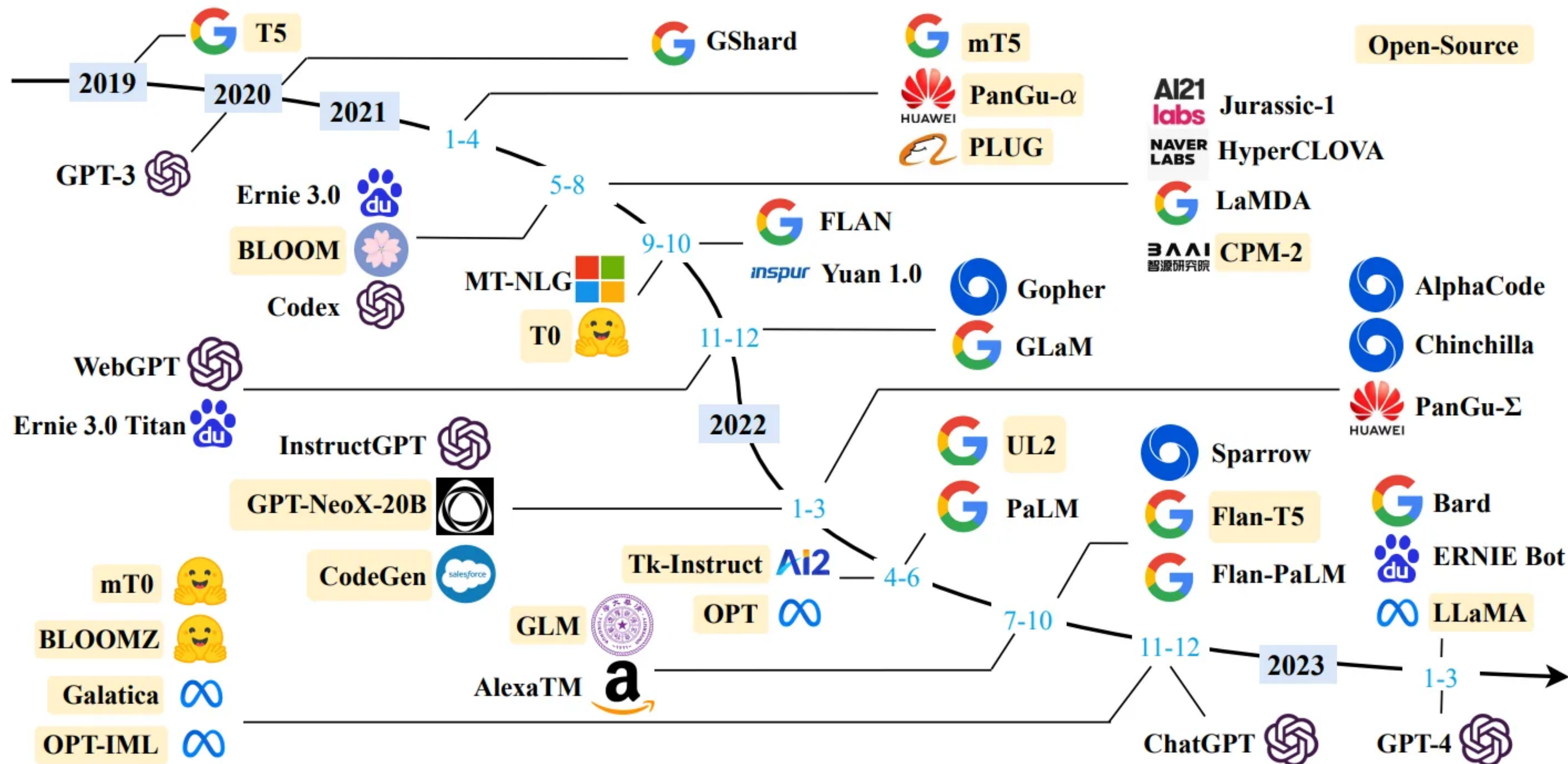


LLM (Large Language Models)



GPT (Generative Pre-Trained Transformer)

2018

GPT1 (0.12B): Improving language understanding by generative pre-training

2019

GPT2 (1.5B): Language Models are Unsupervised Multitask Learners

2020

GPT3 (175B): Language models are few-shot learners

2022

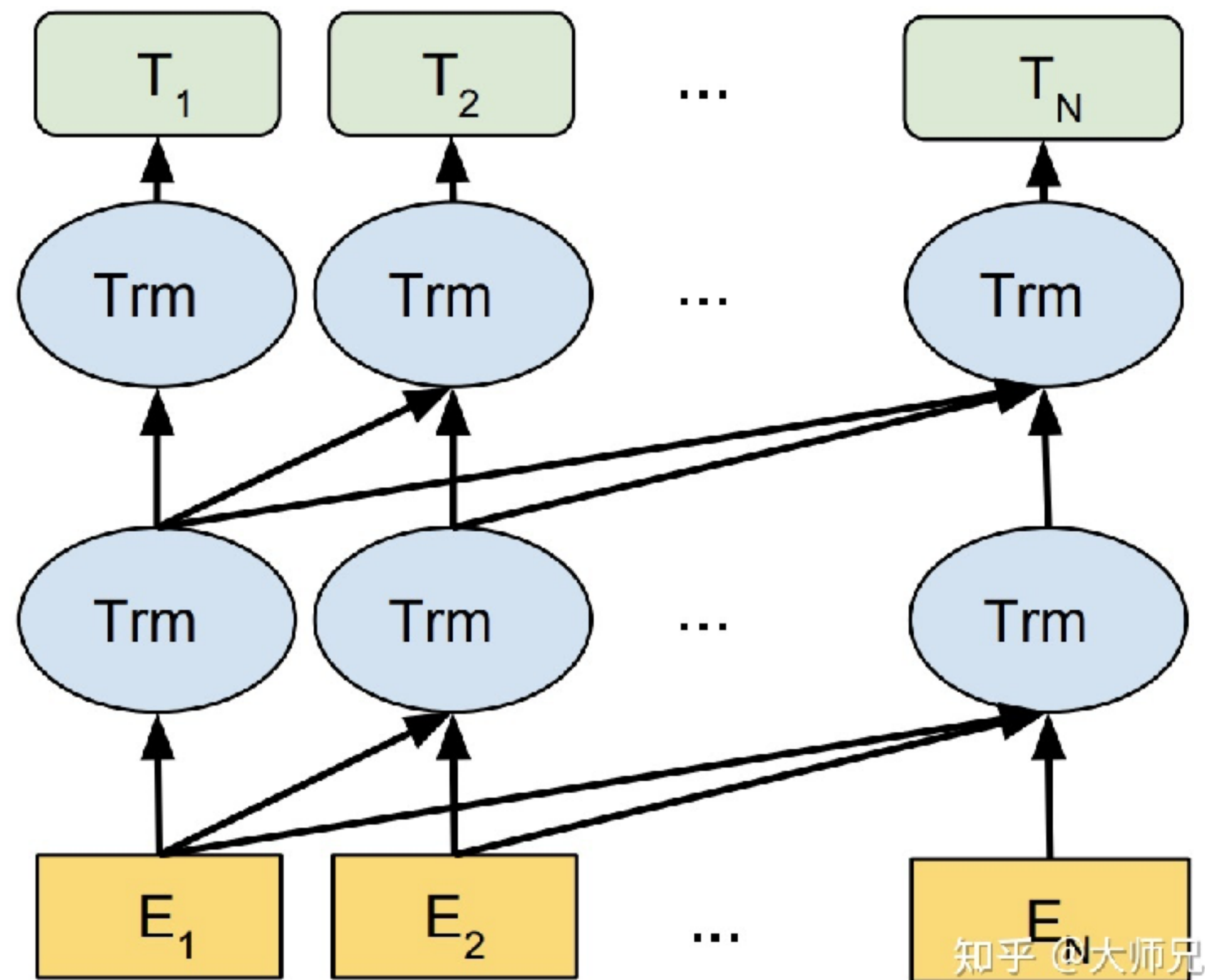
InstructGPT (175B): Training language models to follow instructions with human feedback

ChatGPT (GPT3.5) (175B)

2023

GPT4 (170 trillion)

GPT (Generative Pre-Trained Transformer)



GPT1:

Unsupervised generative pre-training

GPT2:

Zero-shot learning

GPT3:

In-context learning

InstructGPT:

Instruct-tuning with RLHF

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \dots, u_{i-1}; \Theta)$$

An Example of ChatGPT

In the following sample, ChatGPT asks the clarifying questions to debug code.

User this code is not working like i expect — how do i fix it?

```
resultWorkerErr := make(chan error)
defer close(resultWorkerErr)
go func() {
    defer cancel()
    resultWorkerErr <- b.resultWorker(ctx)
}()

err := b.worker(ctx)
cancel()
if err == nil {
    return <-resultWorkerErr
}
return multierror.Append(err, <-resultWorkerErr)
```

ChatGPT It's difficult to say what's wrong with the code without more context. Can you provide more information about what the code is supposed to do and what isn't working as expected? Also, is this the entire code or just a part of it?

User just part of it — the error never surfaces. I think there is something wrong with the channel

ChatGPT It's difficult to say without more information about what the code is supposed to do and what's happening when it's executed. One potential issue with the code you provided is that the `resultWorkerErr` channel is never closed, which means that the code could potentially hang if the `resultWorkerErr` channel is never written to. This could happen if `b.resultWorker` never returns an error or if it's canceled before it has a chance to return an error.

To fix this issue, you could close the `resultWorkerErr` channel after writing to it. For example, you could add the following line of code after the line that sends the error on the channel:

```
close(resultworkerErr)
```

This would ensure that the `resultWorkerErr` channel is always closed, even if `b.resultWorker` never returns an error. Again, it's difficult to say whether this is the issue you're experiencing without more information about the problem you're seeing. Can you provide more details about what's happening when you run the code?

InstructGPT

Training language models to follow instructions with human feedback

Long Ouyang* **Jeff Wu*** **Xu Jiang*** **Diogo Almeida*** **Carroll L. Wainwright***

Pamela Mishkin* **Chong Zhang** **Sandhini Agarwal** **Katarina Slama** **Alex Ray**

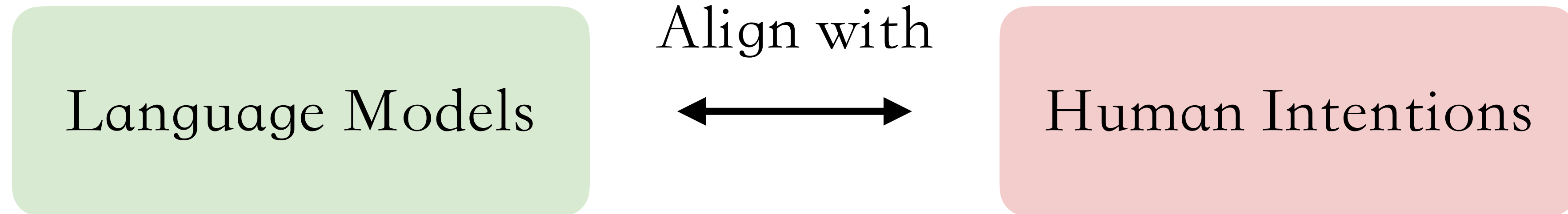
John Schulman **Jacob Hilton** **Fraser Kelton** **Luke Miller** **Maddie Simens**

Amanda Askell[†] **Peter Welinder** **Paul Christiano^{*†}**

Jan Leike* **Ryan Lowe***

OpenAI

Method



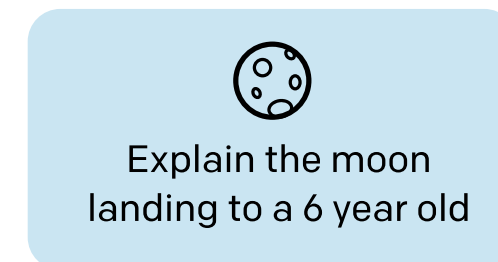
具体方法： Finetune language models with human feedback

Workflow

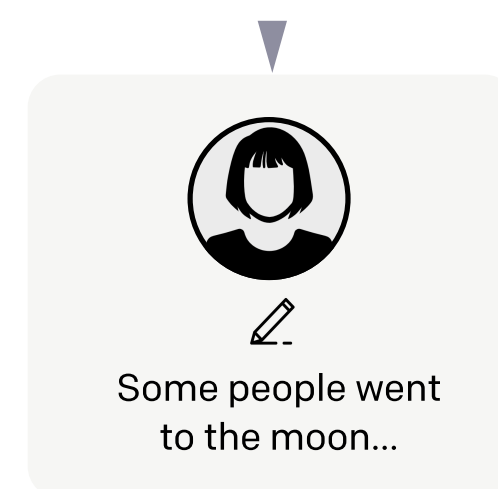
Step 1

**Collect demonstration data,
and train a supervised policy.**

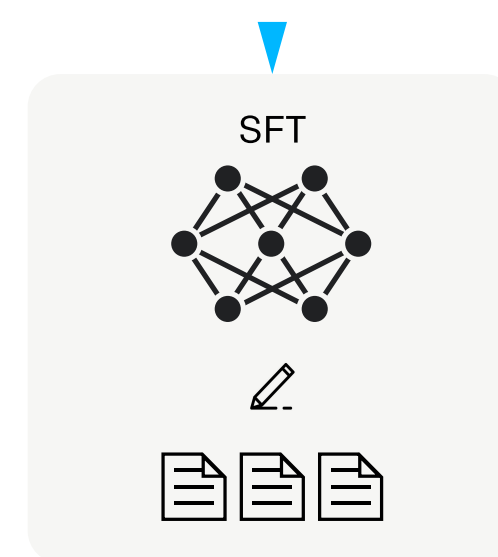
A prompt is
sampled from our
prompt dataset.



A labeler
demonstrates the
desired output
behavior.



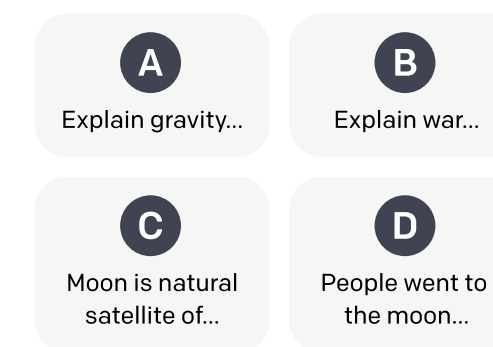
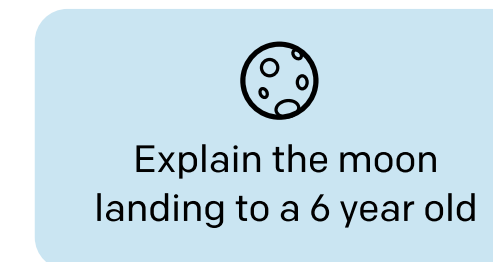
This data is used
to fine-tune GPT-3
with supervised
learning.



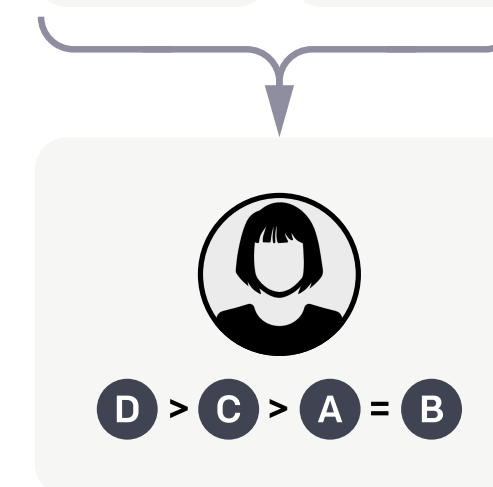
Step 2

**Collect comparison data,
and train a reward model.**

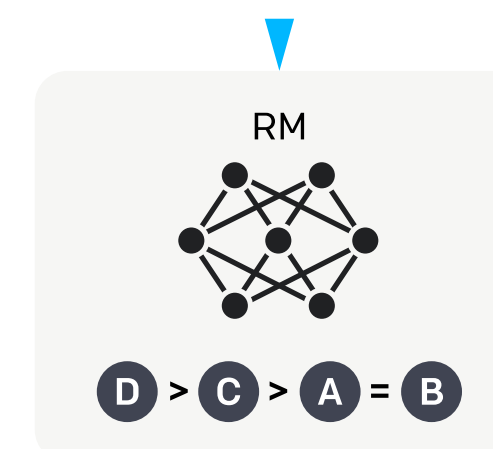
A prompt and
several model
outputs are
sampled.



A labeler ranks
the outputs from
best to worst.



This data is used
to train our
reward model.



Step 3

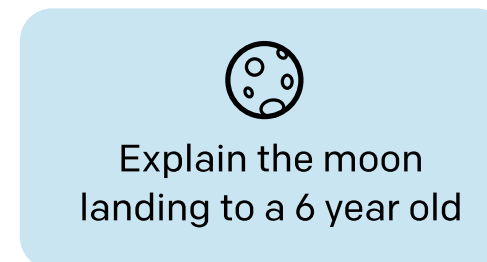
**Optimize a policy against
the reward model using
reinforcement learning.**

Workflow

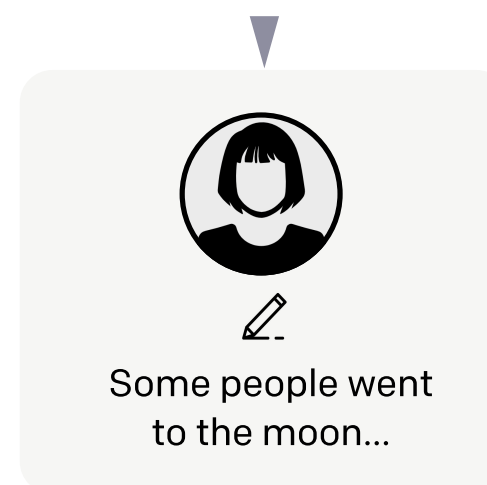
Step 1

**Collect demonstration data,
and train a supervised policy.**

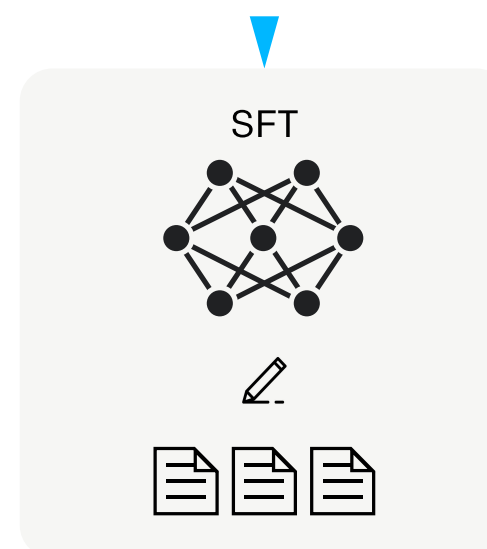
A prompt is
sampled from our
prompt dataset.



A labeler
demonstrates the
desired output
behavior.



This data is used
to fine-tune GPT-3
with supervised
learning.



Step 1: Collect demonstration data, and train a supervised policy.

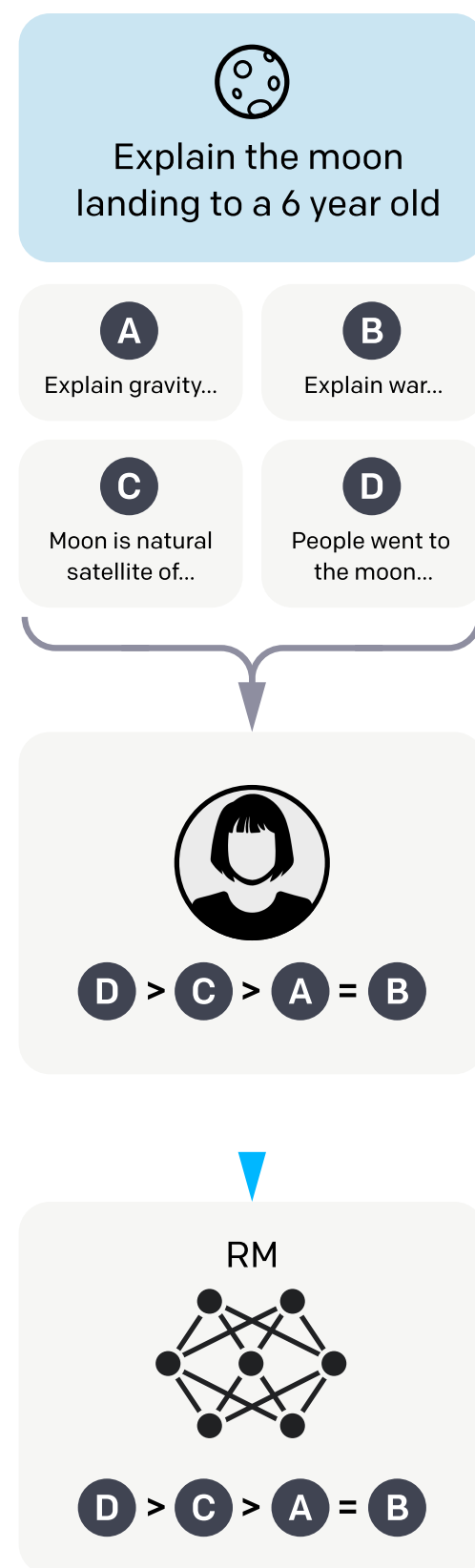
- Labelers demonstrate desired behavior on input prompts.
- Pretrained GPT-3 model is fine-tuned using supervised learning on this data.

Workflow

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 2: Collect comparison data, and train a reward model.

- Collect dataset of model output comparisons.
- Labelers indicate preferred output for a given input.
- Train reward model to predict human-preferred output.

Workflow

Step 3

Optimize a policy against the reward model using reinforcement learning.

Step 3: Optimize a policy against the reward model using Proximal Policy Optimization (PPO).

- Utilize RM output as scalar reward.
- Fine-tune supervised policy with PPO algorithm to optimize reward.

Models

Self-supervised Fine-tuning (SFT)

- Fine-tune GPT-3 on labeler demonstrations.
 - Trained for 16 epochs
 - Use cosine learning rate decay
 - Residual dropout of 0.2
- Choose the final SFT model based on the RM score on the validation set.
- More epochs enhance RM score and human preference ratings, despite overfitting.

Models

Reward Model (RM)

- Labelers **rank** the output of SFT model from high to low.
- Based on SFT model with final unembedding layer removed, train the reward model.
- The reward model takes **prompt and rank** as input, and output **scalar reward**.

Pairwise ranking loss

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

$r_\theta(x, y)$ Scalar output of the reward model for prompt x and answer y

Models

Proximal Policy Optimization (PPO)

PPO

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

Reinforce Learning

π_{ϕ}^{RL} Learned RL policy

π^{SFT} Supervised trained model

Models

Proximal Policy Optimization (PPO)

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

Origin GPT3 objective

Reinforce Learning

π_{ϕ}^{RL} Learned RL policy

π^{SFT} Supervised trained model