Toloka

Overview of Reinforcement Learning from Human Feedback

Generative AI meets human guidance

Nikita Pavlichenko
Research Scientist

About Me

Nikita Pavlichenko

Research Scientist

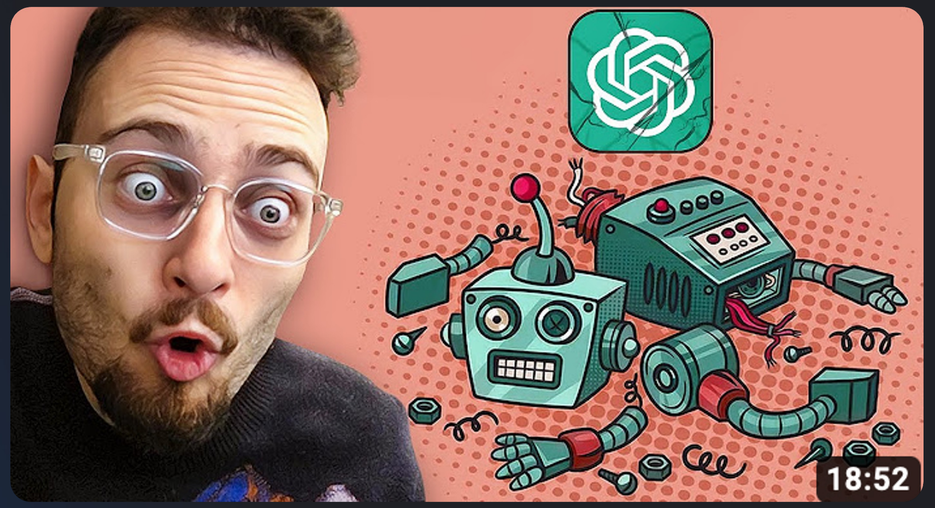
pavlichenko@Toloka.ai

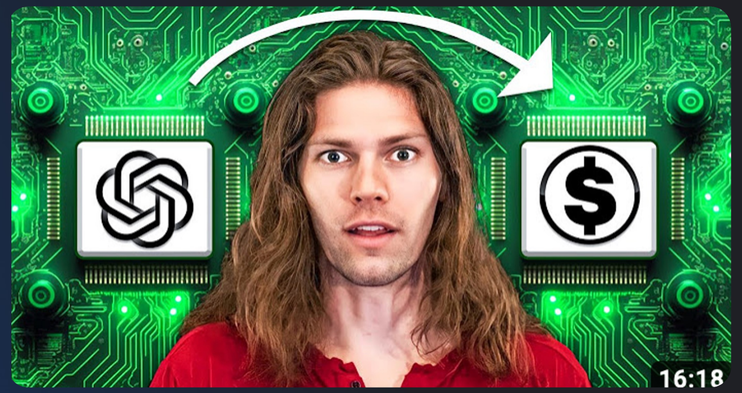
- I’m responsible for RLHF research at Toloka
- Working on Instruct Stable Diffusion and open-source ChatGPT

www.toloka.ai | LinkedIn | GitHub

Why?

Everyone Talks about ChatGPT





Toloka

RLHF in ChatGPT

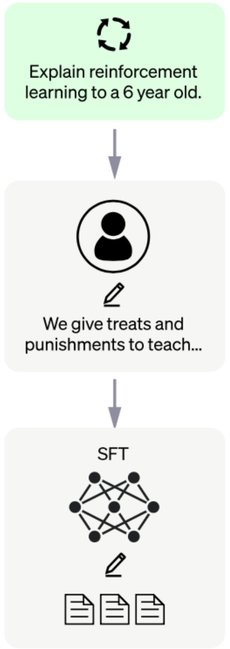
Step 1

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.



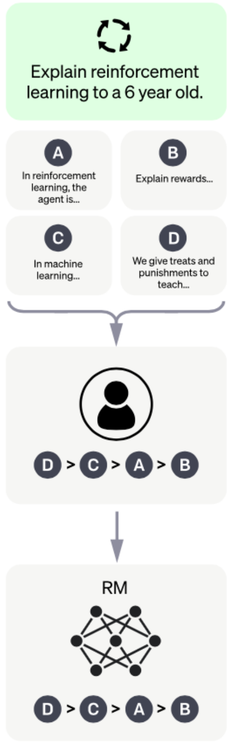
Step 2

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.

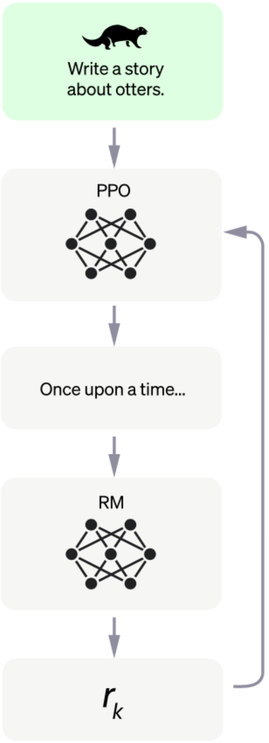
A new prompt is sampled from the dataset.

The PPO model is initialized from the supervised policy.

The policy generates an output.

The reward model calculates a reward for the output.

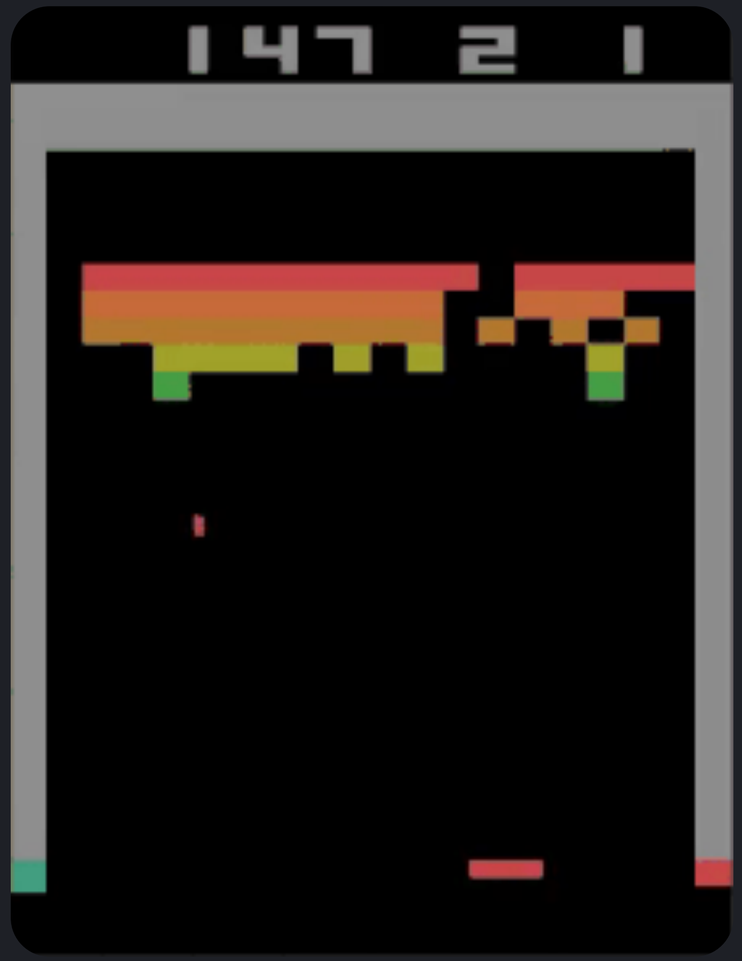
The reward is used to update the policy using PPO.



Reinforcement Learning

RL Motivation

- We want to teach model to do some actions in environment
- Most common example is playing computer games (e.g. Atari) because they are interpretable, easy to implement, and have a well defined reward



Definitions

- **Agent** does actions discretized in time
- At each timestamp, agent observes a current **state** s_t
  (current coordinates, what is showing on the screen, etc.)
- At each timestamp, agent can do some **actions** a_t based on s_t according to **policy**
  (this is what our neural network will predict)
- We want to **maximize the reward** the agent gets in the **episode**
  (until the environment stops, e.g., a car crashes, game ends, etc.)

- Note that in real life is not so easy to define $r$in Atari games it's just the game score but in reality, we must do the reward engineering that will then define how good out agent will perform

RL SOTA

- We can start with baselines like sampling a lot of state-action pairs, computing the reward, an choosing the best actions and then changing our sampling policy that defines a probability distribution over actions that the agent should take in a given state
- We want to use gradient optimization because it works. There are several techniques like Q-learning but SOTA is the **policy gradient**

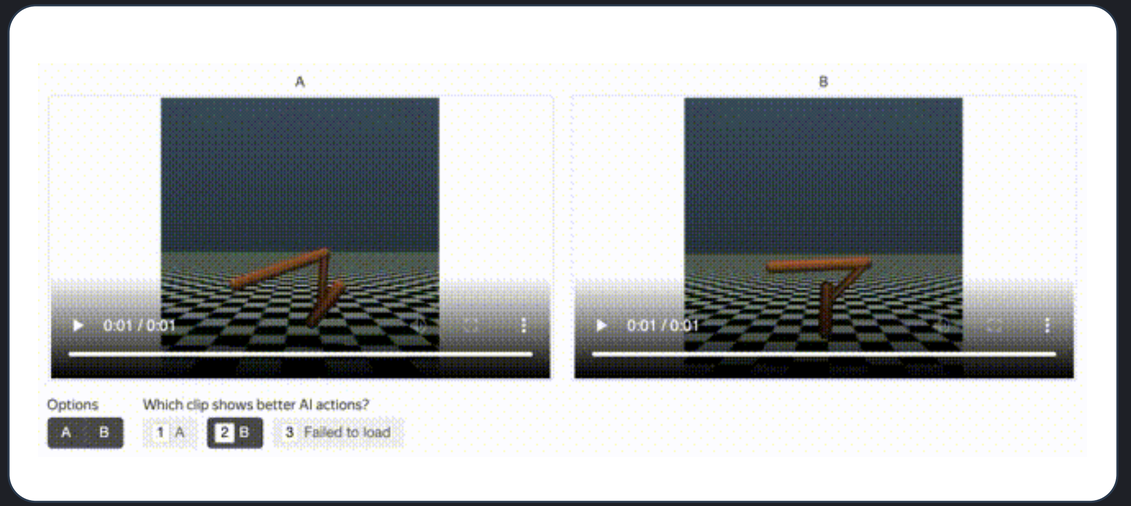
$$J(\theta) = E_{r\sim\pi_\theta} R(\tau) = \sum_{\tau} P(\tau;\theta) R(\tau)$$

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

- Such kind of optimization is difficult and overall optimization in RL is difficult for many reasons, so a well-established baseline is **Proximal Policy Optimization (PPO)**

Reinforcement Learning from Human Feedback

- We know what RL is and what algorithm to use, so performance only depends on the reward engineering and it's extremely difficult to define a good reward function
- We know what agent should do but it's hard to mathematically guide it to show a desired behaviour
- We can show agent's trajectories to humans and ask them to decide which behaviour is closer to a desired one
- Train a separate network that predicts how good action is in each state

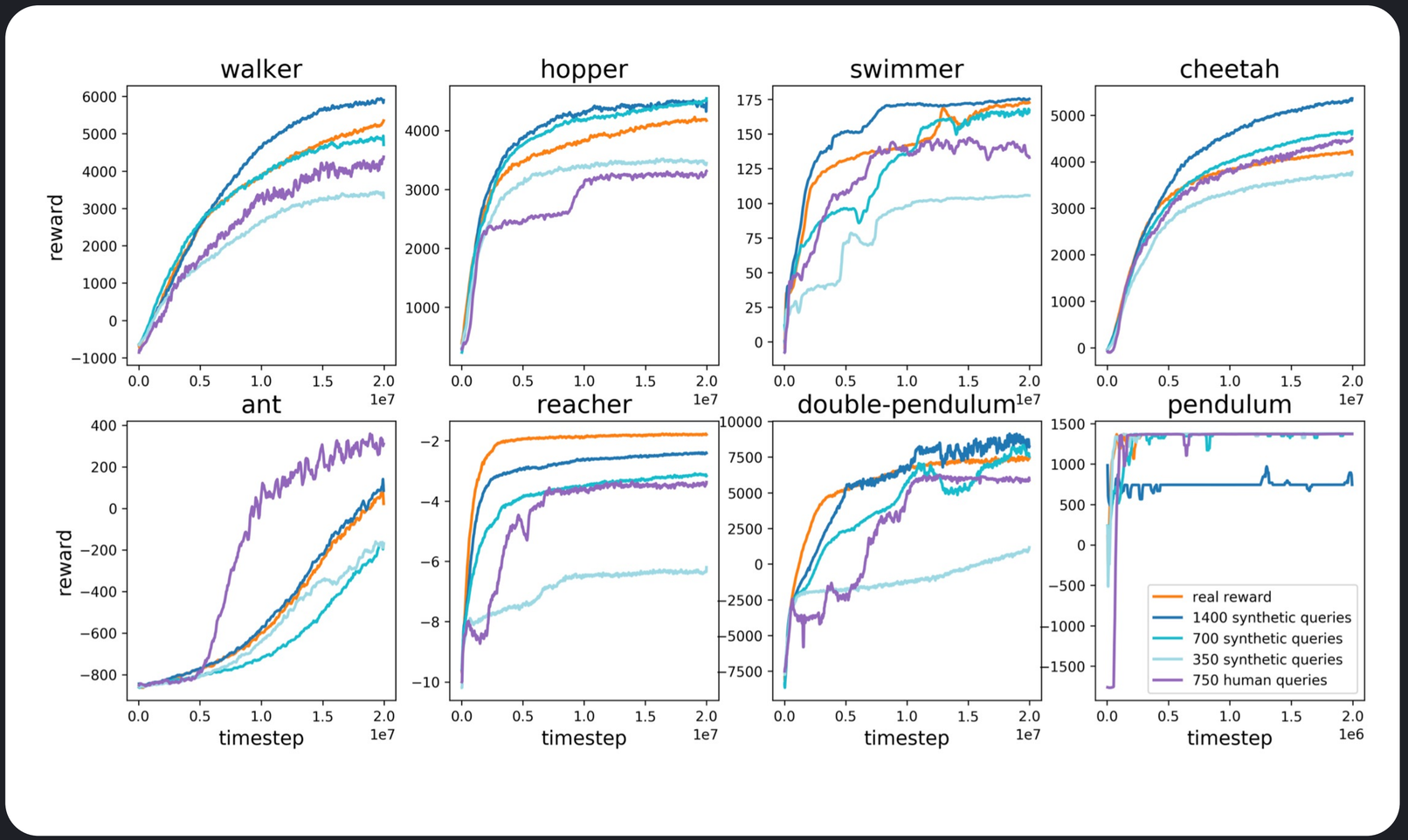


$$\hat{P}\left[\sigma^1 \succ \sigma^2\right] = \frac{\exp \sum \hat{r}(s_t^1, a_t^1)}{\exp \sum \hat{r}(s_t^1, a_t^1) + \exp \sum \hat{r}(s_t^2, a_t^2)}$$

$$\mathcal{L}(\hat{r}) = -\sum_{(\sigma^1, \sigma^2, \mu)} \mu(1) \log \hat{P}\left[\sigma^1 \succ \sigma^2\right] + \mu(2) \log \hat{P}\left[\sigma^2 \succ \sigma^1\right]$$

Reinforcement Learning from Human Feedback

- Agent needs less training steps to start doing something useful
- Overall results are better



Why Is This Relevant?
Who Cares about Walking Robots and Atari Games?

NLP: RL for Machine Translation

- We can think about MT as an RL task
- Model takes some actions (generates words) and gets rewards for them (BLUE score)

$$\nabla_\theta RL = \mathbb{E}_{\mathrm{p}_\theta(\mathrm{y}|\mathrm{x})}[R(y)\nabla_\theta \log p_\theta(y|x)]$$

- We can iterate through multiple source texts and generate translations and in this way, we can get an estimate of a gradient:

$$\widehat{\nabla_\theta RL} = R(\hat{y})\nabla_\theta \log p_\theta(\hat{y}|x), \qquad \hat{y} = p_\theta(y|x)$$

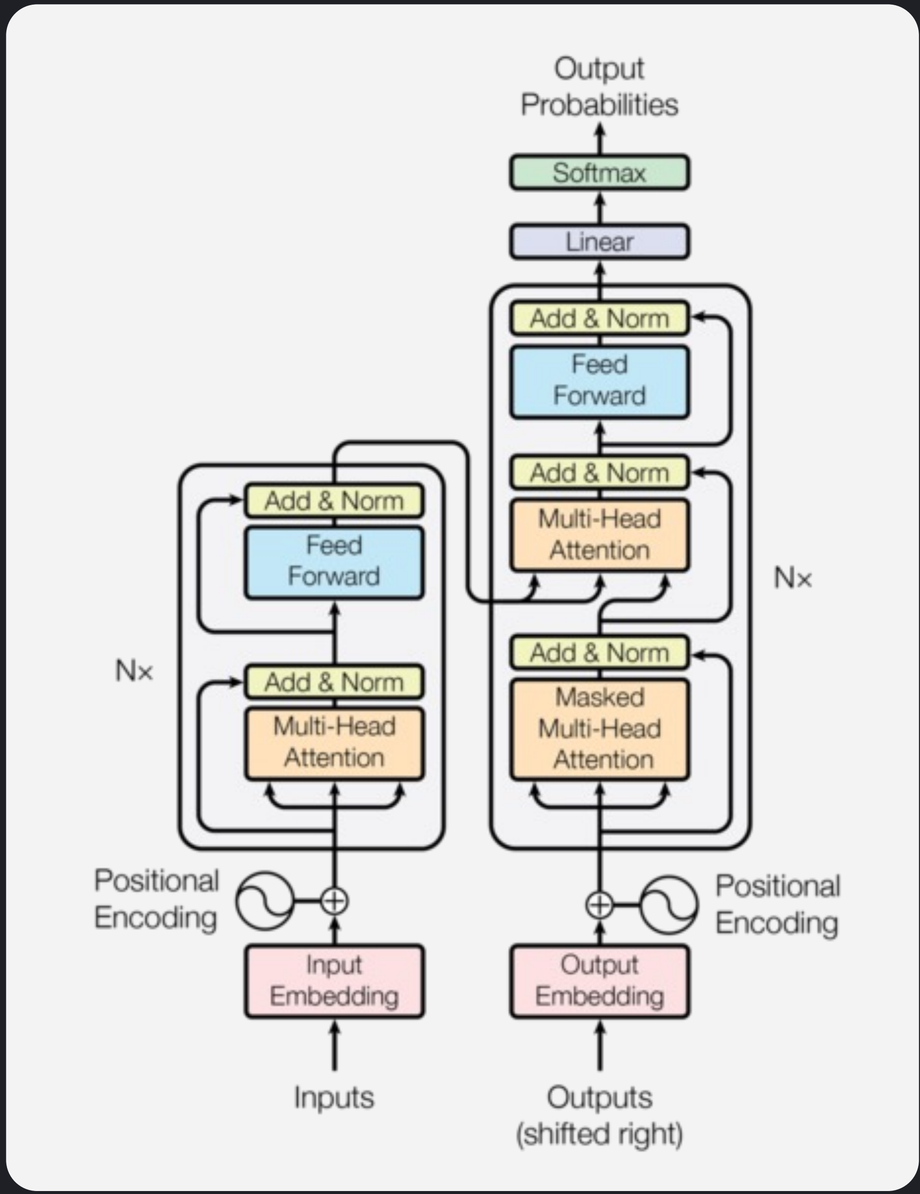
- So, usually it's done as follows: we first train the model on a standard per-token cross-entropy loss and then we "polish" the model with RL on a desired metric
- Everybody knows that BLUE is bad and human evaluations are good, so maybe we can learn the reward function from human feedback first and then do RL… See where it's going?

InstructGPT:
What Is RLHF Right Now

Toloka

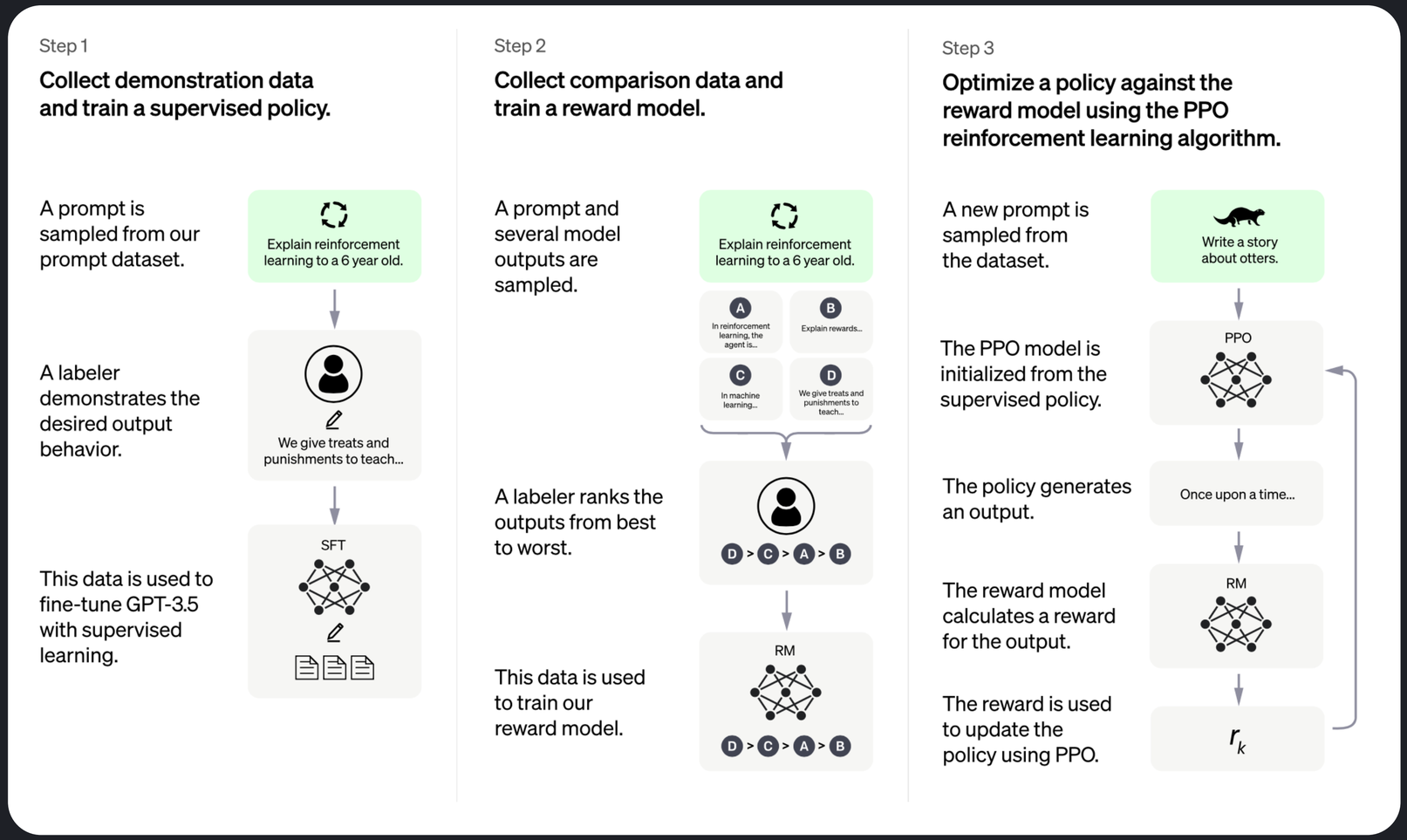
Large Language Models

- Language models take an input sequence and predict the next token (word). That’s it
- They are trained on a large text corpus, usually row web crawl
- They are capable of few-zero-shot learning
- Training to predict the next token based on all the internet is different from the desired objective: follow human’s instructions carefully and safely



InstructGPT

— InstructGPT applies RLHF to make GPT-3 follow human instructions



Why Human Feedback Is Important?

A very natural question is:

— Why can’t we just fine-tune the model on good examples?

There are several reasons why:

1. It’s very hard to collect enough data since there should be a lot of examples written by actual humans. So, all the data the model sees is written by humans → extremely expensive and hard

2. The model only sees good examples and doesn’t know that bad are bad, so nothing stops it to generate something irrelevant from its memory

Why Supervised Pretraining Is Important

- RL suffers from a low start: in the beginning, reward does not increase for a long time and stays on plateau
- Longer it stays on plateau, more compute resources we will spend
- It can stay there for a really long time
- When the model generates outputs completely different from the desired ones, it’s difficult to collect feedback and train the reward model because both examples are completely irrelevant 99% of the time

RLHF Applications

— Generative models are trained on a raw unlabeled data

— They produce good results but they are often different from what the people want

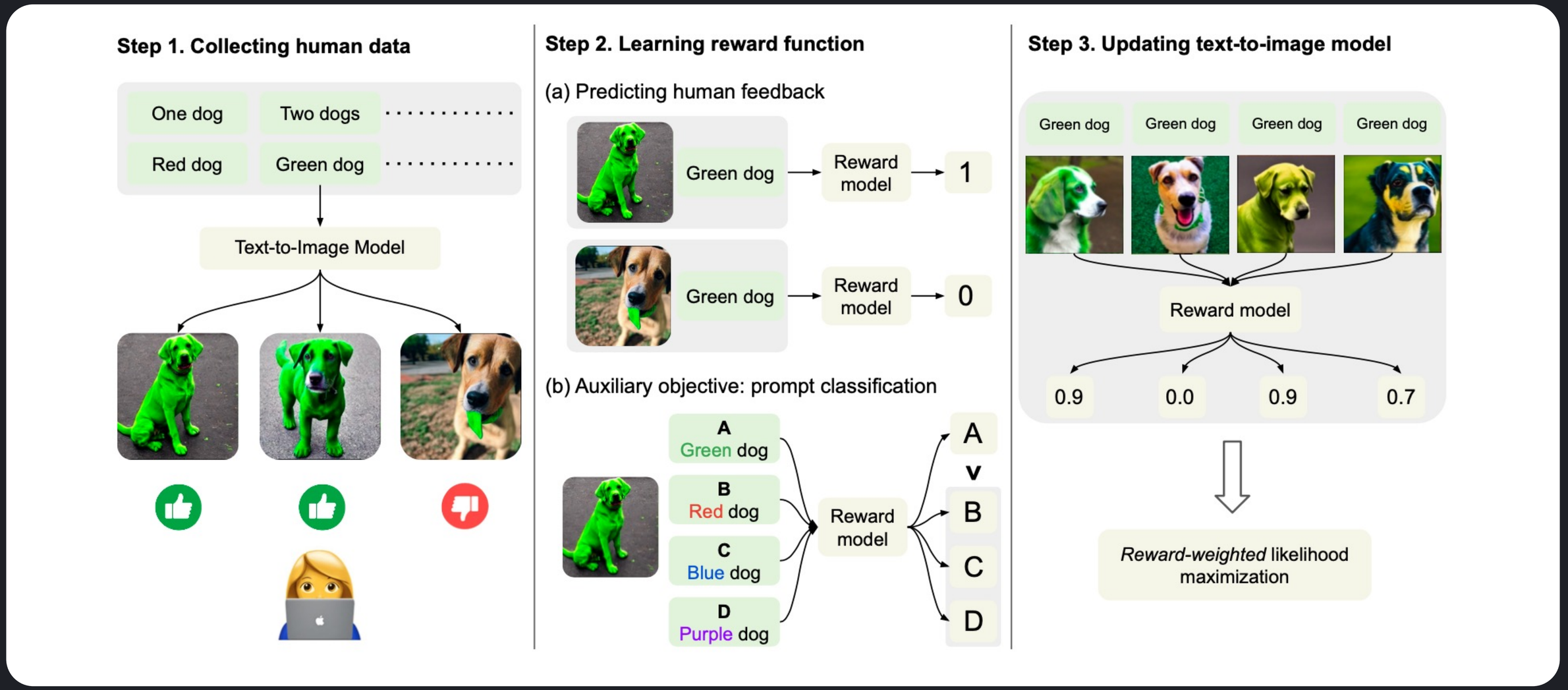
— RLHF can teach a generative model to do what people want

Examples:

— Diffusion that generates aesthetically pleasant images

— TTS that uses a desired intonation, etc

Instruct Stable Diffusion



RLHF Challenges

Most challenges lay in the data collection pipeline:

- Data collection and quality control is difficult
- What makes InstructGPT (ChatGPT) good is creative tasks and expert annotation that are difficult

Questions