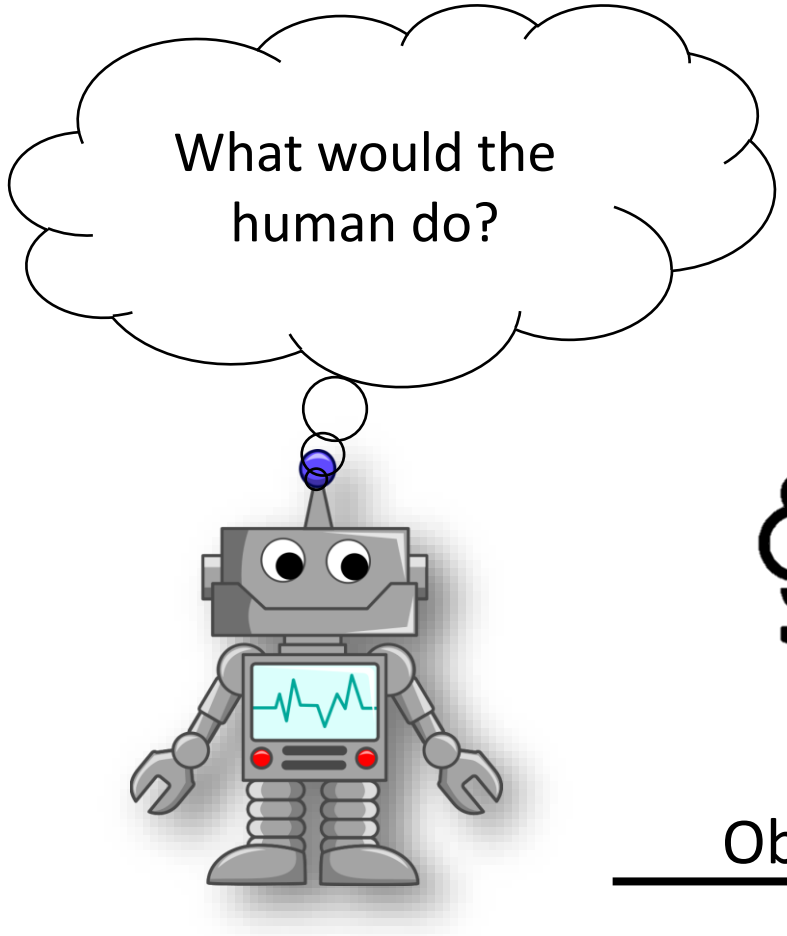
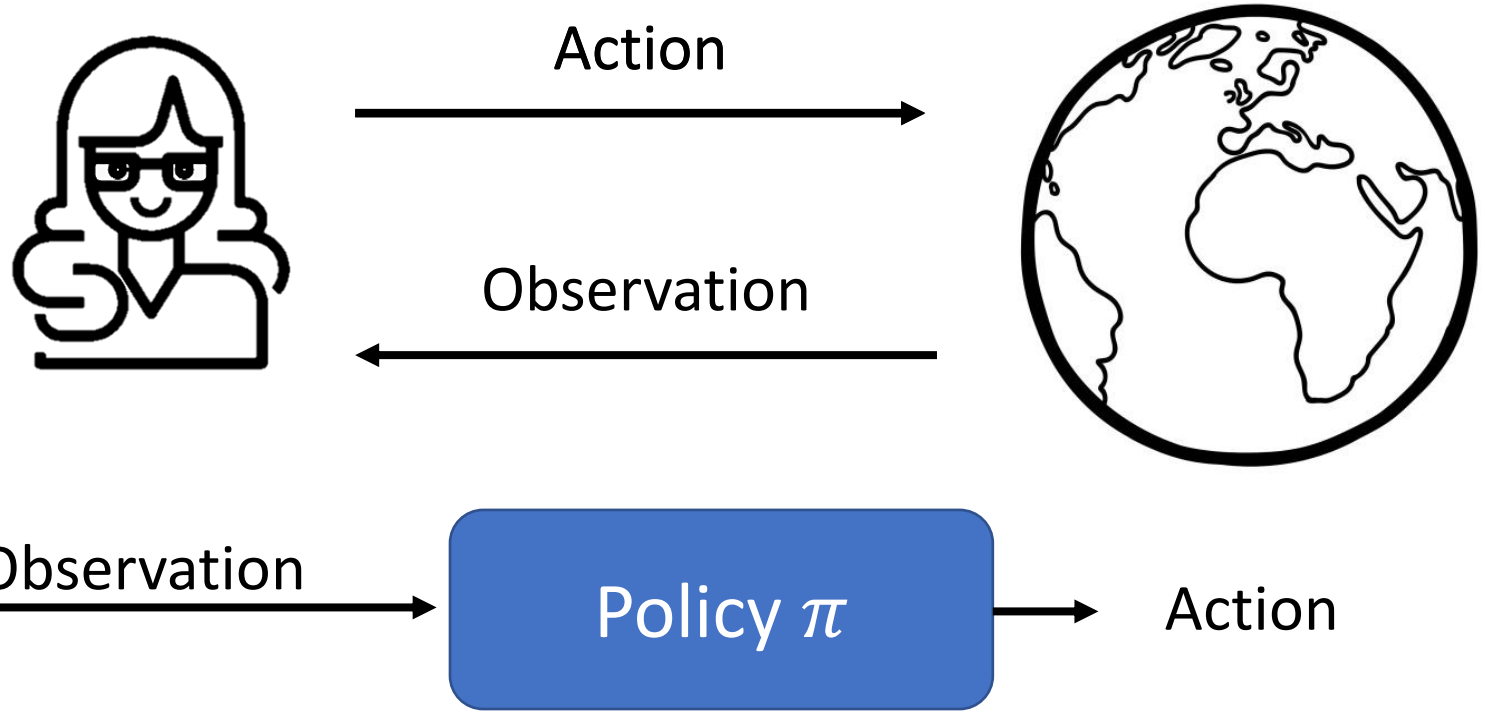
Large Language Models and RL from Human Feedback

Instructor: Daniel Brown

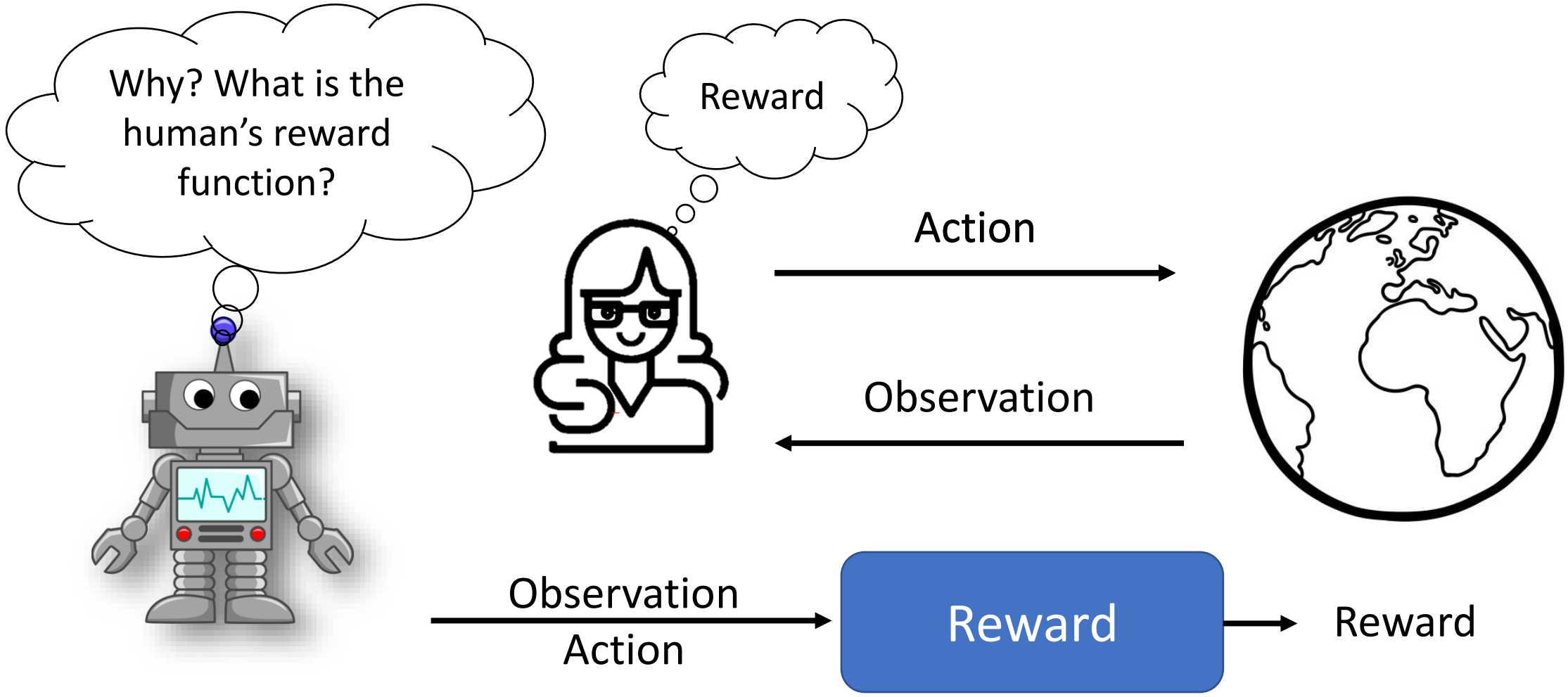
[Some slides adapted from Ana Marasovic, SpinningUp in Deep RL, and others]

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

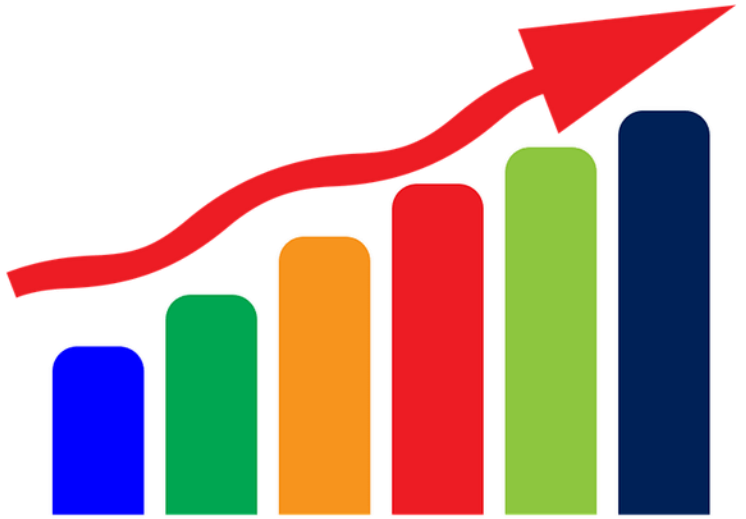
1. ~~Typically couldn't do much better than the demonstrator.~~

   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!

write me a limerick about the university of utah

There once was a school up on high,
Where the mountains and red colors vie,
At Utah they cheer,
With loud voices you'll hear,
As the Utes let their spirit fly high!

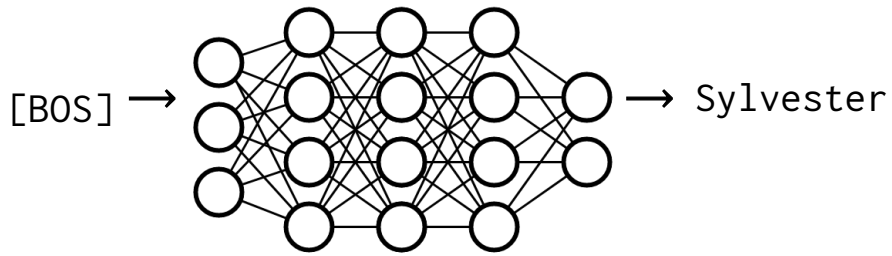
High-Level Recipe

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

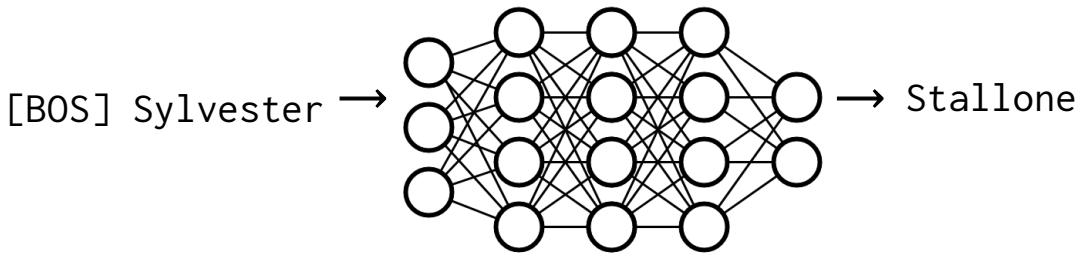
Preliminaries: Language Models

- 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.

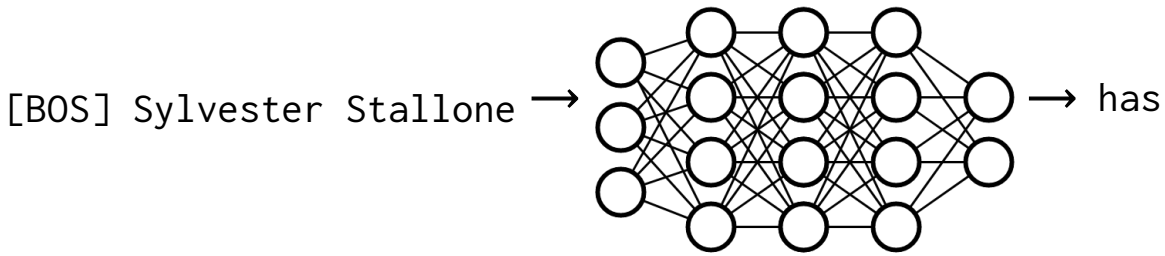
Neural language modeling



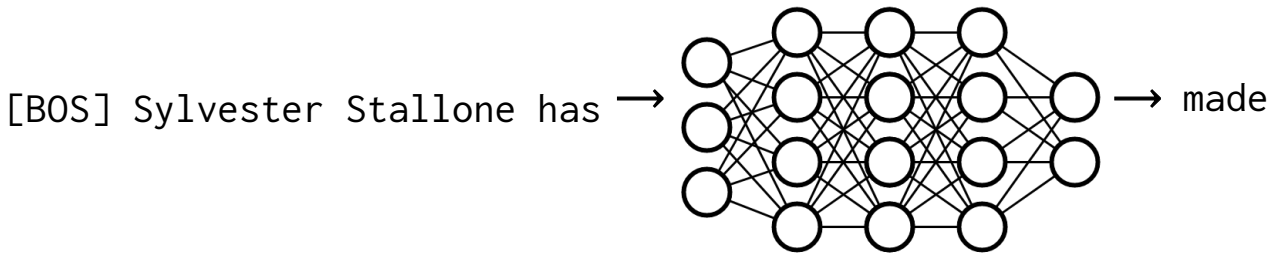
Neural language modeling

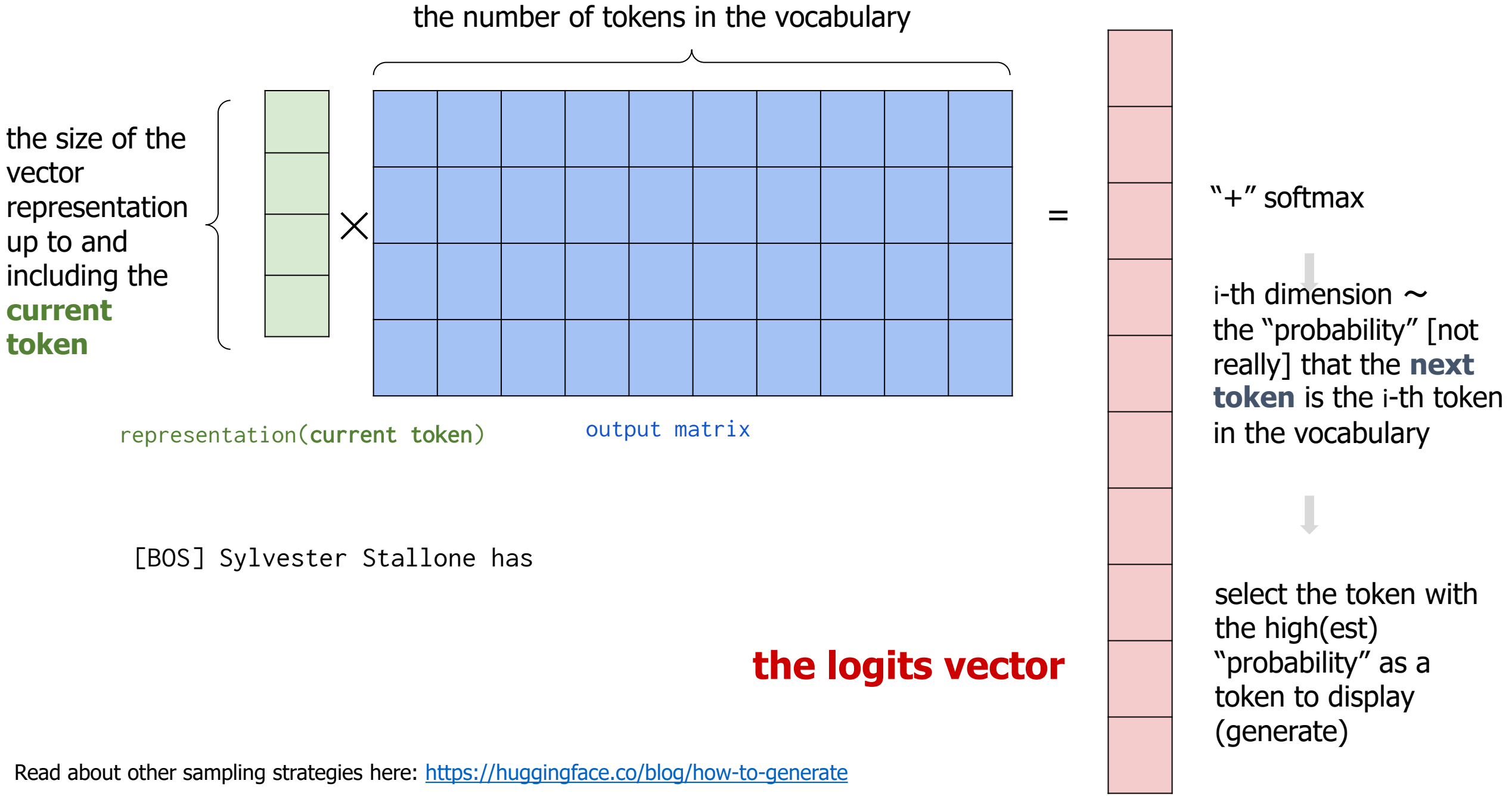


Neural language modeling



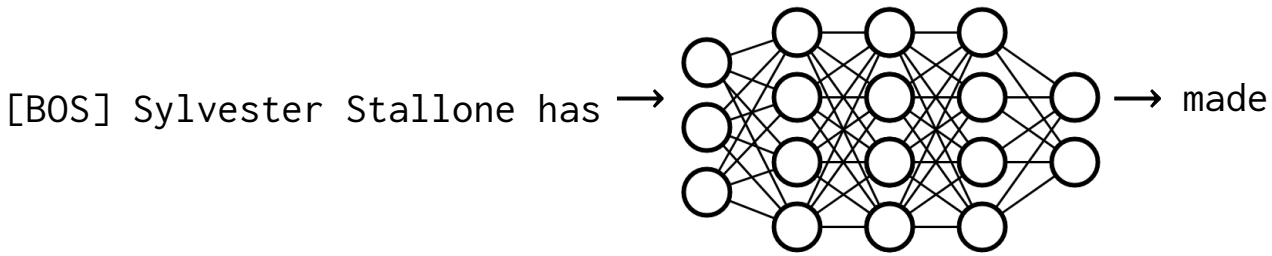
Neural language modeling





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

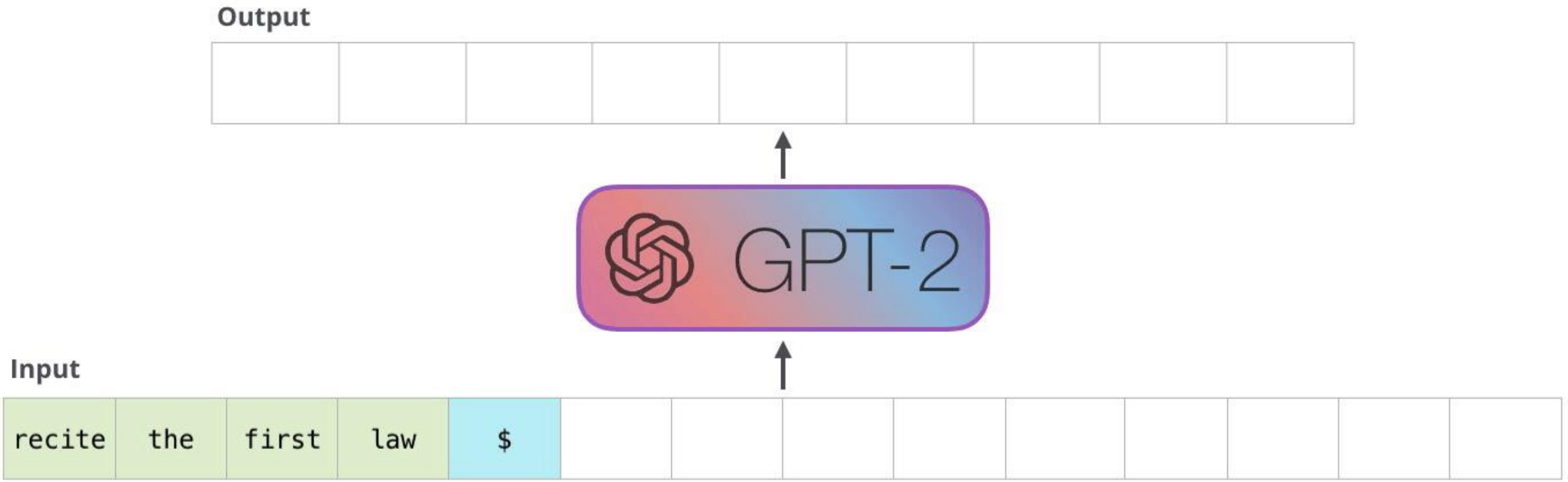
Neural language modeling

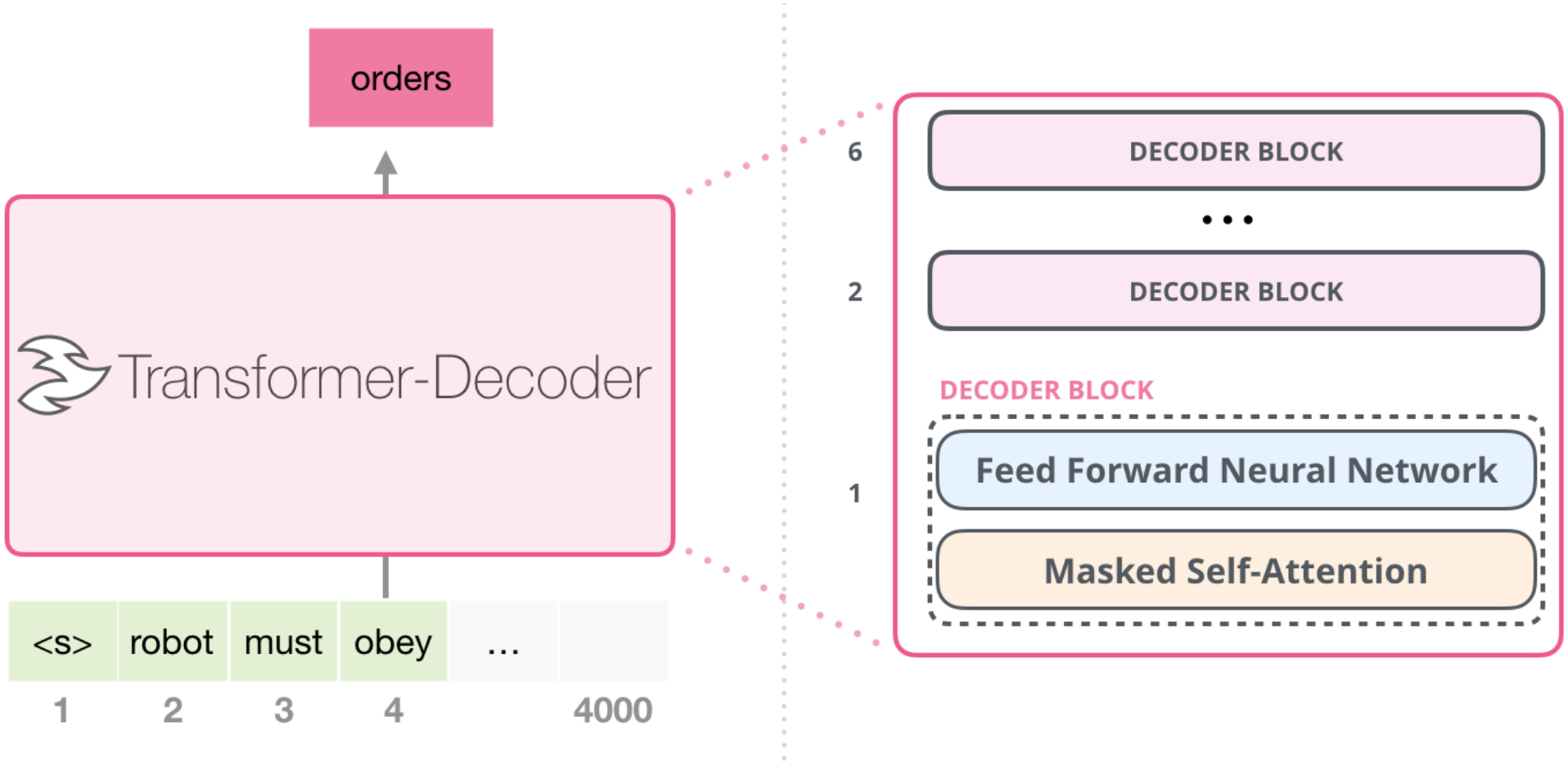


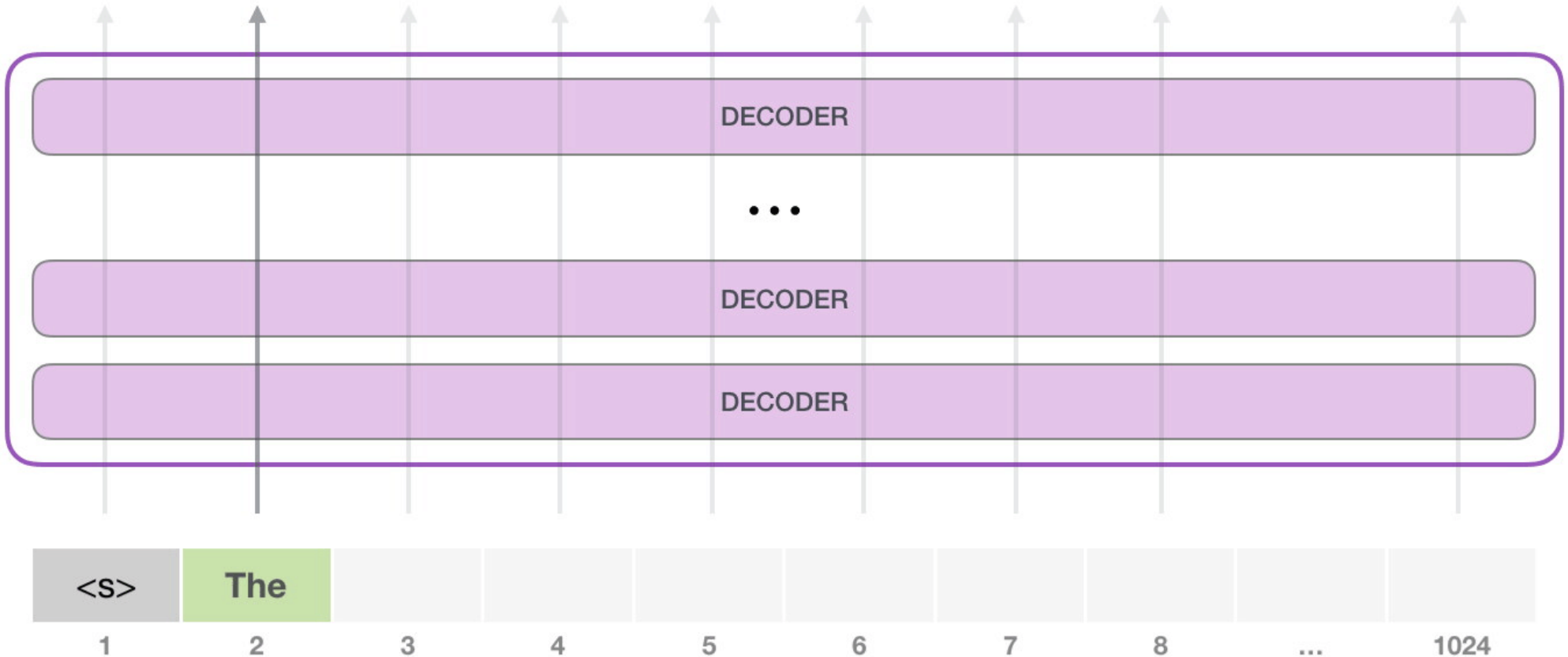
Problems:

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

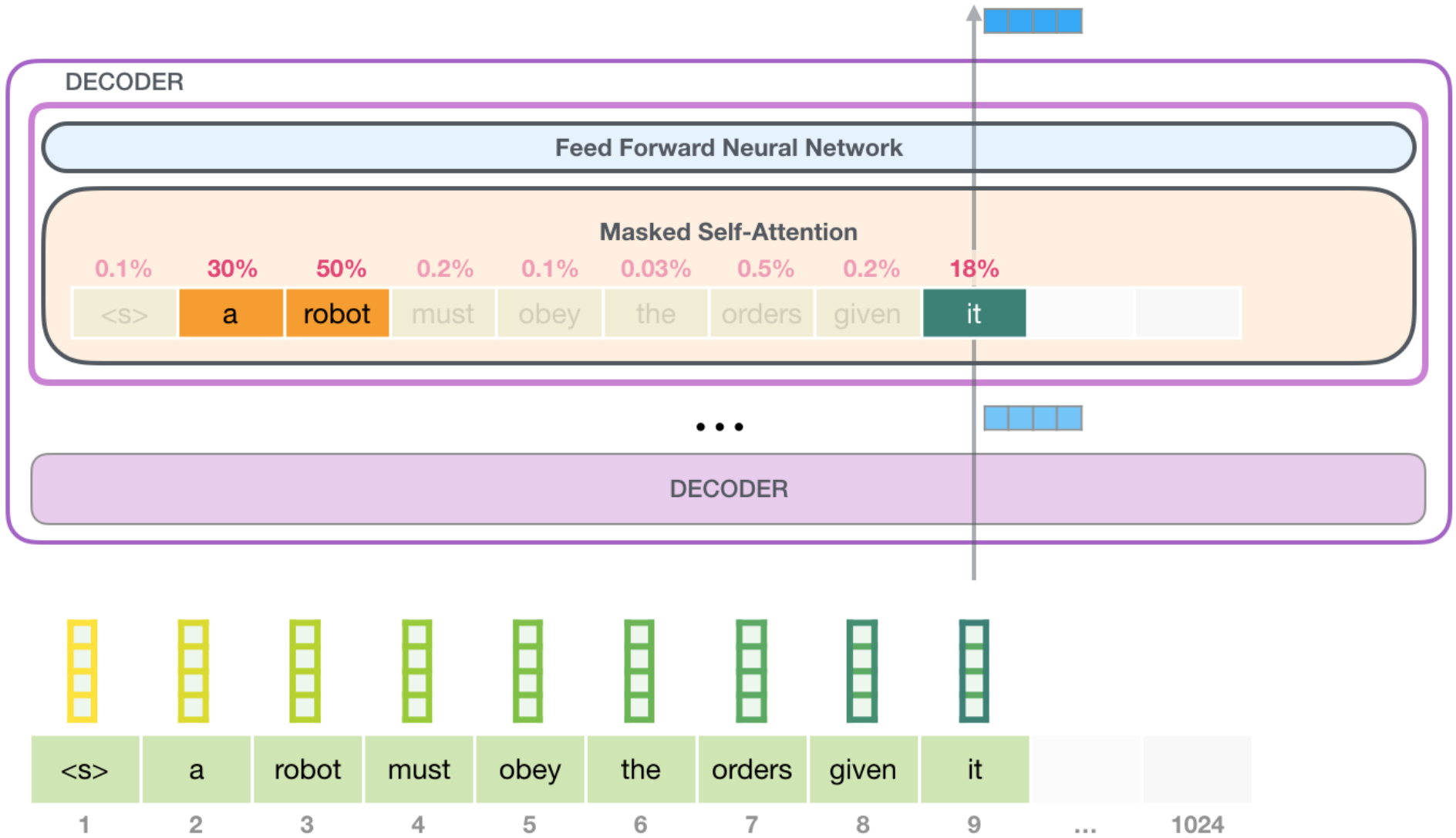
Large Language Models

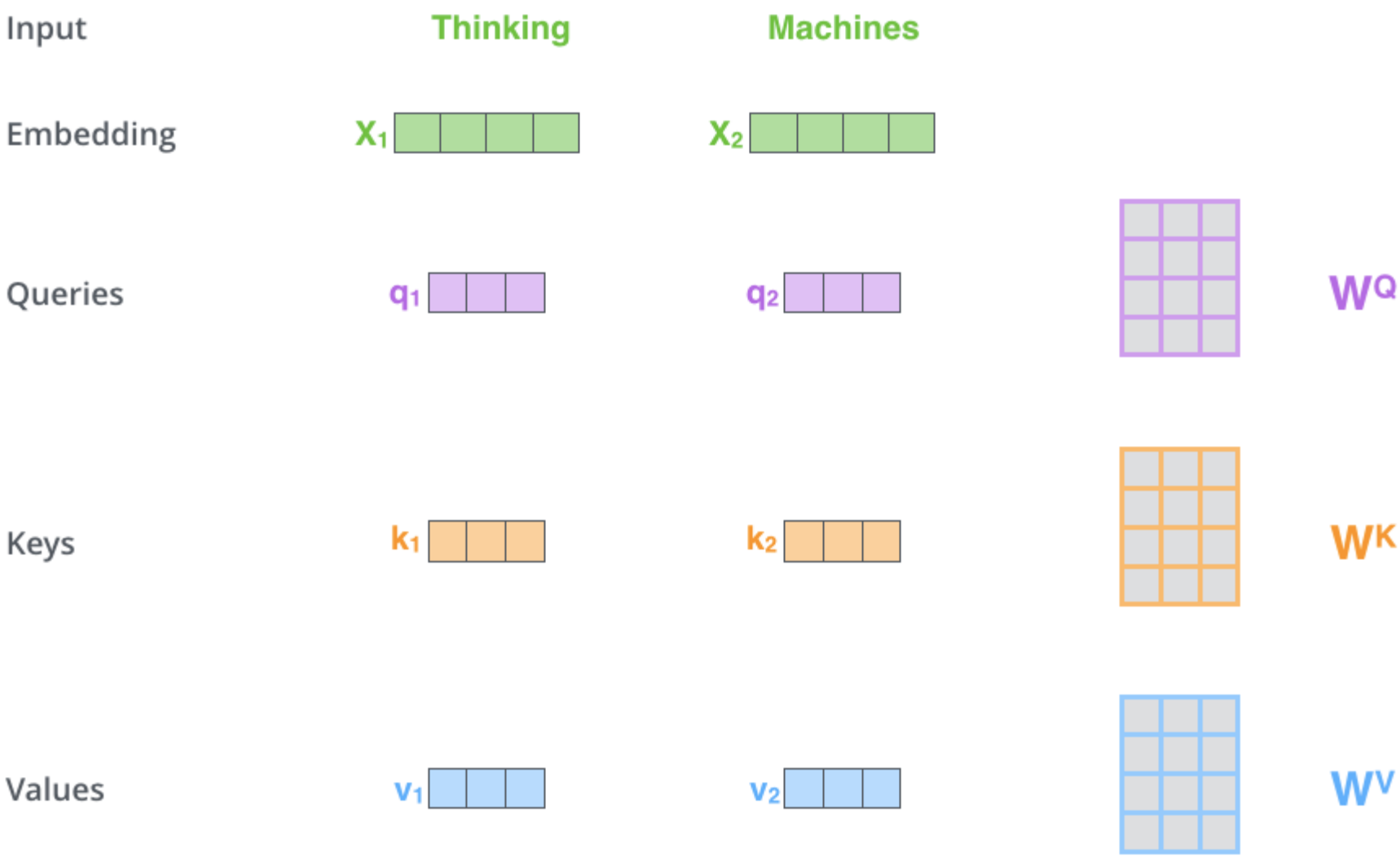



orders
Transformer-Decoder
<s>
robot
must
obey
...
1
2
3
4
4000
6
2
1
DECODER BLOCK
DECODER BLOCK
DECODER BLOCK
Feed Forward Neural Network
Masked Self-Attention

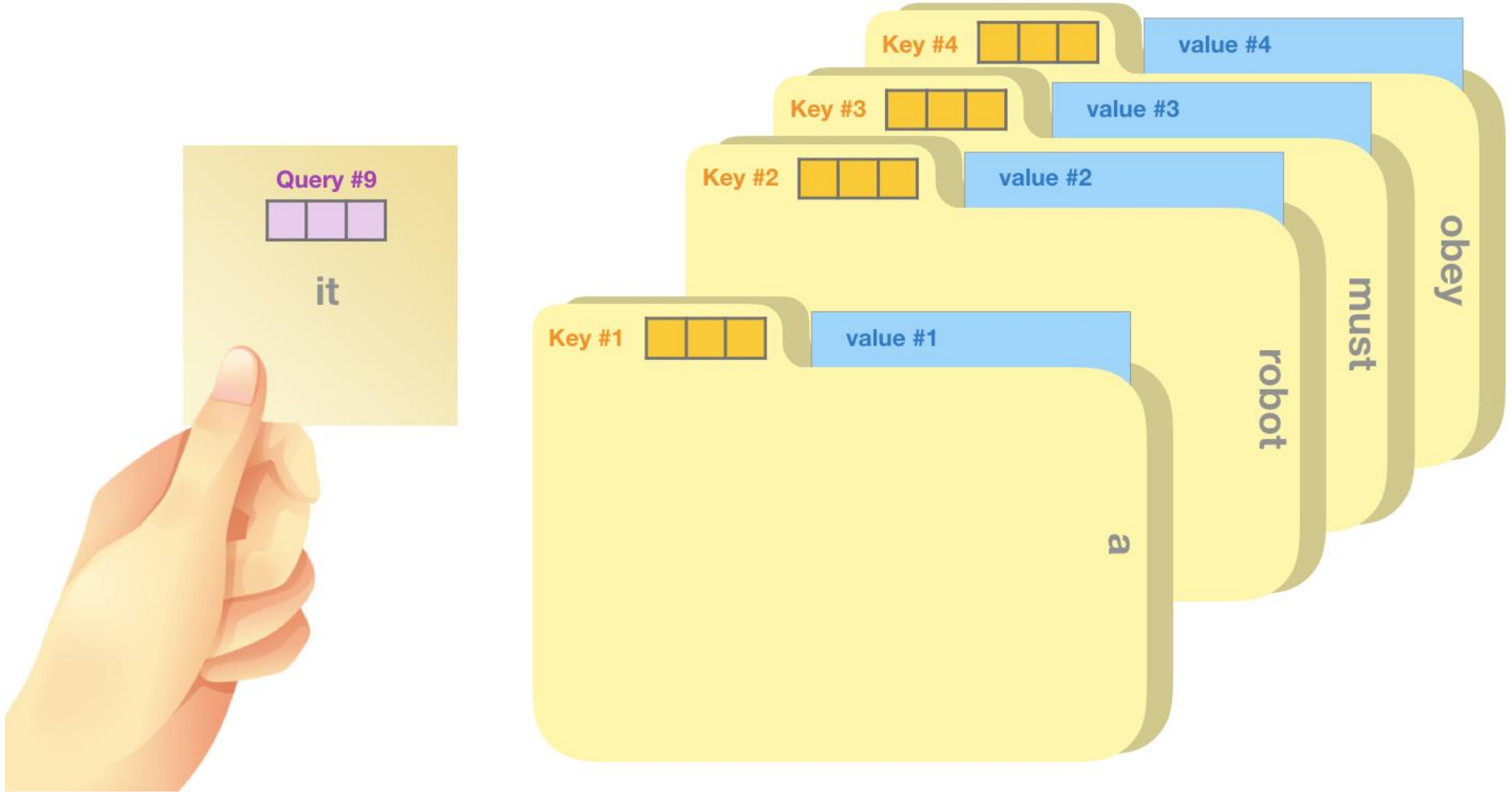

DECODER
DECODER
DECODER
<s>
The
1
2
3
4
5
6
7
8
...
1024


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

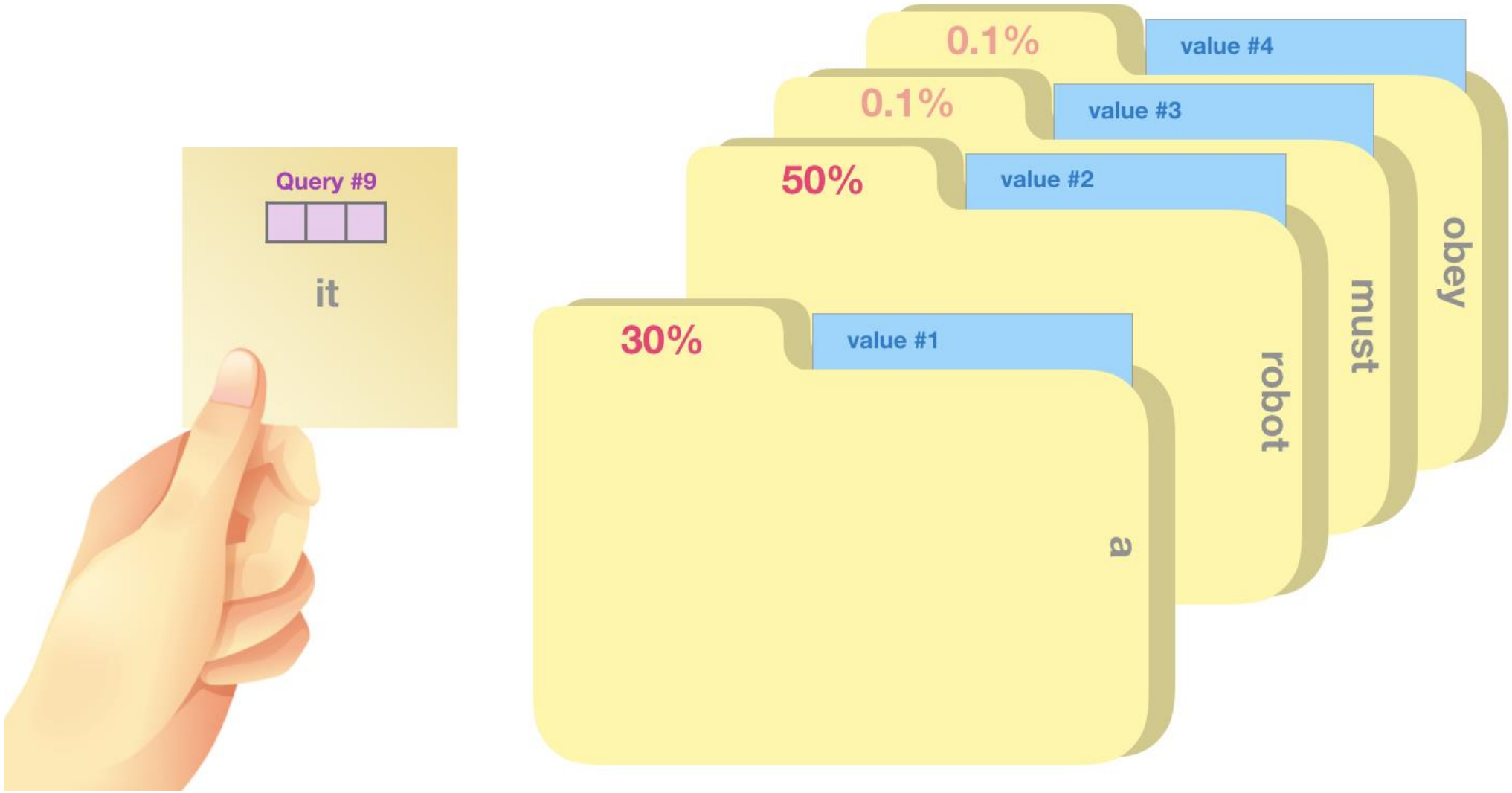

DECODER
Feed Forward Neural Network
Masked Self-Attention
0.1%
30%
50%
0.2%
0.1%
0.03%
0.5%
0.2%
18%
<s>
a
robot
must
obey
the
orders
given
it
DECODER
<s>
a
robot
must
obey
the
orders
given
it
1
2
3
4
5
6
7
8
9
...
1024

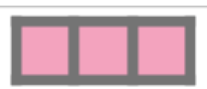

Input
Thinking
Machines
Embedding
X_1
X_2
Queries
q_1
q_2
W^Q
Keys
k_1
k_2
W^K
Values
v_1
v_2
W^V

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.

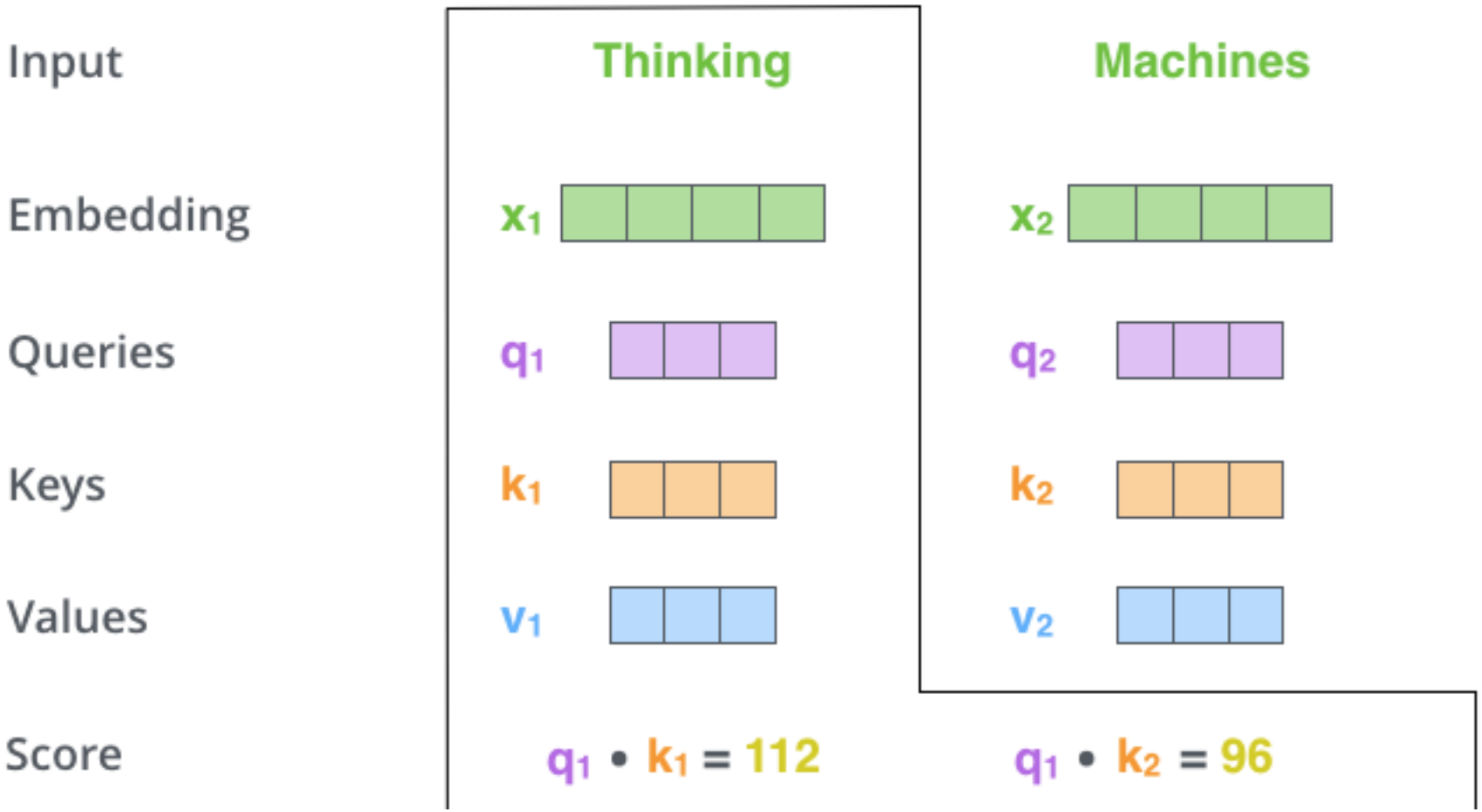


Word	Value vector	Score	Value X Score
<s>		0.001	
a		0.3	
robot		0.5	
must		0.002	
obey		0.001	
the		0.0003	
orders		0.005	
given		0.002	
it		0.19	
		Sum:	

Self attention: An example

For each word, compute the self attention.

First compute the dot product of its query vector with the key vector of all words in the sentence

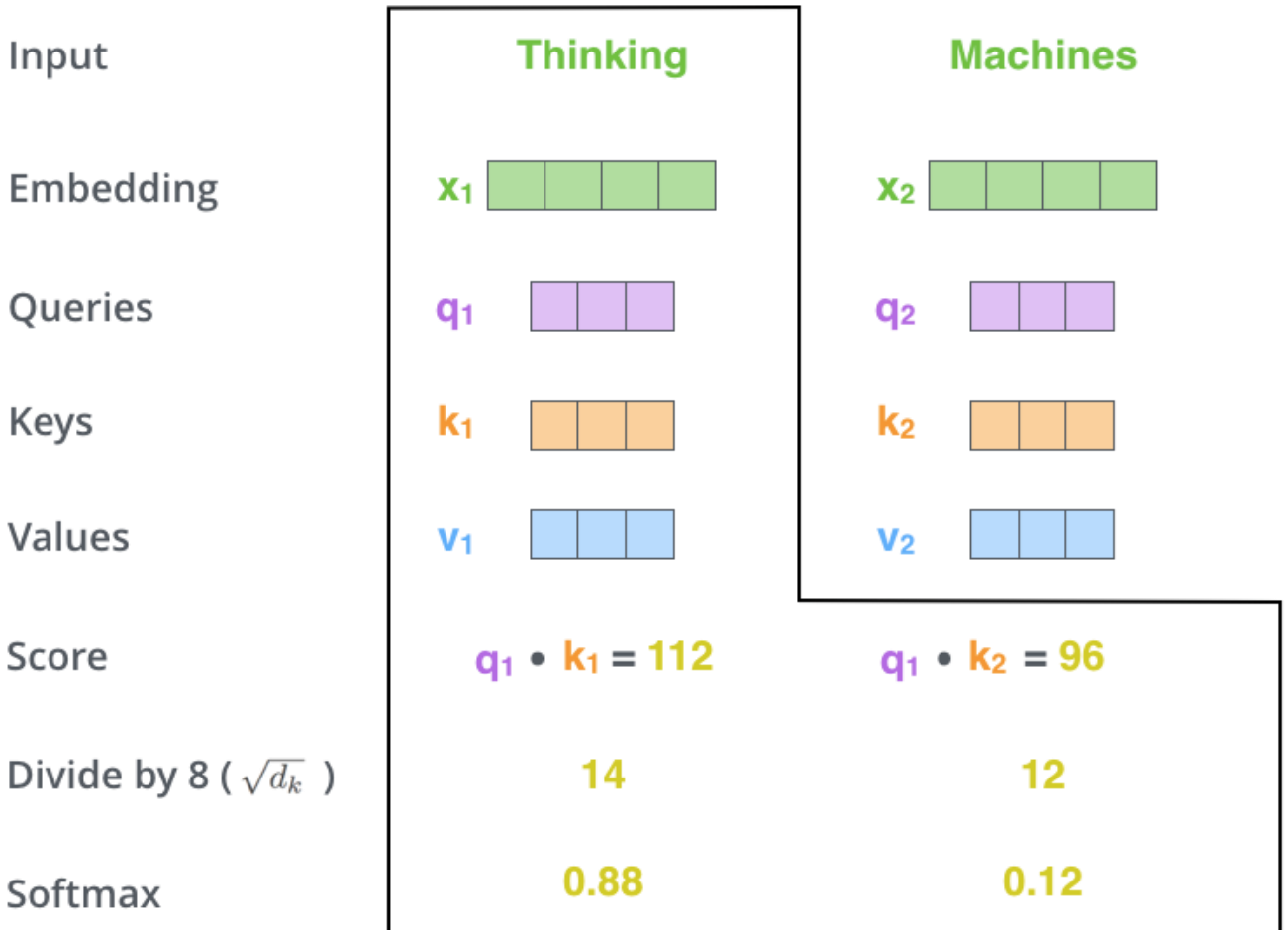


Images from https://jalammar.github.io/illustrated-transformer/

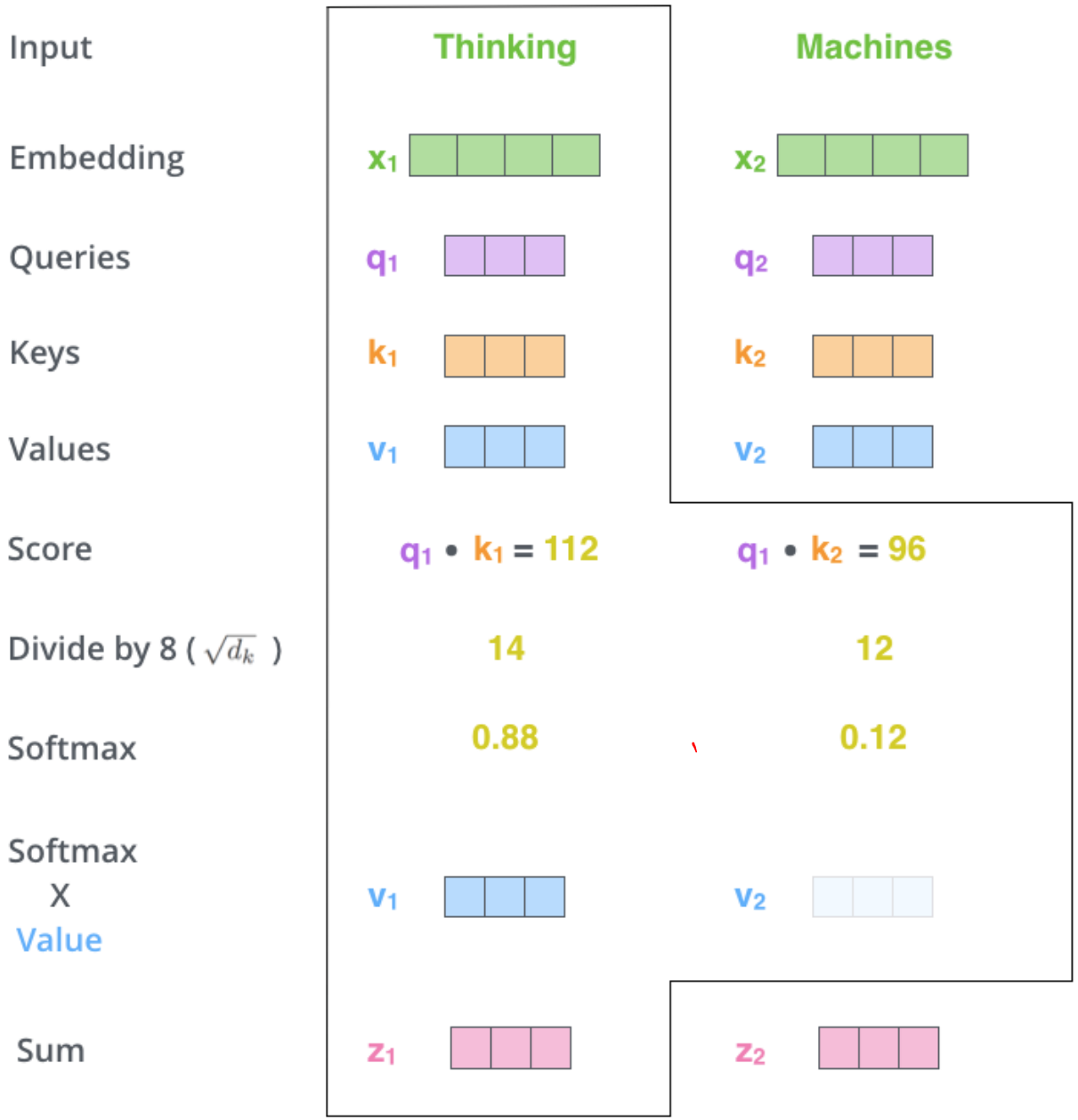
Self attention: An example

For each word, compute the self attention.

Then, normalize



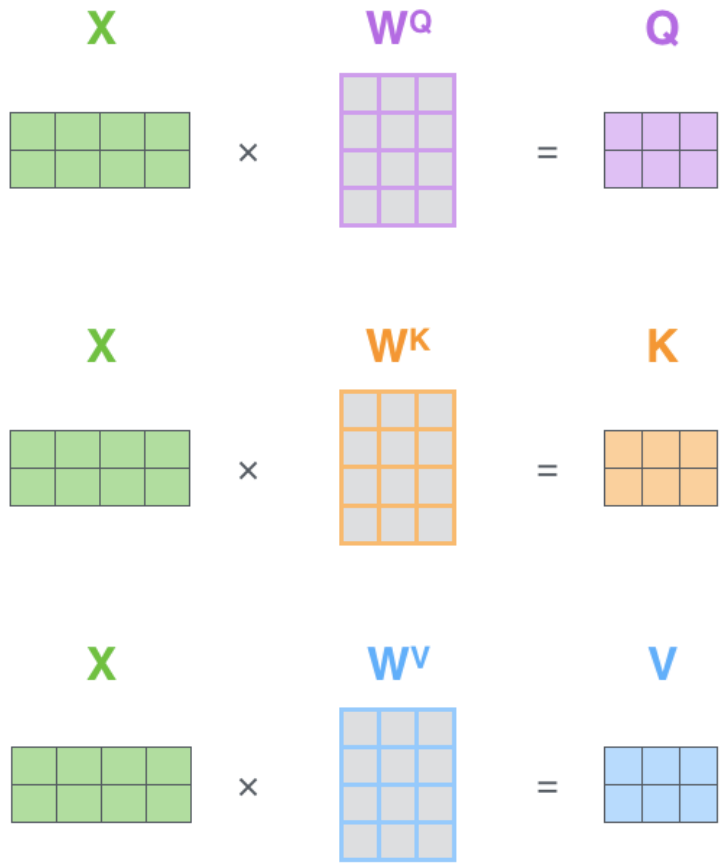
Images from https://jalammar.github.io/illustrated-transformer/

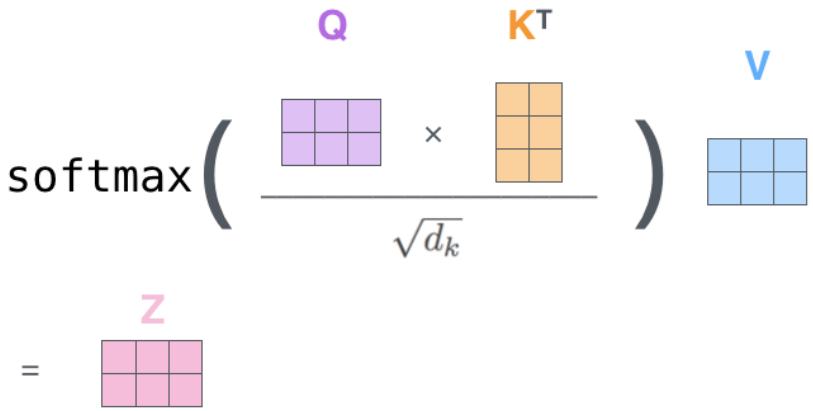


Use the attention probabilities to weigh the value vectors to produce the output vector for that word

Images from https://jalammar.github.io/illustrated-transformer/

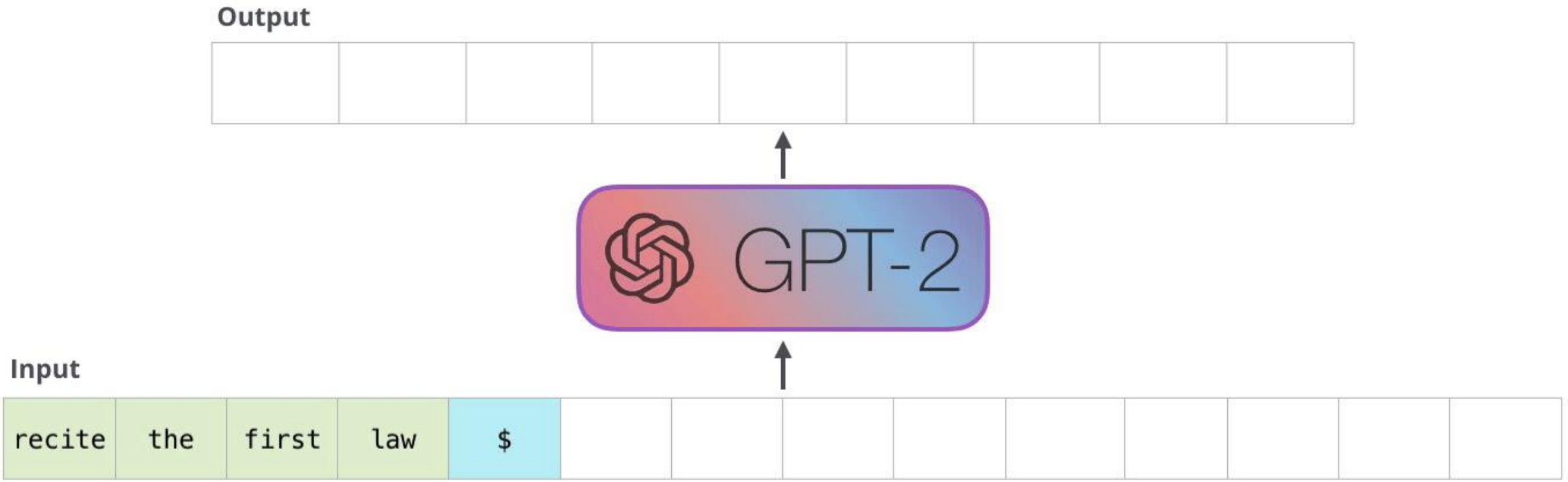
Self attention: Illustrated

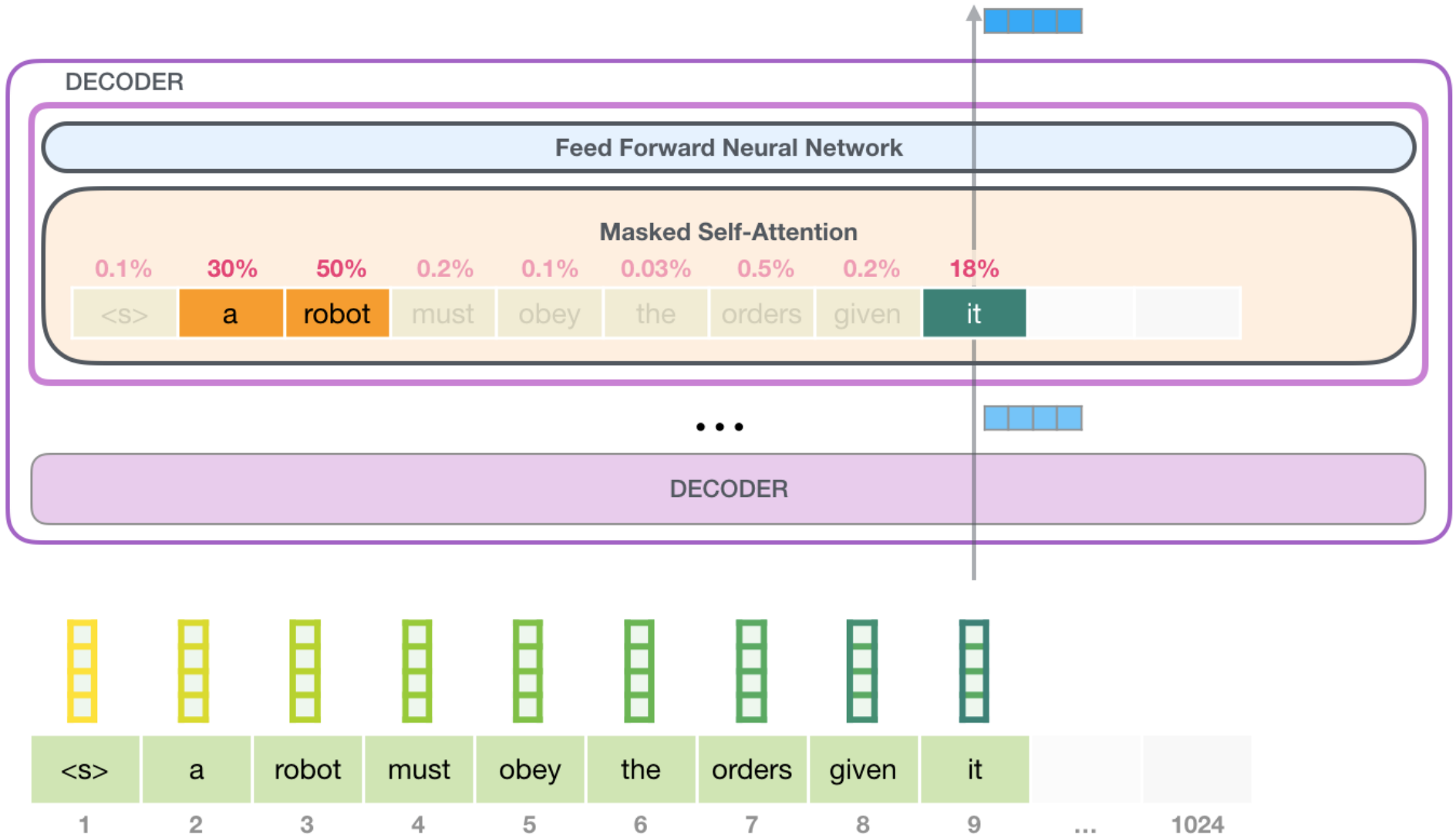


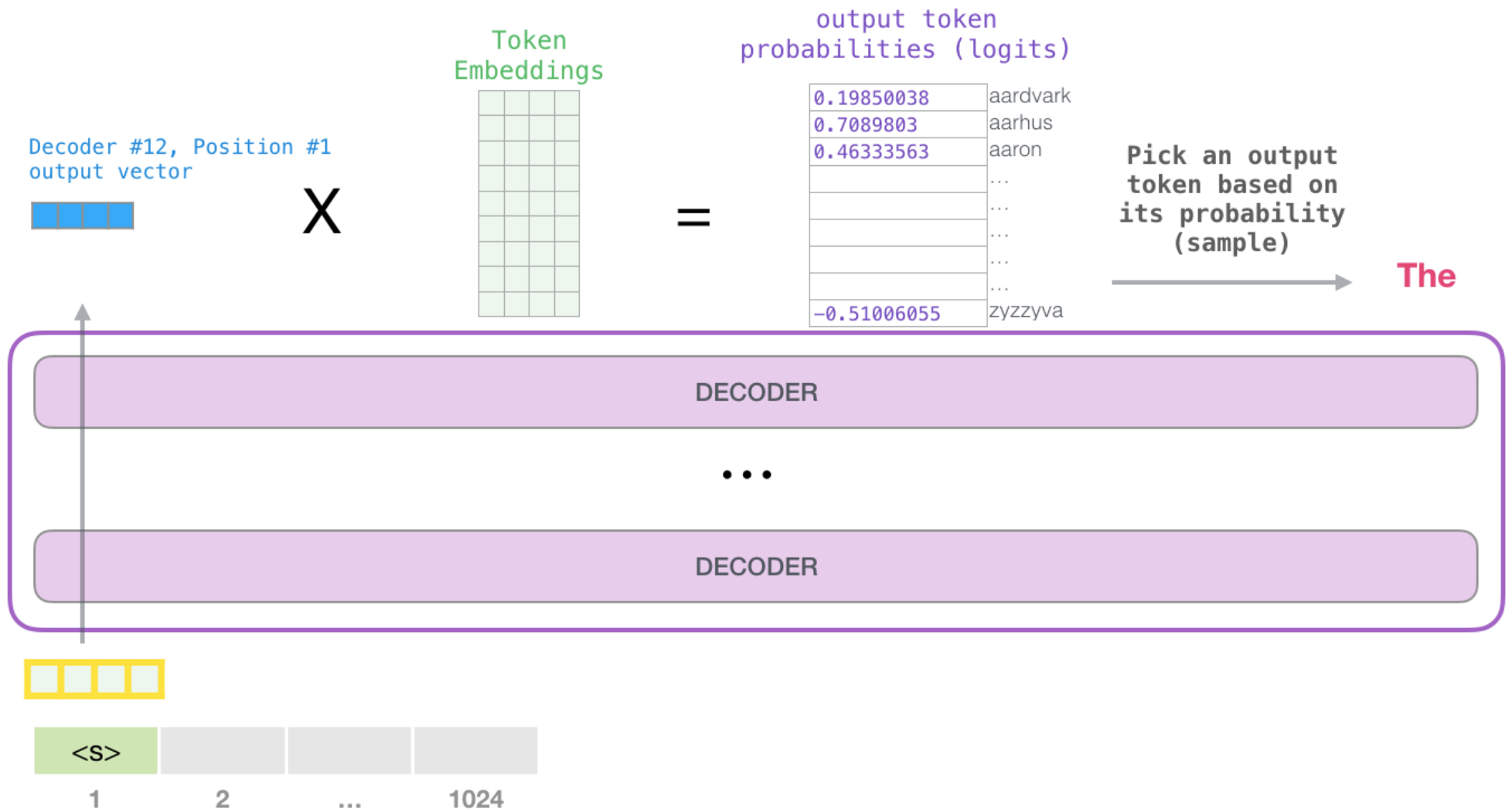


Images from https://jalammar.github.io/illustrated-transformer/

Large (Transformer-Based) Language Models




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


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

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!

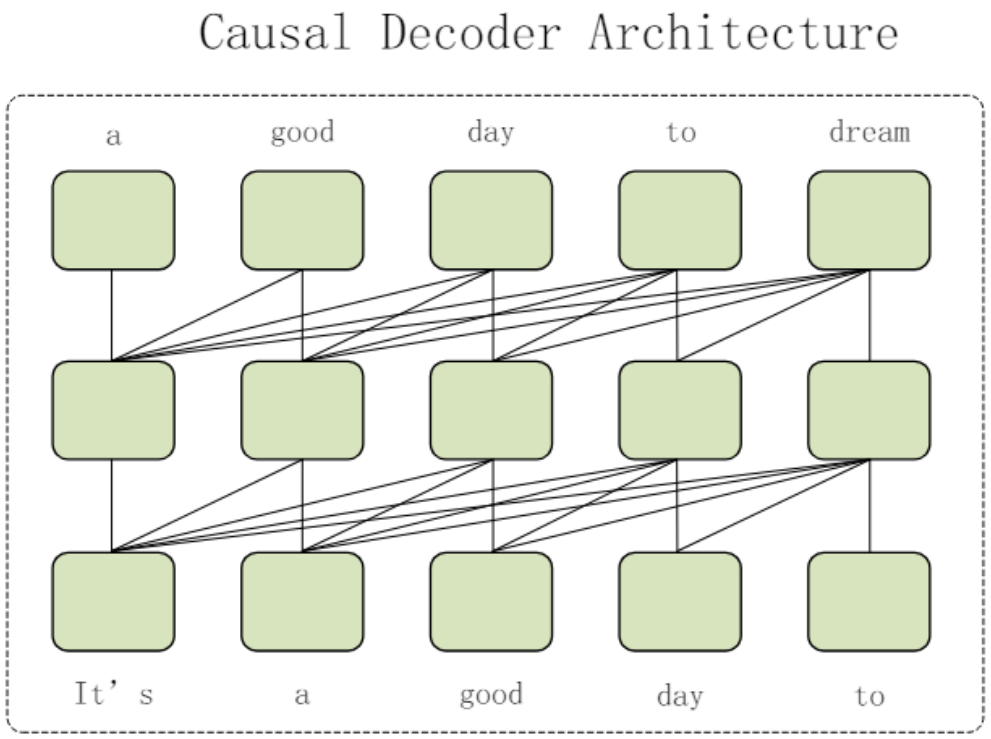
Table 1
Commonly used corpora information.

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 “

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ✗

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

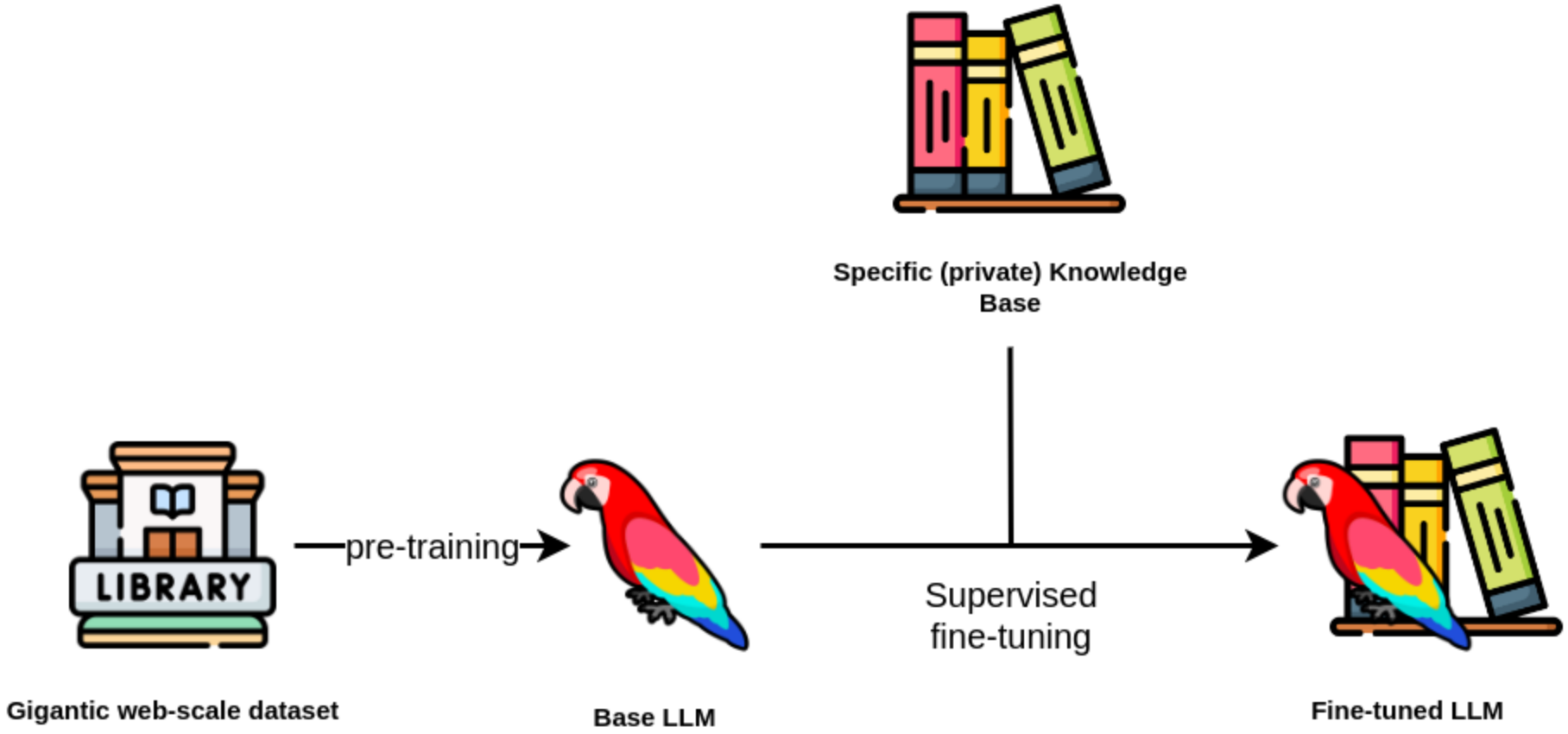
A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

https://arxiv.org/abs/2201.11903

High-Level Recipe for ChatGPT

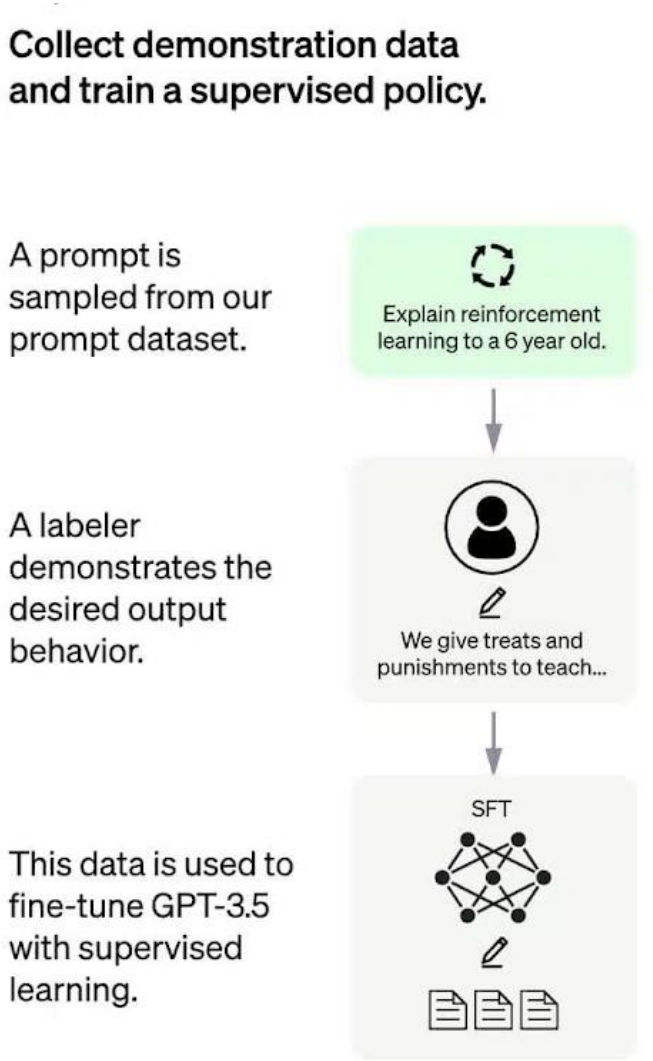
1. Unsupervised pre-training
2. **Supervised finetuning (behavioral cloning) from human demonstrations**
3. Collect preference rankings over outputs to train a reward function
4. Perform policy gradient updates using RL with learned reward

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)

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

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.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3.5 with supervised learning.

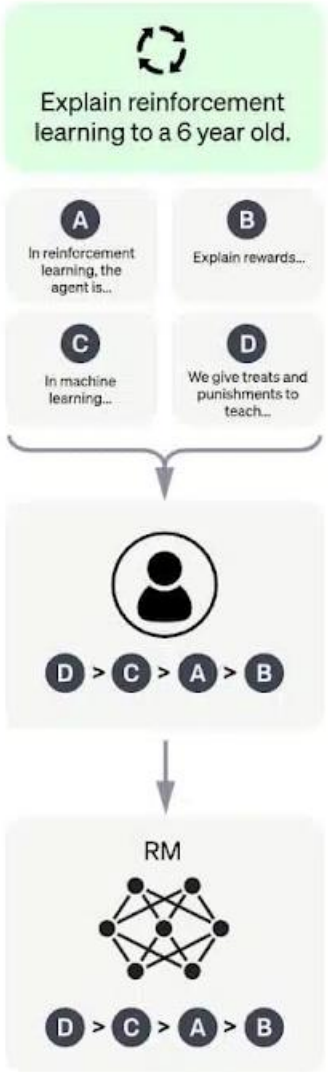
Step 2

Collect comparison data and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

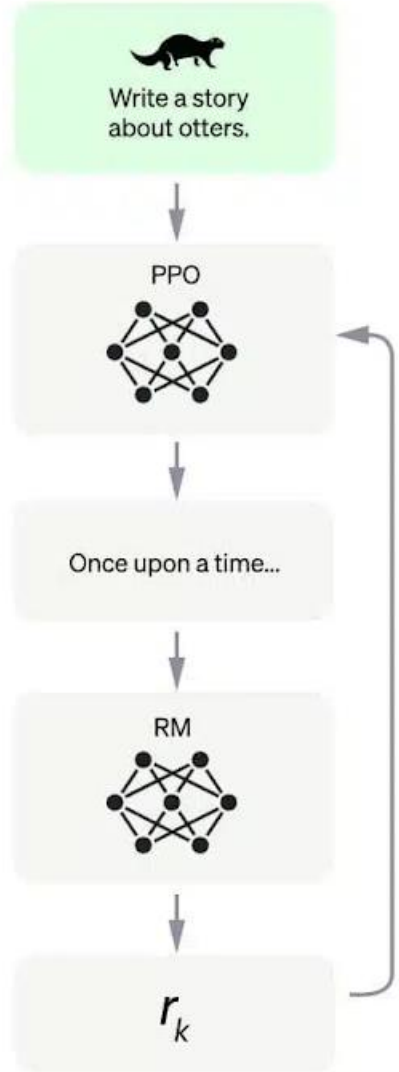
A new prompt is sampled from the dataset.

The PPO model is initialized from the supervised policy.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



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

$$\tau_1 \prec \tau_2$$

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

Bradley-Terry pairwise ranking loss

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

Loss from the InstructGPT paper

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

After training the reward, normalize it by subtracting the average reward of the labeler demonstrations before doing RL (make demos have mean reward of zero).

Optimizing the learned reward

- The paper calls this a (contextual) bandit problem. Why?

- Is this correct?

Contextual bandit setting?

- Context is prompt. Action is response. Reward based on prompt+completion. Then episode ends.

$$\begin{aligned}\text{objective}\,(\phi) = & E_{(x,y)\sim D_{\pi_\phi^{\mathrm{RL}}}}\left[r_\theta(x,y) - \beta \log\left(\pi_\phi^{\mathrm{RL}}(y \mid x)/\pi^{\mathrm{SFT}}(y \mid x)\right)\right] + \\ & \gamma E_{x\sim D_{\text{pretrain}}}\left[\log(\pi_\phi^{\mathrm{RL}}(x))\right]\end{aligned} \tag{2}$$

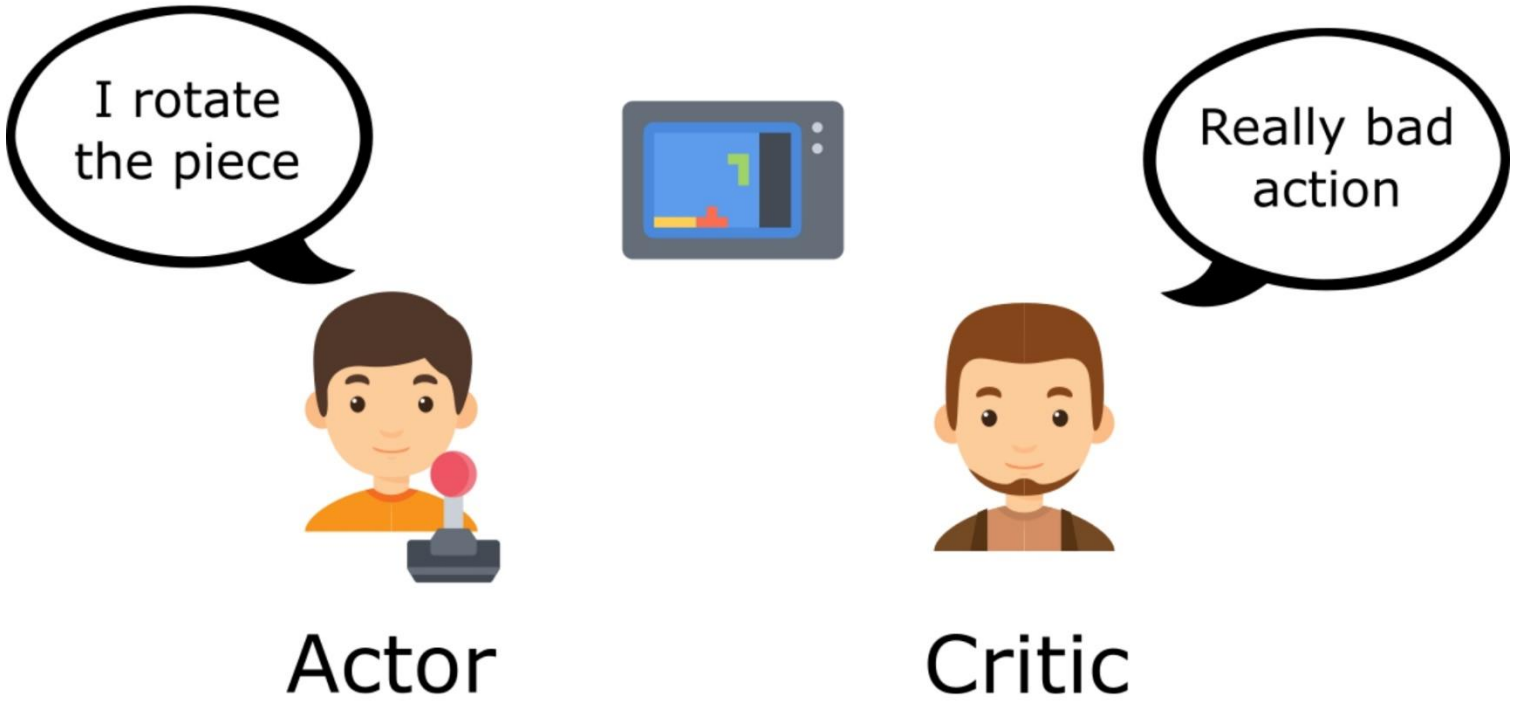
Contextual bandit setting?

- Context is prompt. Action is response. Reward based on prompt+completion. Then episode ends

$$\text{objective}\,(\phi) = E_{(x,y)\sim D_{\pi_\phi^{\text{RL}}}}\left[r_\theta(x,y) - \beta \log\left(\pi_\phi^{\text{RL}}(y \mid x)/\pi^{\text{SFT}}(y \mid x)\right)\right] + \gamma E_{x\sim D_{\text{pretrain}}}\left[\log(\pi_\phi^{\text{RL}}(x))\right] \quad (2)$$

- But we also have a sequence of actions. How does the reward work?
  - Per-token divergence penalty
  - Final token reward based on prompt+completion
- Credit assignment is a problem…

PPO and GRPO



Instructor: Daniel Brown --- University of Utah

Proximal Policy Iteration (PPO)

- Measure how much we are changing policy compared with previous policy using a ratio:

$$ratio_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}$$

- Clip policy gradient update based on this ratio:

$$\theta_{k+1} = \arg\max_\theta \mathop{\mathrm{E}}_{s,a\sim\pi_{\theta_k}} [L(s, a, \theta_k, \theta)]$$

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

Algorithm 1 PPO-Clip

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

for $k = 0, 1, 2, \ldots$ **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:

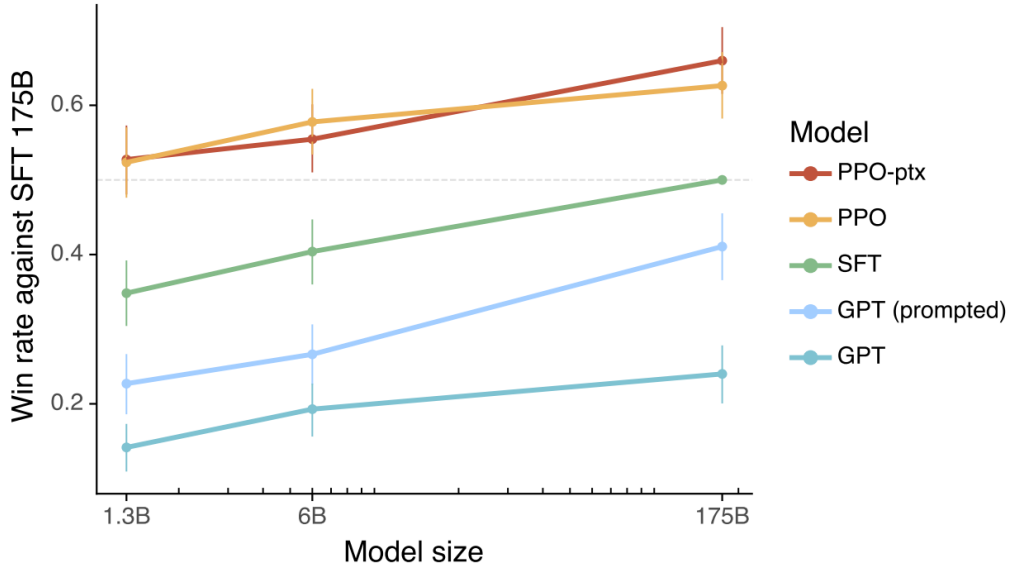
$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

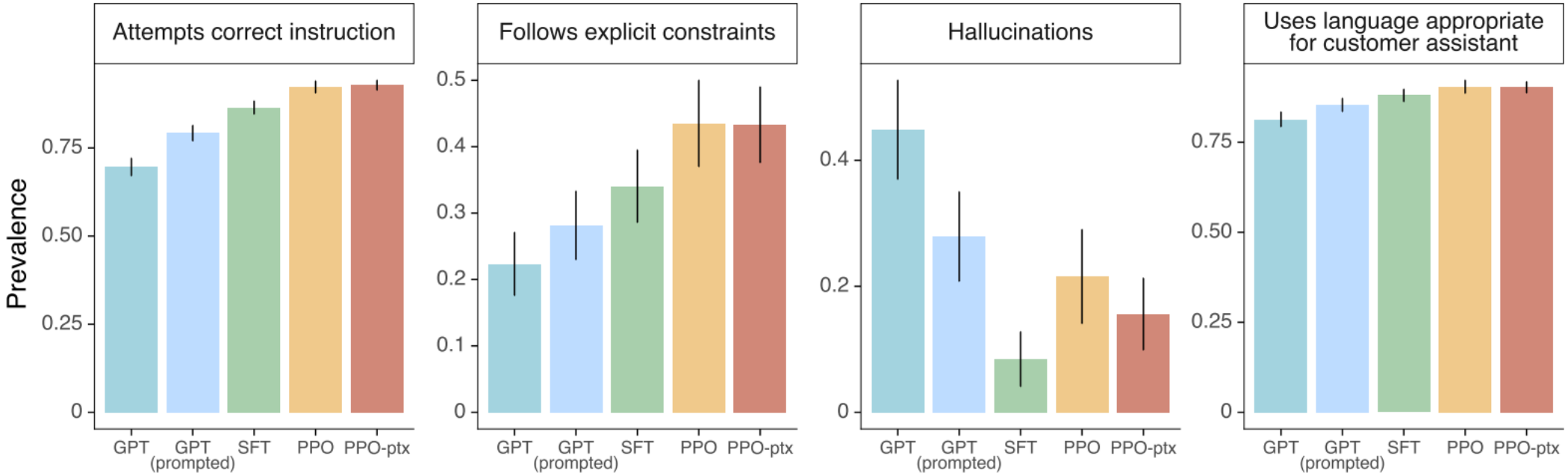
typically via some gradient descent algorithm.

end for

Results

- Labelers prefer InstructGPT over GPT-3





Interesting questions/findings

- Alignment tax. Is alignment a cure or a mask/filter?
- Generalization to foreign language instruction following and coding
- Whose values are we aligning to?
- Do LLMs actually learn to be helpful or just to mimic helpfulness? Does this distinction even make sense or matter?