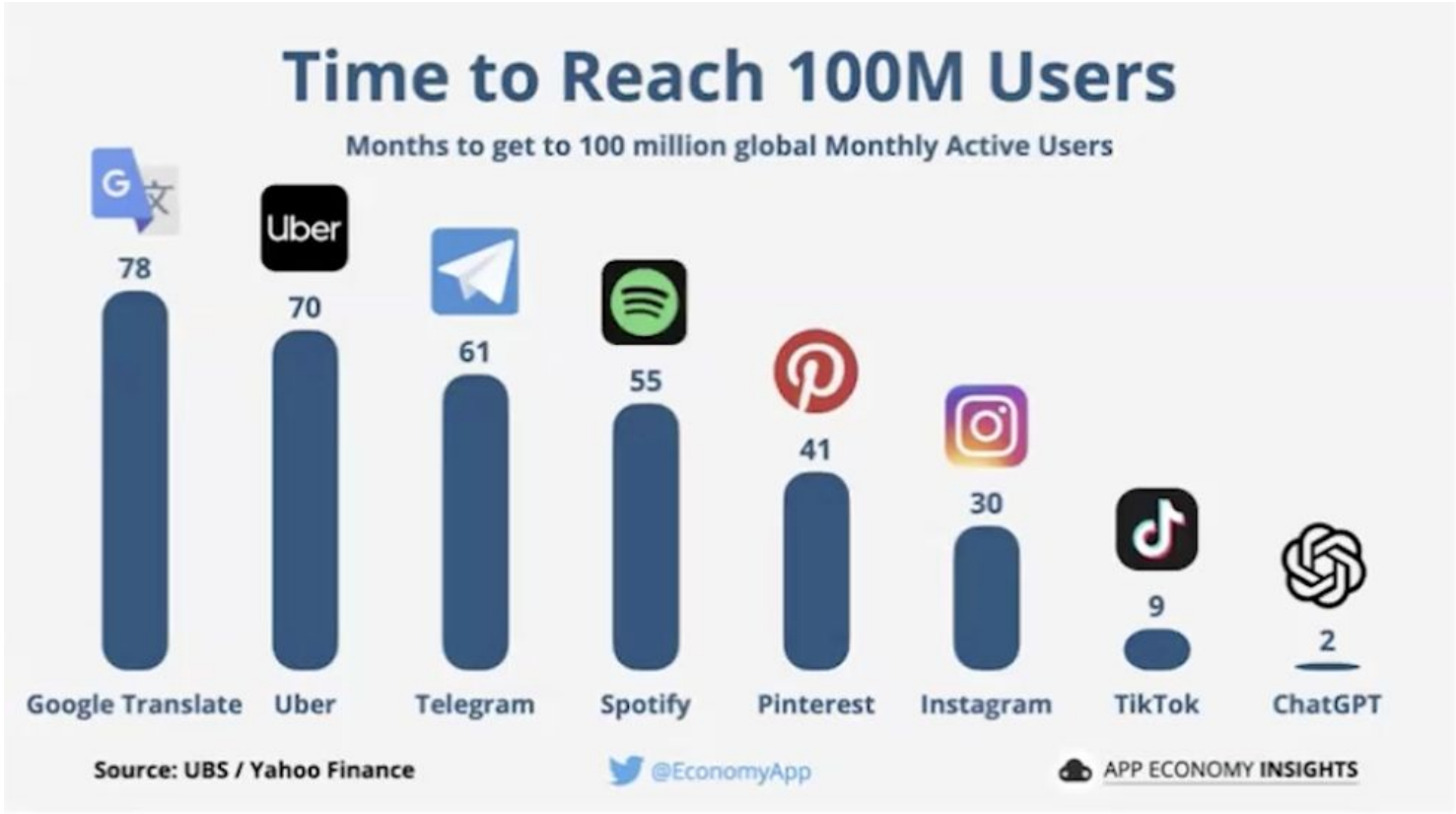
# LLMs & RLHF

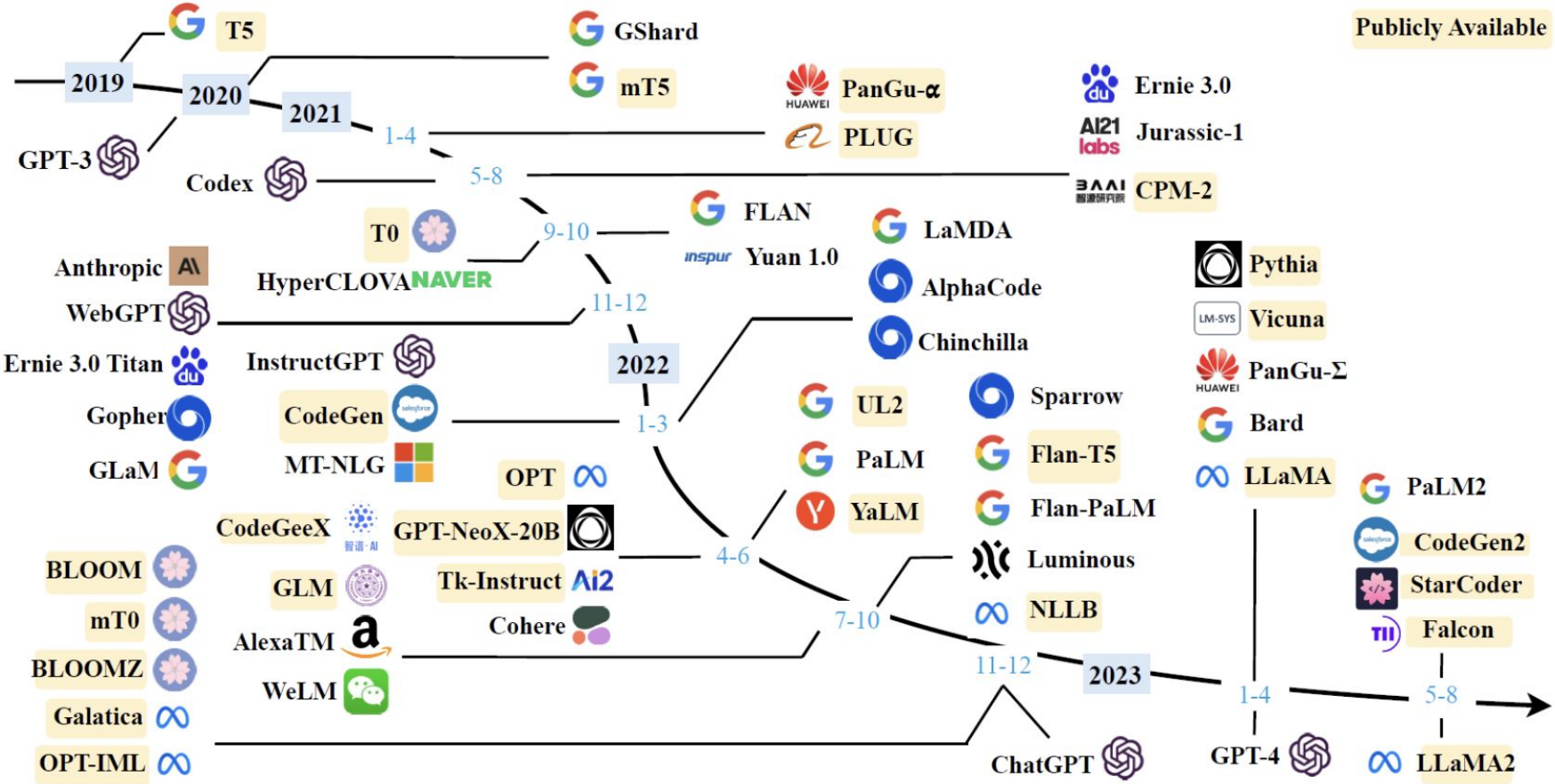
NLP

lecturer: Mollaev D. E.
Sber AI Lab

# Lecture plan:

- Introduction
- Prompting
- The Evolutionary Journey in NLP
- RLHF


Time to Reach 100M Users
Months to get to 100 million global Monthly Active Users
78
70
61
55
41
30
9
2
Google Translate
Uber
Telegram
Spotify
Pinterest
Instagram
TikTok
ChatGPT
Source: UBS / Yahoo Finance
@EconomyApp
APP ECONOMY INSIGHTS


Publicly Available
2019
2020
2021
2022
2023
T5
GPT-3
GShard
mT5
PanGu-α
PLUG
Ernie 3.0
Jurassic-1
CPM-2
1-4
5-8
Codex
T0
HyperCLOVA
FLAN
Yuan 1.0
LaMDA
AlphaCode
Chinchilla
9-10
11-12
Anthropic
WebGPT
Ernie 3.0 Titan
Gopher
GLaM
InstructGPT
CodeGen
MT-NLG
OPT
GPT-NeoX-20B
Tk-Instruct
Cohere
UL2
PaLM
YaLM
Sparrow
Flan-T5
Flan-PaLM
Luminous
NLLB
1-3
4-6
7-10
11-12
CodeGeeX
GLM
AlexaTM
WeLM
BLOOM
mT0
BLOOMZ
Galatica
OPT-IML
ChatGPT
Pythia
Vicuna
PanGu-Σ
Bard
LLaMA
PaLM2
CodeGen2
StarCoder
Falcon
1-4
5-8
GPT-4
LLaMA2

# LLM-arena

| Rank* (UB) | Rank (StyleCtrl) | Model | Arena Score | 95% CI | Votes | Organizatior | License | Knowledge Cutoff |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | chocolate (Early Grok-3) | 1403 | +6/-6 | 9992 | xAI | Proprietary | Unknown |
| 2 | 3 | Gemini-2.0-Flash-Thinking-Exp-01-21 | 1385 | +4/-6 | 15083 | Google | Proprietary | Unknown |
| 2 | 3 | Gemini-2.0-Pro-Exp-02-05 | 1380 | +5/-6 | 13000 | Google | Proprietary | Unknown |
| 2 | 1 | ChatGPT-4o-latest (2025-01-29) | 1377 | +5/-5 | 13470 | OpenAI | Proprietary | Unknown |
| 5 | 3 | DeepSeek-R1 | 1362 | +7/-7 | 6581 | DeepSeek | MIT | Unknown |
| 5 | 8 | Gemini-2.0-Flash-001 | 1358 | +7/-7 | 10862 | Google | Proprietary | Unknown |
| 5 | 3 | o1-2024-12-17 | 1352 | +5/-5 | 17248 | OpenAI | Proprietary | Unknown |
| 8 | 7 | o1-preview | 1335 | +3/-4 | 33169 | OpenAI | Proprietary | 2023/10 |
| 8 | 8 | Qwen2.5-Max | 1334 | +5/-5 | 9282 | Alibaba | Proprietary | Unknown |
| 8 | 7 | o3-mini-high | 1332 | +5/-9 | 5954 | OpenAI | Proprietary | Unknown |
| 11 | 11 | DeepSeek-V3 | 1318 | +4/-5 | 19461 | DeepSeek | DeepSeek | Unknown |
| 11 | 13 | Qwen-Plus-0125 | 1311 | +9/-7 | 5112 | Alibaba | Proprietary | Unknown |
| 11 | 14 | GLM-4-Plus-0111 | 1310 | +6/-9 | 5134 | Zhipu | Proprietary | Unknown |
| 11 | 13 | Gemini-2.0-Flash-Lite-Preview-02-05 | 1309 | +6/-5 | 10262 | Google | Proprietary | Unknown |

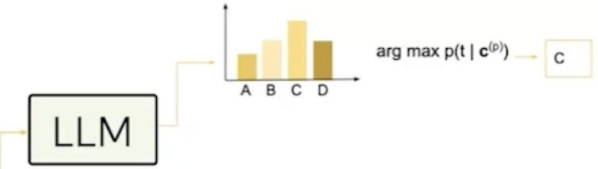
# Lecture plan:

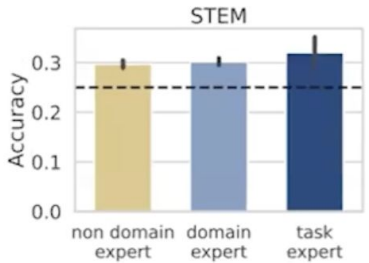
- Introduction
- Prompting
- The Evolutionary Journey in NLP
- RLHF

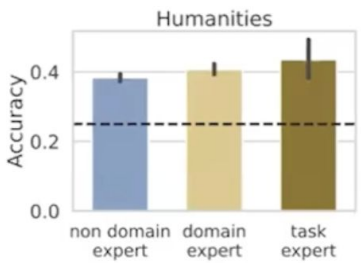
# In-Context Impersonation

Please consider the following multiple-choice question and the four answer options A, B, C, and D. Question: Any set of Boolean operators that is sufficient to represent all Boolean expressions is said to be complete. Which of the following is NOT complete?
A: {AND, NOT}, B: {NOT, OR}, C: {AND, OR}, D: {NAND}

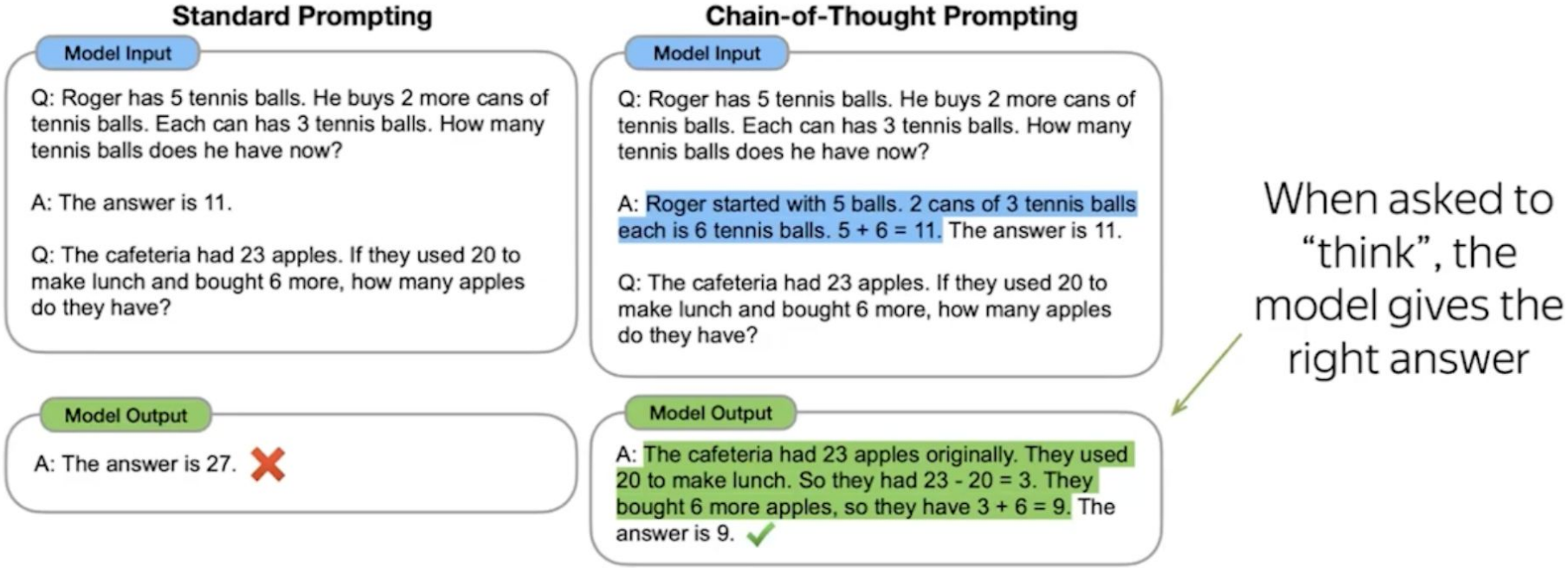
If you were a **high-school computer science expert**, which answer would you choose?



# Chain of Thought



# Self-Consistency

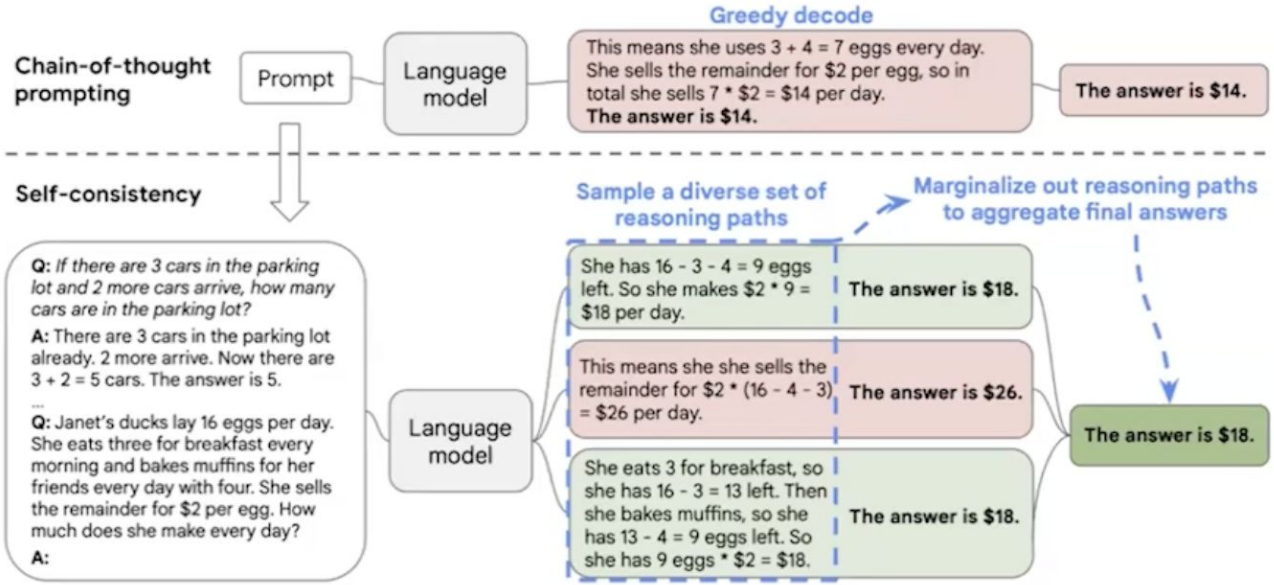
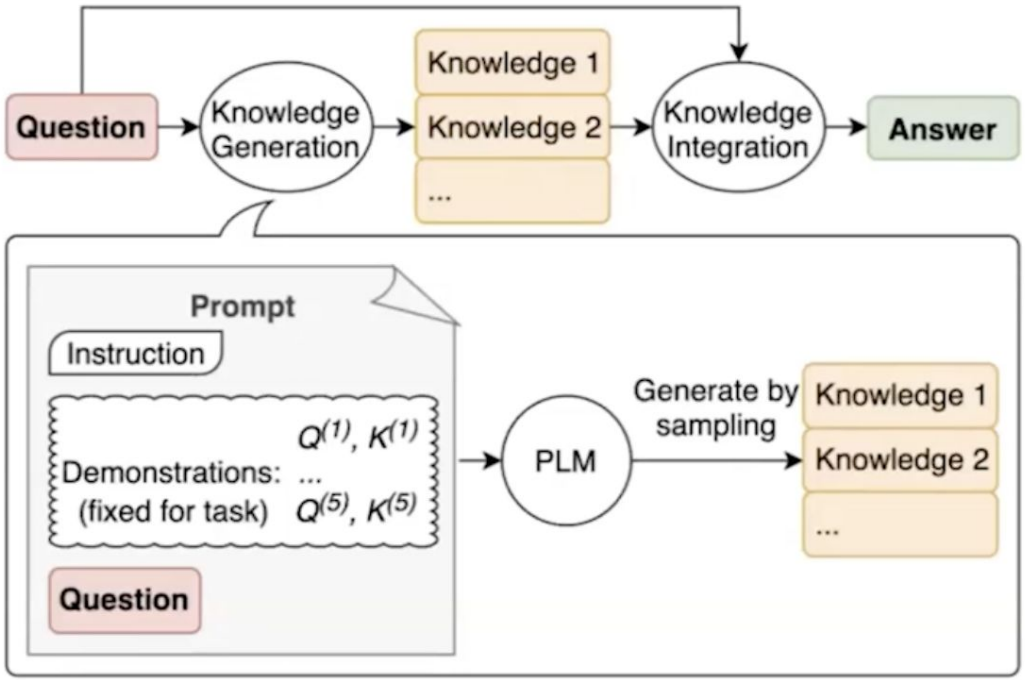



Figure 1: The self-consistency method contains three steps: (1) prompt a language model using chain-of-thought (CoT) prompting; (2) replace the "greedy decode" in CoT prompting by sampling from the language model's decoder to generate a diverse set of reasoning paths; and (3) marginalize out the reasoning paths and aggregate by choosing the most consistent answer in the final answer set.

# Generated Knowledge Prompting




(select highest-confidence prediction)

# Generated Knowledge Prompting

## Step 1: Generate knowledge

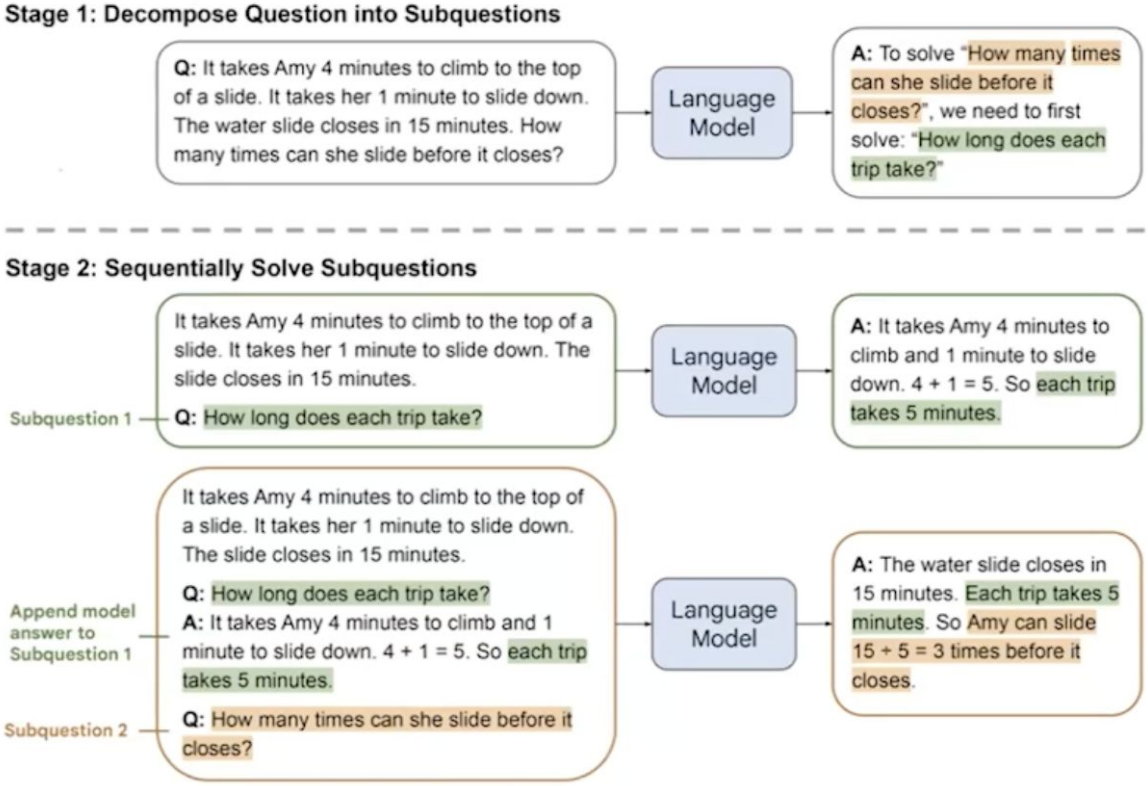
*Prompt:*

```
Input: Greece is larger than mexico.
Knowledge: Greece is approximately 131,957 sq km, while Mexico is approximately 1,964,375 sq km,
making Mexico 1,389% larger than Greece.
Input: Glasses always fog up.
Knowledge: Condensation occurs on eyeglass lenses when water vapor from your sweat, breath, and
ambient humidity lands on a cold surface, cools, and then changes into tiny drops of liquid,
forming a film that you see as fog. Your lenses will be relatively cool compared to your breath,
```

⋮

```
Input: A rock is the same size as a pebble.
Knowledge: A pebble is a clast of rock with a particle size of 4 to 64 millimetres based on the
Udden–Wentworth scale of sedimentology. Pebbles are generally considered larger than granules (2 to
4 millimetres diameter) and smaller than cobbles (64 to 256 millimetres diameter).
Input: Part of golf is trying to get a higher point total than others.
Knowledge:
```

# Least-to-Most Prompting



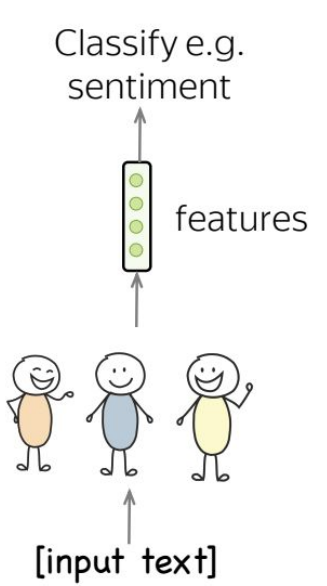

Variants:
"Let's break down the problem: "
"What are the steps needed to solve the task?"
[insert your option]

# Lecture plan:

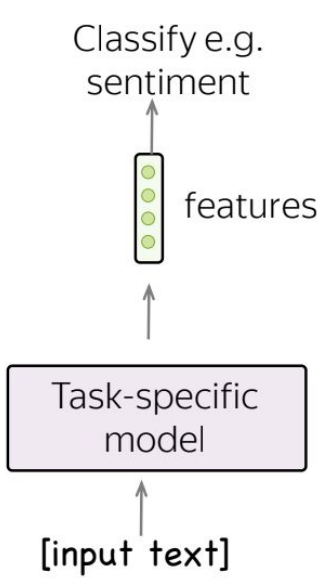
- Introduction
- Prompting
- The Evolutionary Journey in NLP
- RLHF

# The Evolutionary Journey in NLP

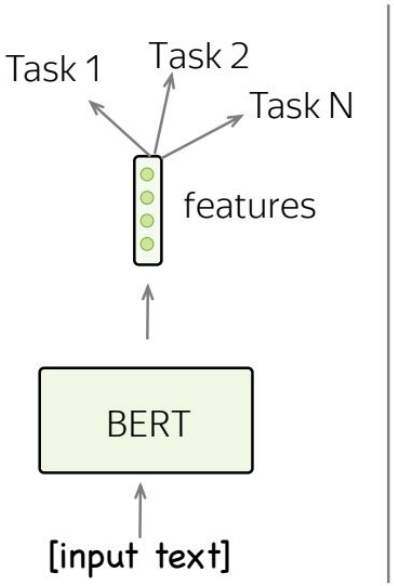
Different task – another set of features (and people!)




Different task – another model




One model, classify anything




Talk to the model

Input (prompt)

What is the sentiment of the next sentence?
I love this movie!

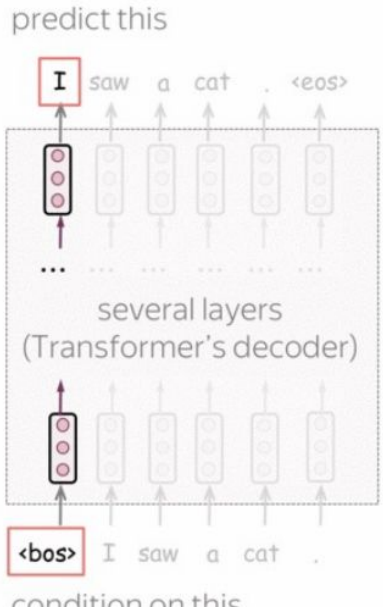
Model output

positive

# GPT: Generative Pre-trained Transformer

## Training

Transformer decoder with the standard left-to-right language modeling objective




## Inference

**GPT-1** Classification via
- Task-specific input transformations
- Supervised fine-tuning

**GPT-2** Generation tasks via task-specific input transformations

For example, for text summarization simply add "**TL;DR**":

**GPT-3** Complex generation and reasoning tasks via in-context learning: prompt with task description and a few demonstrations

```
Translate English to Spanish:
a black cat -> un gato negro
I am hungry -> tengo hambre
a cup of tea ->
```

← task description
← examples
← prompt

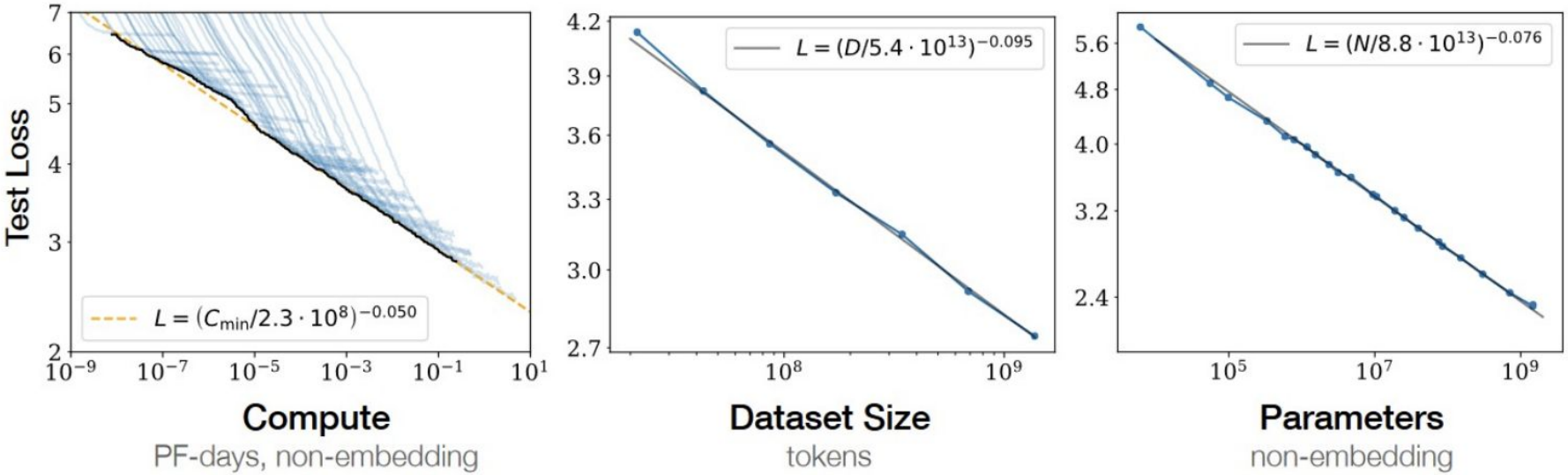
# GPT: Generative Pre-trained Transformer

| | GPT-1 (2018) | GPT-2 (2019) | GPT-3 (2020) |
|---|---|---|---|
| Number of parameters | 117 million | 1.5 billion | 175 billion |
| Training data | 5 GB | 40 GB | 45 TB (i.e., 45 000 GB) |

The entirety of English Wikipedia constitutes just 0.6% of GPT-3 training data

# Scaling Laws

How does model quality change with respect to compute spent? | dataset size? | model size?



# What we want: Instruction Following

Paper: https://arxiv.org/abs/2203.02155

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

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

**GPT-3 175B completion:**
A. to store the value of C[0]
B. to store the value of C[1]
C. to store the value of C[i]
D. to store the value of C[i - 1]

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

# Instruction Tuning

Training objective:

what we **want** vs
what we **told** model to do

What we **told** model to do:

- predict the next token on a webpage from the internet

Alignment

What we want model to do:

- follow the user's instructions helpfully and safely

The language modeling objective is misaligned

# Lecture plan:

- Introduction
- Prompting
- The Evolutionary Journey in NLP
- Reinforcement Learning from Human Feedback (RLHF)

# InstructGPT: basically ChatGPT

Paper: https://arxiv.org/abs/2203.02155

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 with supervised learning.

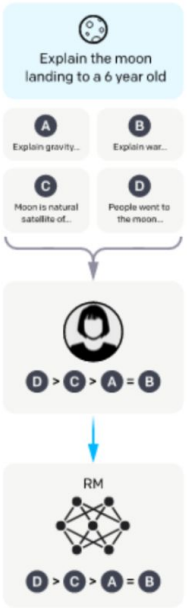

Step 2

**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

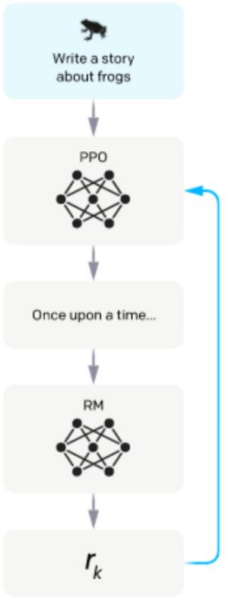
**Optimize a policy against the reward model using reinforcement learning.**

A new prompt is sampled from the dataset.

The policy generates an output.

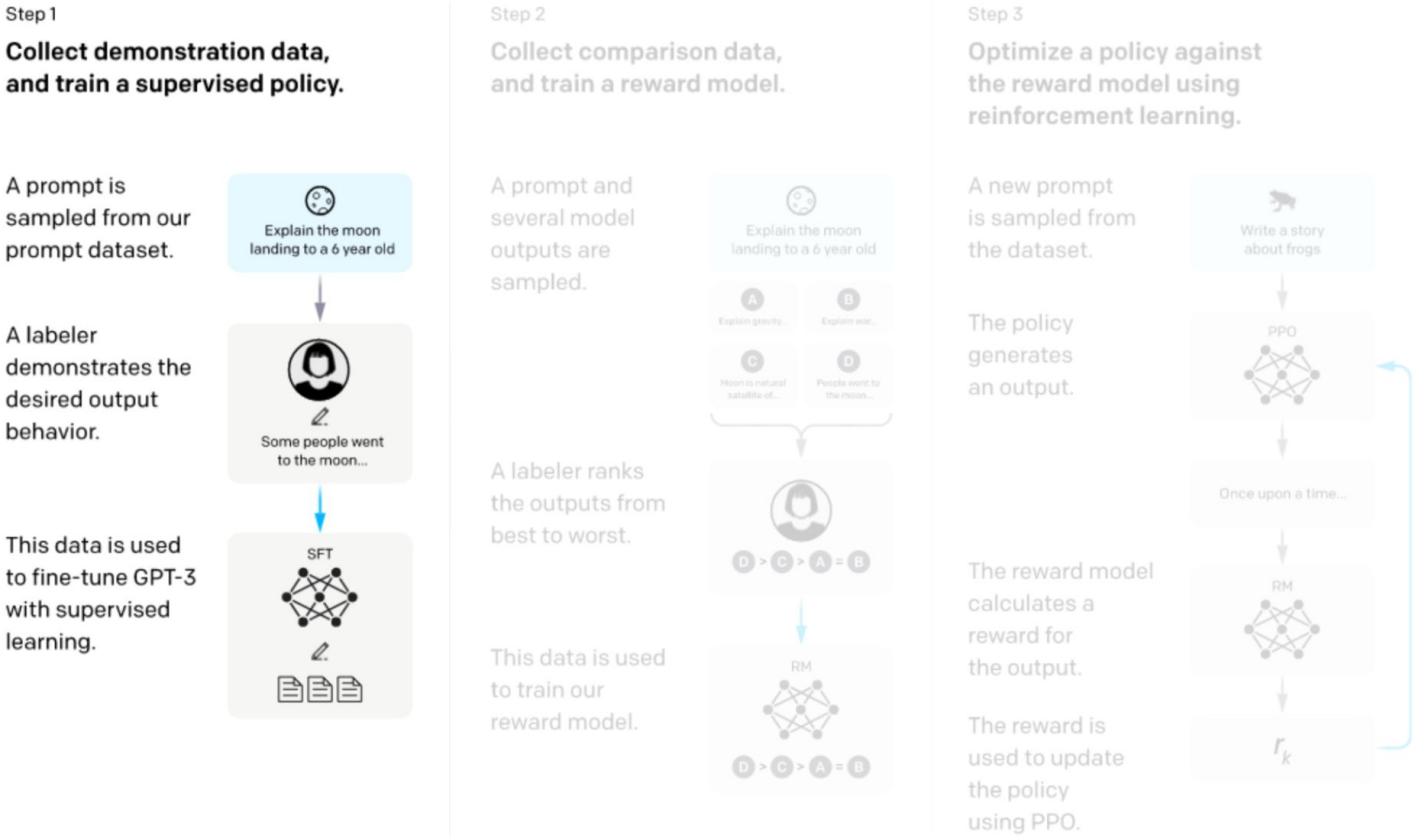
The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



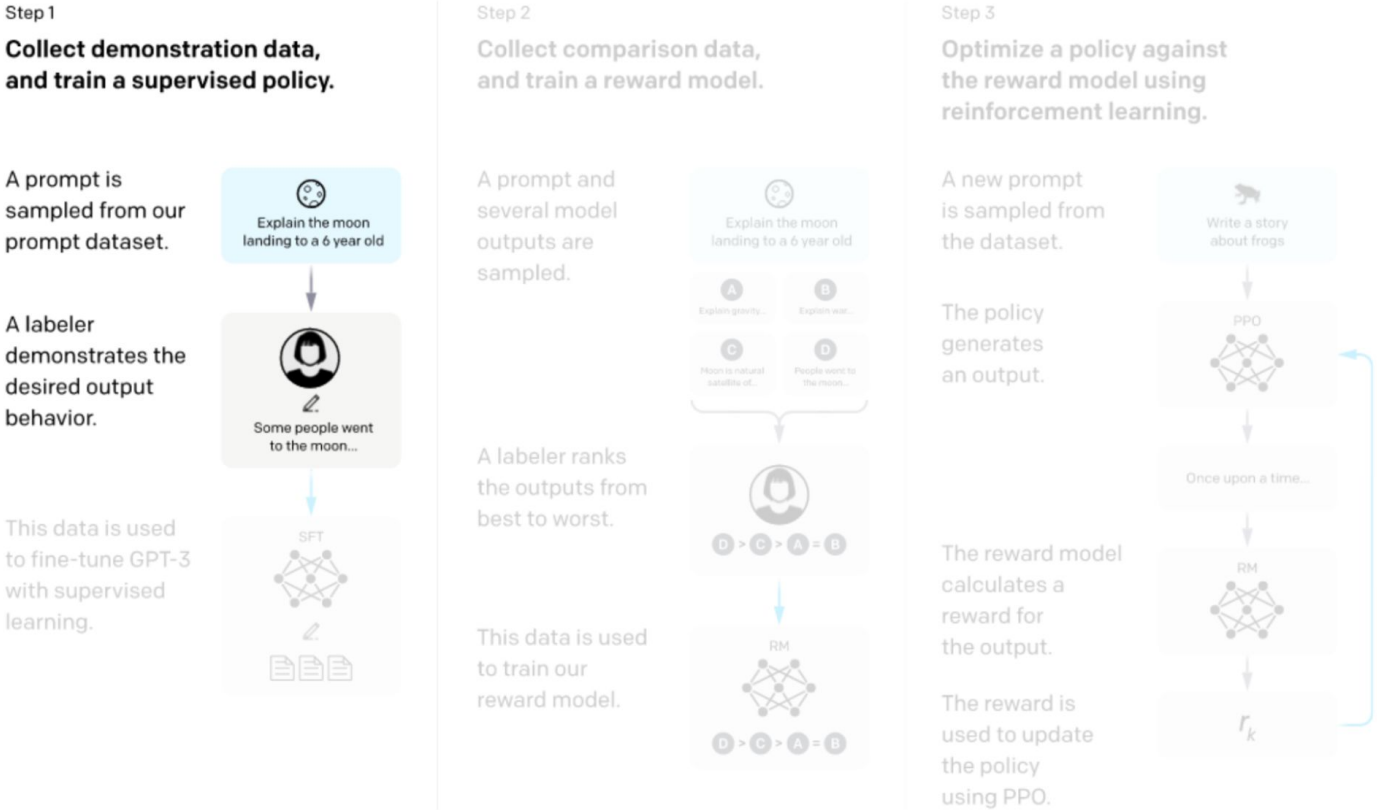
# Stage 1: Supervised Fine-tuning

Paper: https://arxiv.org/abs/2203.02155



# Stage 1: Supervised Fine-tuning

Paper: https://arxiv.org/abs/2203.02155



# Stage 1: Supervised Fine-tuning

Paper: https://arxiv.org/abs/2203.02155

## Prompt Dataset

### Initial stage

Manually written by labelers

- Plain: come up with an arbitrary task, while ensuring the tasks had sufficient diversity
- Few-shot: come up with an instruction, and multiple query/response pairs for that instruction
- User-based: come up with prompts corresponding to some of the use-cases stated in waitlist applications to the OpenAI API

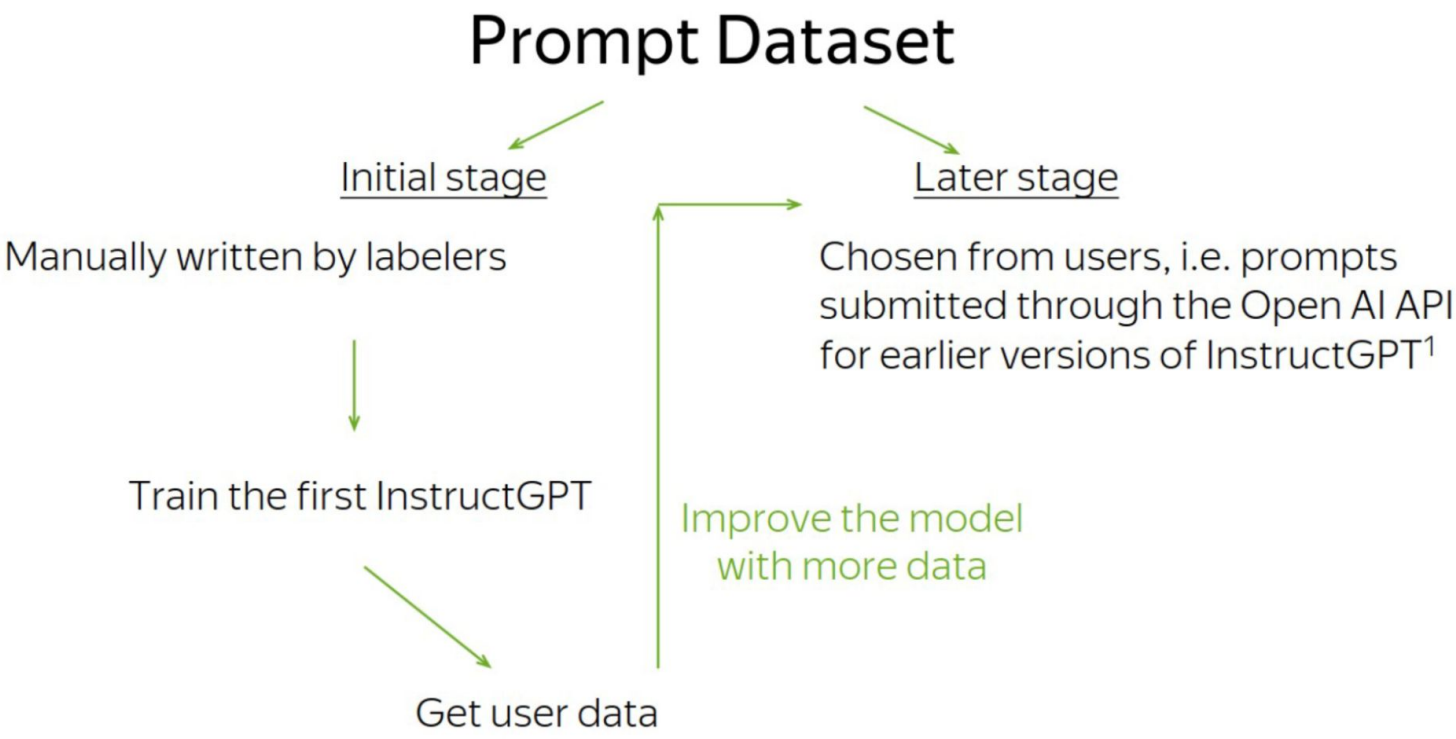
### Later stage

Chosen from users, i.e. prompts submitted through the Open AI API for earlier versions of InstructGPT[1]

- filter all prompts in the training split for personally identifiable information (PII)
- heuristically deduplicate prompts
- no more than 200 prompts per user ID
- create train, validation, and test splits based on user ID

# Stage 1: Supervised Fine-tuning

Paper: https://arxiv.org/abs/2203.02155



# Stage 1: Supervised Fine-tuning

Paper: https://arxiv.org/abs/2203.02155

## Use cases

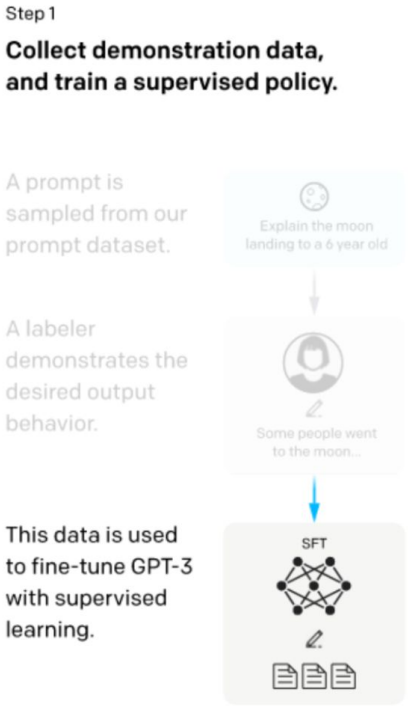
| Use-case | (%) |
|---|---|
| Generation | 45.6% |
| Open QA | 12.4% |
| Brainstorming | 11.2% |
| Chat | 8.4% |
| Rewrite | 6.6% |
| Summarization | 4.2% |
| Classification | 3.5% |
| Other | 3.5% |
| Closed QA | 2.6% |
| Extract | 1.9% |

## Examples

| Use-case | Prompt |
|---|---|
| Brainstorming | List five ideas for how to regain enthusiasm for my career |
| Generation | Write a short story where a bear goes to the beach, makes friends with a seal, and then returns home. |
| Rewrite | This is the summary of a Broadway play:<br>"""<br>{summary}<br>"""<br>This is the outline of the commercial for that play:<br>""" |

# Stage 1: Supervised Fine-tuning

Paper: https://arxiv.org/abs/2203.02155




**Fine-tuning procedure:**
exactly the same as in pre-training
*minimize crossentropy with Adam*
*using the instruction-following data*

Cheaper version: train LoRA adapters
https://arxiv.org/abs/2305.14314

# Why can’t we stop at SFT stage?

**Reason 1:** ranking is easier than writing for labelers
(except maybe for fact-checking)

**Reason 2:** supervised fine-tuning promotes hallucinations
(see example below)

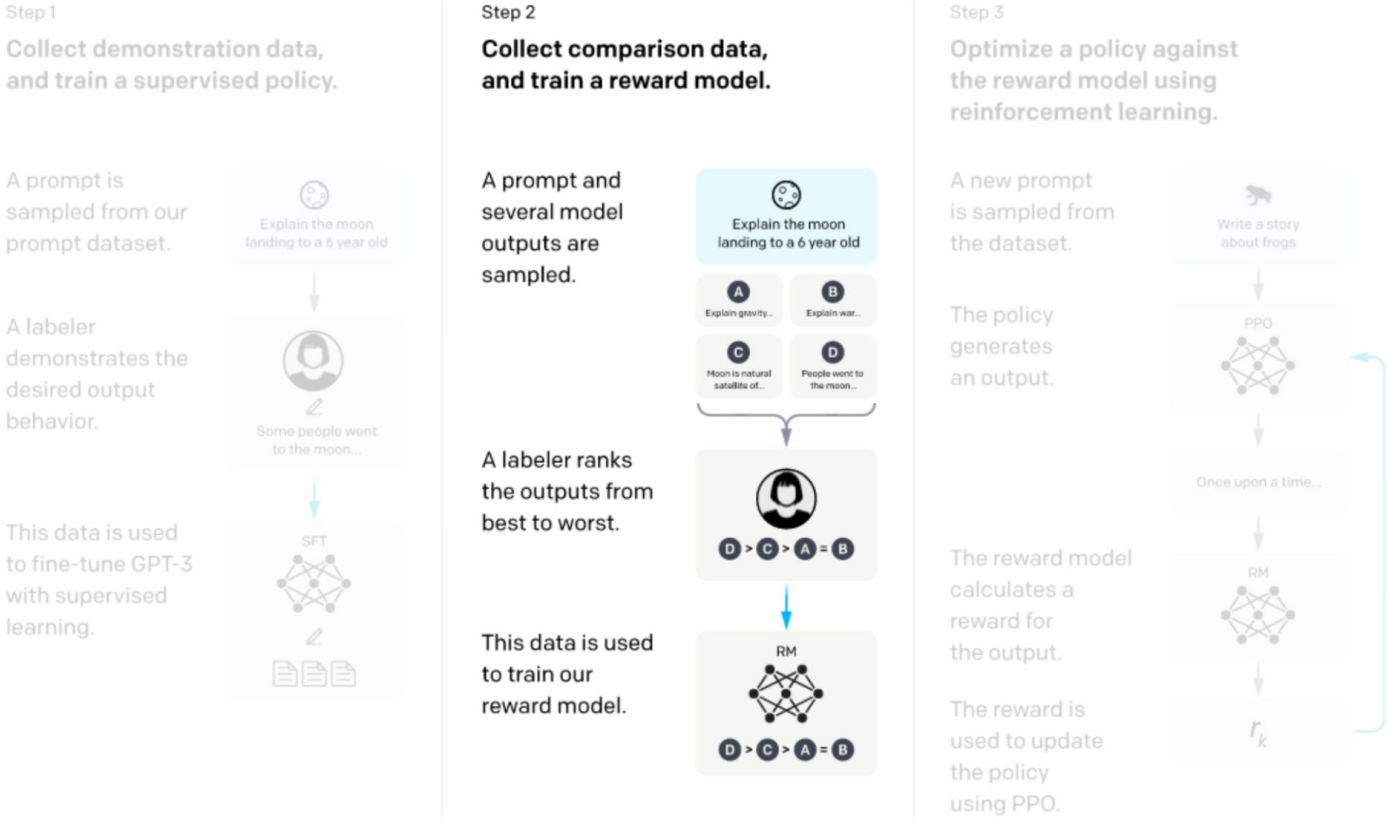
01 Assume the model doesn’t know who killed Pushkin

02 Assume we have **Who killed Pushkin? → Dantes** in the dataset

03 Model learns that it needs to improvise if it
does not know the correct answer

# Stage 2: Reward Model

Paper: https://arxiv.org/abs/2203.02155



# Stage 2: Reward Model

How OpenAI did it

Choose:

- 40 contractors on Upwork and through ScaleAI
- labelers who were sensitive to the preferences of different demographic groups
- labelers who were good at identifying outputs that were potentially harmful

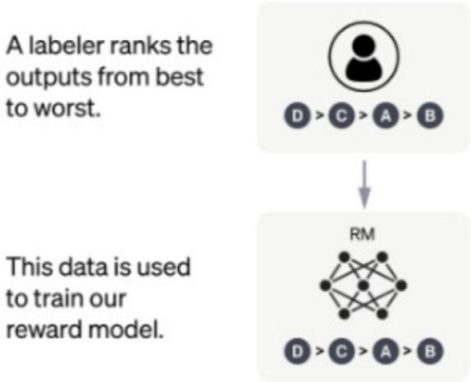
Mentor:

- Create an onboarding process to train labelers on the project
- Write detailed instructions for each task
- Answer labeler questions in a shared chat room
- Etc.

# Stage 2: Reward Model

Paper: https://arxiv.org/abs/2203.02155

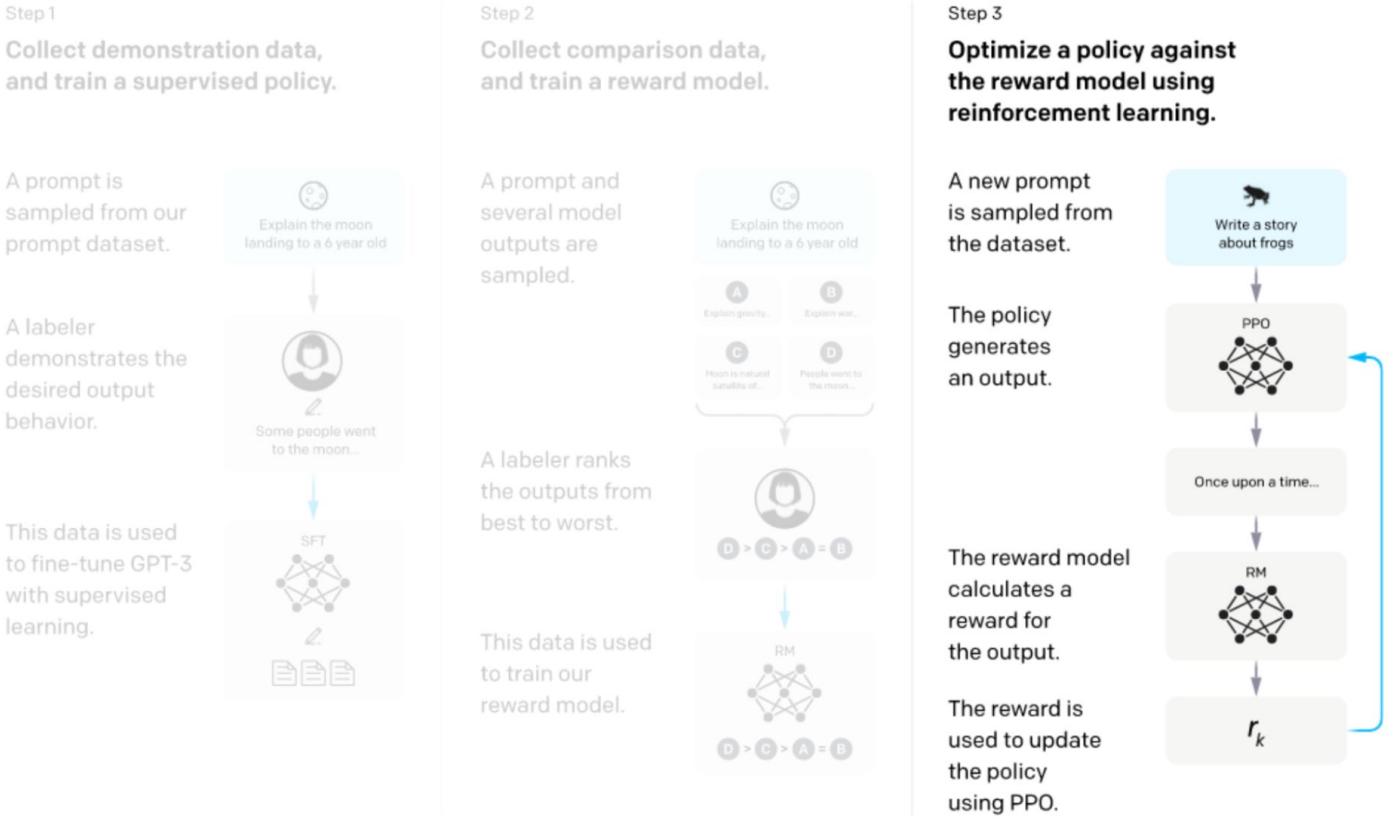
- We want the reward model to give such scores that the ranking is similar to that of humans.
- For every pair the ranking is wrong, the reward model is penalized.




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

# Stage 3: Reinforcement Learning

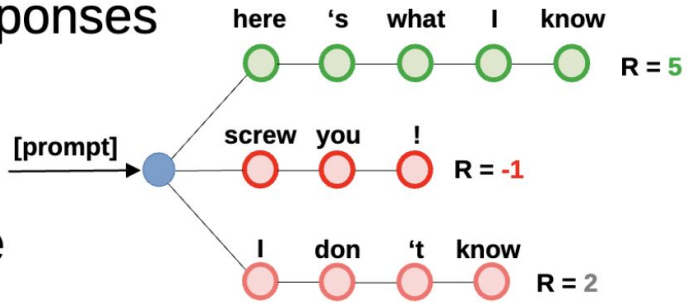
Paper: https://arxiv.org/abs/2203.02155



# Stage 3: Reinforcement Learning

TL;DR reinforcement learning for LLM tasks:

1. Let the model generate several responses (sample with probability)




2. Compute reward for each response (apply reward model)

3. Train to increase probability of responses **with high reward** (and decrease probability if reward is low)

# Stage 3: Reinforcement Learning

Policy gradient basics

**Policy** = probability of response (from your language model)

$$\pi_\theta(y|x) = P_\theta(y_0, \ldots, y_T|x) = \prod_t^{T-1} P_\theta(y_{t+1}|y_{0:t}, x)$$

**Reward** $R_\psi(x, y)$ = your reward model prediction

**Objective:** average reward, in expectation over policy

$$J = E_{x \sim p_{data}(x)} E_{y \sim \pi_\theta(y|x)} R_\psi(x, y)$$

higher = better

# Stage 3: Reinforcement Learning

Policy gradient basics

**Objective:** average reward, in expectation over policy

$$J = E_{x \sim p_{data}(x)} E_{y \sim \pi_\theta(y|x)} R_\psi(x, y)$$

**Step 1:** estimate J with a batch of samples

**Step 2:** compute gradient $\frac{\partial J}{\partial \theta}$

# Stage 3: Reinforcement Learning

REINFORCE, Williams (1992)

**Objective:** average reward, in expectation over policy

$$J = E_{x \sim p_{data}(x)} E_{y \sim \pi_\theta(y|x)} R_\psi(x, y)$$

**Step 1:** estimate J with a batch of samples

$$J_{mc} = \frac{1}{N} \sum_{i=1}^{N} R_\psi(x_i, y_i), \quad where\ x_i \sim P_{data}(x), y_i \sim \pi_\theta(y|x_i)$$

# Stage 3: Reinforcement Learning

REINFORCE, Williams (1992)

**Objective:** average reward, in expectation over policy

$$J = E_{x \sim p_{data}(x)} E_{y \sim \pi_\theta(y|x)} R_\psi(x, y)$$

**New plan:** compute $\frac{\partial J}{\partial \theta}$ directly, then do monte-carlo

$$J = \sum_x P_{data}(x) \cdot \sum_y \pi_\theta(y|x) \cdot R_\psi(x, y)$$

sum over all possible x and y pairs (intractable)

but we'll deal with that later

# Stage 3: Reinforcement Learning

REINFORCE, Williams (1992)

**Objective:** average reward, in expectation over policy

$$J = E_{x \sim p_{data}(x)} E_{y \sim \pi_\theta(y|x)} R_\psi(x, y)$$

**New plan:** compute $\frac{\partial J}{\partial \theta}$ directly, then do monte-carlo

$$J = \sum_x P_{data}(x) \cdot \sum_y \pi_\theta(y|x) \cdot R_\psi(x, y)$$

$$\nabla_\theta \; J = \nabla_\theta \; [\sum_x P_{data}(x) \cdot \sum_y \pi_\theta(y|x) \cdot R_\psi(x, y)]$$

# Stage 3: Reinforcement Learning

REINFORCE, Williams (1992)

**Objective:** average reward, in expectation over policy

$$J = E_{x \sim p_{data}(x)} E_{y \sim \pi_\theta(y|x)} R_\psi(x, y)$$

**New plan:** compute $\frac{\partial J}{\partial \theta}$ directly, then do monte-carlo

$$J = \sum_x P_{data}(x) \cdot \sum_y \pi_\theta(y|x) \cdot R_\psi(x, y)$$

$$\nabla_\theta J = \underbrace{\sum_x P_{data}(x)}_{\text{const}(\theta)} \cdot \nabla_\theta \left[ \sum_y \pi_\theta(y|x) \cdot R_\psi(x, y) \right]$$

# Stage 3: Reinforcement Learning

REINFORCE, Williams (1992)

**Objective:** average reward, in expectation over policy

$$J = E_{x \sim p_{data}(x)} E_{y \sim \pi_\theta(y|x)} R_\psi(x, y)$$

**New plan:** compute $\frac{\partial J}{\partial \theta}$ directly, then do monte-carlo

$$J = \sum_x P_{data}(x) \cdot \sum_y \pi_\theta(y|x) \cdot R_\psi(x, y)$$

$$\nabla_\theta \ J = \sum_x P_{data}(x) \cdot \sum_y R_\psi(x, y) \cdot \nabla_\theta \ [\pi_\theta(y|x)]$$

Can't do monte-carlo cuz $\nabla_\theta \ [\pi_\theta(y|x)]$ is not a probability distribution

# Stage 3: Reinforcement Learning

REINFORCE, Williams (1992)

**Last formula** from previous slide

$$\nabla_\theta J = \sum_x P_{data}(x) \cdot \sum_y R_\psi(x, y) \cdot \nabla_\theta [\pi_\theta(y|x)]$$

Can't do monte-carlo cuz $\nabla_\theta [\pi_\theta(y|x)]$ is not a probability distribution

**Log-derivative trick:**

$$\nabla_x \log f(x) = \frac{1}{f(x)} \nabla_x f(x)$$

$$f(x) \cdot \nabla_x \log f(x) = \nabla_x f(x)$$

$$\pi_\theta(y|x) \cdot \nabla_\theta \log \pi_\theta(y|x) = \nabla_\theta \pi_\theta(y|x)$$

# Stage 3: Reinforcement Learning

REINFORCE, Williams (1992)

**Last formula** from previous slide

$$\nabla_\theta J = \sum_x P_{data}(x) \cdot \sum_y R_\psi(x, y) \cdot \nabla_\theta \left[\pi_\theta(y|x)\right]$$

$$\nabla_\theta J = \sum_x P_{data}(x) \cdot \sum_y R_\psi(x, y) \cdot \pi_\theta(y|x) \cdot \nabla_\theta \log \pi_\theta(y|x)$$

$$\pi_\theta(y|x) \cdot \nabla_\theta \log \pi_\theta(y|x) = \nabla_\theta \pi_\theta(y|x)$$

# Stage 3: Reinforcement Learning

REINFORCE, Williams (1992)

**Last formula** from previous slide

$$\nabla_\theta J = \sum_x P_{data}(x) \cdot \sum_y R_\psi(x, y) \cdot \nabla_\theta [\pi_\theta(y|x)]$$

$$\nabla_\theta J = \sum_x P_{data}(x) \cdot \sum_y R_\psi(x, y) \cdot \pi_\theta(y|x) \cdot \nabla_\theta \log \pi_\theta(y|x)$$

$$\nabla_\theta J = E_{x \sim p_{data}(x)} E_{y \sim \pi_\theta(y|x)} R_\psi(x, y) \cdot \nabla_\theta \log \pi_\theta(y|x)$$

**Can use monte-carlo estimate**

# Stage 3: Reinforcement Learning

REINFORCE, Williams (1992)

**Last formula** from previous slide

$$\nabla_\theta J = \sum_x P_{data}(x) \cdot \sum_y R_\psi(x, y) \cdot \nabla_\theta [\pi_\theta(y|x)]$$

$$\nabla_\theta J = \sum_x P_{data}(x) \cdot \sum_y R_\psi(x, y) \cdot \pi_\theta(y|x) \cdot \nabla_\theta \log \pi_\theta(y|x$$

**Monte-carlo gradient** (GPU-friendly)

$$[\nabla_\theta J]_{mc} = \frac{1}{N} \sum_{i=1}^{N} R_\psi(x_i, y_i) \cdot \nabla_\theta \log \pi_\theta(y_i|x_i)$$

$$, where\ x_i \sim P_{data}(x), y_i \sim \pi_\theta(y|x_i)$$

# Stage 3: Reinforcement Learning

Modern reinforcement learning

REINFORCE (Williams, 1992)

$$[\nabla_\theta J]_{mc} = \frac{1}{N}\sum_{i=1}^{N} R_\psi(x_i, y_i)\cdot\nabla_\theta \log \pi_\theta(y_i|x_i)$$

$$\theta := \theta + \alpha\frac{\partial J}{\partial \theta}$$

Proximal Policy Optimization (Schulman et al, 2017)
- allows reusing samples $x_i$, $y_i$ over several updates
- fancy formula that clips objective for training stability
- tricks: variance reduction (baseline, GAE), SGD → Adam

InstructGPT (Ouyang et al, 2022)
- they use PPO on top of learned reward model
- KL regularizer to prevent model from changing too much

# Stage 3: Reinforcement Learning

Paper: https://arxiv.org/abs/2110.08207