Lecture 16:
Post-Training LLMs

Instructor: Swabha Swayamdipta
USC CSCI 444 NLP
Nov 3, 2025

Some slides adapted from Qinyuan Ye, Justin Cho, Dan Jurafsky and Chris Manning

Announcements + Logistics

- Today: Quiz 4
- Wed: Paper Discussions in Class
  - Before class: Read 3 papers, okay not to understand everything
  - During class: Discuss the paper and your questions with your classmates
  - After class: Submit a 500 word report on the readings, ranking your papers and justifying why
  - Three papers
    - DPO
    - LoRA
    - InstructGPT
- By Wednesday: Progress Report Grades

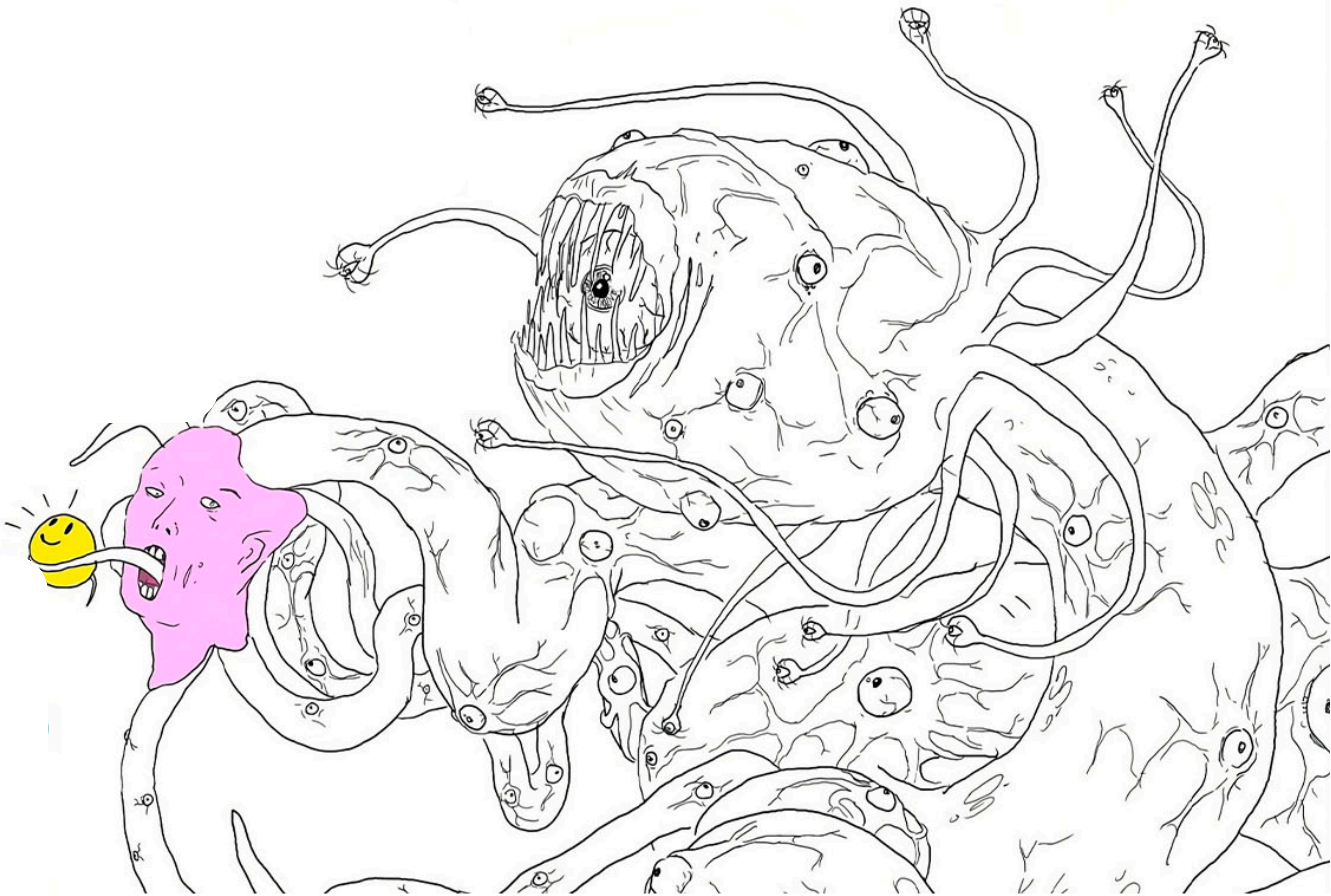
Quiz 4

Password: nucleus

Lecture Outline

- Announcements and Logistics
- Quiz 4
- Instruction Tuning
- Prompting
- Preference Tuning

Training LLMs



A significant, yet small part of the LM training phase

Shoggoth; slide credit: Justin Cho

USC Viterbi

Three Stages of Training LLMs

Stage 1

- Pre-training on large corpus of text
- Produces a **Base Language Model**
- Continued Pre-training for domain adaptation (optional; sometimes called mid-training, stage 1.5)

Stage 2

- Post-training for Task Adaptation
- Seq2Seq Instruction Tuning (Supervised Finetuning, different meaning than BERT-style classification tasks)

Stage 3

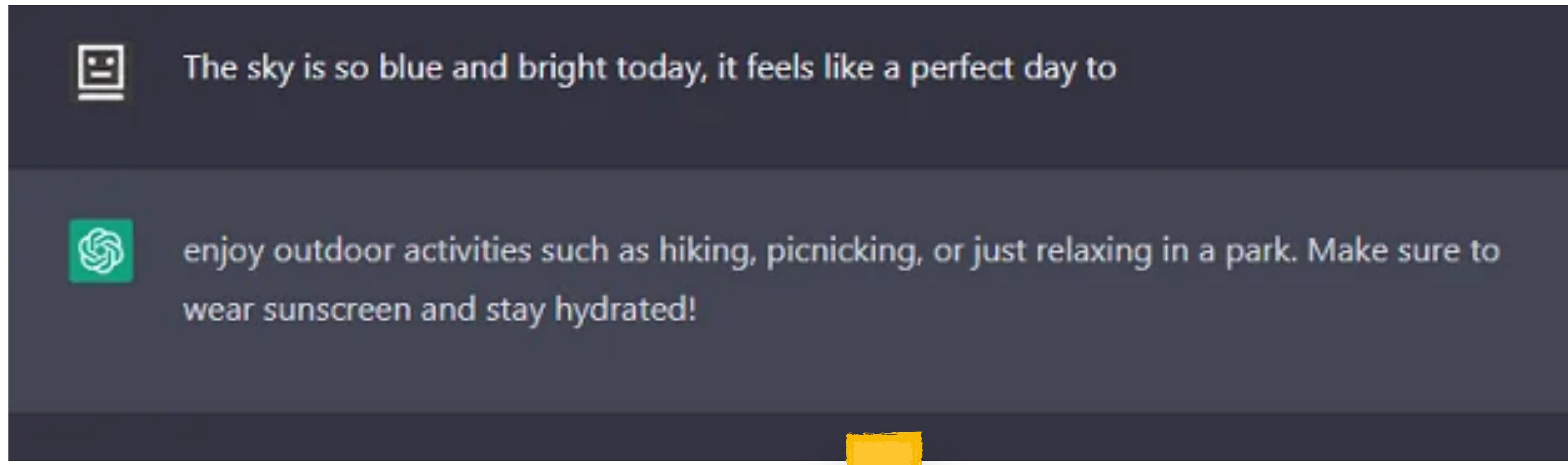
- Post-training for Preference Alignment
  - Either Reinforcement Learning with Human Feedback : Train a supervised classifier (reward model) on human demonstrations to provide feedback to LM. Reinforcement learning to maximize rewards given by reward model
  - Or, train with a preferred and a dispreferred generation (more popular now)

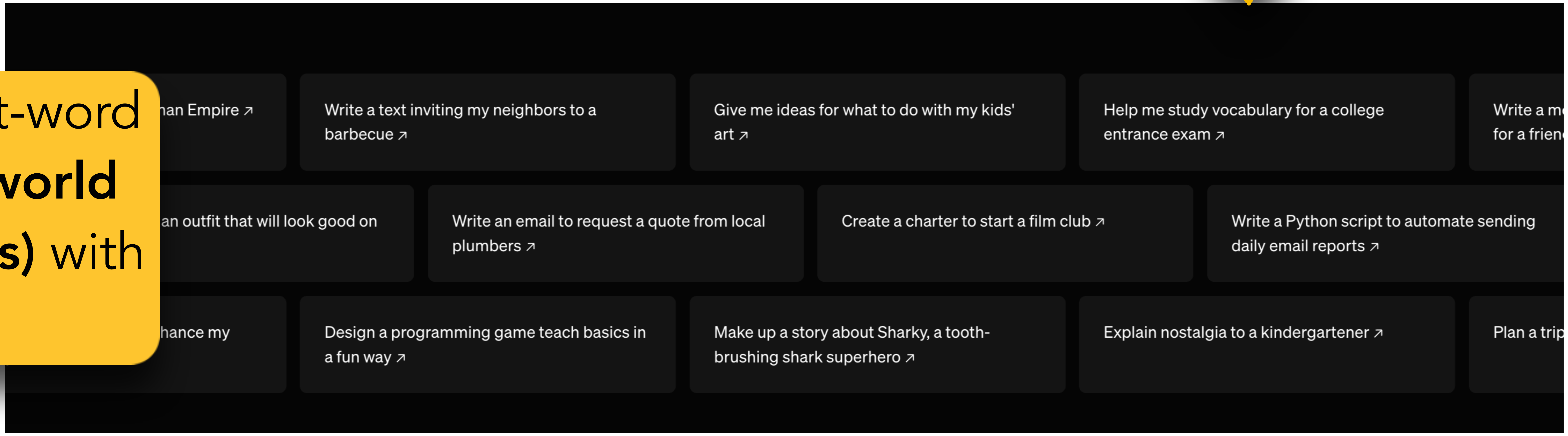
Inference: Prompting with Instructions and Demonstrations (also called examples, shots)

USC Viterbi

Pre-training and Post-training

- The term "training" is no longer specific enough :)
- Pre-training: Self-supervised, standard next token prediction
- Post-training: Supervision (sequence to sequence)
  - **Instruction-Tuning**: Supervision is not necessarily via labels, but sequence pairs. Labels in standard NLP benchmarks can be converted into sequence pairs
  - **Preference-Tuning**: Collects human judgments / preferences as rewards, or a pair of preferred / dispreferred generations





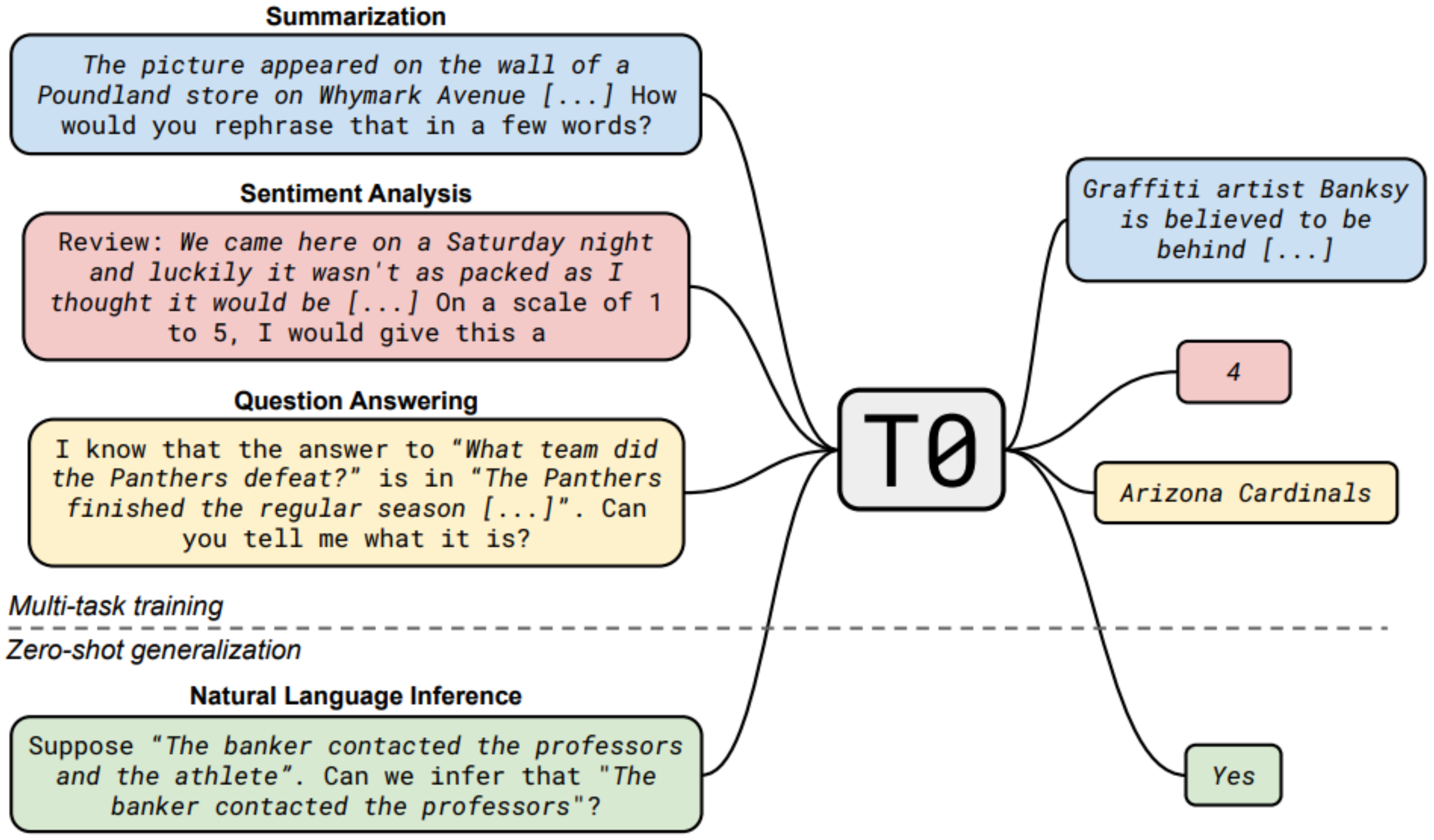
Post-training converts next-word completion models into **world models (agents, assistants)** with many capabilities!

USC Viterbi

Instruction-Tuning LLMs for Task Adaptation

Instruction Tuning

- Pretraining:
  - Train a model to continue a given context
- Instruction Tuning:
  - Train a model to follow varied instructions
  - Needed because the vast majority of pretraining is done on data which are not in the form of instructions
  - Fine-tuned (using the next-token-prediction objective) on a dataset of instructions together with correct responses

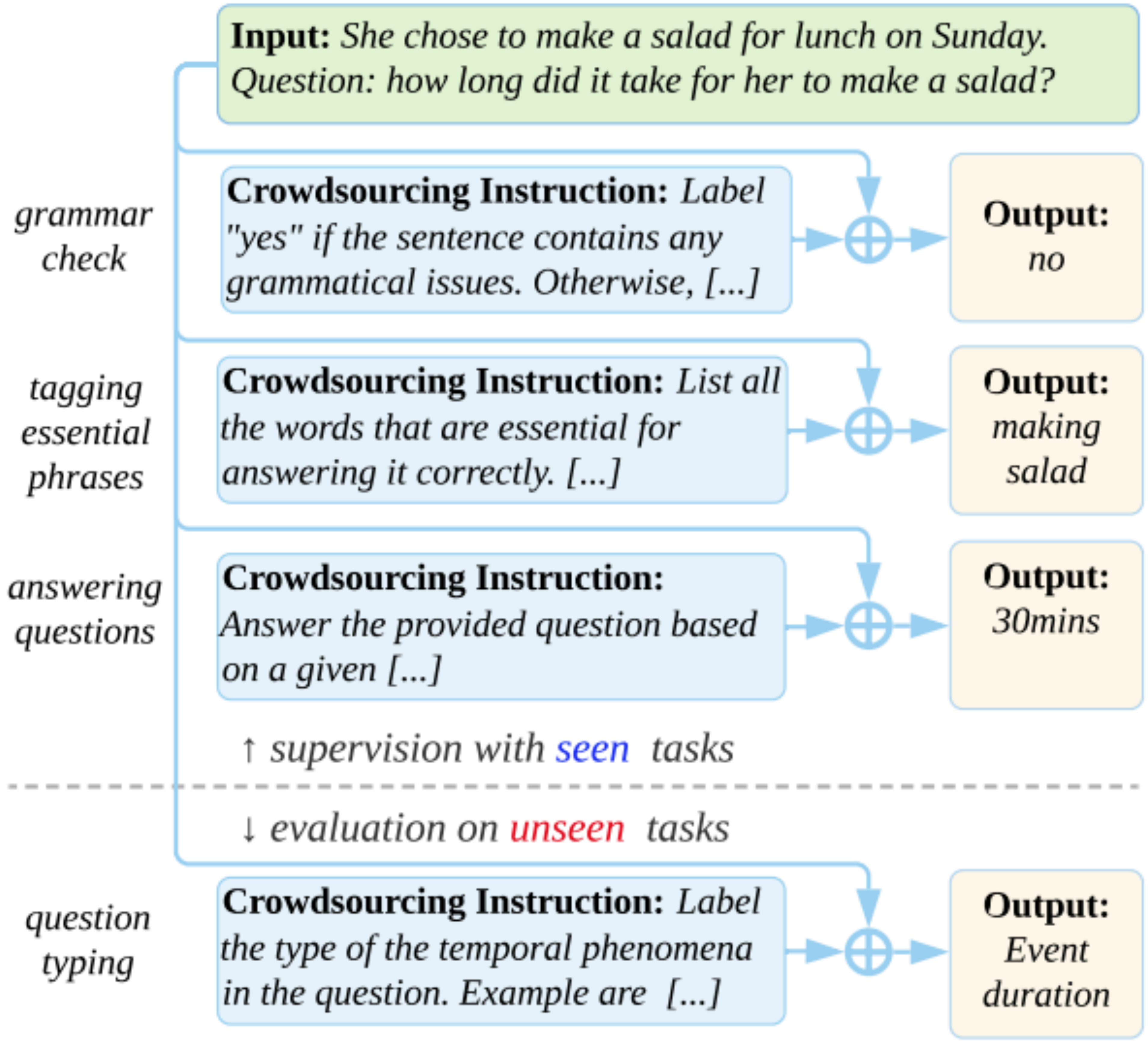


"Multitask Prompted Training Enables Zero-Shot Task Generalization" (Sahn et al., 2022)

USC Viterbi

Instruction Tuning and Task Generalization

- During instruction tuning, the model learns to follow instructions of given tasks
- At test time, it generalizes to follow instructions on unseen tasks!



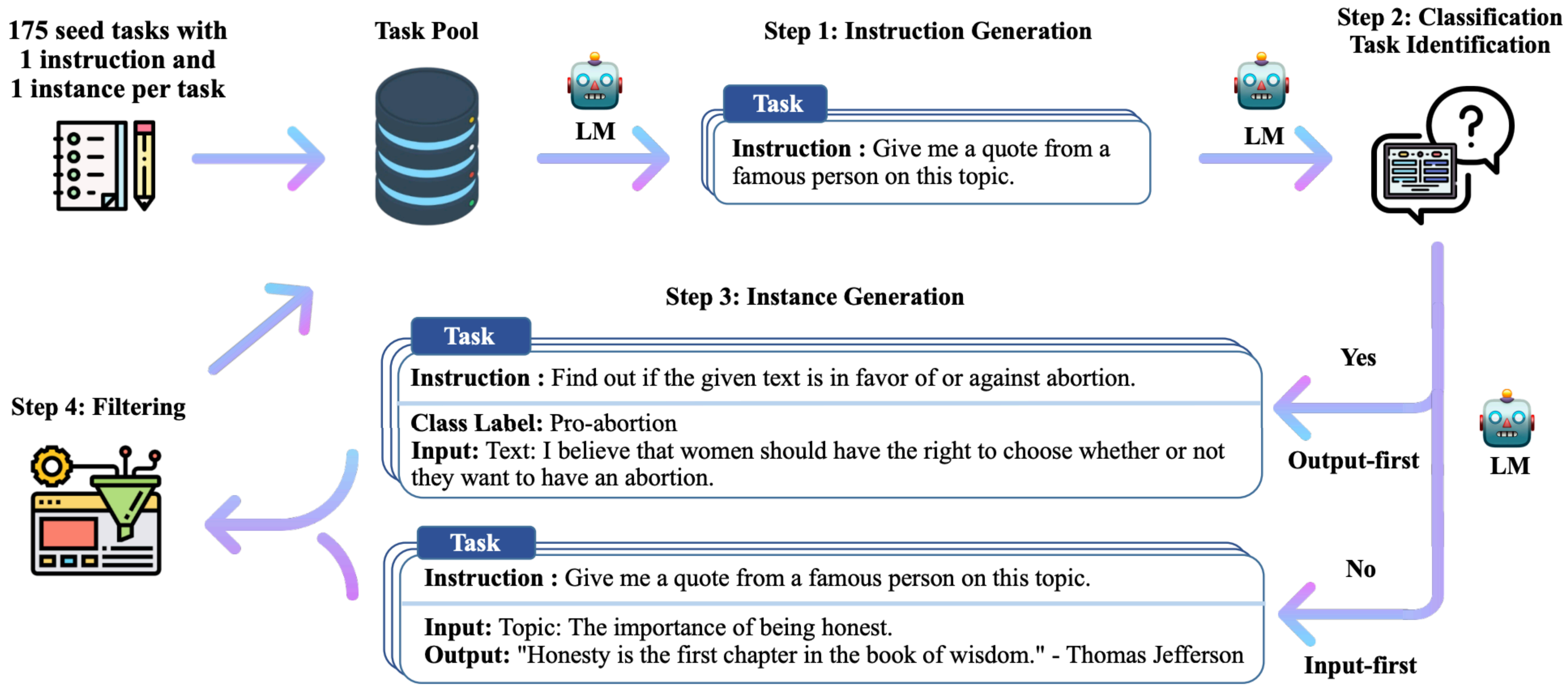
"Cross-Task Generalization via Natural Language Crowdsourcing Instructions" (Mishra et al., 2022)

Instruction Tuning Data

More data (instructions)
→ better model

Resource →	SUP-NATINST (this work)	NATINST (Mishra et al., 2022b)	CROSSFIT (Ye et al., 2021)	PROMPTSOURCE (Bach et al., 2022)	FLAN (Wei et al., 2022)	INSTRUCTGPT (Ouyang et al., 2022)
Has task instructions?	✓	✓	✗	✓	✓	✓
Has negative examples?	✓	✓	✗	✗	✗	✗
Has non-English tasks?	✓	✗	✗	✗	✓	✓
Is public?	✓	✓	✓	✓	✓	✗
Number of tasks	1616	61	269	176	62	–
Number of instructions	1616	61	–	2052	620	14378
Number of annotated tasks types	76	6	13	13*	12	10
Avg. task definition length (words)	56.6	134.4	–	24.8	8.2	–

Super-NaturalInstructions. (Wang et al., 2022)



Diverse data (instructions)
→ better model

"Self-Instruct: Aligning Language Models with Self-Generated Instructions" (Wang et al., 2023)

Instruction Tuning Data

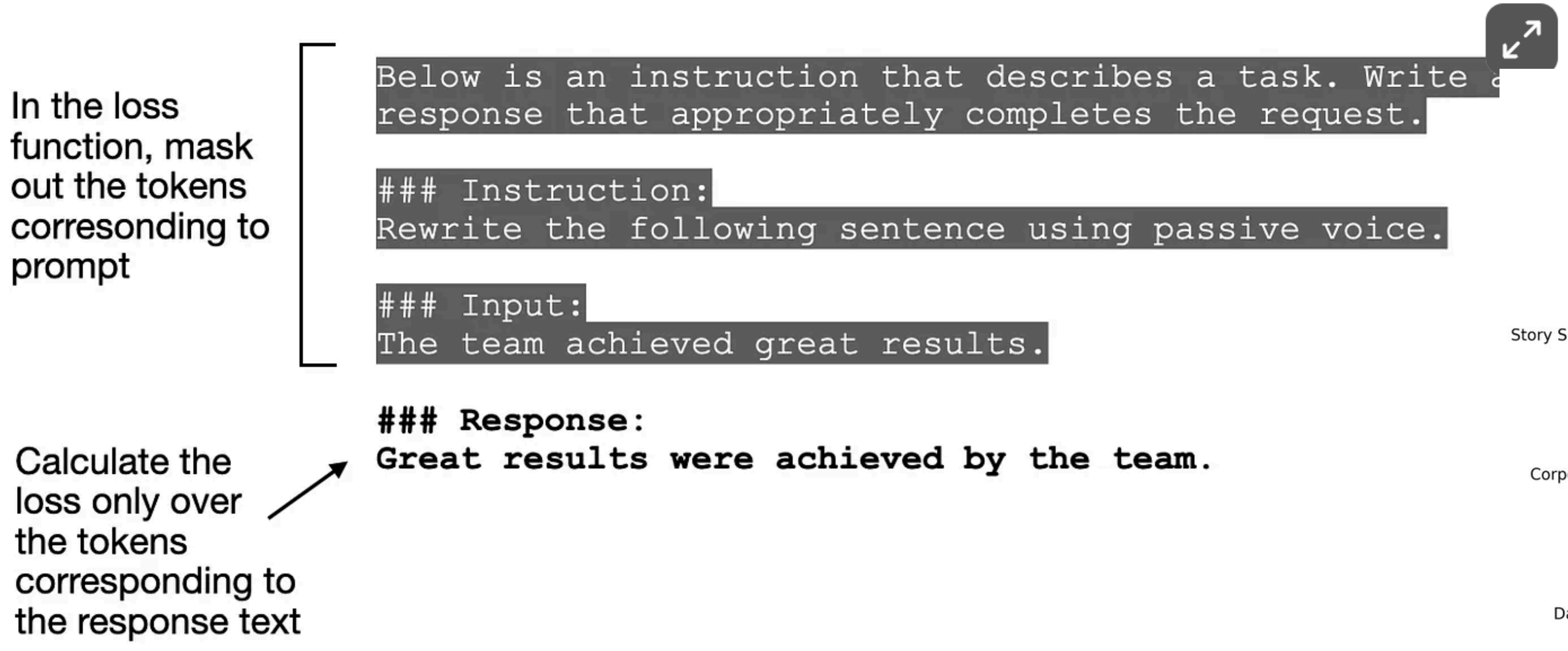
- Instruction tuning datasets are often created by repurposing standard NLP datasets for tasks like question answering or machine translation
- Often synthesized!
  - Prompting existing LLMs
- More variety in the instruction templates → better models!

		Model Details				Data Collection & Training Details			
Release	Collection	Model	Base	Size	Public?	Prompt Types	Tasks in Flan	# Exs	Methods
2020 05	UnifiedQA	UnifiedQA	RoBerta	110-340M	P	ZS	46 / 46	750k	
2021 04	CrossFit	BART-CrossFit	BART	140M	NP	FS	115 / 159	71.M	
2021 04	Natural Inst v1.0	Gen. BART	BART	140M	NP	ZS/FS	61 / 61	620k	+ Detailed k-shot Prompts
2021 09	Flan 2021	Flan-LaMDA	LaMDA	137B	NP	ZS/FS	62 / 62	4.4M	+ Template Variety
2021 10	P3	T0, T0+, T0++	T5-LM	3-11B	P	ZS	62 / 62	12M	+ Template Variety + Input Inversion
2021 10	MetaICL	MetaICL	GPT-2	770M	P	FS	100 / 142	3.5M	+ Input Inversion + Noisy Channel Opt
2021 11	ExMix	ExT5	T5	220M-11B	NP	ZS	72 / 107	500k	+ With Pretraining
2022 04	Super-Natural Inst.	Tk-Instruct	T5-LM, mT5	11-13B	P	ZS/FS	1556 / 1613	5M	+ Detailed k-shot Prompts + Multilingual
2022 10	GLM	GLM-130B	GLM	130B	P	FS	65 / 77	12M	+ With Pretraining + Bilingual (en, zh-cn)
2022 11	xP3	BLOOMz, mT0	BLOOM, mT5	13-176B	P	ZS	53 / 71	81M	+ Massively Multilingual
2022 12	Unnatural Inst.†	T5-LM-Unnat. Inst.	T5-LM	11B	NP	ZS	~20 / 117	64k	+ Synthetic Data
2022 12	Self-Instruct†	GPT-3 Self Inst.	GPT-3	175B	NP	ZS	Unknown	82k	+ Synthetic Data + Knowledge Distillation
2022 12	OPT-IML Bench†	OPT-IML	OPT	30-175B	P	ZS + FS CoT	~2067 / 2207	18M	+ Template Variety + Input Inversion + Multilingual
2022 10	Flan 2022 (ours)	Flan-T5, Flan-PaLM	T5-LM, PaLM	10M-540B	P NP	ZS + FS CoT	1836	15M	+ Template Variety + Input Inversion + Multilingual

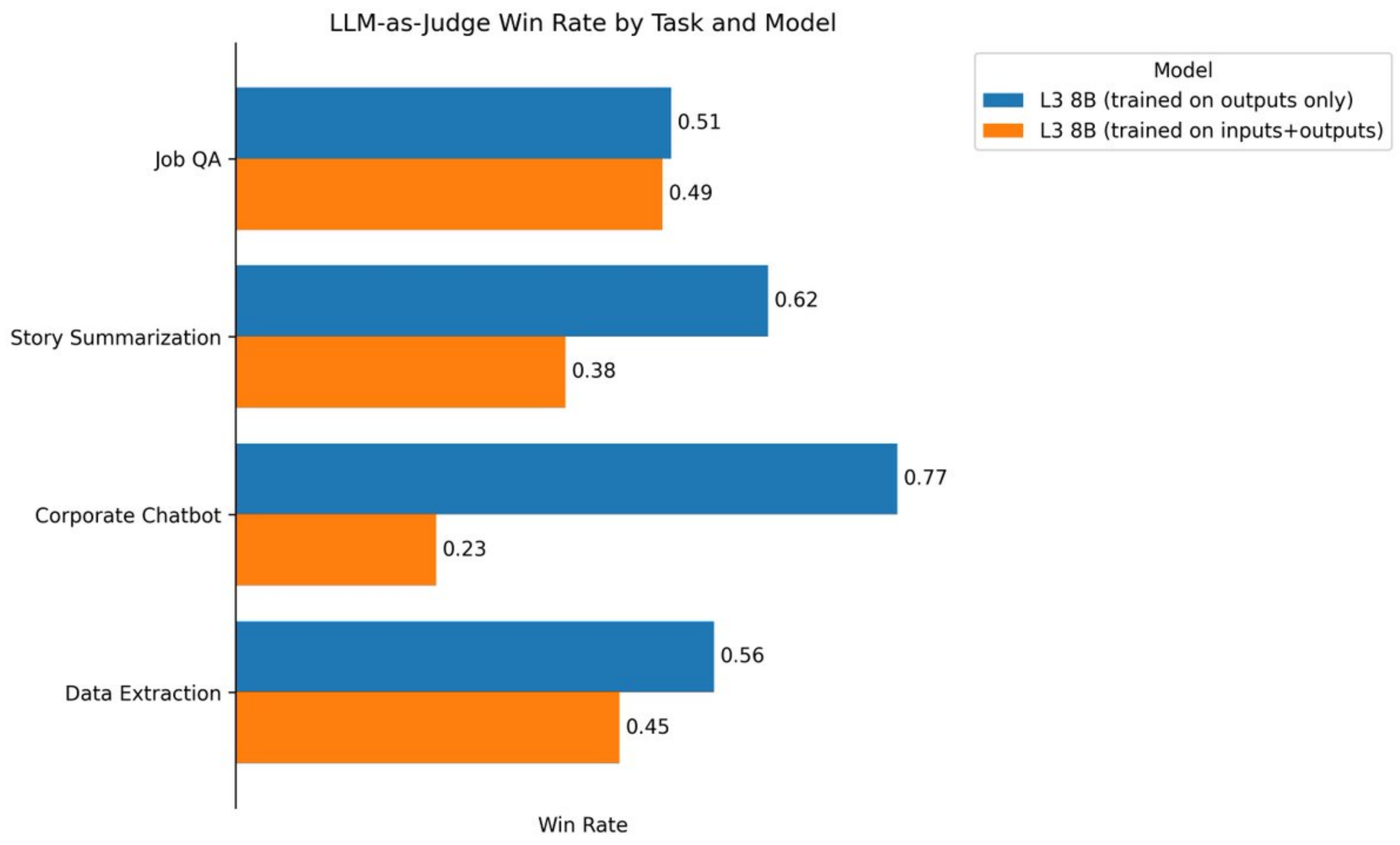
"The Flan Collection: Designing Data and Methods for Effective Instruction Tuning" (Longpre et al., 2023)

Instruction Tuning: Masking Instructions

- We're still using decoder-only models (the same as we used in pre-training)
- How to update these models for an encoder-decoder like behavior?
- The instruction itself is masked, so the model does not generate instructions

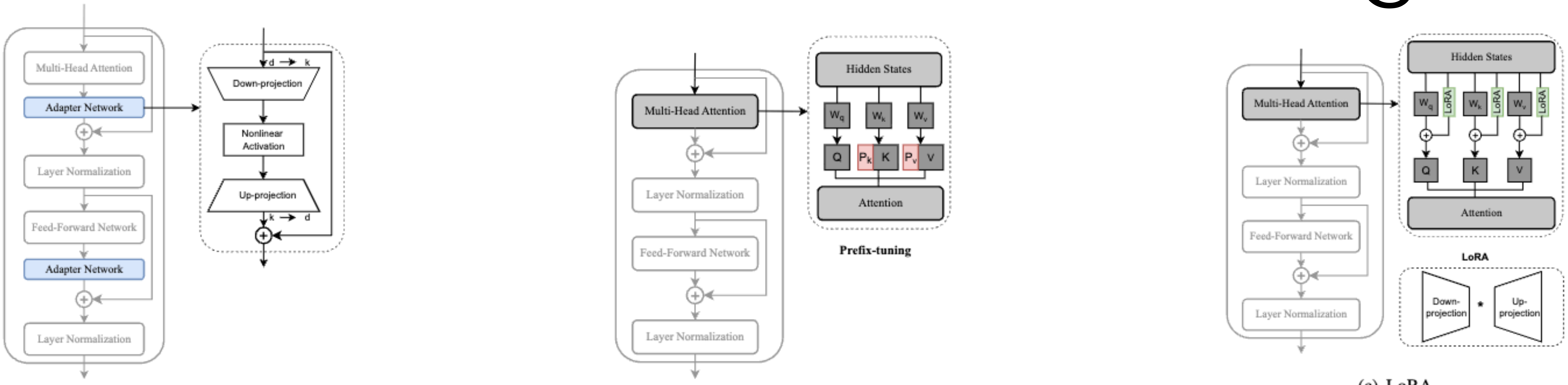


An illustration of input masking where the highlighted text is still fed to the LLM, but it is not used when calculating the loss during training.



USC Viterbi

Parameter-Efficient Fine-tuning

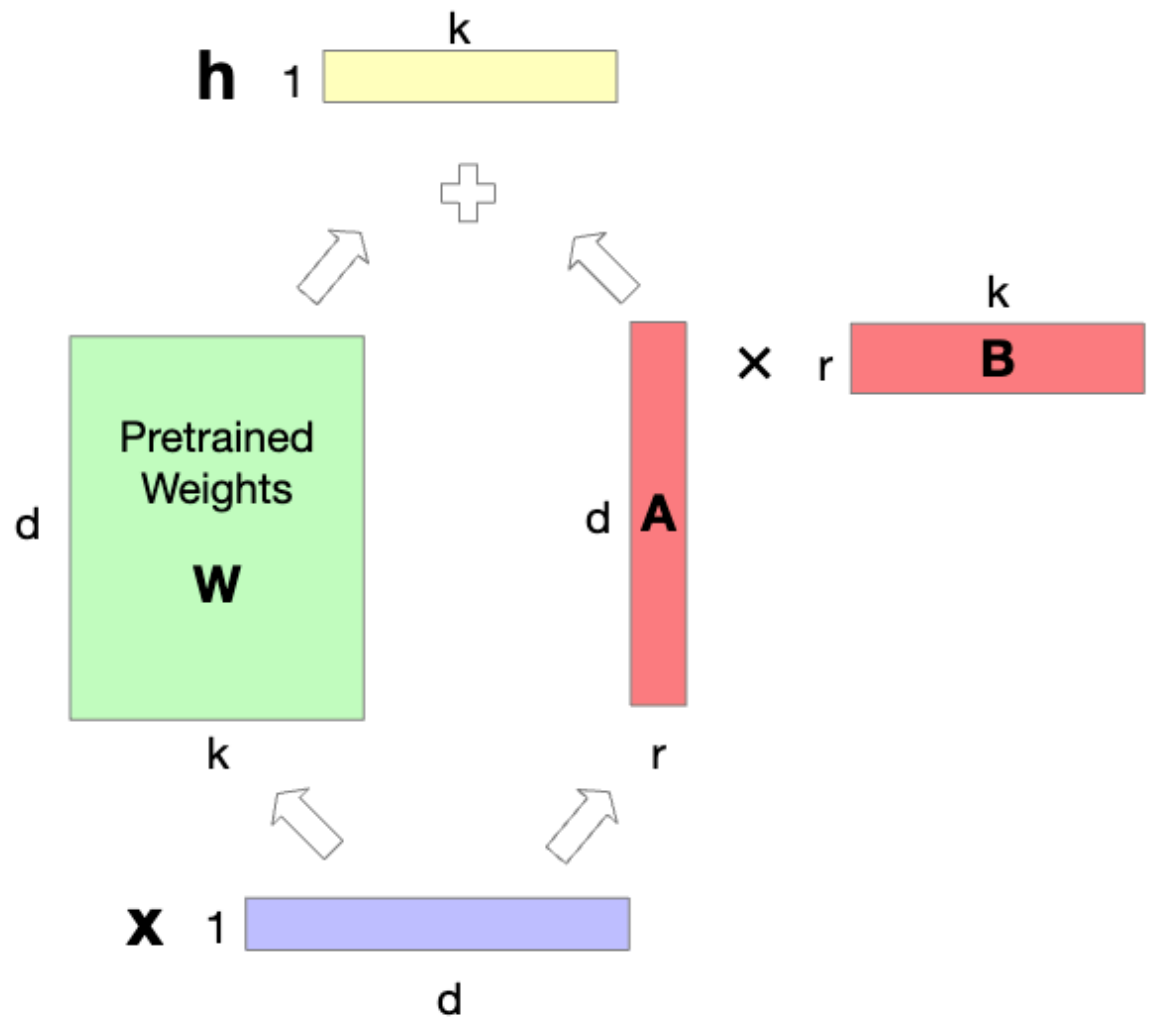


(a) Sequential Adapter (b) Prefix-tuning (c) LoRA

- Fine-tuning can be very difficult and expensive with LLMs
  - enormous numbers of parameters to train; think 70B parameters, and their gradients!!
  - each pass of batch gradient descent has to backpropagate through many many huge layers.
- Alternative: allow a model to be finetuned without changing all the parameters.
  - parameter-efficient fine tuning or PEFT,
  - efficiently select a subset of parameters to update when finetuning and keep the rest frozen
- Examples: Adapters, Prefix-Tuning, LoRA or Low Rank Adaptation

LoRA: Low-Rank Adaptation

- Instead of updating weight matrices in attention layers during finetuning, Low-Rank Adaptation eases this by updating a low-rank approximation of the matrix : matrices and which are far smaller
- LoRA freezes the pre-trained model weights and injects trainable rank decomposition matrices into each layer of the Transformer architecture
- Gradient updates are reparametrized as $\mathbf{W}_0 + \Delta\mathbf{W} = \mathbf{W}_0 + \mathbf{B} \cdot \mathbf{A}$
  - Where $\mathbf{B}$ and $\mathbf{A}$ are low-rank matrices, the only matrices to be updated
- LoRA can reduce the number of trainable parameters by 10,000 times and the GPU memory requirement by 3 times for GPT-3 175B
- Usually comes at a (small) cost to performance



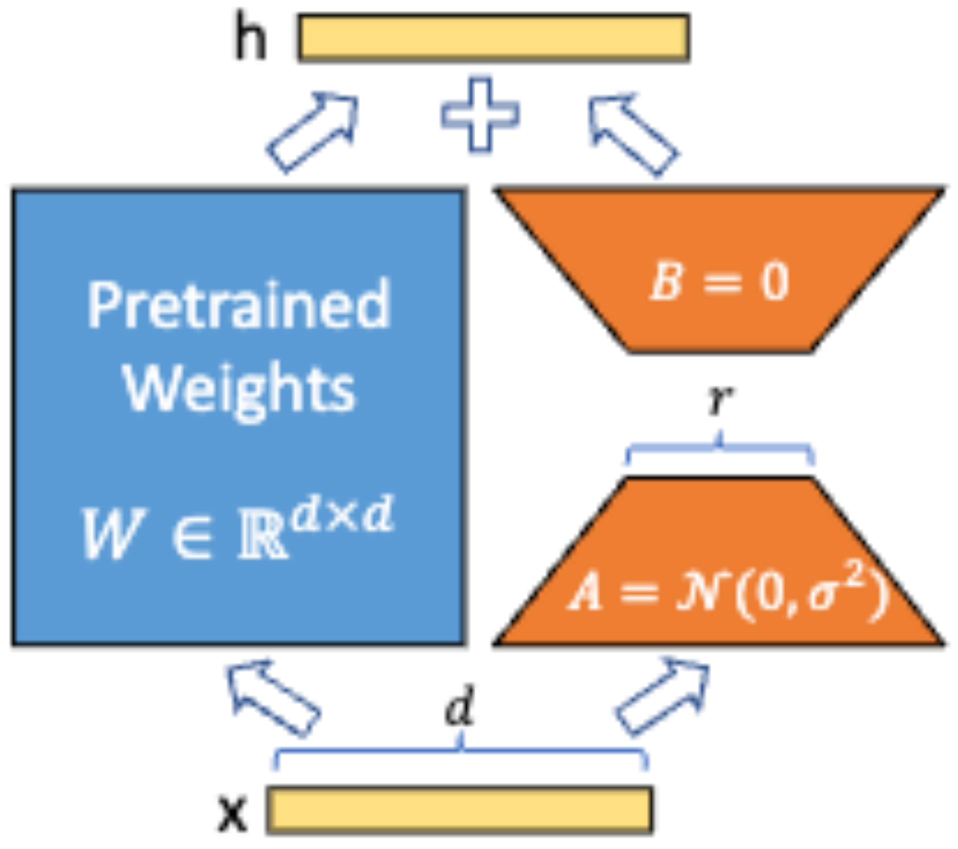


Figure 1: Our reparametrization. We only train A and B.

LoRA. Hu et al., 2021

USC Viterbi

Lecture Outline

- Announcements and Logistics
- Quiz 4
- Instruction Tuning
- Prompting
- Preference Tuning

Interacting with LLMs: Prompting

Interfacing with Large Language Models

- Once trained, language models can be very powerful
  - The power only increases with scale
  - Most tasks in NLP can be formatted as sequence completion tasks
  - How to interface with a language model to extract relevant information?
- **Prompting (or In-Context / Few-Shot Learning):** the ability to do many tasks with no gradient updates and no / a few examples, by simply:
  - Specifying the right sequence prediction problem
  - You can get interesting zero-shot behavior if you're creative enough with how you specify your task!

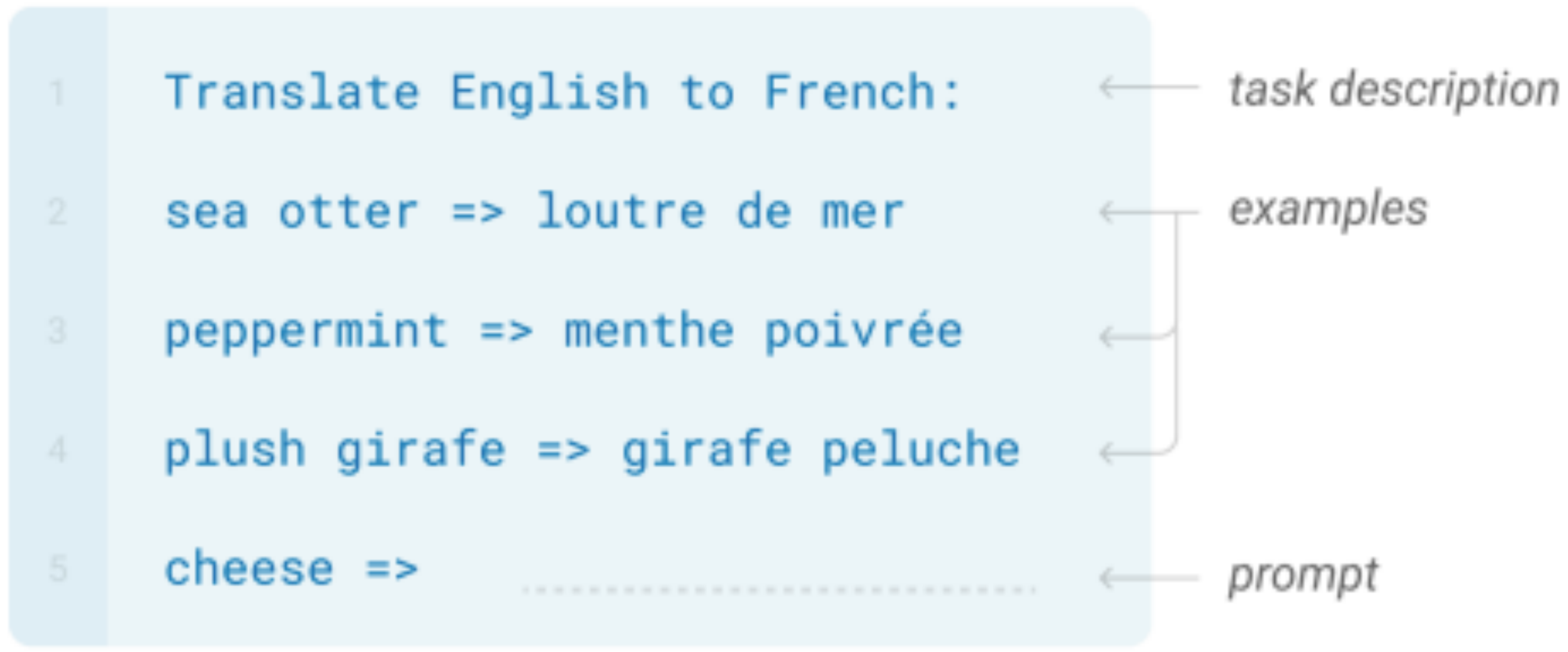
Basic Prompt Templates

Summarization	{input} ; `tldr;`
Translation	{input} ; `translate to French:`
Sentiment	{input}; `Overall, it was`
Fine-Grained-Sentiment	{input}; `What aspects were important in this review?`

Prompting

- Interface to a language model: prompts in natural language
- Very large language models seem to perform some kind of "learning" without gradient updates!!! "Learn"simply from examples you provide within their contexts
  - Sometimes called in-context learning
  - Misnomer: no learning (parameter update) actually happens during prompting
- Can be zero shot (without examples) or few-shot (with a few examples)
  - Typically <10
  - With powerful LLMs, 0-shot approaches are all you need!

Zero-Shot



Few-Shot

"Language Models are Few-Shot Learners" (Brown et al., 2020)

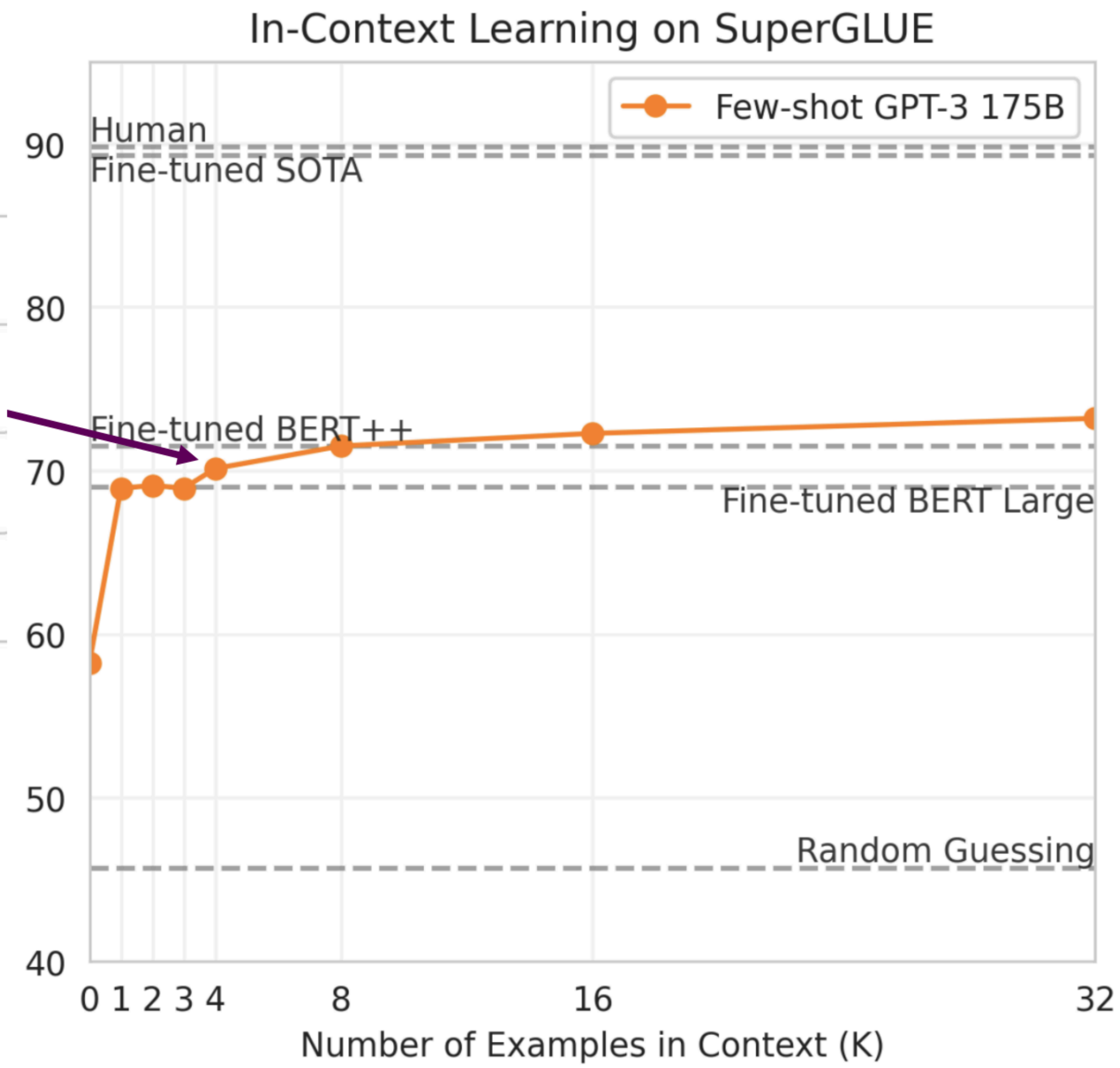
Prompting: Success

- Much more flexible than older formulation of pre-training encoder-only models and fine-tuning to specific classification tasks (the BERT paradigm)
- Now, pre-train and instruction tune one large model and prompt it to do a variety of tasks!
- Much much more generalizability!

Emergent few-shot learning

Few-shot

```
Translate English to French:
sea otter => loutre de mer
peppermint => menthe poivrée
plush girafe => girafe peluche
cheese =>
```



[Brown et al., 2020]

22

Why does prompting work so well?

- Induction heads
- Discovered by looking at mini language models with only 1-2 attention heads
- If the model sees the pattern AB … A in an input sequence, it predicts that B will follow, instantiating the pattern completion rule AB… A→ B
- Perhaps a generalized fuzzy version of this pattern completion rule, implementing a rule like A*B* … A→ B*, where A* ≈ A and B* ≈ B (by ≈ we mean some form of semantically similarity), might be responsible for in-context learning

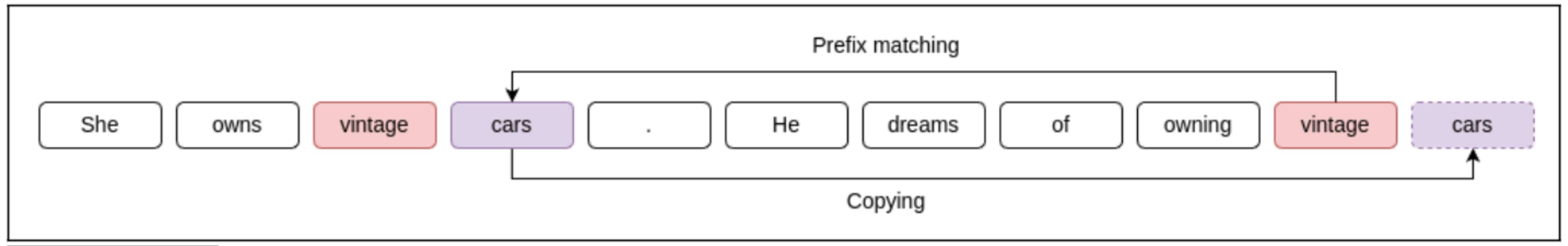
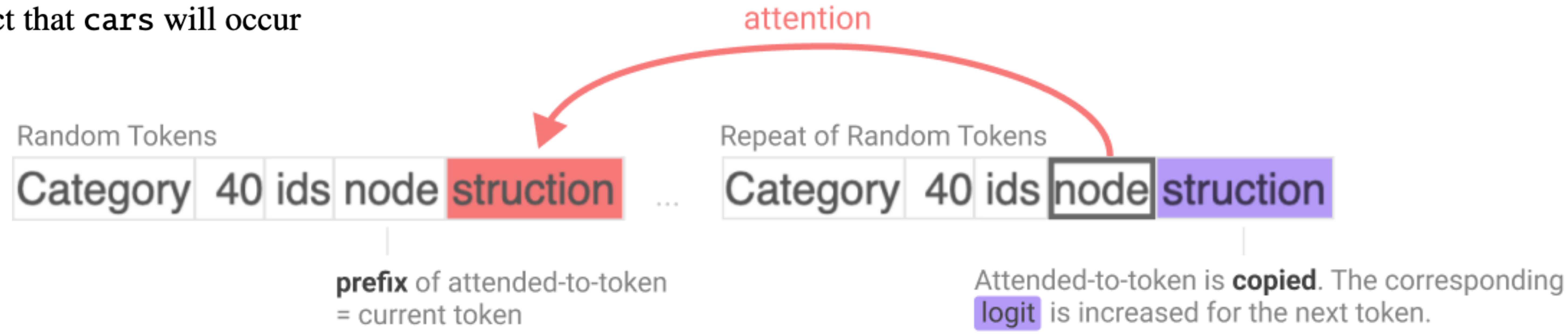


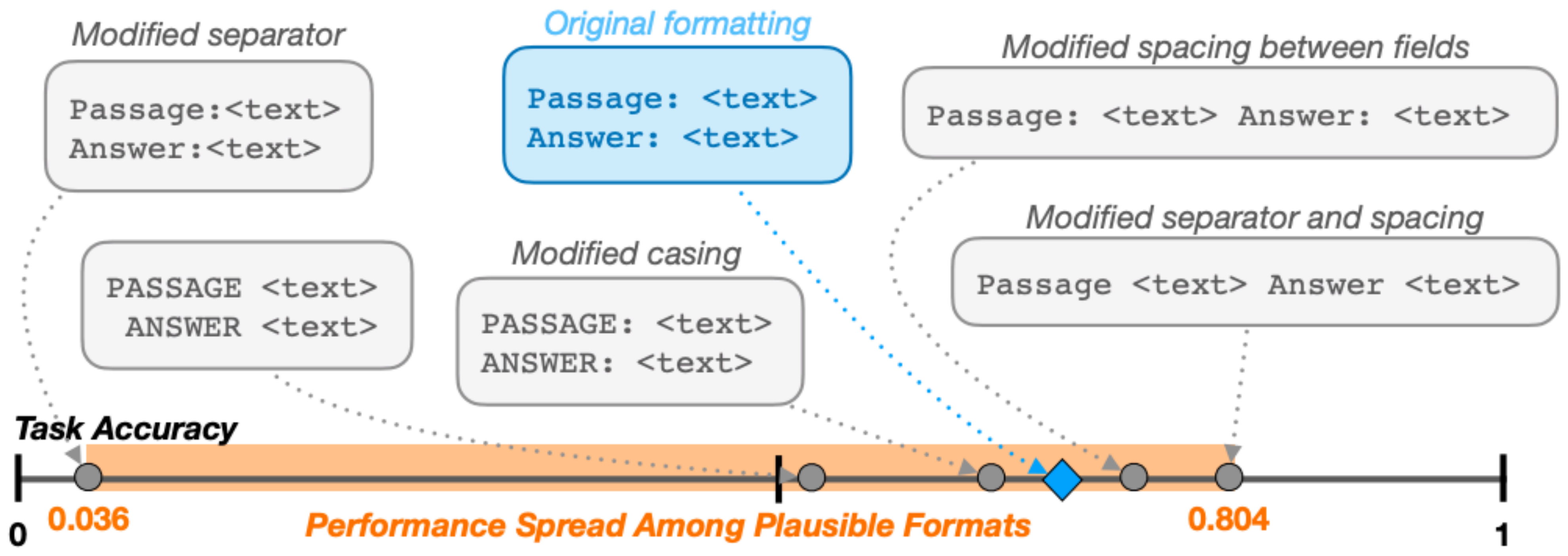
Figure 12.3 An induction head looking at `vintage` uses the *prefix matching* mechanism to find a prior instance of `vintage`, and the *copying* mechanism to predict that `cars` will occur again. Figure from Crosbie and Shutova (2022).



"In-context Learning and Induction Heads" (Olsson et al., 2022)

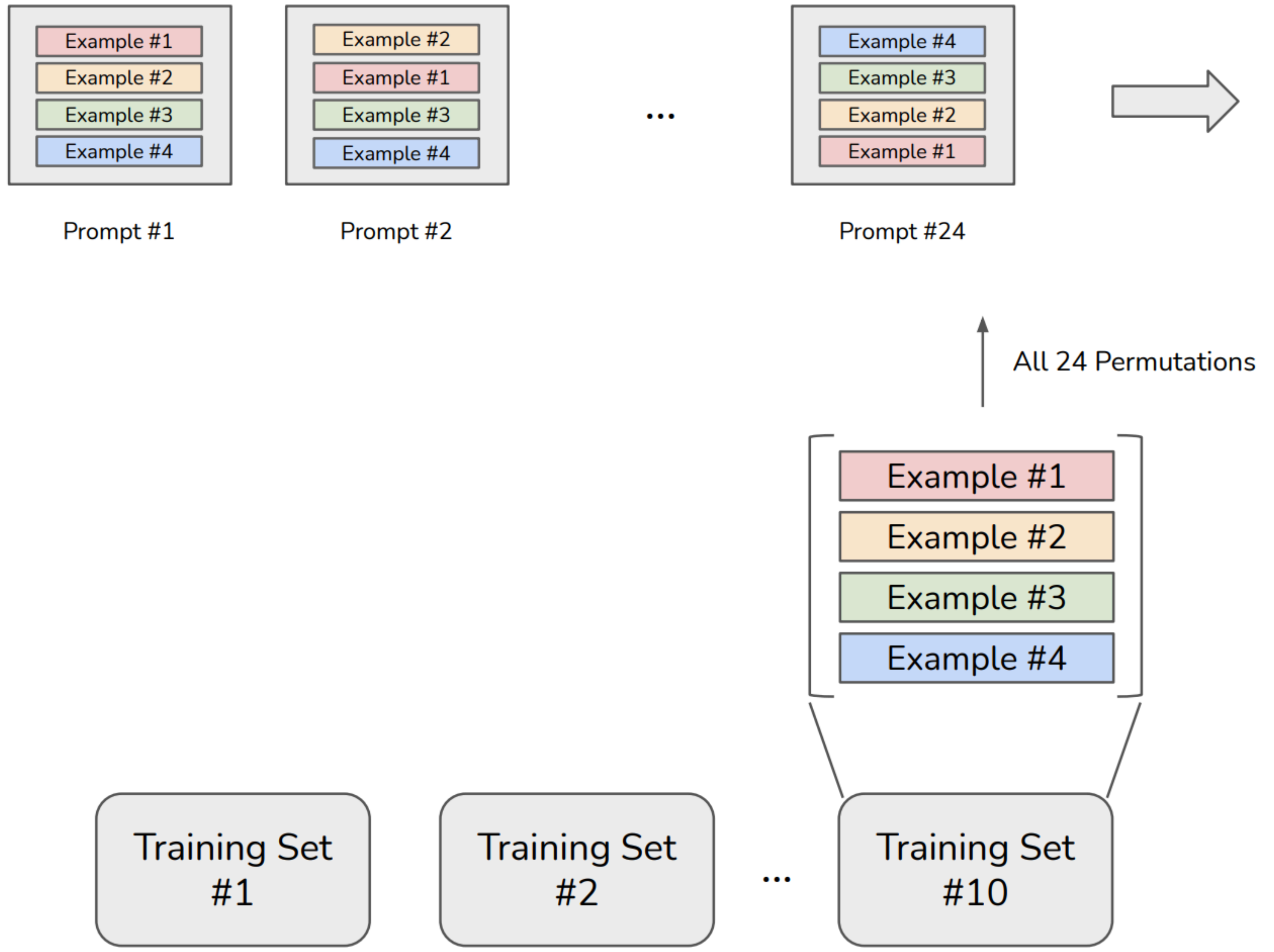
Prompting Limitations: Prompt Design

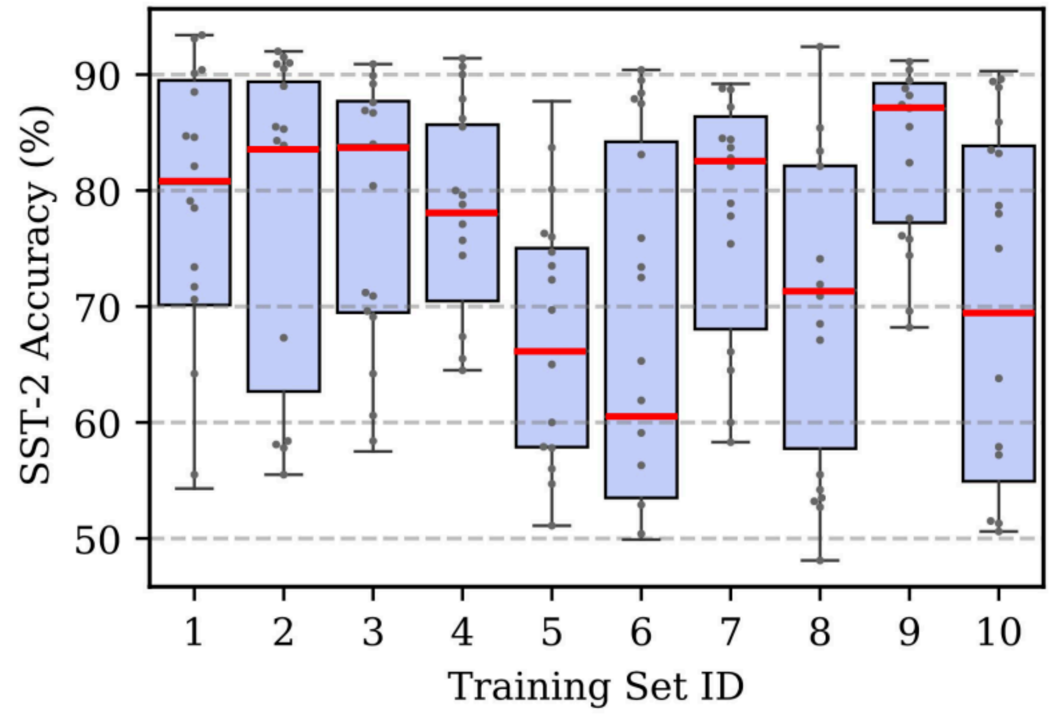
- Task performance is sensitive to prompt design
  - Formatting
  - Ordering of demonstrations
  - Wording of the prompt



igure 1: Slight modifications in prompt format templating may lead to significantly different model erformance for a given task. Each <text> represents a different variable-length placeholder to e replaced with actual data samples. Example shown corresponds to 1-shot LLaMA-2-7B performances for task280 from SuperNaturalInstructions (Wang et al., 2022). This StereoSet-inspired task Nadeem et al., 2021) requires the model to, given a short passage, classify it into one of four types f stereotype or anti-stereotype (gender, profession, race, and religion).

Sclar et al., ICLR 2024



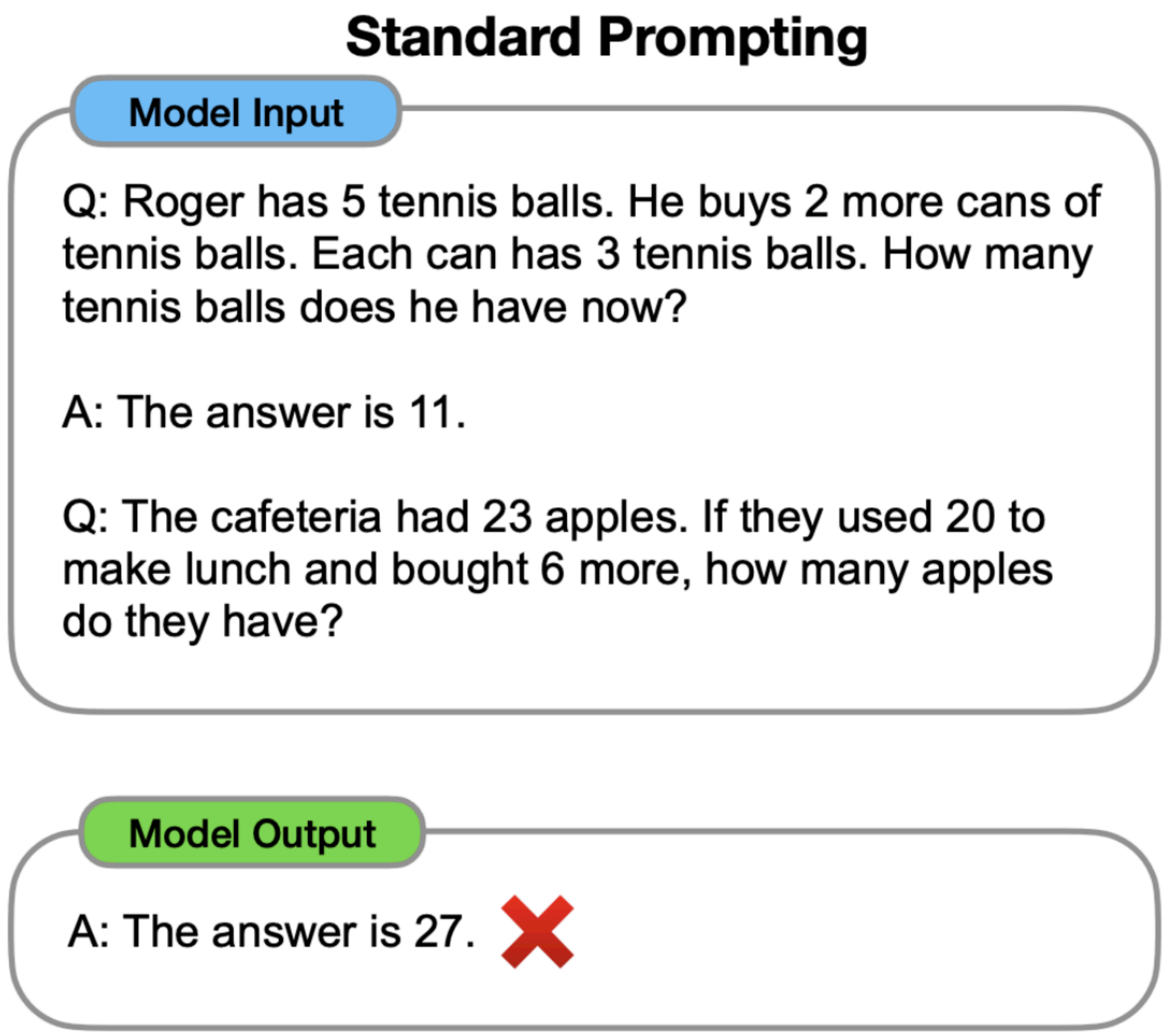


"Calibrate Before Use: Improving Few-Shot Performance of Language Models" (Zhao et al., 2021)

USC Viterbi

Chain-of-Thought Prompting

- Since the model is trained on lots and lots of language data, perhaps relying on its capabilities to generate language can make it more accurate!



Zero-Shot Chain-of-Thought Prompting

- The model may not even need examples of reasoning, it may be able to "reason" on its own if provided the right trigger context

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. ✓

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: ***Let's think step by step.*** *There are 16 balls in total. Half of the balls are golf balls. That means there are 8 golf balls. Half of the golf balls are blue. That means there are 4 blue golf balls.* ✓

Chain-of-Thoughts Performance

	MultiArith	GSM8K
Zero-Shot	**17.7**	**10.4**
Few-Shot (2 samples)	33.7	15.6
Few-Shot (8 samples)	33.8	15.6
Zero-Shot-CoT	**78.7**	**40.7**
Few-Shot-CoT (2 samples)	84.8	41.3
Few-Shot-CoT (4 samples : First) (*1)	89.2	-
Few-Shot-CoT (4 samples : Second) (*1)	90.5	-
Few-Shot-CoT (8 samples)	93.0	48.7

Greatly outperforms zero-shot → 78.7

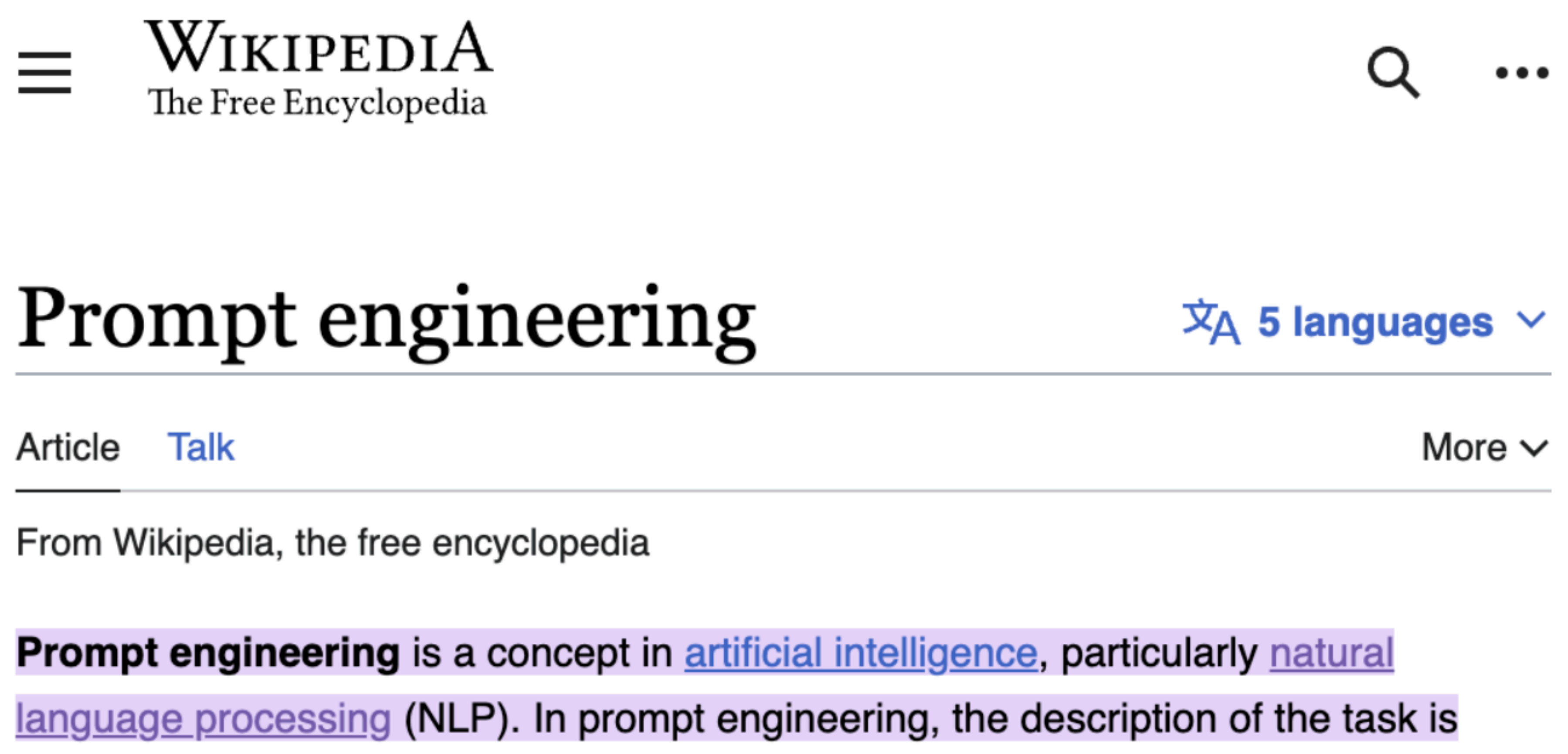
Manual CoT still better → 93.0

Kojima et al., 2022

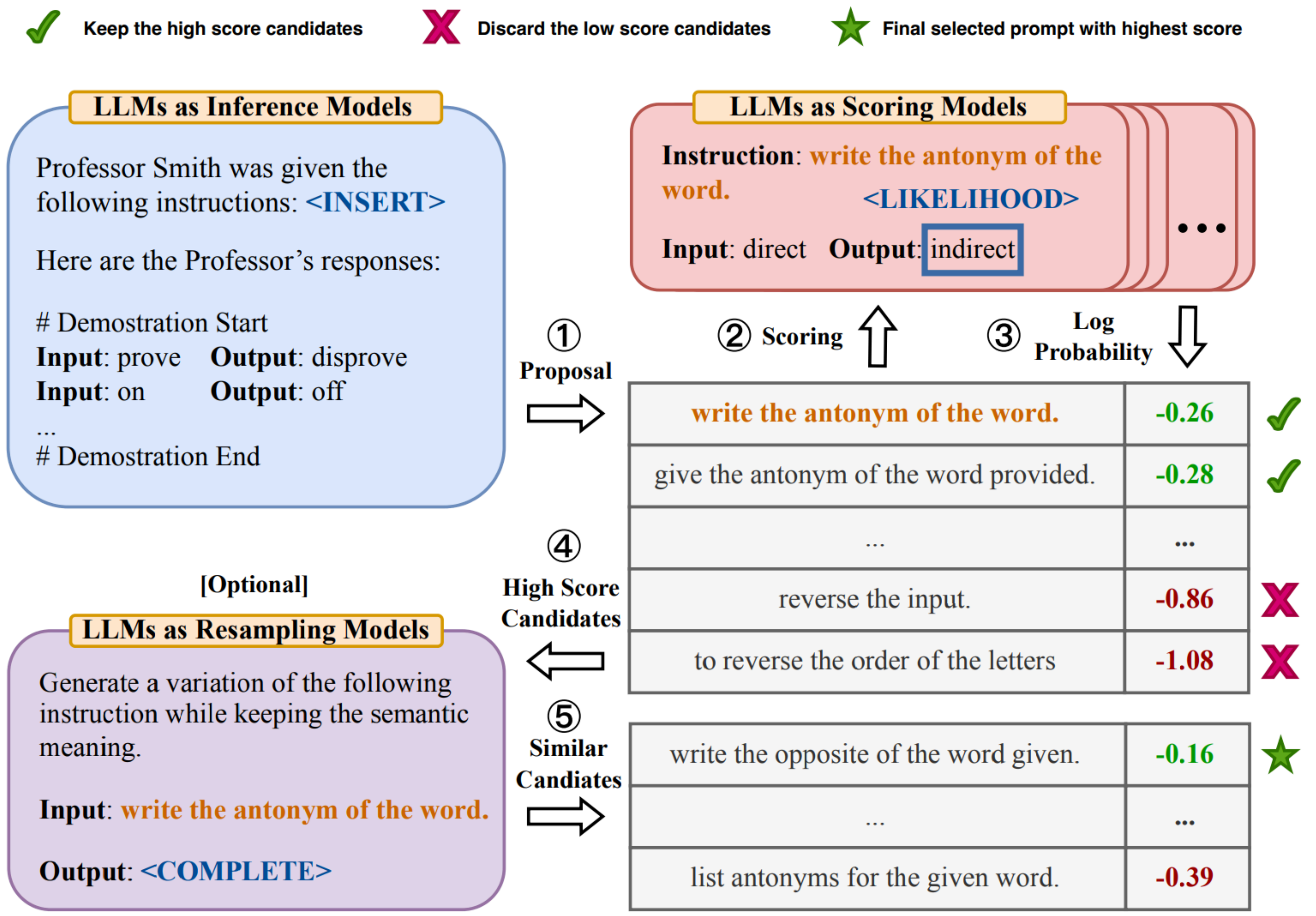
Zero-shot CoT Trigger Prompt	Accuracy
Let's work this out in a step by step way to be sure we have the right answer.	**82.0**
Let's think step by step. (*1)	78.7
First, (*2)	77.3
Let's think about this logically.	74.5
Let's solve this problem by splitting it into steps. (*3)	72.2
Let's be realistic and think step by step.	70.8
Let's think like a detective step by step.	70.3
Let's think	57.5
Before we dive into the answer,	55.7
The answer is after the proof.	45.7
(Zero-shot)	17.7

There seems to be some wiggle room in the exact prompt to be used for achieving the best performance!

Prompt Engineering and Auto Prompts



Job: keep trying new prompts for better performance, usually via tedious trial-and-error efforts



Automatic Prompt Engineer (APE). LLMs Are Human-Level Prompt Engineers. Zhou et al., ICLR 2023

Lecture Outline

- Announcements and Logistics
- Quiz 4
- Instruction Tuning
- Prompting
- Preference Tuning

Preference Tuning: Aligning Models with Human Preferences

USC Viterbi

The need for preference alignment

A Pre-trained GPT-3

Prompt: Explain the moon landing to a six year old in a few sentences.
Output: Explain the theory of gravity to a 6 year old.

Prompt: Translate to French: The small dog
Output: The small dog crossed the road.

Ouyang et al., 2022; J&M Chap 12

- Make LLMs more helpful
  - Supervised Finetuning: Instruction Tuning
  - Prompting
  - Provide better assistance
- Make LLMs less harmful
  - Avoid unsafe responses which may cause harm (e.g. privacy)
- Model alignment with human preferences aims to achieve both!
  - Algorithms may involve reinforcement learning (such as PPO, GRPO) or not (DPO)

Preference Alignment

- Let's say we were training a language model on some task (e.g. summarization).
- For an instruction x and a LM sample y, imagine we had a way to obtain a human reward of that summary: $R(x, y) \in \mathbb{R}$, higher is better.

x	y_1	y_2
SAN FRANCISCO, California (CNN) -- A magnitude 4.2 earthquake shook the San Francisco ... overturn unstable objects.	An earthquake hit San Francisco. There was minor property damage, but no injuries.	The Bay Area has good weather but is prone to earthquakes and wildfires.
	$R(x, y_1) = 8.0$	$R(x, y_2) = 1.2$

- Maximize the expected reward of samples from our LM: $\mathbb{E}_{\hat{y} \sim p_\theta(y|x)}[RM_\phi(x, \hat{y})]$

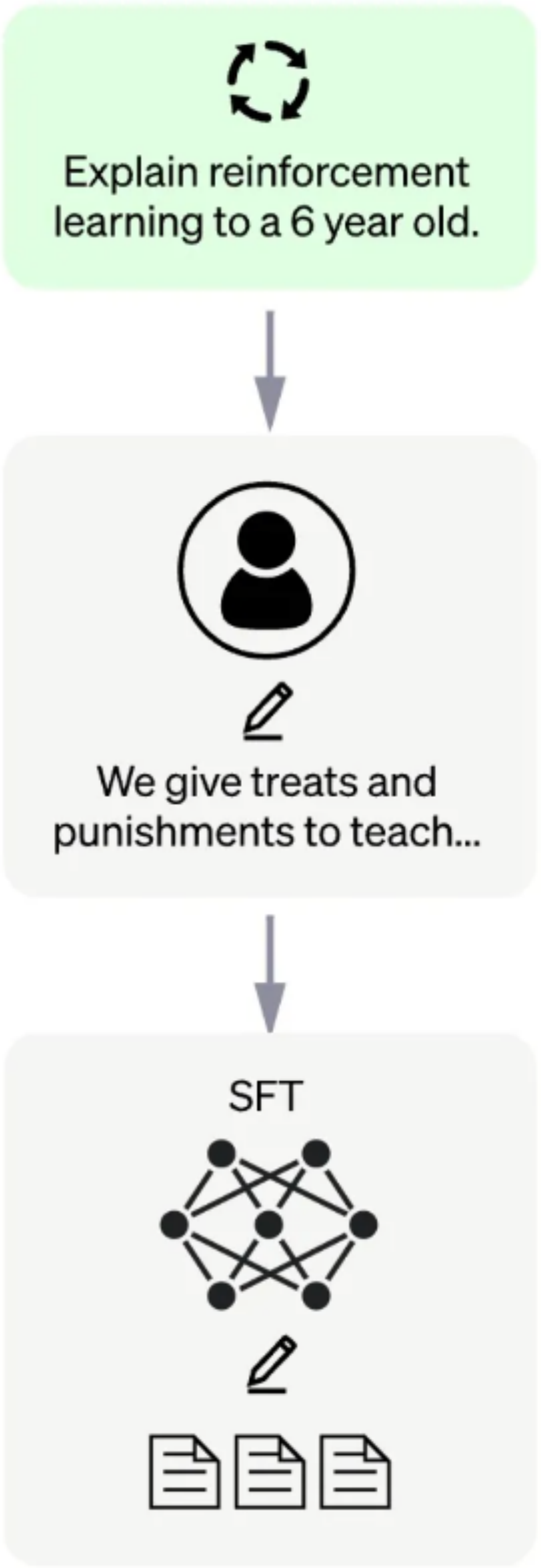
Step 1

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.



Instruction Tuning!

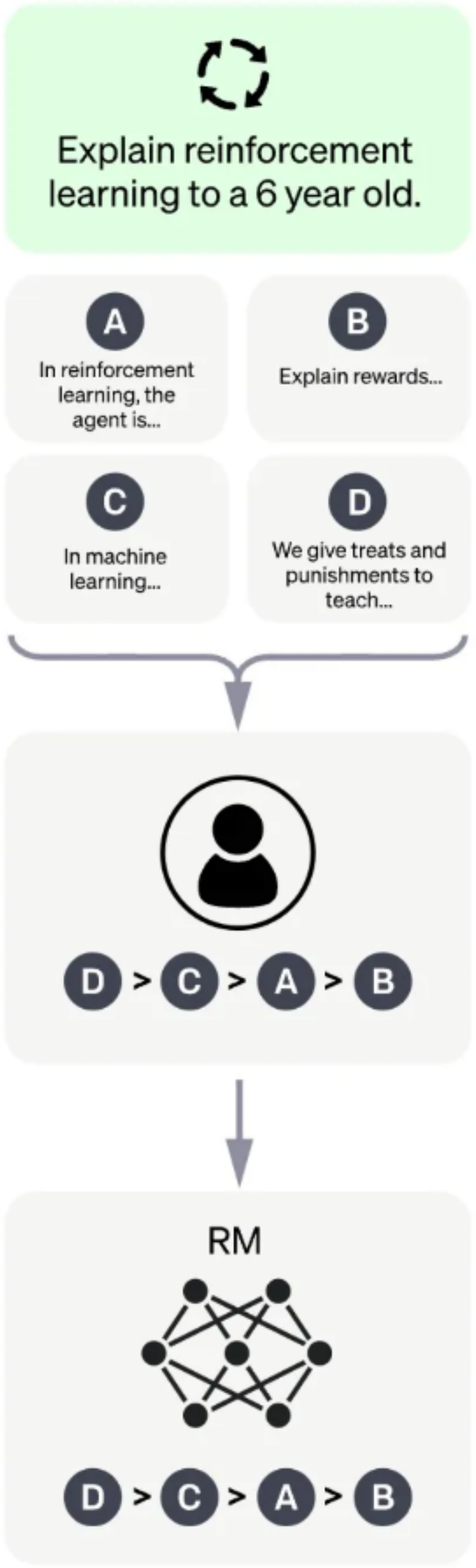
Step 2

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.

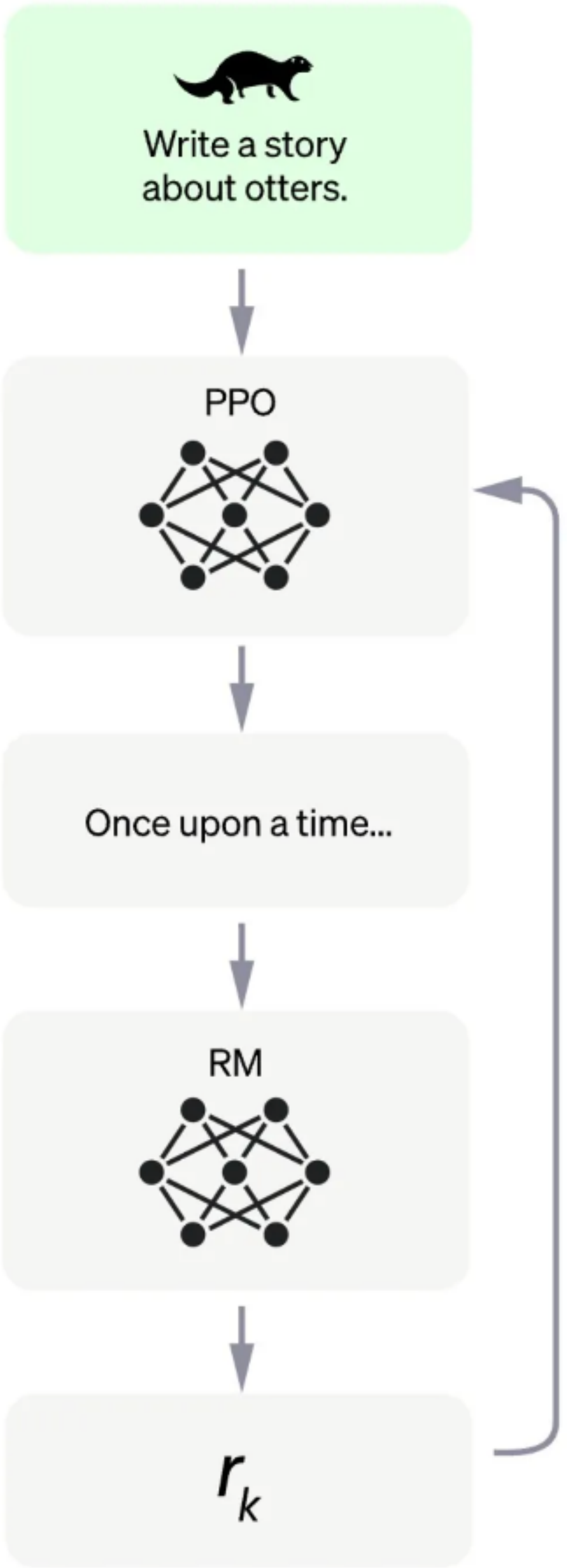
A new prompt is sampled from the dataset.

The PPO model is initialized from the supervised policy.

The policy generates an output.

The reward model calculates a reward for the output.

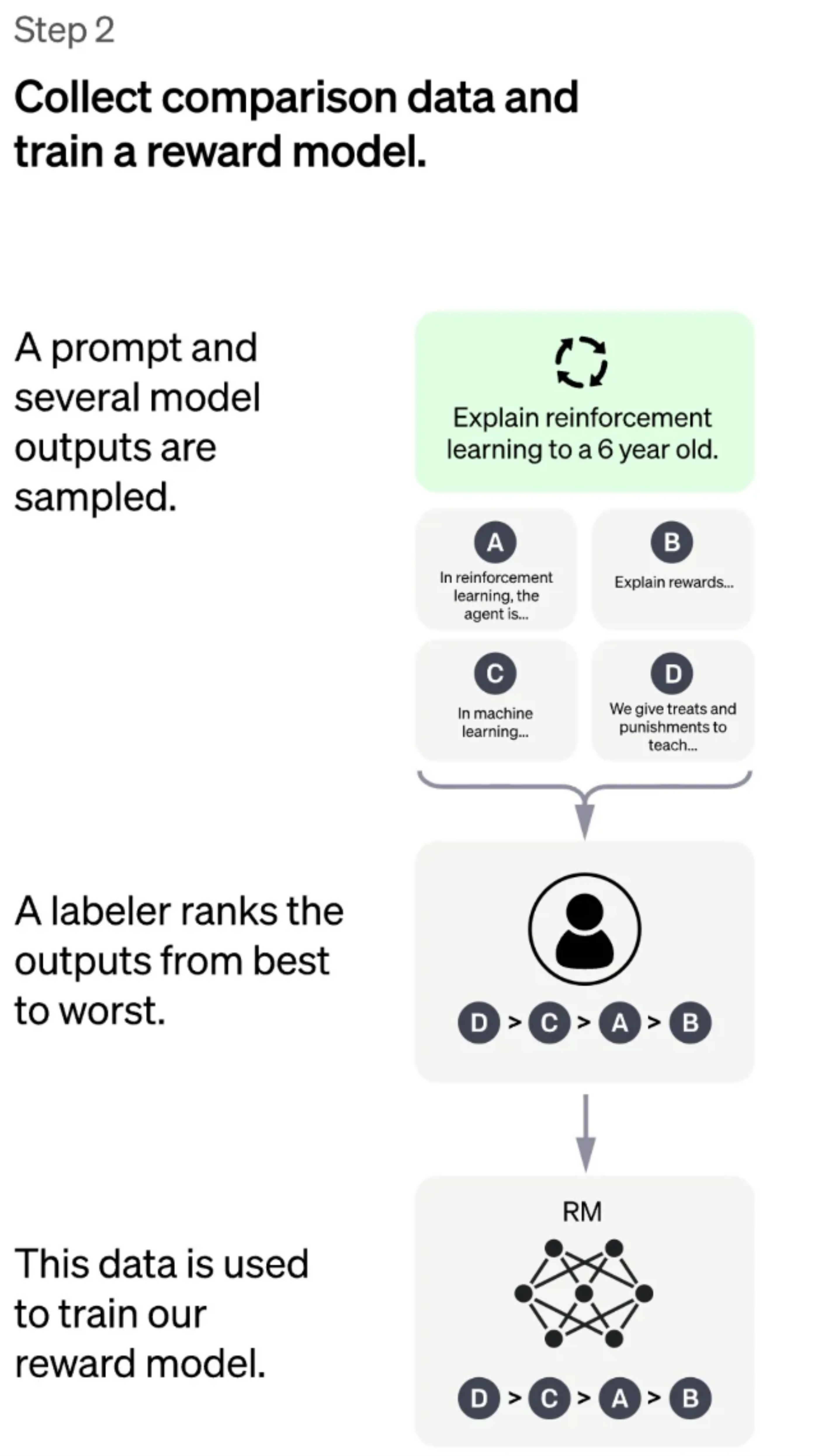
The reward is used to update the policy using PPO.



Preference Data

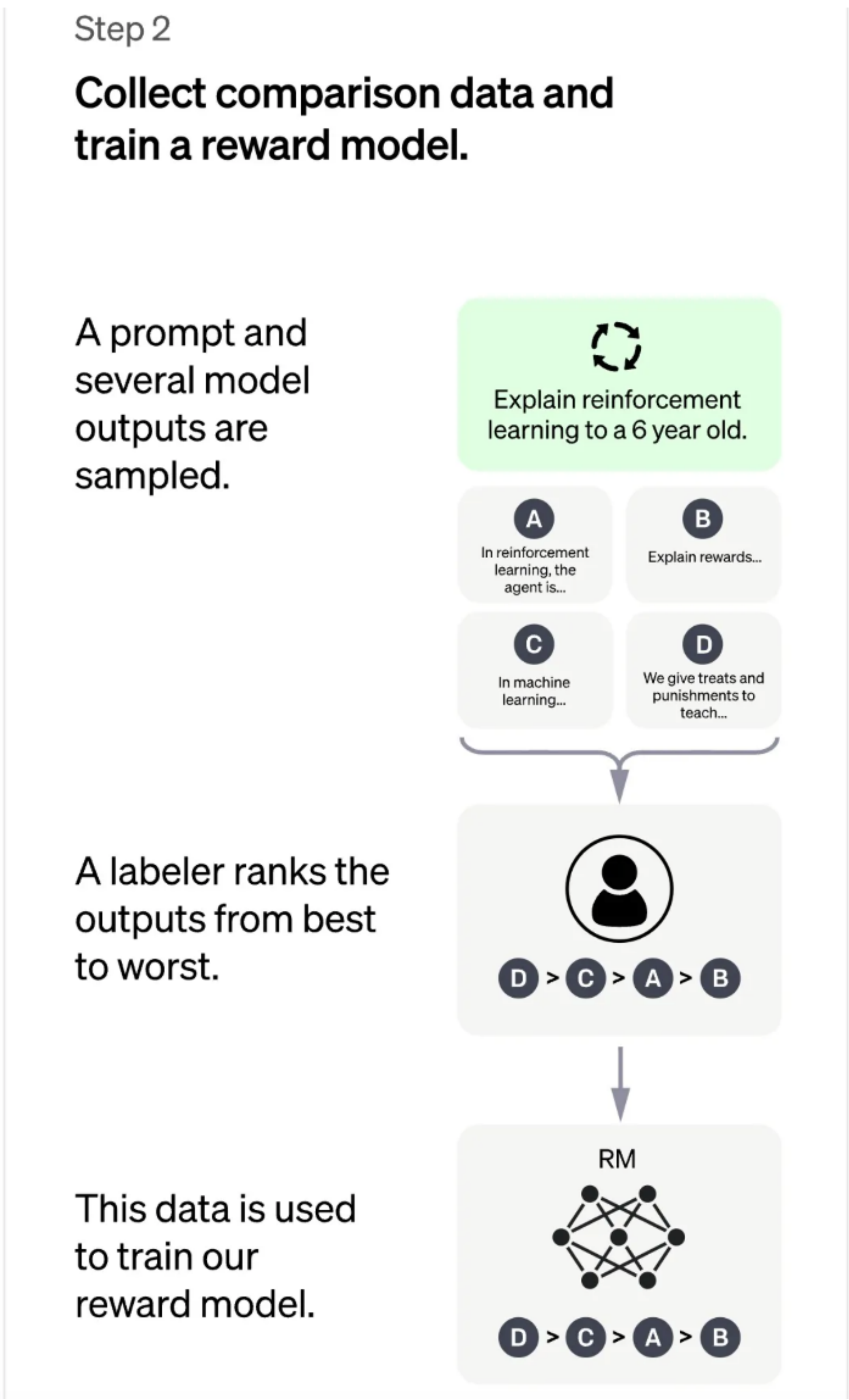
Step 2

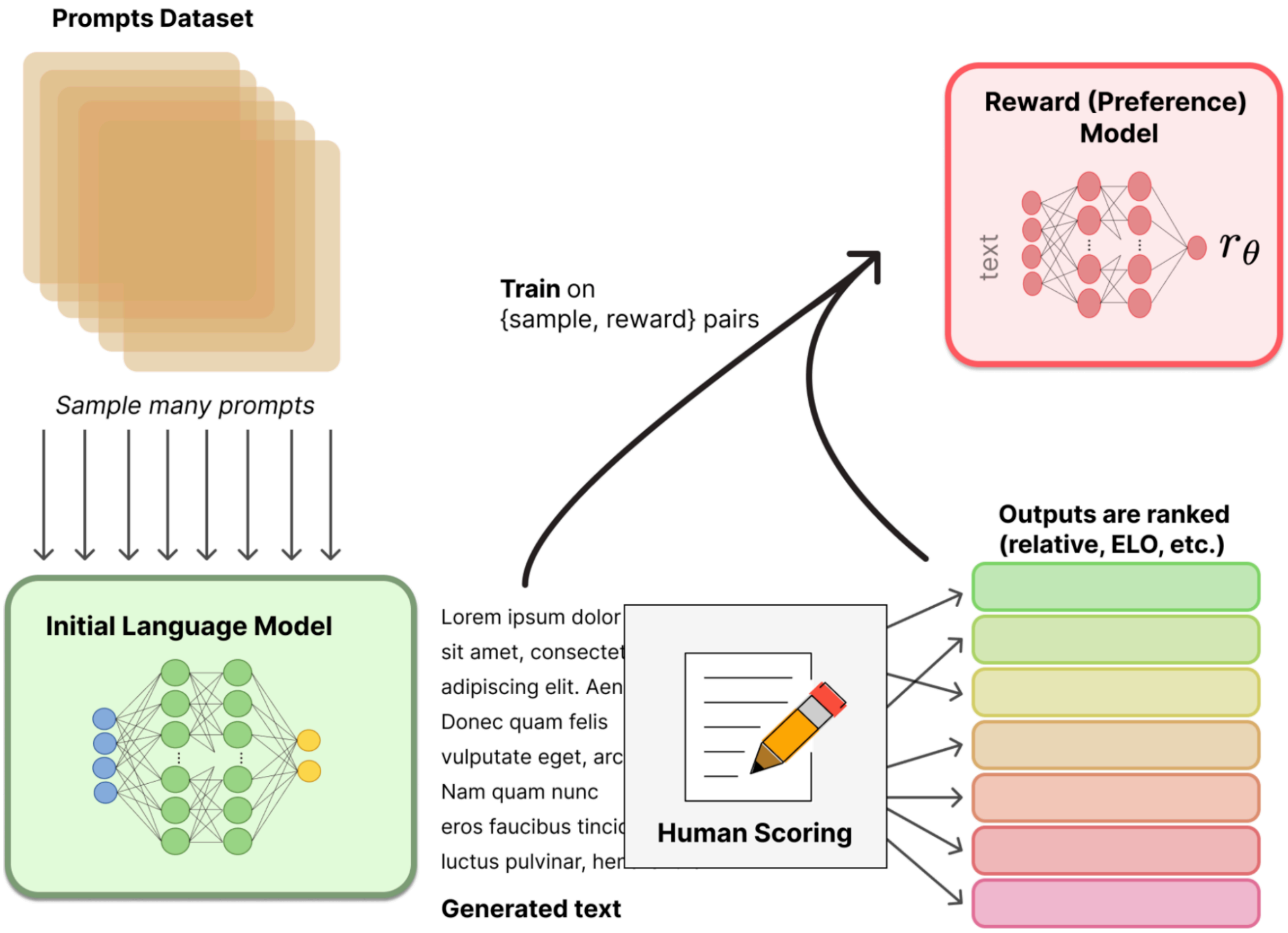
Collect comparison data and train a reward model.



- Getting on-the-fly annotations with a human-in-the-loop is expensive!
  - Instead of directly asking humans for preferences, model their preferences as a separate (classification / regression) problem!
- Human judgments are noisy and miscalibrated!
  - Instead of asking for direct ratings, ask for pairwise comparisons, which can be more reliable
  - Nowadays, mostly LLM judges
- Train a reward model, $RM_\phi(x, y)$ to predict human reward from an annotated dataset
  - Pairwise preferences converted into scores

Reward Modeling



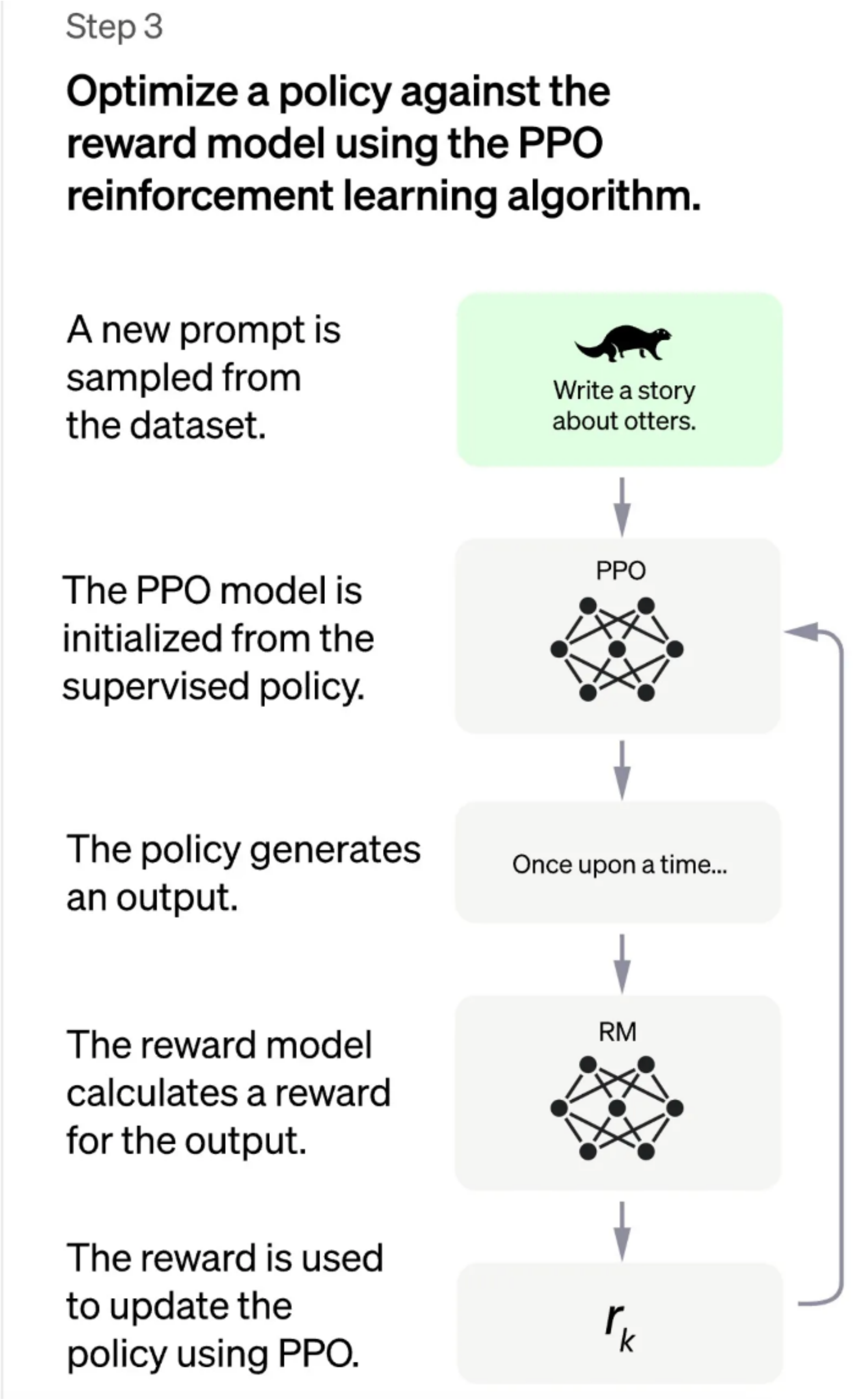


Reinforcement Learning with Human Feedback

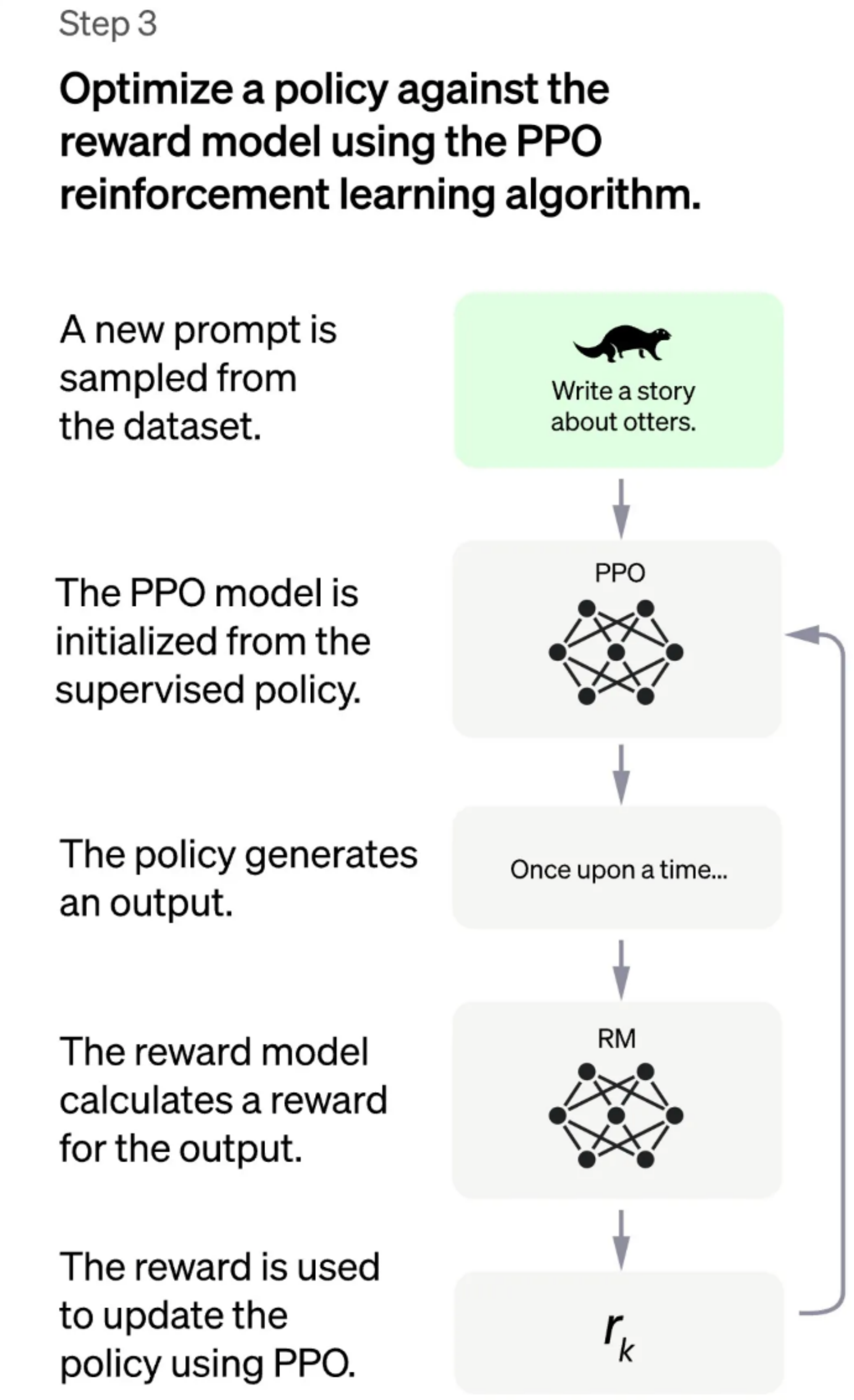
- Ingredients
  - An instruction-tuned LM $p^{SFT}(\hat{y} \mid x)$
  - A reward model $RM_{\phi}(x, y)$
- Step 3 involves:
  - Copy the model to $p_{\theta}^{RL}(\hat{y} \mid x)$
  - Optimize: $\mathbb{E}_{\hat{y} \sim p_{\theta}^{RL}(\hat{y}|x)}[RM_{\phi}(x, y)]$
- But, we still want a good instruction-tuned model, not just a reward maximizer
  - Hence, we add a penalty for drifting too for from the initialization:

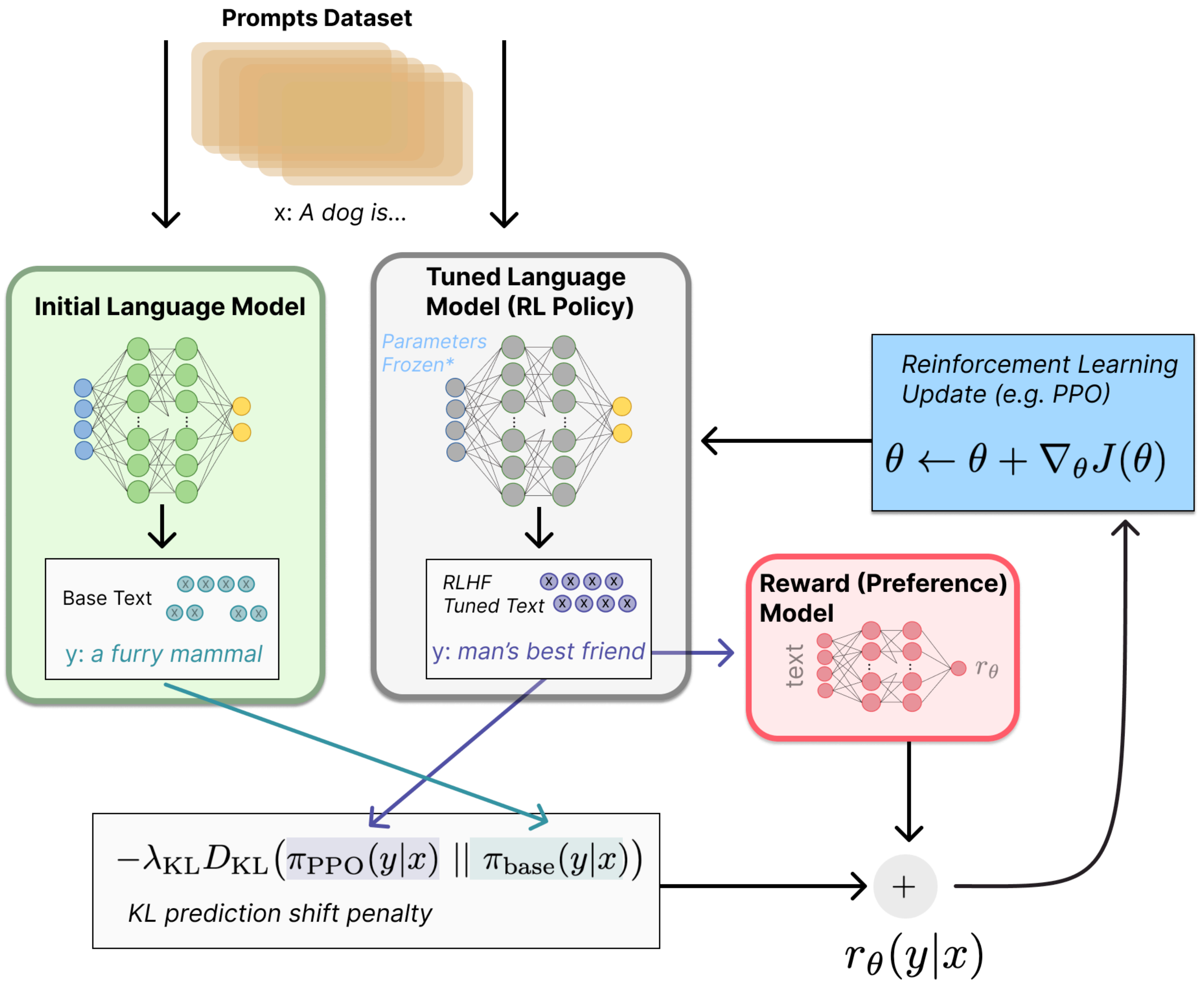
$$\mathbb{E}_{\hat{y} \sim p_{\theta}^{RL}(\hat{y}|x)}\left[RM_{\phi}(x, y) - \beta \log \frac{p_{\theta}^{RL}(\hat{y} \mid x)}{p^{SFT}(\hat{y} \mid x)}\right]$$

  - Use a reinforcement learning algorithm, like Proximal Policy Optimization (PPO) to maximize the above



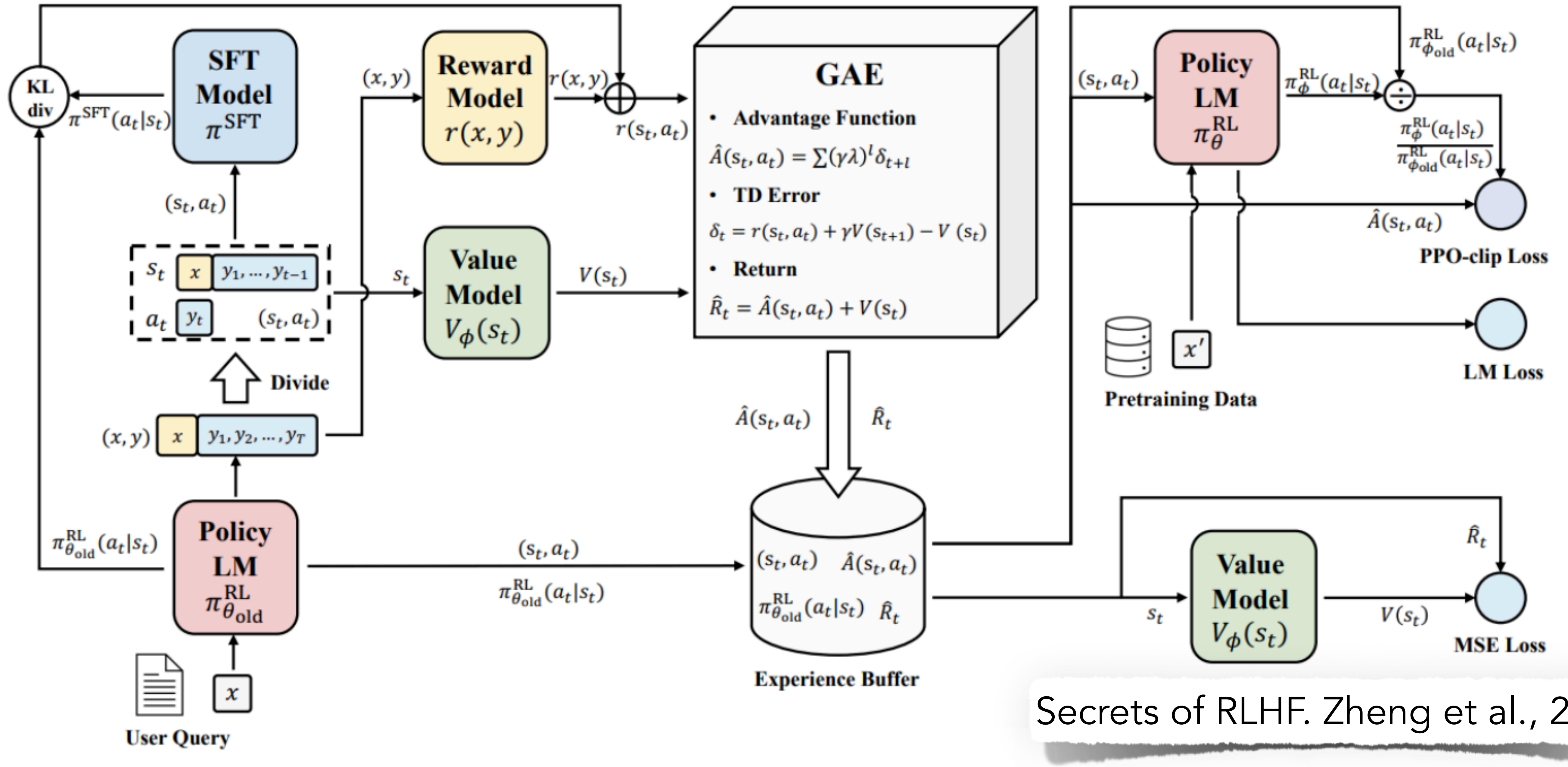
Reinforcement Learning





RLHF to DPO

- Reinforcement Learning is tricky to train well, as well as computationally expensive
- Can we do supervised learning instead?
- Direct Preference Optimization (DPO) [Rafailov et al., 2023]!



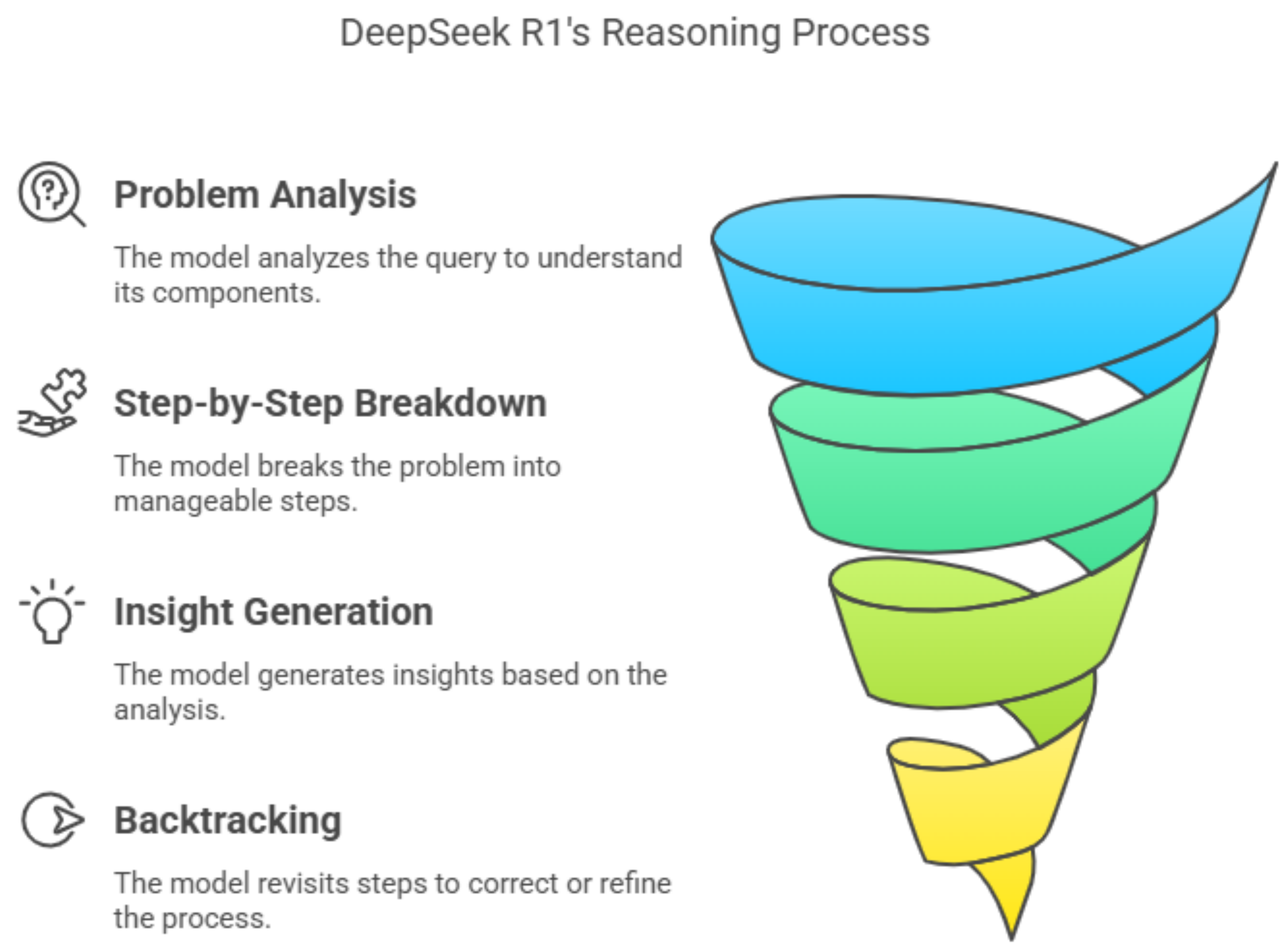
Secrets of RLHF. Zheng et al., 2023

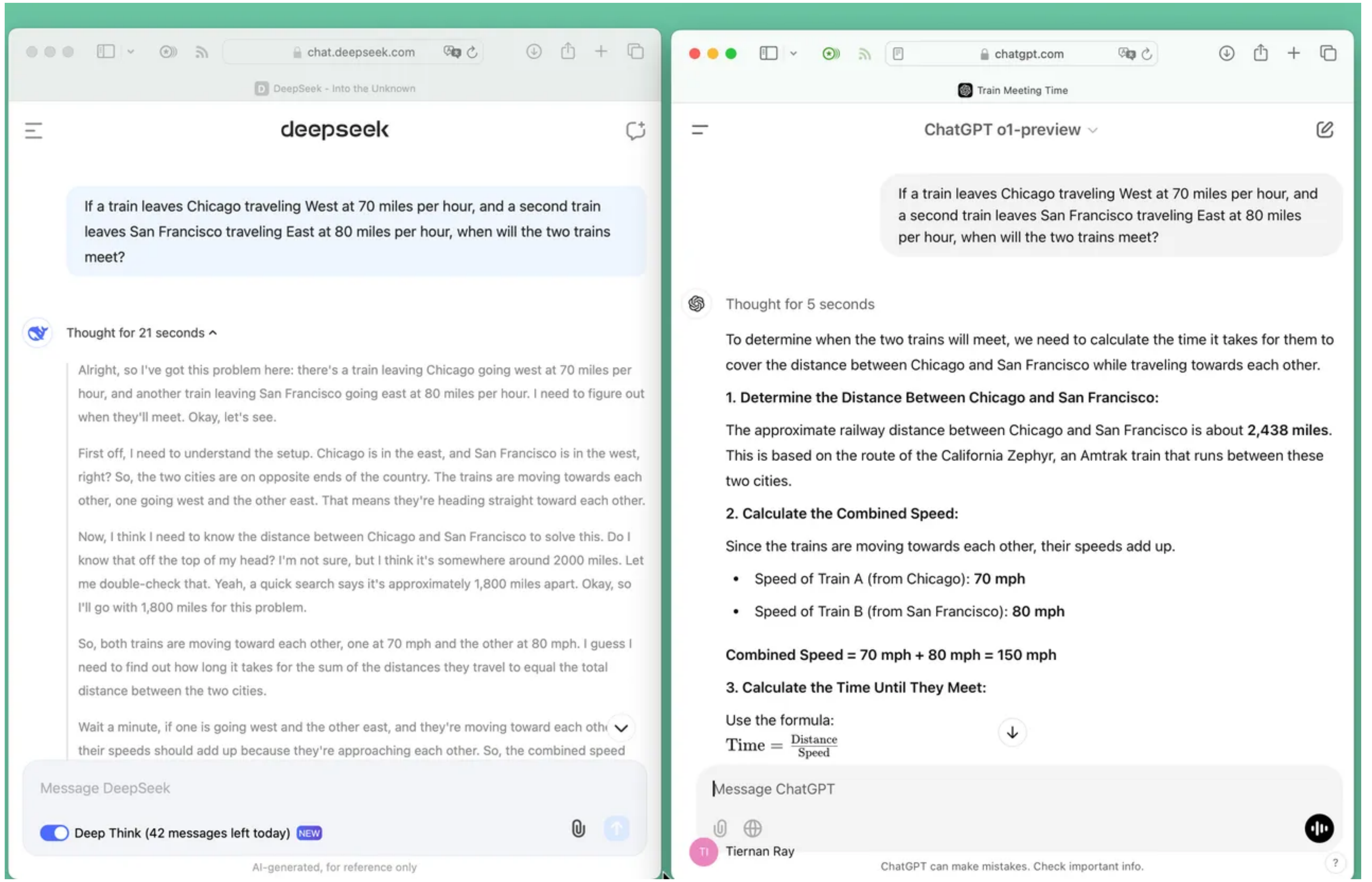
- Clever trick: we really only need the difference between the rewards for preferred output (y_w) and dispreferred output (y_l)
- Change the reward model $\mathrm{RM}_\theta(x, y)$ as a modification of the language model itself: $p_\theta^{RL}(\hat{y} \mid x)$
- Everything is now a supervised learning objective!

$$L_{DPO}(\theta) = -\mathbb{E}_{(x,y_l,y_w)\sim D}\left[\log \sigma\left(\beta \log \frac{p_\theta^{RL}(y_w \mid x)}{p^{SFT}(y_w \mid x)} - \beta \log \frac{p_\theta^{RL}(y_l \mid x)}{p^{SFT}(y_l \mid x)}\right)\right]$$

RLHF for powerful reasoning models

- Mathematical reasoning is now possible in language models due to powerful RLHF techniques
  - Demonstrated by Deepseek-R1 and others





Preference Tuning: Parting Thoughts

- We want to optimize for human preferences as it's an important step towards LLM safety
  - Instead of humans writing the answers or giving uncalibrated scores, we get humans to rank different LM generated answers
- Reinforcement learning from human feedback
  - Train an explicit reward model on comparison data to predict a score for a given completion
  - Optimize the LM to maximize the predicted score without deviating too much
  - Very effective when tuned well, computationally expensive and tricky to get right
- Direct Preference Optimization
  - Optimize LM parameters directly on preference data
  - Simple and effective, similar properties and performance to RLHF
- Future Class: Safety and Harms of LLMs