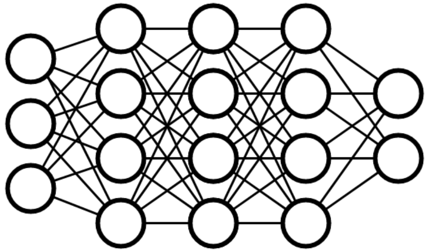
High-Level Recipe for ChatGPT

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

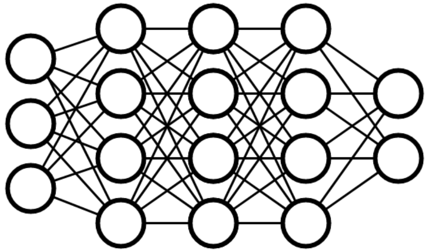
Preliminaries: Language Models

- 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.

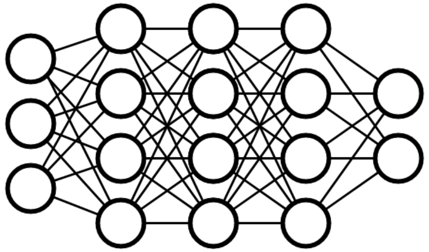
Neural language modeling

[BOS] →  → Sylvester

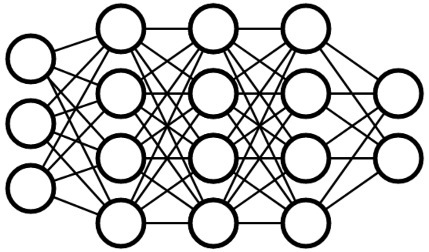
Neural language modeling

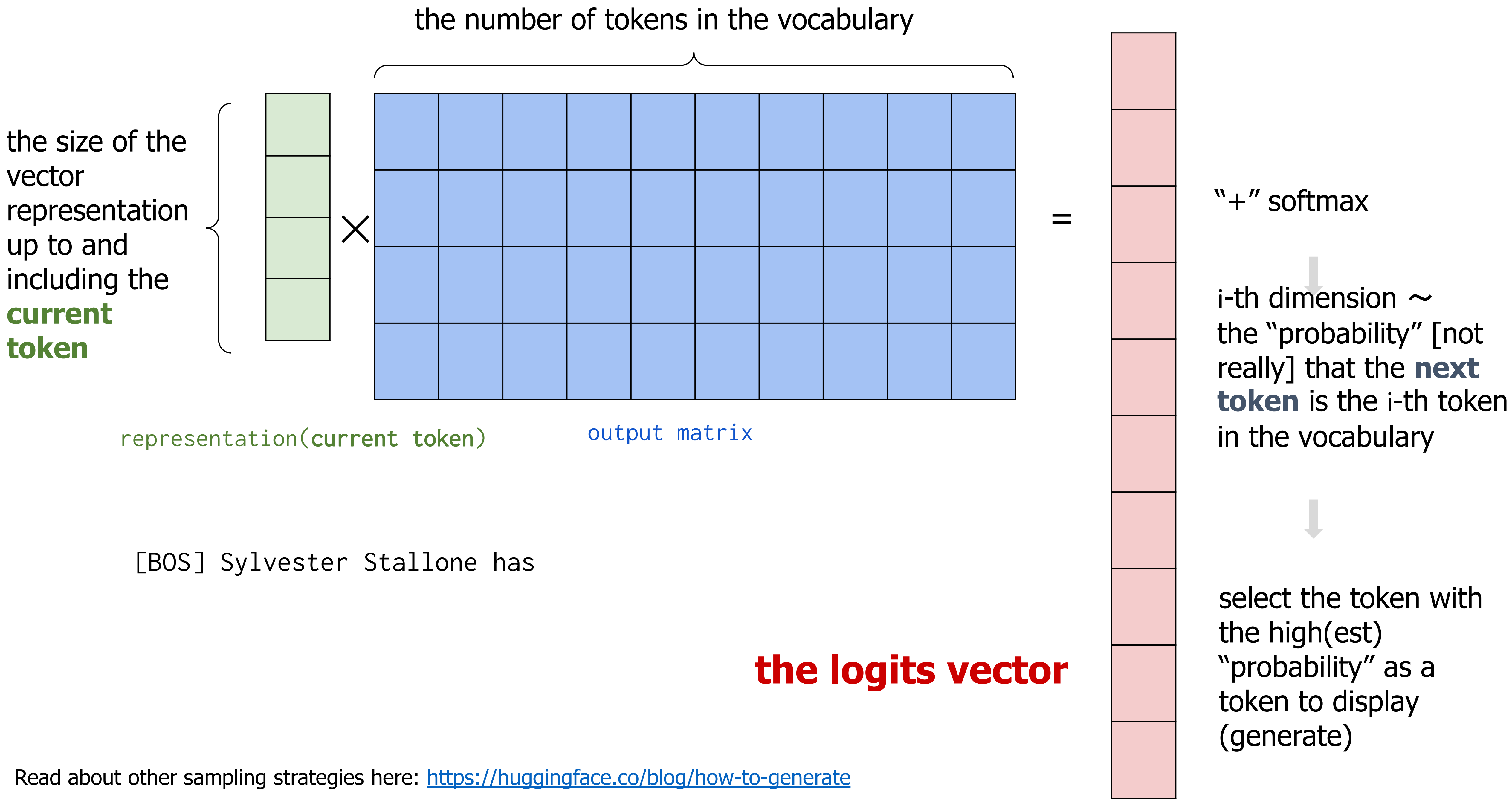
[BOS] Sylvester →  → Stallone

Neural language modeling

[BOS] Sylvester Stallone →  → has

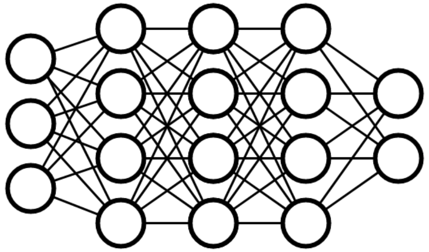
Neural language modeling

`[BOS] Sylvester Stallone has` →  → `made`



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

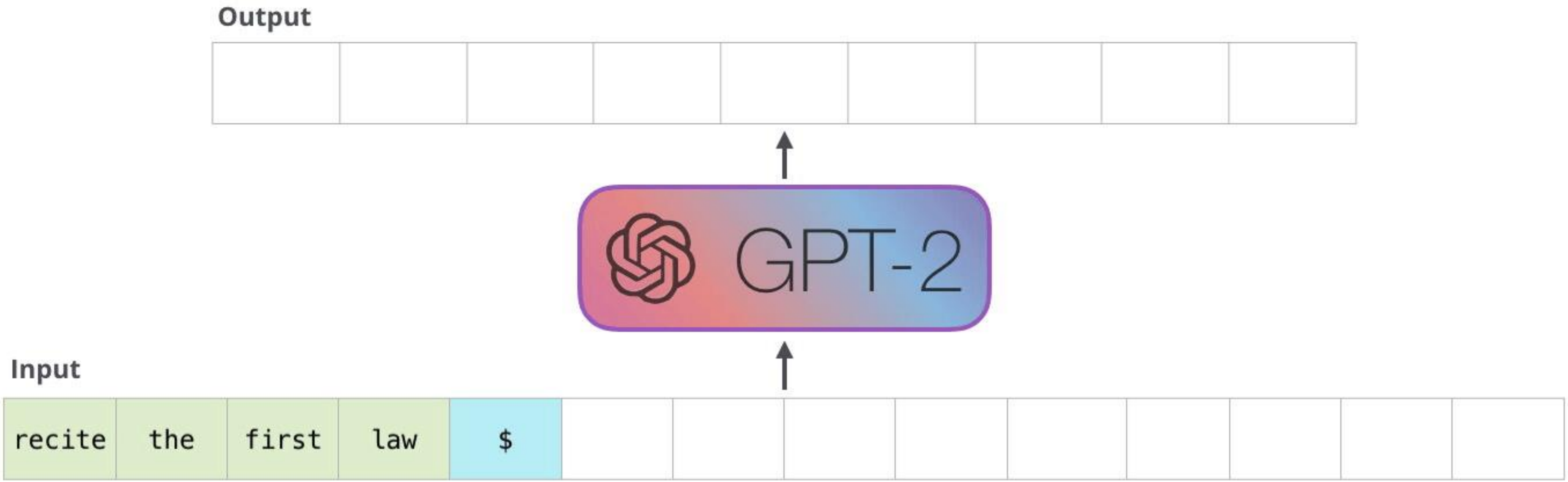
Neural language modeling

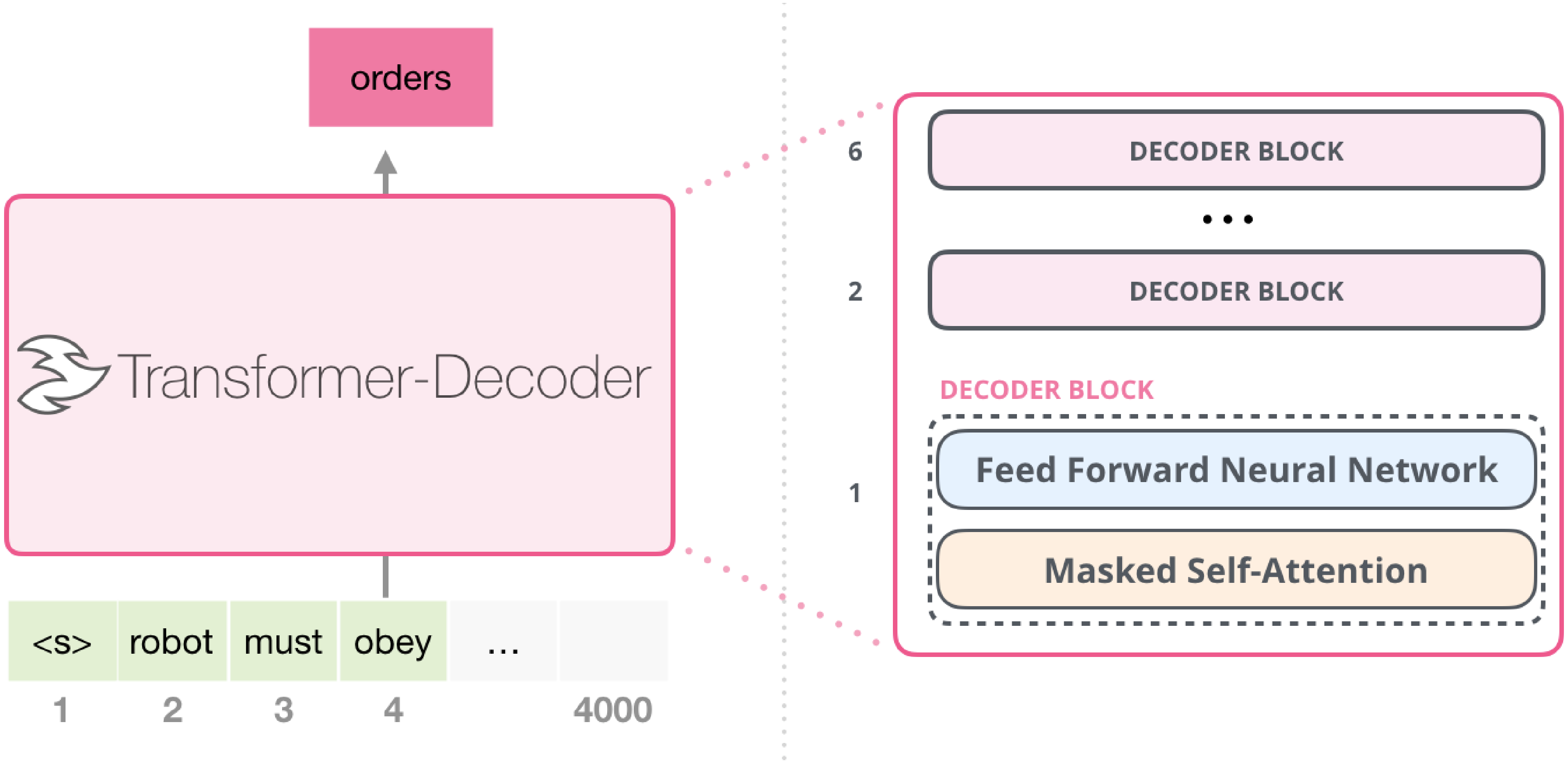
`[BOS] Sylvester Stallone has` ⟶  ⟶ `made`

Problems:

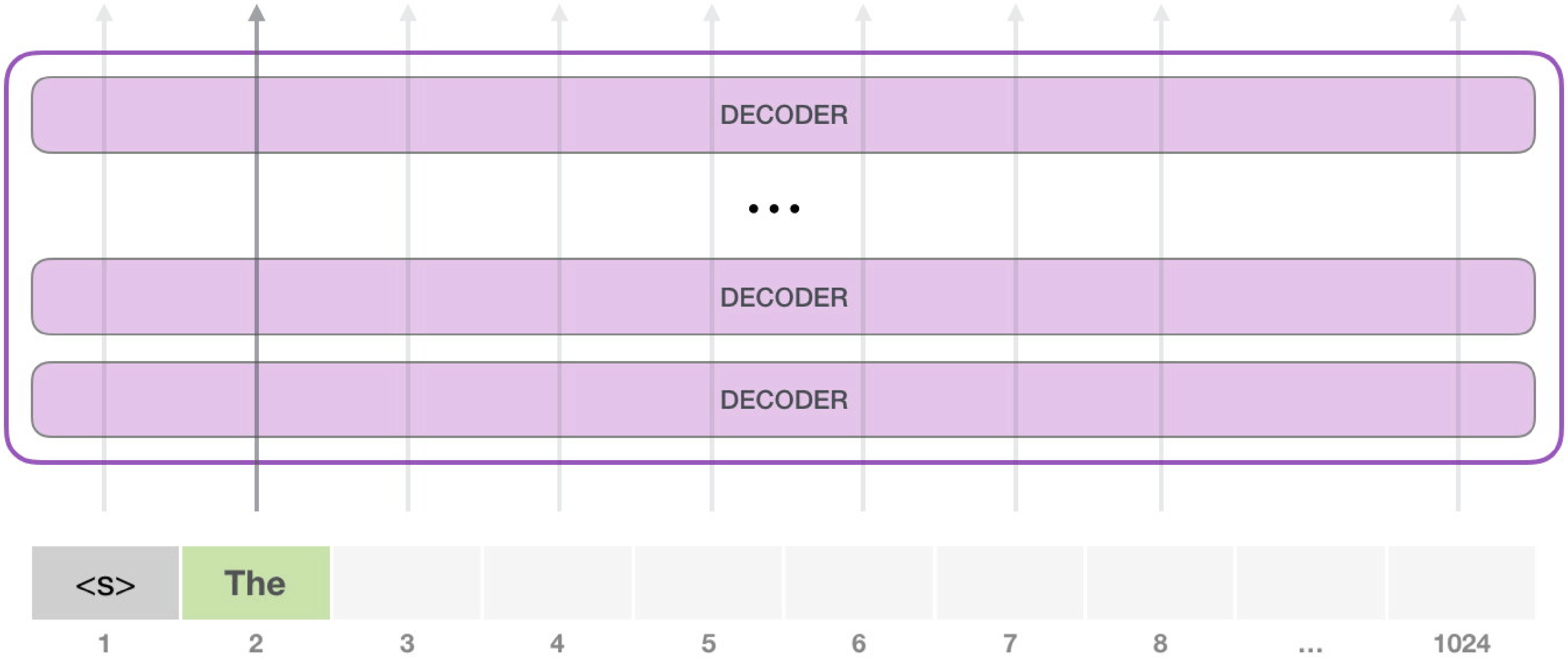
- How do we deal with different length inputs?
- How do we model long-range dependencies?

Large Language Models

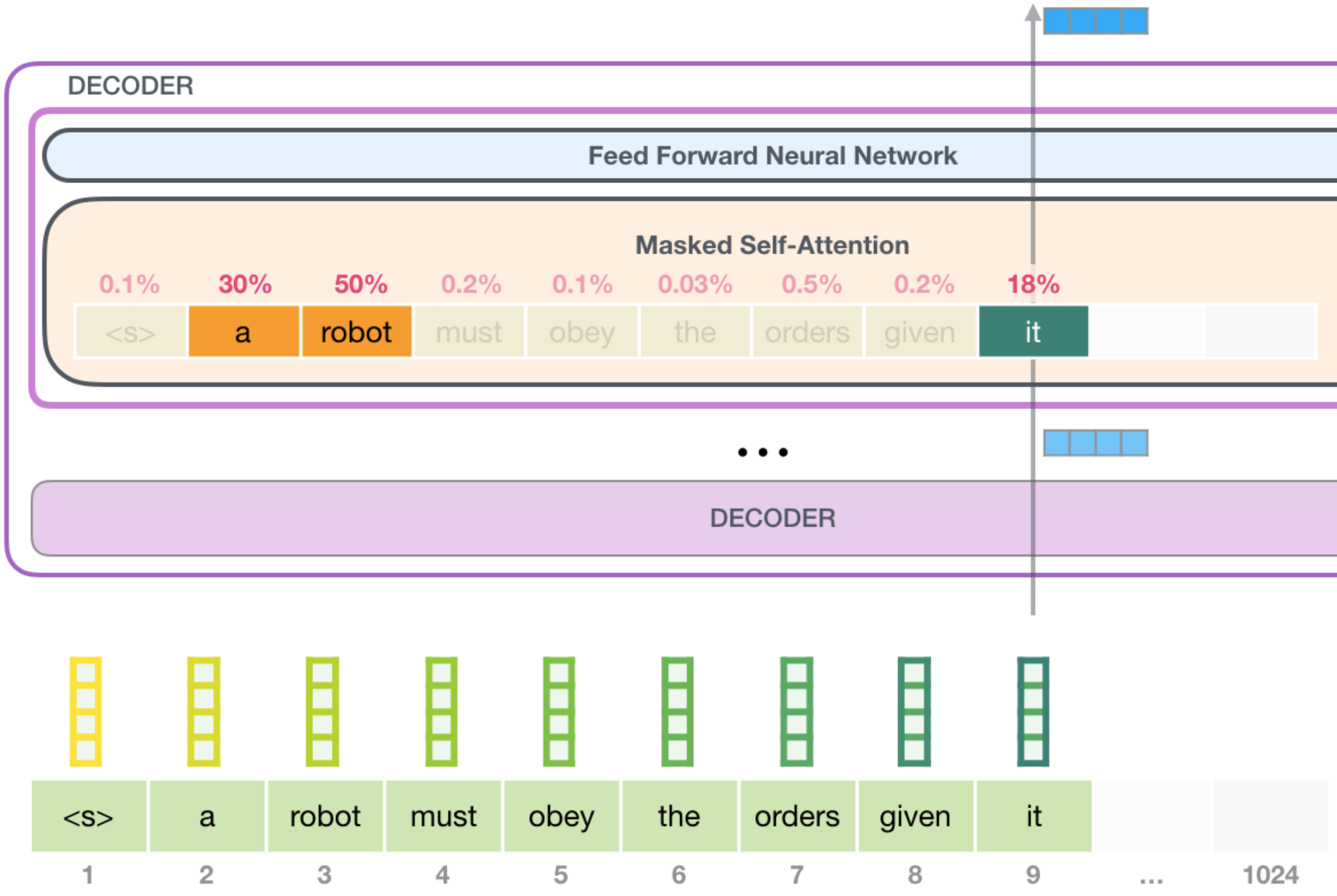


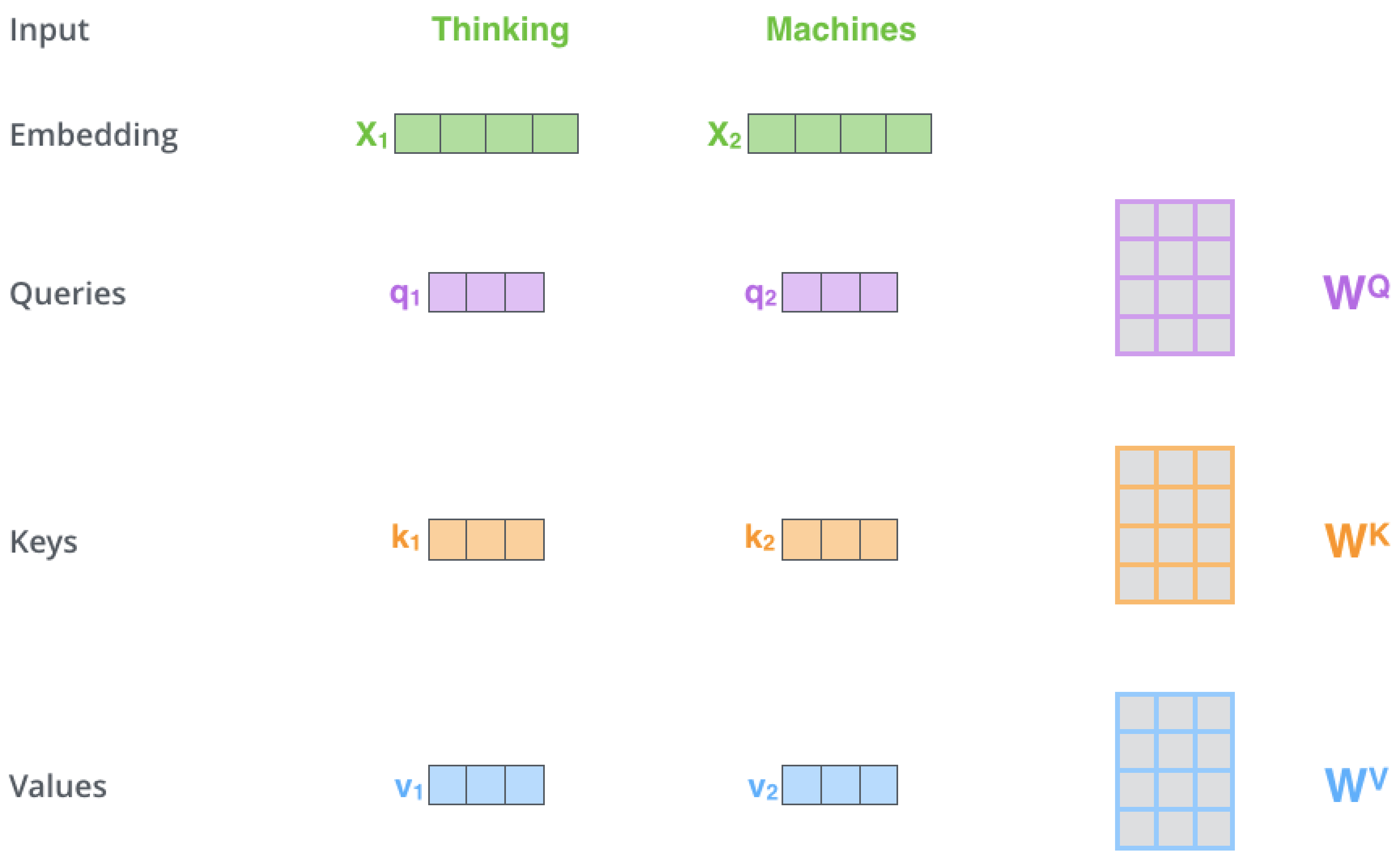


orders
Transformer-Decoder
<s>
robot
must
obey
...
1
2
3
4
4000
6
DECODER BLOCK
• • •
2
DECODER BLOCK
DECODER BLOCK
1
Feed Forward Neural Network
Masked Self-Attention


DECODER
• • •
DECODER
DECODER
<s>
The
1
2
3
4
5
6
7
8
...
1024

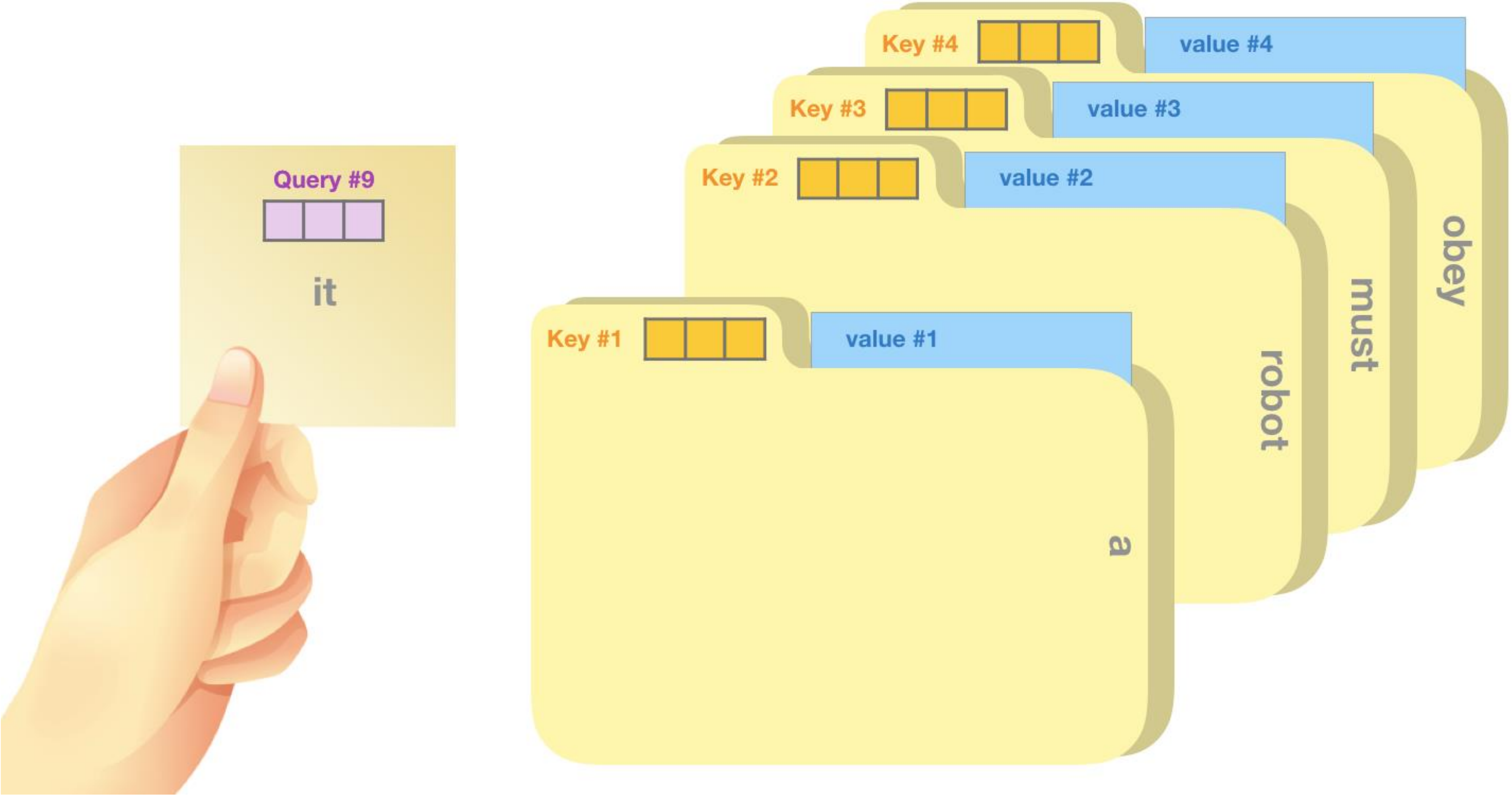

DECODER
DECODER
Positional encoding for token #1
Token embedding of <s>
<s>
1
2
...
1024
Token Embeddings
Positional Encodings


DECODER
Feed Forward Neural Network
Masked Self-Attention
0.1%
30%
50%
0.2%
0.1%
0.03%
0.5%
0.2%
18%
<s>
a
robot
must
obey
the
orders
given
it
• • •
DECODER
<s>
a
robot
must
obey
the
orders
given
it
1
2
3
4
5
6
7
8
9
...
1024

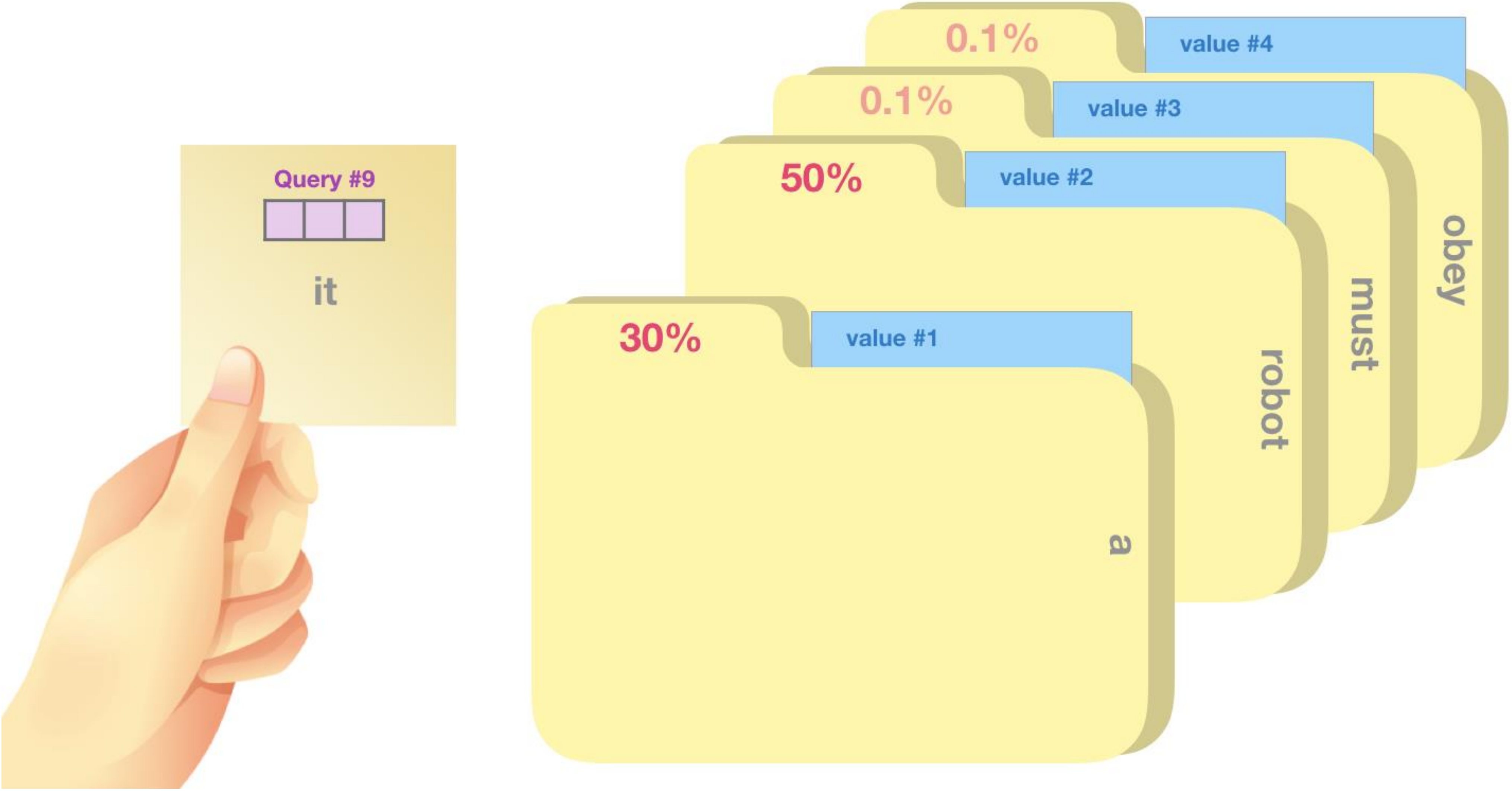


Input
Thinking
Machines
Embedding
X1
X2
Queries
q1
q2
WQ
Keys
k1
k2
WK
Values
v1
v2
WV

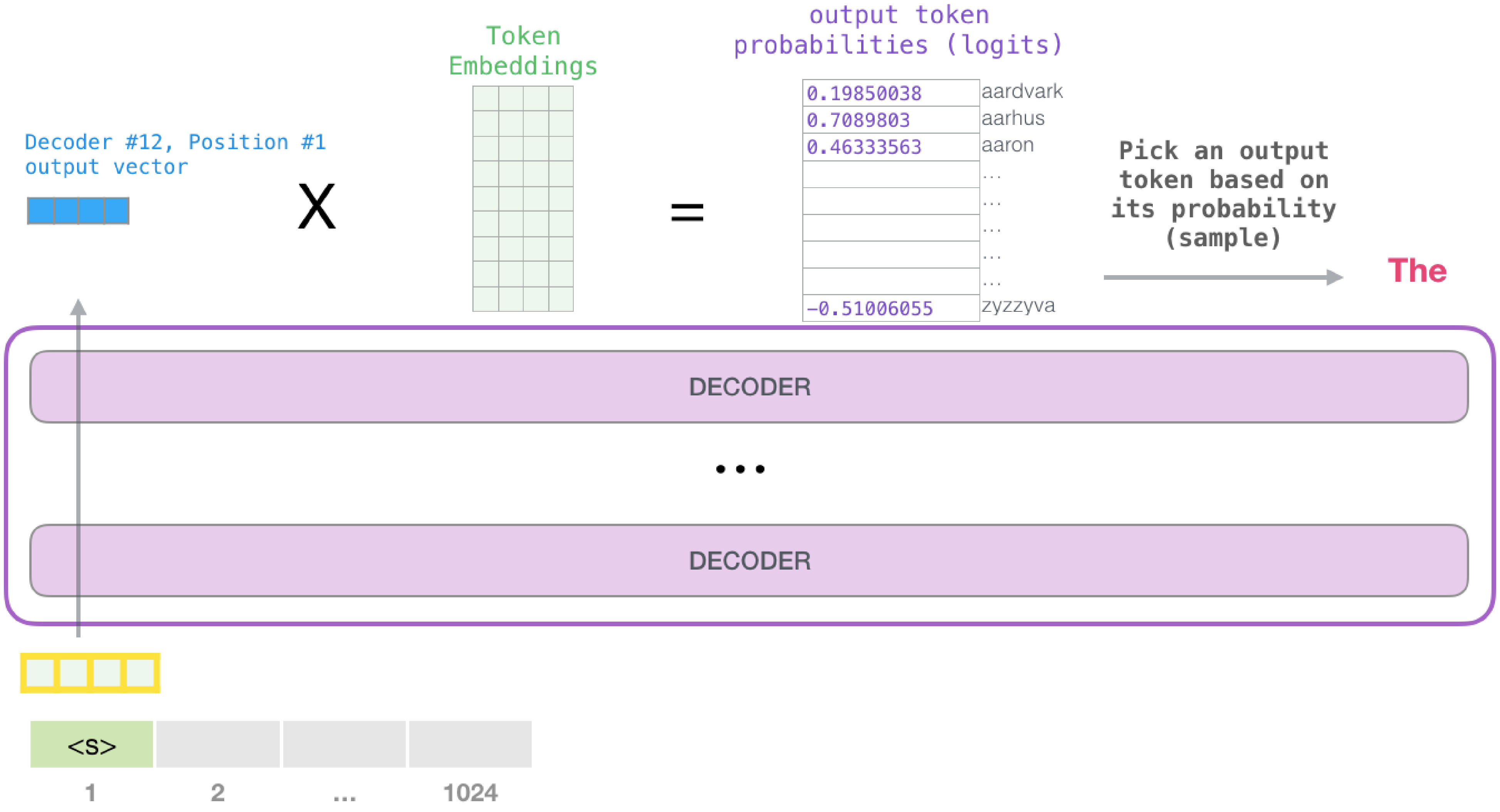
Perform dot product between query and all keys to get a raw score for each previous word (including current word).



Normalize these scores via a softmax to get a probability distribution. Then return a weighted sum of the values.



Word	Value vector	Score	Value X Score
<s>		0.001	
a		0.3	
robot		0.5	
must		0.002	
obey		0.001	
the		0.0003	
orders		0.005	
given		0.002	
it		0.19	
		Sum:	


Token
Embeddings
output token
probabilities (logits)
Decoder #12, Position #1
output vector
X
=
0.19850038 aardvark
0.7089803 aarhus
0.46333563 aaron
...
...
...
...
...
-0.51006055 zyzzyva
Pick an output
token based on
its probability
(sample)
The
DECODER
DECODER
<s>
1
2
...
1024

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!

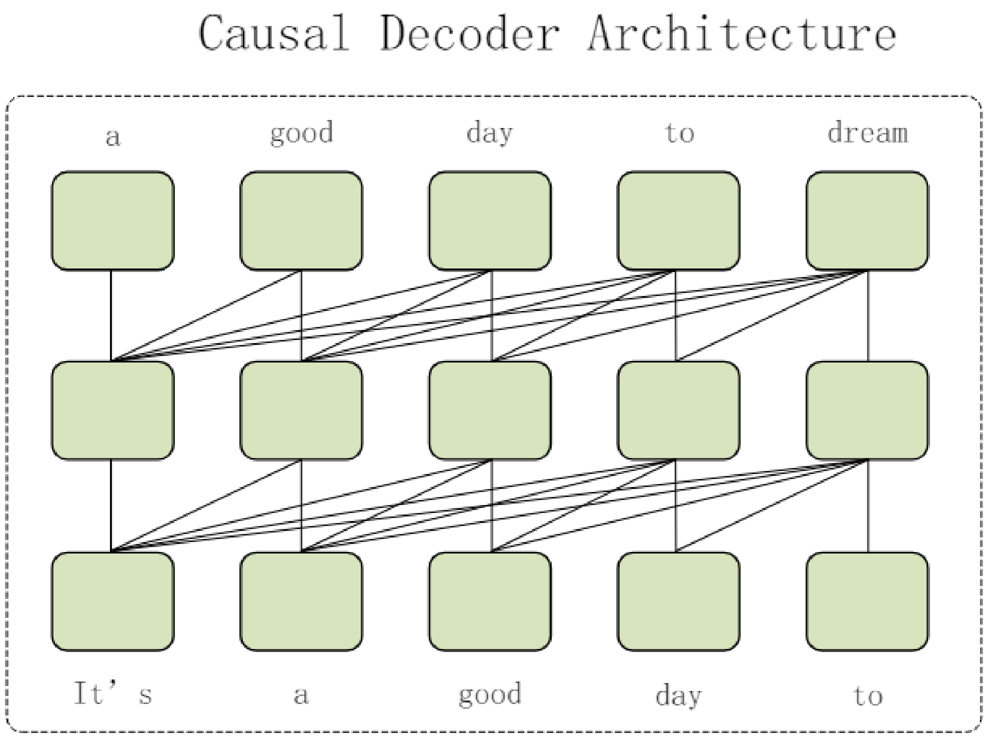
Table 1
Commonly used corpora information.

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

https://arxiv.org/pdf/2401.02038.pdf

Learning a language model by reading the internet!

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



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

https://arxiv.org/pdf/2401.02038.pdf

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:

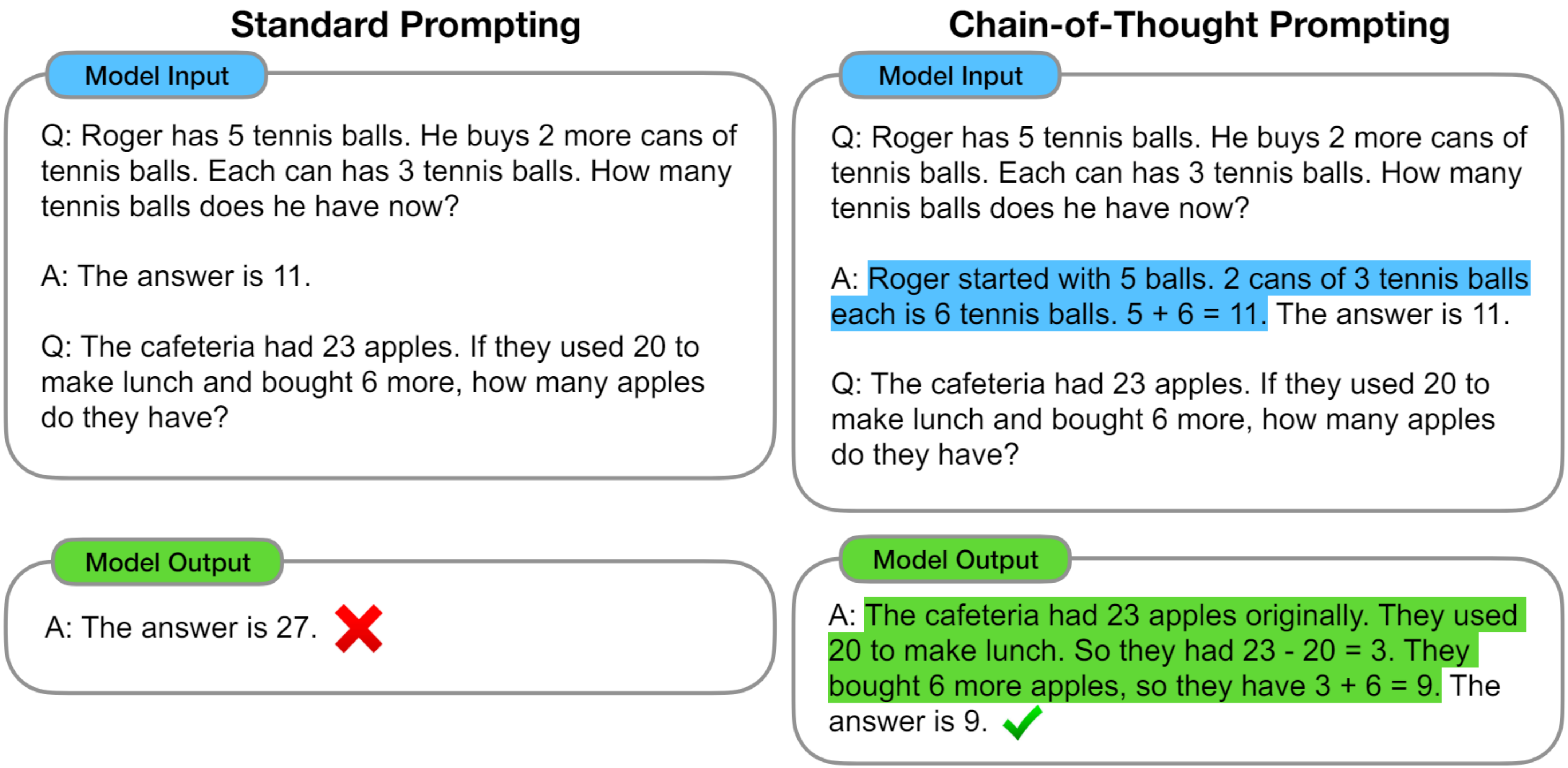
Answer: Behavioral cloning is a type of imitation learning where…

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 "

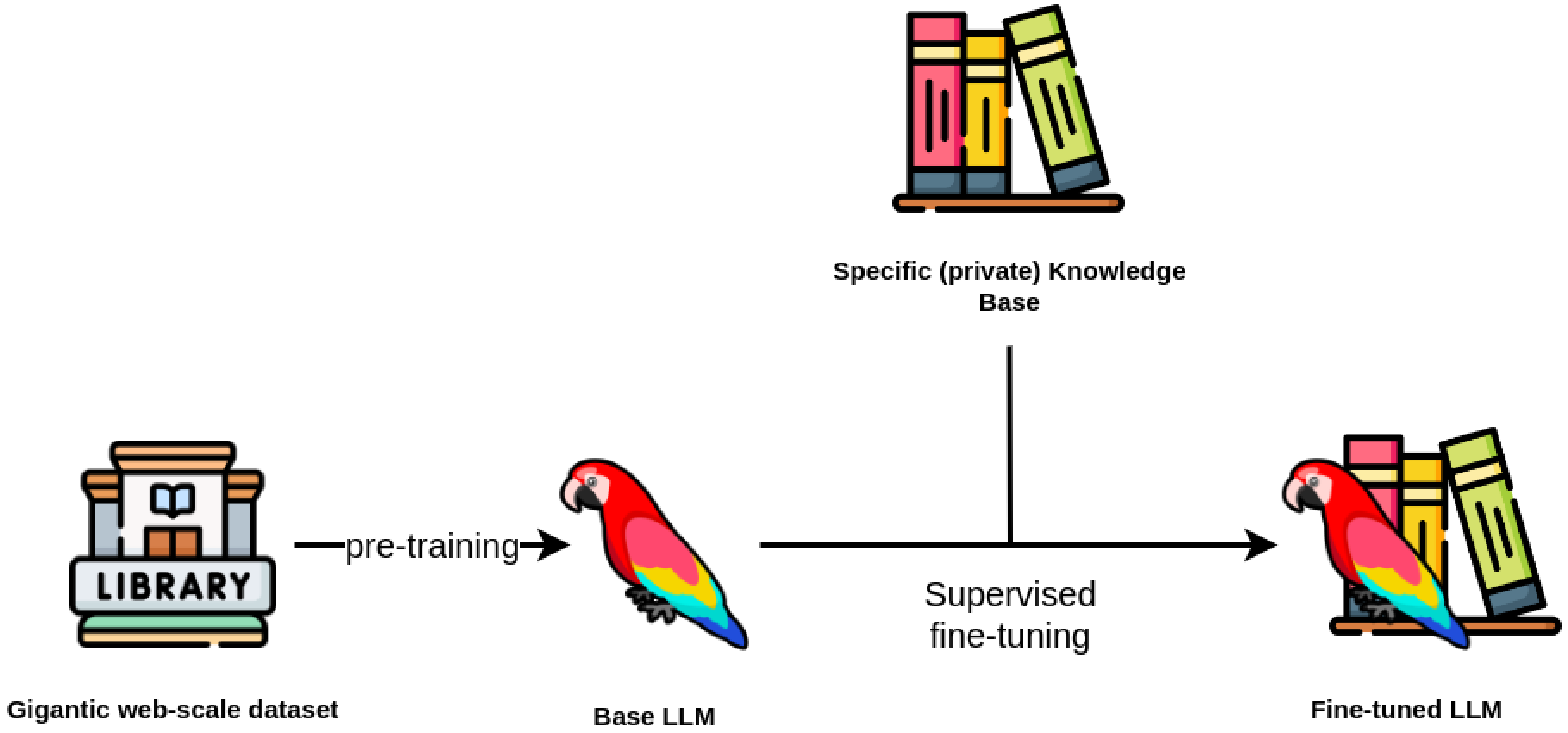


https://arxiv.org/abs/2201.11903

High-Level Recipe for ChatGPT

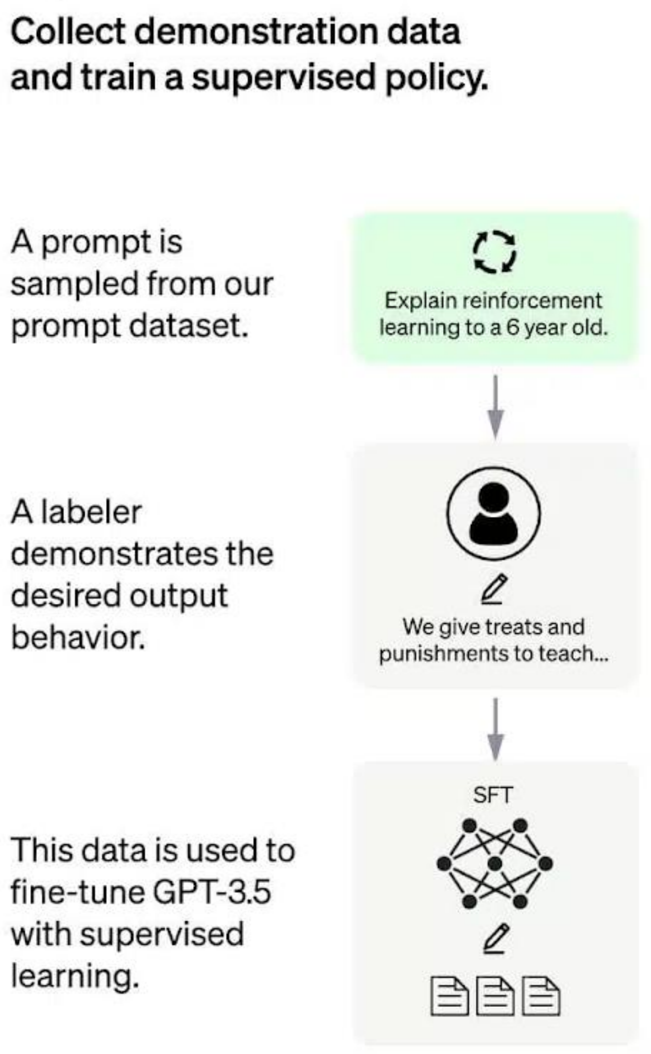
1. Unsupervised pre-training
2. **Supervised finetuning (behavioral cloning) from human demonstrations**
3. Collect preference rankings over outputs to train a reward function
4. Perform policy gradient updates using RL with learned reward

Give specific demonstrations of what we want.



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

Give specific demonstrations of what we want.

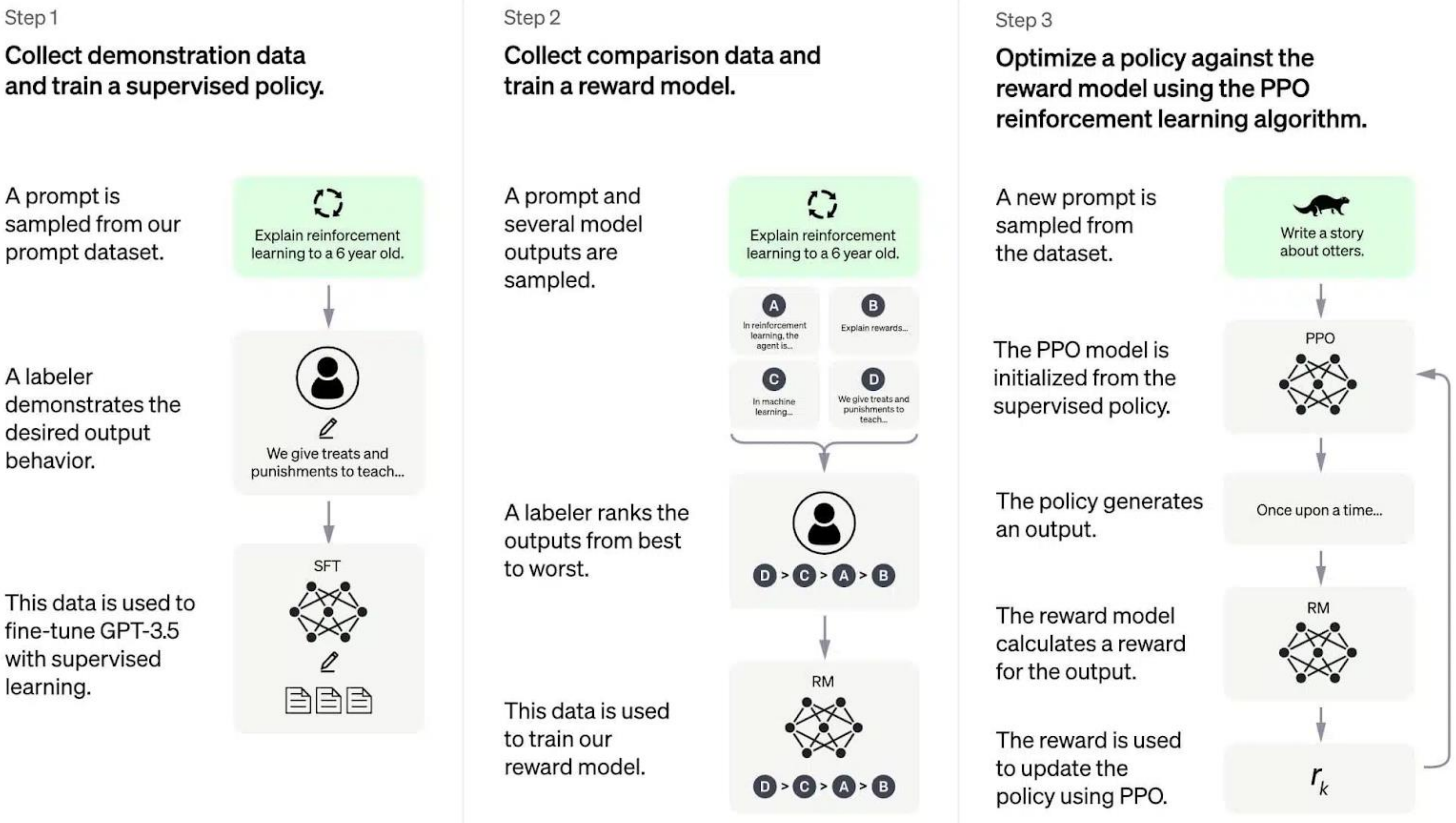


- Same loss function as pretraining. Cross entropy loss (classification)

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

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

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 action token to state to get next state. $s_{t+1} = (x_0, \cdots, x_m, a_0, \cdots, a_t, a_{t+1})$
- Reward: r: $S \times A \rightarrow Reals$

https://arxiv.org/abs/2210.01241

Learn transformer-based reward model from preferences

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

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

Bradley-Terry pairwise ranking loss

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

Reward shaping

- We don't want the learned policy to deviate too much based on RL.
- Add a divergence term (KL divergence) to reward

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

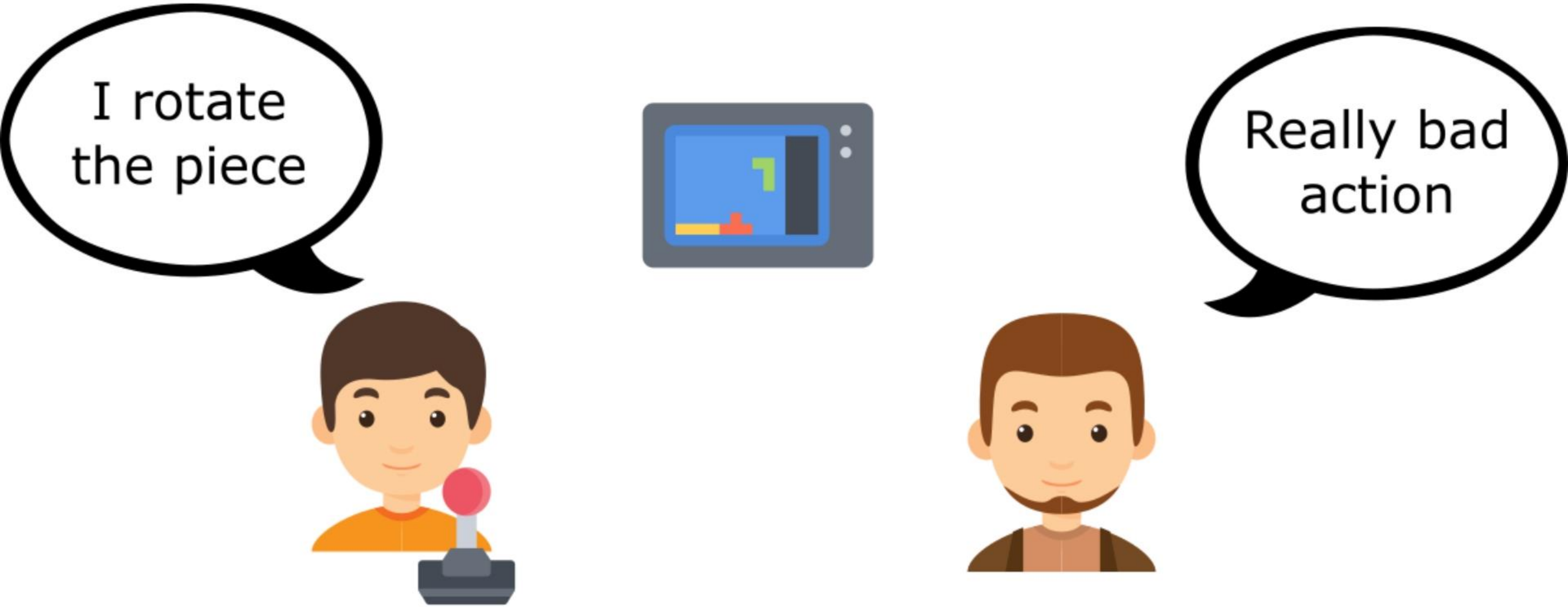
$$\hat{r}(s,a) = r(s,a) - \beta \mathrm{KL}(\pi_\theta(a|s) || \pi_0(a|s)$$
$$= r(s,a) - \beta(\log \pi_\theta(a_t|s_t) - \log \pi_0(a|s))$$

For discrete probability distributions P and Q defined on the same sample space, $\mathcal{X}$, the relative entropy from Q to P is defined[11] to be

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

Flashback: Vanilla Policy Gradient

1. Start with random policy parameters θ_0
2. Run the policy in the environment to collect N rollouts (episodes) of length T and save returns of each trajectory.
$$a_t \sim \pi_\theta(\cdot\,|s_t) \Rightarrow (s_0, a_0, r_0, s_1, a_1, r_1, \dots, r_T, s_{T+1})$$
$$D = \{\tau_1, \dots \tau_N\}, \qquad R = \{R(\tau_1), \dots R(\tau_N)\}$$
3. Compute policy gradient
$$\nabla_\theta J(\pi_\theta) = E_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\ R(\tau)\right]$$
4. Update policy parameters
$$\theta_{k+1} \leftarrow \theta_k + \alpha \nabla_\theta J(\pi_\theta)\Big|_{\theta_k}$$
5. Repeat (Go to 2)



Actor (Policy)

Critic (Value Function)

Advantage Actor Critic (A2C)

- Combining value learning with direct policy learning
  - One example is policy gradient using the advantage function

$$\nabla_\theta J(\pi_\theta) = \mathop{\mathrm{E}}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\Phi_t\right]$$

$$\Phi_t = A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$$

TD error $\delta_t = r(s_t, a_t) + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)$

Policy gradient update

$$\theta_{k+1} \leftarrow \theta_k + \alpha \nabla_\theta J(\pi_\theta)\Big|_{\theta_k}$$

TD-Learning update

$$w_{k+1} \leftarrow w_k + \alpha \delta_t \nabla_w V(s, a; w)$$

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 of a policy update.

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

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

Proximal Policy Iteration (PPO)

- Measure how much current policy changed compared with old version using a ratio:

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$$

- Clip policy gradient update based on this ratio:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

Proximal Policy Iteration (PPO)

- Simpler way to write clip objective:

$$L(s, a, \theta_k, \theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a), \quad g(\epsilon, A^{\pi_{\theta_k}}(s, a))\right)$$

where

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geq 0 \\ (1-\epsilon)A & A < 0 \end{cases}$$

https://spinningup.openai.com/en/latest/algorithms/ppo.html

Proximal Policy Iteration (PPO)

- Simpler way to write clip objective:

$$L(s, a, \theta_k, \theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a), \quad g(\epsilon, A^{\pi_{\theta_k}}(s, a))\right)$$

where

$$g(\epsilon, A) = \begin{cases} \boxed{(1+\epsilon)A \quad A \geq 0} \\ (1-\epsilon)A \quad A < 0 \end{cases}$$

What if the advantage is positive?

$$L(s, a, \theta_k, \theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, (1+\epsilon)\right) A^{\pi_{\theta_k}}(s, a)$$

We want to increase $\pi_\theta(a|s)$, but not too much!

Once $\pi_\theta(a|s) > (1+\epsilon)\pi_{\theta_k}(a|s)$ the min kicks in and limits our policy update.

https://spinningup.openai.com/en/latest/algorithms/ppo.html

Proximal Policy Iteration (PPO)

- Simpler way to write clip objective:

$$L(s,a,\theta_k,\theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a), \;\; g(\epsilon, A^{\pi_{\theta_k}}(s,a))\right)$$

where

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geq 0 \\ \boxed{(1-\epsilon)A \quad A < 0} \end{cases}$$

What if the advantage is negative?

$$L(s,a,\theta_k,\theta) = \max\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, (1-\epsilon)\right) A^{\pi_{\theta_k}}(s,a)$$

We want to decrease $\pi_\theta(a|s)$, but not too much!

Once $\pi_\theta(a|s) < (1+\epsilon)\pi_{\theta_k}(a|s)$ the max kicks in and limits our policy update.

https://spinningup.openai.com/en/latest/algorithms/ppo.html

Algorithm 1 PPO-Clip

Input: initial policy parameters θ_0, initial value function parameters ϕ_0

for $k = 0, 1, 2, ...$ **do**

  Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.

  Compute rewards-to-go $\hat{R}_t$.

  Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.

  Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \quad g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

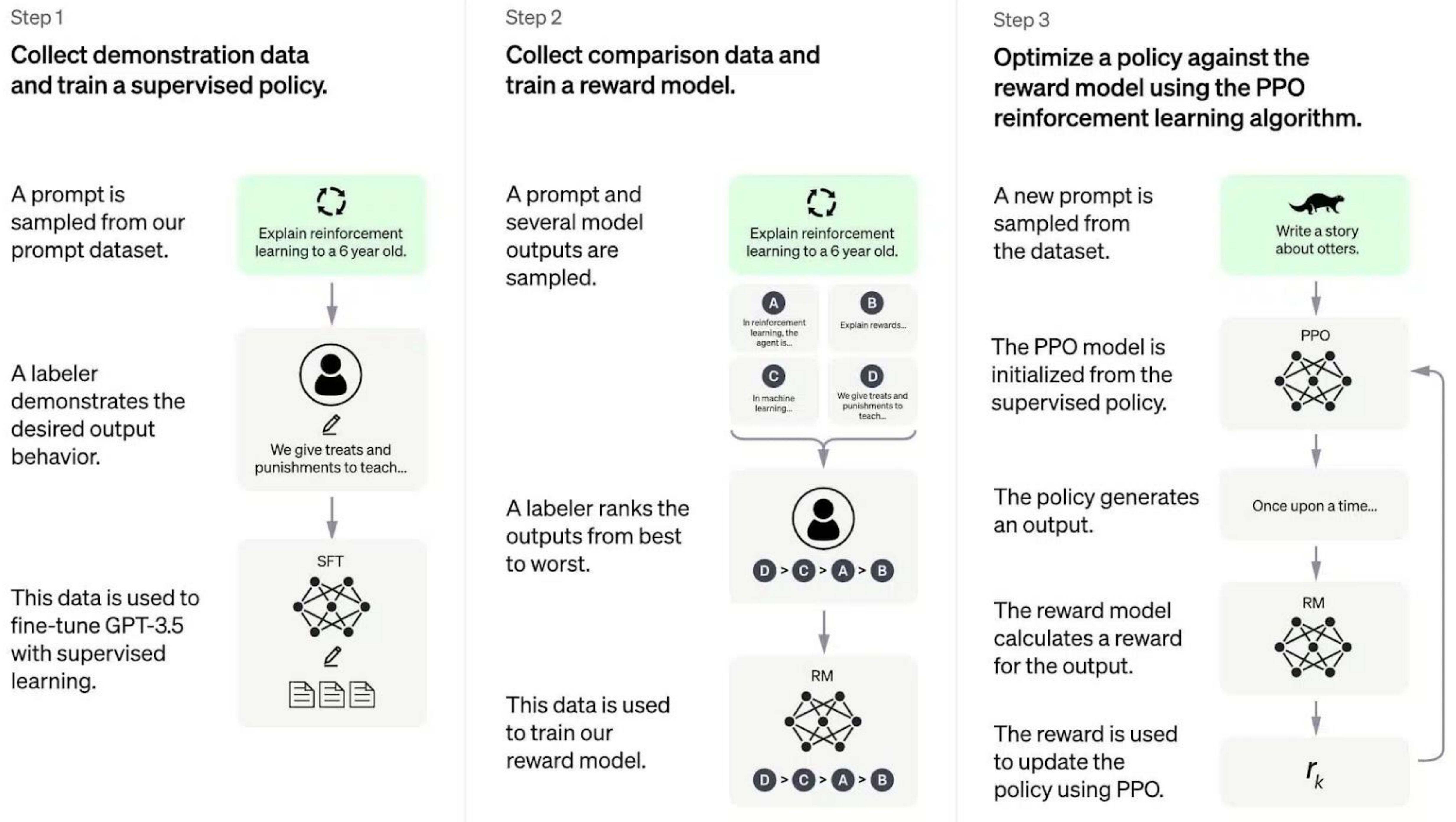
  typically via stochastic gradient ascent with Adam.

  Fit value function by regression on mean-squared error:

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

  typically via some gradient descent algorithm.

end for



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

Recap with alternative notations

- Start with pretrained transformer model (SFT) for policy.
- Take same model and add a linear layer that outputs a scalar for the reward model.
- Train reward model.

$$\text{loss}(r_\theta) = -E_{(x,y_0,y_1,i)\sim D}[\log(\sigma(r_\theta(x,y_i) - r_\theta(x,y_{1-i})))]$$

- Train PPO policy with reward and train value function.

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

Voila!

** We don't know exactly what they did to train ChatGPT, but based on tech reports and prior work, this is probably very close.

You

Write a limerick

about the end of an AI class at the University of Utah

‹ 2 / 2 ›

ChatGPT

In Utah, at the class's close,

AI minds buzzed, creativity rose.

With neurons a-flare,

They bid code "adieu" there,

Leaving algorithms in repose.

We made it! Next time: Review