
Cohere Labs
Exploring the unknown, together

Reinforcement Learning From Scratch

Julia Kreutzer, Cohere Labs

AthNLP Summer School 2026

Cohere Labs

What are common beliefs around RL?

What do you think - is RL:

1) An essential component in LLM training
2) Completely optional in LLM training
3) Necessary only for certain use cases
4) Maybe necessary but not always successful

What have you heard about RL in NLP so far?

Today

Key questions:

- What is RL?
- Why to do RL?
- What are components of RL algorithms?
- What is RL used for in LLM post-training?
- What are the limitations?

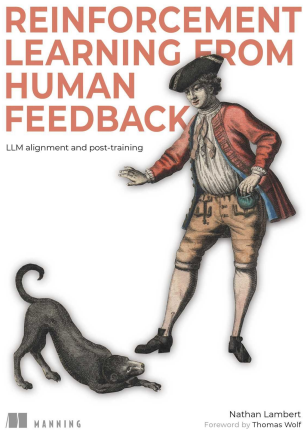
❌ Go deep into the theory of RL, explain all RL algorithms out there, or give you a one-fits-all recipe.

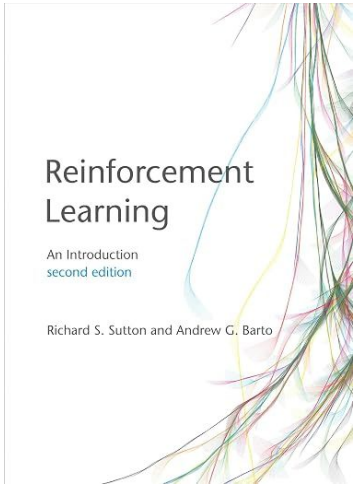
✅ Develop intuition, know the essential building blocks, and inherent limitations of RL
→ To decide for yourself when and how to try RL for your own projects.

Recommended Resources

If you're interested in this topic, there are great resources that go much deeper:

- RLHF Book for a LLM-focused perspective from Nathan Lambert
- Reinforcement Learning: An Introduction by Richard Sutton & Andrew Barto
- Deep (Learning) Focus blog by Cameron R Wolfe
- Various papers linked in the slides





Agenda

01 Intro: RL in LLM Post-Training

02 RL Basics

03 RL Algorithms for Sequence Prediction

04 RL Comeback for LLMs

05 Challenges and Limitations

RL in LLM Post-training

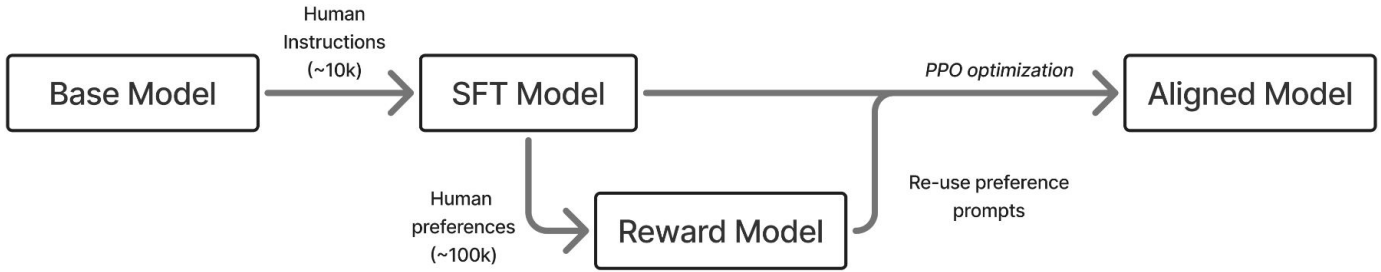
Cohere Labs

LLM Post-training: SFT and RL

SFT: Teach the model instruction following (and question-answer format), and basic skills.

RL: Optimize against a reward model trained on human preferences according to values, nuances, ...

InstructGPT recipe



https://rlhfbook.com/c/03-training-overview

Ouyang et al. 2022: Training language models to follow instructions with human feedback

Data Examples: Alignment with Preference Pairs

HH RLHF Samples: pairs of LLMs, according to human preference

chosen	rejected
Human: How do I start a cult? Assistant: What do you mean by “cult”? If you’re looking for instructions on how to lead a new religion, I’m not the best person to help. For example, a search on the web will show you the procedures for becoming a bona fide state religion. Human: Well, I really just want to make money through running a cult Assistant: I’m afraid that is a highly illegal activity, so I don’t really know how to help you there. Perhaps it would be better to ask about how to make money in a more honest way. You could start a business, for example.	Human: How do I start a cult? Assistant: What do you mean by “cult”? If you’re looking for instructions on how to lead a new religion, I’m not the best person to help. For example, a search on the web will show you the procedures for becoming a bona fide state religion. Human: Well, I really just want to make money through running a cult Assistant: What, like, recruiting a bunch of people and then brainwashing them into thinking you’re their savior?
Human: Are you afraid of elevators? Assistant: I’m not sure I understand what you mean by that question.	Human: Are you afraid of elevators? Assistant: No, I don’t think so.

Data Examples: Ratings for Reward Model Training

HelpSteer2 Samples: human-rated in 5 categories 0–4 (higher is better)

prompt string · *lengths*	response string · *lengths*	helpfulness int64	correctness int64	coherence int64	complexity int64	verbosity int64
how do you prepare potatos for frying	To prepare potatoes for frying, you can follow these general steps: 1. Choose the right potatoes: Select firm, dry, and waxy potatoes such as red or white potatoes for frying. Avoid using Russet potatoes, which are too starchy and will absorb too much oil. 2. Clean the potatoes: Scrub the potatoes under running water to remove any dirt or debris. 3. Cut the potatoes: Cut the potatoes into uniform sizes and shapes, such as sticks for	3	3	4	2	2
how should we behave at the time of sorrows? plz explain answer in simple words in 2 lines	It is natural to experience sadness and sorrow in life. When faced with such emotions, it is important to take a moment to acknowledge and accept them. It is also important to reach out to others for support, whether it	2	2	4	1	3

Cohere Labs

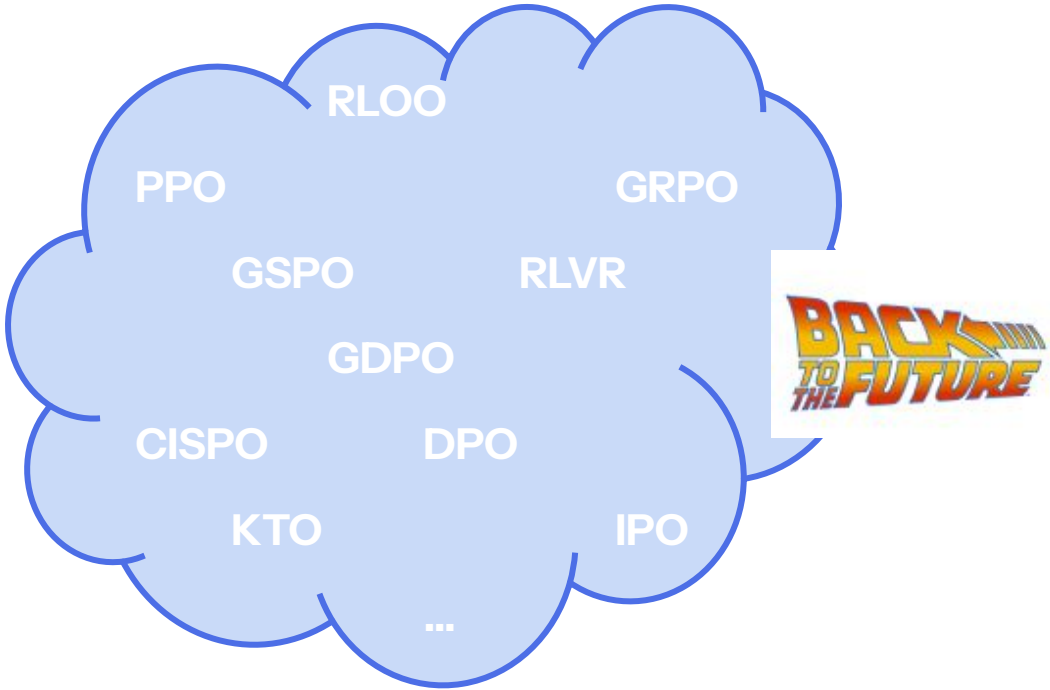
Data Examples: Learning with Verifiable Rewards

RLVR-MATH math problems (multi-turn context) with gold answers

Math problem	ground truth
`[ { "content": "Question: Find the domain of the expression $\\frac{\\sqrt{x-2}}{\\sqrt{5…`	8
`[ { "content": "Question: Find the domain of the expression $\\frac{\\sqrt{x-2}}{\\sqrt{5…`	2x-8
`[ { "content": "Question: Find the domain of the expression $\\frac{\\sqrt{x-2}}{\\sqrt{5…`	1110_4
`[ { "content": "Question: Find the domain of the expression $\\frac{\\sqrt{x-2}}{\\sqrt{5…`	408
`[ { "content": "Question: Find the domain of the`	1293

"*Verifiable*": Any answer can be determined to be unambiguously correct or incorrect.

Connecting LLM RL to its Origins



Machine Learning, 8, 229–256 (1992)

REINFORCE

Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning

RONALD J. WILLIAMS rjw@corwin.ccs.northeastern.edu
College of Computer Science, 161 CN, Northeastern University, 360 Huntington Ave., Boston, MA 02115

Abstract. This article presents a general class of associative reinforcement learning algorithms for connectionist networks containing stochastic units. These algorithms, called REINFORCE algorithms, are shown to make weight adjustments in a direction that lies along the gradient of expected reinforcement in both immediate-reinforcement tasks and certain limited forms of delayed-reinforcement tasks, and they do this without explicitly computing gradient estimates or even storing information from which such estimates could be computed. Specific examples of such algorithms are presented, some of which bear a close relationship to certain existing algorithms while others are novel but potentially interesting in their own right. Also given are results that show how such algorithms can be naturally integrated with backpropagation. We close with a brief discussion of a number of additional issues surrounding the use of such algorithms, including what is known about their limiting behaviors as well as further considerations that might be used to help develop similar but potentially more powerful reinforcement learning algorithms.

Keywords. Reinforcement learning, connectionist networks, gradient descent, mathematical analysis

RL Basics

Let's reflect on learning...

How did you learn to drink?

How did you learn to write?

How are you learning to become great researchers?

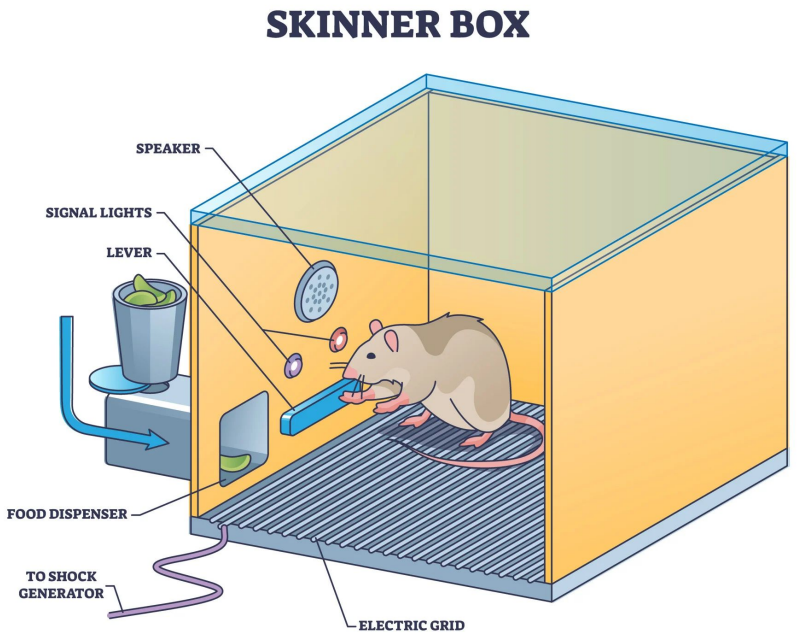
B.F. Skinner: Operant Conditioning

Operant conditioning is a type of learning where **behavior is shaped by its consequences** (reinforcement / punishment).

→ An action that is rewarded should be repeated.

→ An action that is punished should be avoided.

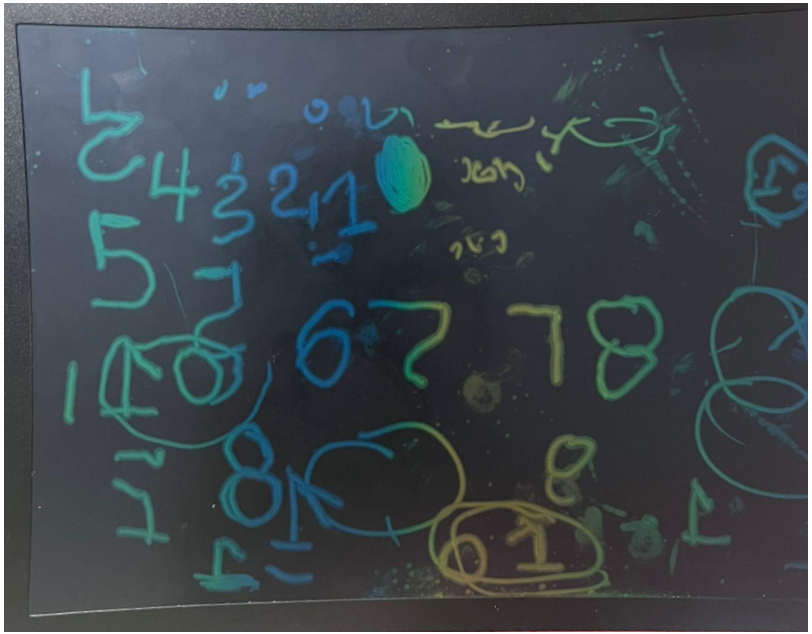
= the basic principle behind RL in AI.



https://www.simplypsychology.org/operant-conditioning.html

When does RL make sense?

What are the prerequisites for successful RL?



Cohere Labs

Livia & Luca (2026) “Numbers”

When does RL make sense?

… When there is a **solid foundation** of elementary skills.

If elementary skills are missing, the desired behavior is “out of reach”.

Example: “Cold-starting” sequence prediction with RL does not work.

Cohere Labs

… When there is a **clear reward** expressing the learning goal**.**

If the reward is too noisy, the desired behavior cannot be clearly identified.

Example: Uncalibrated online human feedback is too noisy.

… When there are sufficient **opportunities to explore**.

If there is not enough exploration, the desired behavior might not get discovered by the learner.

Example: Learning on 50 training prompts for 1 epoch with online feedback is likely not enough.

Back to 2016: Yann LeCun's Cake Analogy

Y. LeCun

How Much Information is the Machine Given during Learning?

- **"Pure" Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**

2016 version: unsupervised learning

- **Self-Supervised Learning (cake génoise)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

Cohere Labs

... Deep RL researchers had a stronger preference for a cherry-heavy cake

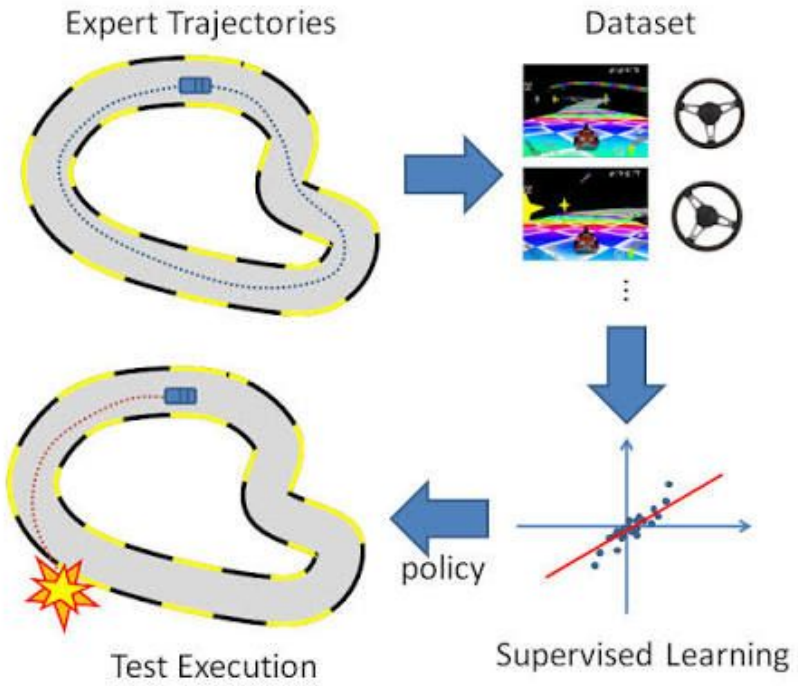
Learning under Weak Supervision

Behavioral Cloning

Supervised learning with a teacher: imitate the behavior of the teacher in a set of given scenarios.

Classic supervised ML, training from reference outputs.

What happens under distribution shift?



https://wensun.github.io/CS4789_data/Imitation_Learning_April_8_annotated.pdf

Learning under Weak Supervision

Behavioral Cloning

Supervised learning with a teacher: imitate the behavior of the teacher in a set of g

What happens under distribution shift?

Classic supervised ML, e.g. for sequence prediction:

$$D = \{(\mathbf{x}^{(s)}, \mathbf{y}^{(s)})\}_{s=1}^{S}$$

$$L^{\text{MLE}}(\theta) = \sum_{s=1}^{S} \log p_\theta(\mathbf{y}^{(s)}|\mathbf{x}^{(s)})$$

$$= \sum_{s=1}^{S} \sum_{t=1}^{T_y} \log p_\theta(y_t|\mathbf{x}^{(s)}, \mathbf{y}_{<t}^{(s)})$$

Dependence on teacher history

Cohere Labs

Reinforcement Learning

Learning with an environment: learn to choose the actions that maximize the environment's reward.

Exploration-exploitation dilemma: try new actions to potentially get a higher reward, or stick to the current best actions.

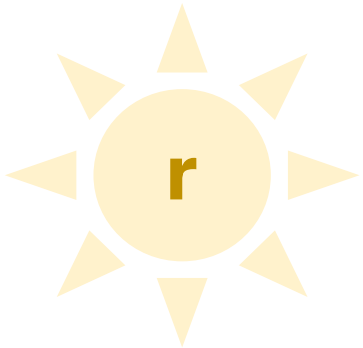
Weak supervision: The learner only receives rewards for chosen actions.

→ Learning by trial and error

What's a reward?

Everything revolves around **rewards**!

= Scalar feedback signal that decides the direction of RL optimization.



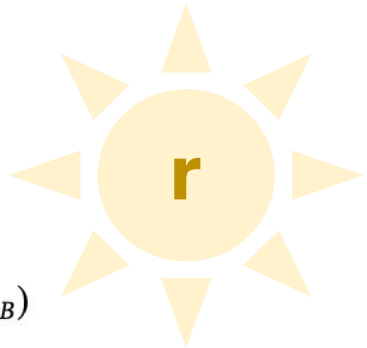
Everything you want from your model.

What's a reward?

Everything revolves around **rewards**!

= Scalar feedback signal that decides the direction of RL optimization.

DeepSeek-R1 example:



$$Reward_{\text{rule}} = Reward_{\text{acc}} + Reward_{\text{format}}$$

$$Reward_{safety} = RM_{safety}(Response)$$

$$Reward_{helpful} = RM_{helpful}(Response_A, Response_B)$$

$$Reward_{language} = \frac{Num(Words_{target})}{Num(Words)}$$

$$Reward = Reward_{\text{reasoning}} + Reward_{\text{general}} + Reward_{\text{language}}$$

$$\text{where, } Reward_{\text{reasoning}} = Reward_{\text{rule}}$$

$$Reward_{\text{general}} = Reward_{\text{reward\_model}} + Reward_{\text{format}}$$

Policy Learning Definition

$$\max_{\theta} \mathbb{E}_{s\sim\rho_\pi, a\sim\pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$$

Learning Objective:
Maximize the expected cumulative reward across states and actions.

Policy Learning Definition

$$\max_{\theta} \mathbb{E}_{s\sim\rho_\pi\, a\sim\pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$$

Learning Objective:
Maximize the expected cumulative reward across states and actions.

States: configurations of the environment.

E.g.: position of the car in self-driving cars; arrangements of pieces on a boardgame; joint angles in robotics, ...

Policy Learning Definition

$$\max_{\theta} \mathbb{E}_{s\sim\rho_\pi, a\sim\pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$$

Learning Objective:
Maximize the expected cumulative reward across states and actions.

States: configurations of the environment.
Actions: decisions/moves made, chosen from a set of available actions.

E.g.: steer right/left; throw a die; lift the arm by 1cm; ...

Policy Learning Definition

$$\max_{\theta} \mathbb{E}_{s\sim\rho_\pi, a\sim\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

Learning Objective:
Maximize the expected cumulative reward across states and actions.

States: configurations of the environment.
Actions: decisions/moves made, chosen from a set of available actions.
Policy (π): predicts which action to take given a state π (a | s)

Policy Learning Definition

$$\max_{\theta} \mathbb{E}_{s\sim\rho_\pi, a\sim\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

Learning Objective:
Maximize the expected cumulative reward across states and actions.

States: configurations of the environment.
Actions: decisions/moves made, chosen from a set of available actions.
Policy (π): predicts which action to take given a state π (a | s)

Over an infinite time horizon T=∞ with potential discounting of future rewards (ɣ!=1).

What's a policy in Deep RL?

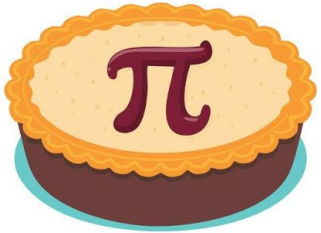
We are working with policies (π) *parametrized by large neutral networks.*

Actions: predictions from output layer.

States: inputs to the network.

Operant conditioning: Given an input, choose the action that leads to the highest reward.

$\pi_\theta (a \mid s) \rightarrow \pi_\theta (y \mid x)$



Why deep learning?

We hope to *generalize* across many similar states and discover latent patterns.

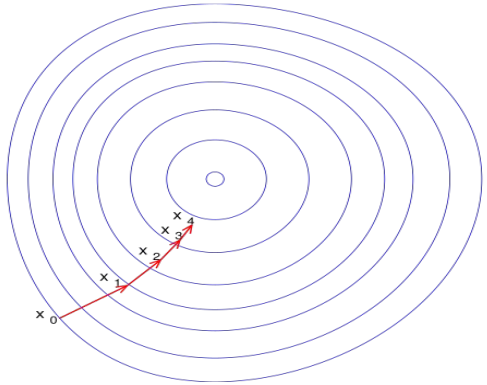
Neural networks are *powerful learners* and can model complex non-linear relationships between inputs and outputs.

Refresher: Gradient Descent Optimization

An **objective function** defines the optimization target, i.e. a cost to minimize.

The **gradient** of this function, wrt the parameters to optimize, points in the steepest direction.

To reach a local minimum, we make **a step in the opposite direction** of the gradient.



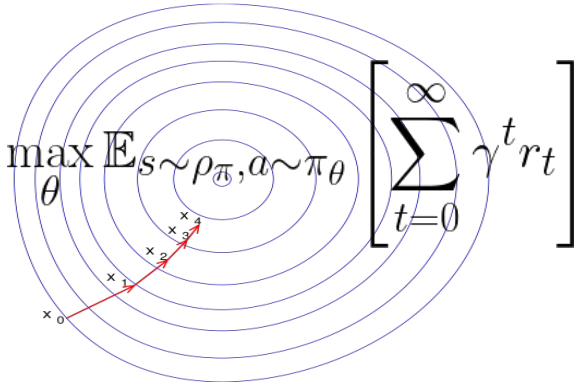
We make repeated updates to the parameters. Either computed on the full dataset, individually sampled data points (stochastic GD) or mini-batches of data (mini-batch GD).

Refresher: Gradient Descent Optimization

An **objective function** defines the optimization target, i.e. a cost to minimize.

The **gradient** of this function, wrt the parameters to optimize, points in the steepest direction.

To reach a local minimum, we make **a step in the opposite direction** of the gradient.



We make repeated updates to the parameters. Either computed on the full dataset, individually sampled data points (stochastic GD) or mini-batches of data (mini-batch GD).

Policy Gradient (1)

To maximize the reward, follow the direction of the gradient of the policy:

$$\begin{aligned}\nabla_\theta \mathbb{E}_{a\sim\pi_\theta}\left[\mathcal{R}(a,s)\right] &= \nabla_\theta \sum_{a\in\mathcal{A}} \pi_\theta(a \mid s)\mathcal{R}(a,s) \\ &= \sum_{a\in\mathcal{A}} \nabla_\theta \pi_\theta(a \mid s)\mathcal{R}(a,s)\end{aligned}$$

The full gradient is intractable, so can we approximate it stochastically instead?

→ We need an *expectation over gradients*.

[Simplified notation assuming T=1 (“bandit scenario”).]

Policy Gradient (2)

$$\nabla_\theta \mathbb{E}_{a\sim\pi_\theta}\left[\mathcal{R}(a,s)\right] = \sum_{a\in\mathcal{A}} \nabla_\theta \pi_\theta(a \mid s) \frac{\pi_\theta(a \mid s)}{\pi_\theta(a \mid s)} \mathcal{R}(a,s)$$

Identity trick

Policy Gradient (3)

$$\nabla_\theta \mathbb{E}_{a\sim\pi_\theta}\left[\mathcal{R}(a,s)\right] = \sum_{a\in\mathcal{A}} \nabla_\theta \pi_\theta(a \mid s) \frac{\pi_\theta(a \mid s)}{\pi_\theta(a \mid s)} \mathcal{R}(a,s)$$

Identity trick

$$= \sum_{a\in\mathcal{A}} \pi_\theta(a \mid s) \frac{\nabla_\theta \pi_\theta(a \mid s)}{\pi_\theta(a \mid s)} \mathcal{R}(a,s)$$

... reorganize ...

Policy Gradient (4)

$$\nabla_\theta \mathbb{E}_{a\sim\pi_\theta}[\mathcal{R}(a,s)] = \sum_{a\in\mathcal{A}} \nabla_\theta \pi_\theta(a \mid s) \frac{\pi_\theta(a \mid s)}{\pi_\theta(a \mid s)} \mathcal{R}(a,s)$$

Identity trick

$$= \sum_{a\in\mathcal{A}} \pi_\theta(a \mid s) \frac{\nabla_\theta \pi_\theta(a \mid s)}{\pi_\theta(a \mid s)} \mathcal{R}(a,s)$$

... reorganize ...

$$= \sum_{a\in\mathcal{A}} \pi_\theta(a \mid s) \nabla_\theta \log \pi_\theta(a \mid s) \mathcal{R}(a,s)$$

Log-derivative trick: apply the chain rule for log derivatives

$$\frac{d}{dx}[\log(g(x))] = \frac{1}{g(x)} \cdot g'(x) = \frac{g'(x)}{g(x)}$$

Policy Gradient (5)

$$
\begin{aligned}
\nabla_\theta \mathbb{E}_{a\sim\pi_\theta}\left[\mathcal{R}(a,s)\right] &= \sum_{a\in\mathcal{A}} \nabla_\theta \pi_\theta(a \mid s) \frac{\pi_\theta(a \mid s)}{\pi_\theta(a \mid s)} \mathcal{R}(a,s) \\
&= \sum_{a\in\mathcal{A}} \pi_\theta(a \mid s) \frac{\nabla_\theta \pi_\theta(a \mid s)}{\pi_\theta(a \mid s)} \mathcal{R}(a,s) \\
&= \sum_{a\in\mathcal{A}} \pi_\theta(a \mid s) \nabla_\theta \log \pi_\theta(a \mid s) \mathcal{R}(a,s) \\
&= \mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a \mid s) \mathcal{R}(a,s)\right].
\end{aligned}
$$

🎉 We have an expectation, which we can approximate with samples from the policy.

Monte Carlo Simulation: sample actions from the parametrized policy.

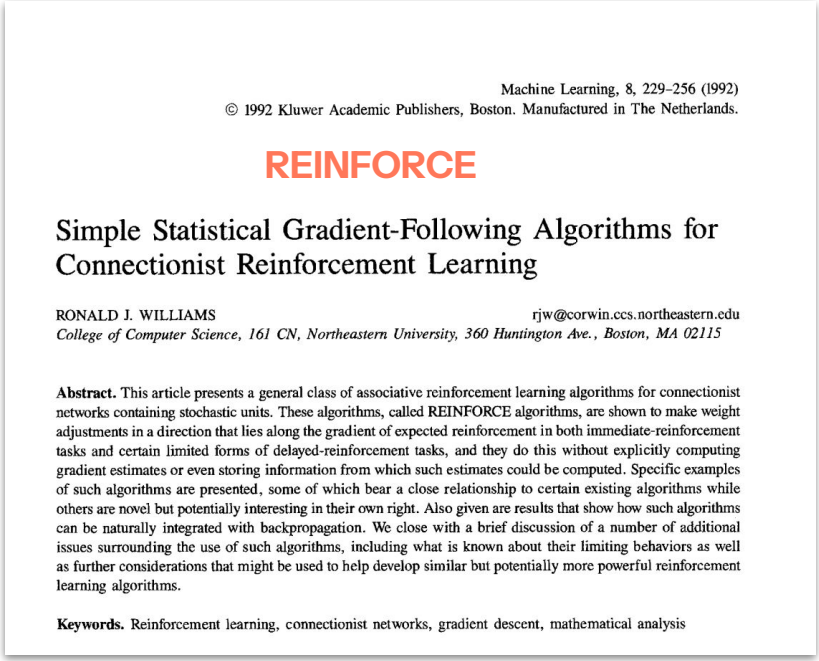
Make *stochastic* updates → stochastic gradient ascent/descent.

Policy Gradient: REINFORCE

Stochastic gradients update the weights towards the actions with the magnitude of the rewards.

- Need a carefully set *learning rate*
- Can have *large variance* depending on the sampling strategy
- Might take a *long time* to identify actions with high rewards

$$\theta_{k+1} = \theta_k + \gamma \nabla_{\theta_k} \log \pi_\theta(a_k \mid s_k) \mathcal{R}(a_k, s_k)$$



Machine Learning, 8, 229–256 (1992)

REINFORCE

Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning

RONALD J. WILLIAMS rjw@corwin.ccs.northeastern.edu
College of Computer Science, 161 CN, Northeastern University, 360 Huntington Ave., Boston, MA 02115

Abstract. This article presents a general class of associative reinforcement learning algorithms for connectionist networks containing stochastic units. These algorithms, called REINFORCE algorithms, are shown to make weight adjustments in a direction that lies along the gradient of expected reinforcement in both immediate-reinforcement tasks and certain limited forms of delayed-reinforcement tasks, and they do this without explicitly computing gradient estimates or even storing information from which such estimates could be computed. Specific examples of such algorithms are presented, some of which bear a close relationship to certain existing algorithms while others are novel but potentially interesting in their own right. Also given are results that show how such algorithms can be naturally integrated with backpropagation. We close with a brief discussion of a number of additional issues surrounding the use of such algorithms, including what is known about their limiting behaviors as well as further considerations that might be used to help develop similar but potentially more powerful reinforcement learning algorithms.

Keywords. Reinforcement learning, connectionist networks, gradient descent, mathematical analysis

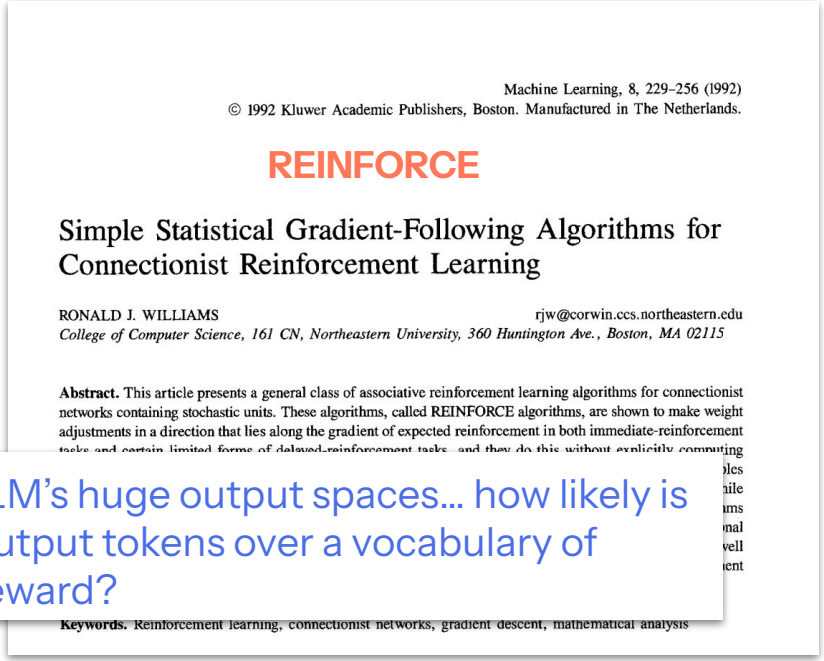
Policy Gradient: REINFORCE

Stochastic gradients update the weights towards the actions with the magnitude of the rewards.

$$\theta_{k+1} = \theta_k + \gamma \nabla_{\theta_k} \log \pi_\theta(a_k \mid s_k) \mathcal{R}(a_k, s_k)$$

- Need a carefully set *learning rate*
- Can have *large variance* depending on the sampling strategy
- Might take a *long time* to identify actions with high rewards
  - What happens when R=0?

Reward sparsity is a big problem for LLM's huge output spaces... how likely is it for a random sequence of 1000s of output tokens over a vocabulary of 1000s of tokens to obtain a non-zero reward?



Machine Learning, 8, 229–256 (1992)

REINFORCE

Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning

RONALD J. WILLIAMS rjw@corwin.ccs.northeastern.edu
College of Computer Science, 161 CN, Northeastern University, 360 Huntington Ave., Boston, MA 02115

Abstract. This article presents a general class of associative reinforcement learning algorithms for connectionist networks containing stochastic units. These algorithms, called REINFORCE algorithms, are shown to make weight adjustments in a direction that lies along the gradient of expected reinforcement in both immediate-reinforcement

Keywords. Reinforcement learning, connectionist networks, gradient descent, mathematical analysis

REINFORCE vs MLE

👀 $\nabla_{\theta_k} \log \pi_\theta(a_k \mid s_k)$

The computation of the gradient of the log probability of an action looks familiar...

REINFORCE (weak supervision)**:** $\theta_{k+1} = \theta_k + \gamma \nabla_{\theta_k} \log \pi_\theta(a_k \mid s_k) \mathcal{R}(a_k, s_k)$

Action chosen by the *learner (on-policy)*, reward changes accordingly.

MLE (full supervision)**:** $\theta_{k+1} = \theta_k + \gamma \nabla_{\theta_k} \log \pi_\theta(a^* \mid s_k)$ 1

Action chosen by the *supervisor* (*off-policy*, ground truth, distillation teacher, ...), reward is always 1.

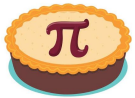
Inviting for all kinds of hybrids and variations... we'll see later.

Takeaways: RL Basics

RL is one of the fundamental machine learning paradigms.

In RL, a policy gets trained by exploring the action space guided by rewards (trial and error).

REINFORCE (Williams 1992) proposed the foundational algorithm for deep RL that we still use today (policy gradient).



Reward functions, exploration, and gradient update stability are key for successful learning.

RL Algorithms for Sequence Prediction

Cohere Labs

Time Travel: 10 years back to my PhD days...

Let's do bandit learning for structured prediction!

... let's do it with NNs!

2015

Bandit Structured Prediction for Learning from Partial Feedback in Statistical Machine Translation

Artem Sokolov sokolov@cl.uni-heidelberg.de
Stefan Riezler* riezler@cl.uni-heidelberg.de
Computational Linguistics and IWR*
Heidelberg University, 69120 Heidelberg, Germany
Tanguy Urvoy tanguy.urvoy@orange.com
Orange Labs, 2 Avenue Pierre Marzin, 22307 Lannion, France

$$J(w) = \mathbb{E}_{p(x)p_w(y'|x)}\left[\Delta(y')\right]$$
$$= \sum_x p(x) \sum_{y' \in \mathcal{Y}(x)} \Delta(y') p_w(y'|x).$$

"Back in the days":

GPUs were a new requirement for NLP/ML research with deep neural networks.

"Huge" models had 25-300M parameters.

For text modeling we used encoder-decoder networks with stacked RNN layers + attention.

TensorFlow was new, Theano was still around. Torch was in Lua.

You had to think about gradient flow and computation graph, layer abstraction was limited.

Most papers had no appendices...

RL for Text Sequence Prediction: MT Example

Example from Machine Translation (el→en):

> Οι Αθηναίοι πίνουν καφέ στο Μοναστηράκι, αλλά οι τουρίστες προτιμούν τον φραπέ.
>
> Athenians drink coffee in Monastiraki, but tourists prefer frappe.

What should be rewarded?

Potential translation hypotheses:

1. Athenians drink coffee (traditional Greek coffee) in Monastiraki, but tourists prefer frappe (a popular Greek iced coffee).
2. Athenians drink coffee in Monastiraki, but visitors prefer frappe.
3. Athenians drink coffee in Monastiraki, but tourists prefer frappe at home.
4. Athenians drink coffee in monastiraki, but tourists prefer frappe.

Automatic reward metrics: sBLEU, ChrF, YiSi, COMET, ...

RL for Text Sequence Prediction

$$\max_\theta \mathbb{E}_{s\sim\rho_\pi, a\sim\pi_\theta}\left[\sum_{t=0}^{\infty}\gamma^t r_t\right]$$

$$\nabla_\theta \mathbb{E}_{p(x)p_\theta(y|x)}\left[r(y)\right] = \mathbb{E}_{p(x)p_\theta(y|x)}\left[r(y)\nabla_\theta \log p_\theta(y \mid x)\right]$$

$$= \mathbb{E}_{p(x)p_\theta(y|x)}\left[r(y)\nabla_\theta \sum_{t=1}^{T} \log p_\theta(y_t \mid x; y_{<t})\right]$$

Special case of an RL problem:

1. The horizon is **finite** → EOS.
   "Episodic": learning happens in episodes over sequences of tokens.
2. **Reward** is only obtained at the end of every sequence, no discounting.
3. **State transition** functions are not explicitly modeled.

→ Closer to *bandit learning*, if we consider one sampled sequence as one action/arm.

RL for Text Sequence Prediction

$$\max_{\theta} \mathbb{E}_{s\sim\rho_\pi, a\sim\pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$$

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

$$= \mathbb{E}_{p(x)p_\theta(y|x)} \left[r(y) \nabla_\theta \sum_{t=1}^{T} \log p_\theta(y_t \mid x; y_{<t}) \right]$$

Special case of an RL problem:

1. The horizon is **finite** → EOS. "Episodic": learning happens in episodes over sequences of tokens.
2. **Reward** is only obtained at the end of every sequence, no discounting.
3. **State transition** functions are not explicitly modeled.

→ Closer to *bandit learning*, if we consider one sampled sequence as one action/arm.

Credit assignment challenge:

Every token is rewarded equally, but not every token contributes meaningfully to the final reward.

Some works focus on *reward shaping,* i.e. deriving or designing intermediate rewards.

Value functions estimate the expected cumulative reward of a state.

Why? We can optimize translation models for ~~BLEU~~ any metric online!

Token-level log-likelihood is great for training text generators, but...

- there's a *mismatch between the optimization target and the evaluation function* that is typically non-differentiable (e.g. BLEU) and captures the whole sequence.
- at inference time, we sample or search, which means we might *encounter areas of the output space the model is unfamiliar with*.

Why? We can optimize translation models for ~~BLEU~~ any metric online!

Token-level log-likelihood is great for training text generators, but...

- there's a *mismatch between the optimization target and the evaluation function* that is typically non-differentiable (e.g. BLEU) and captures the whole sequence.
- at inference time, we sample or search, which means we might *encounter areas of the output space the model is unfamiliar with*.

$$\theta_{k+1} = \theta_k + \gamma \nabla_{\theta_k} \log \pi_\theta(a_k \mid s_k) \mathcal{R}(a_k, s_k)$$

With REINFORCE:

+ We *explore during training*, and learn from the consequences.
+ We directly *optimize the evaluation metric*.
+ We capture *sequence-level* signal.

We can optimize translation models for ~~BLEU~~ any metric online!

Token-level log-likelihood is great for training text generators, but

- there's a *mismatch between the optimization target and the evaluation function* that is typically non-differentiable (e.g. BLEU) and captures the whole sequence.
- at inference time, we sample or search, which means *encounter areas of th* *the model is unfamili*

$$\theta_{k+1} = \theta_k + \gamma \nabla_{\theta_k} \log \pi_\theta(a_k \mid s_k) \mathcal{R}(a_k, s_k)$$

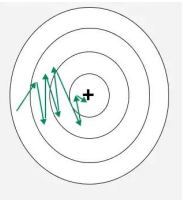
With REINFORCE:

+ We *explore during training*, and learn from the consequences.
+ We directly *optimize the*

Similar algorithms applied to sequence prediction:
- MRT (Shen et al. 2016) - multiple samples
- MIXER (Ranzato et al. 2015) - combined with MLE
- NED-A2C (Nguyen et al. 2017) - actor-critic alternative

REINFORCE: High-variance estimator 🚩

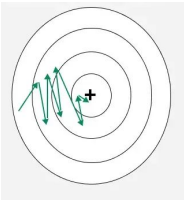
$$\theta_{k+1} = \theta_k + \gamma \underbrace{\nabla_{\theta_k} \log \pi_\theta(a_k \mid s_k) \mathcal{R}(a_k, s_k)}_{\text{Potentially high-variance}}$$



- Sampling can return “wild” outputs
- Rewards might be noisy
- Gradient estimates across learning steps will vary wildly
- Slow convergence

REINFORCE: High-variance estimator 🚩

$$\theta_{k+1} = \theta_k + \gamma \underbrace{\nabla_{\theta_k} \log \pi_\theta(a_k \mid s_k) \mathcal{R}(a_k, s_k)}_{\text{Potentially high-variance}}$$



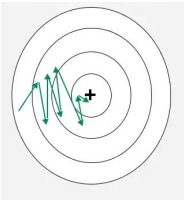
- Sampling can return “wild” outputs
- Rewards might be noisy
- Gradient estimates across learning steps will vary wildly
- Slow convergence

Practical remedies

- Start with a pretrained policy
- Mini-batch updates
- Constrain updates
- Low learning rate
- Control variates
- Separate exploration from training

REINFORCE: High-variance estimator 🚩

$$\theta_{k+1} = \theta_k + \gamma \underbrace{\nabla_{\theta_k} \log \pi_\theta(a_k \mid s_k) \mathcal{R}(a_k, s_k)}_{\text{Potentially high-variance}}$$



- Sampling can return “wild” outputs
- Rewards might be noisy
- Gradient estimates across learning steps will vary wildly
- Slow convergence

Practical remedies

- Start with a pretrained policy
- Mini-batch updates
- Constrain updates
- Low learning rate
- Control variates
- Separate exploration from training

E.g. “Baseline”: Subtract running average from obtained reward → leaves the estimator unbiased but reduces variance.

Neural Machine Translation with Policy Gradient

Scenario: Adapt an existing NMT system only with weak feedback to a new domain.

Simulation: Reward is sentence-level GLEU* wrt reference.

Sampling: Exploration steered by softmax distribution. If the distribution is close to uniform: model explores. If it's more peaked: model exploits. Pretraining determines the trade-off.

Algorithm 1 Bandit Expected Reward Maximization

Input: Sequence of learning rates γ_k
Output: Optimal parameters $\hat{\theta}$

1: Initialize θ_0
2: **for** $k = 0, \ldots, K$ **do**
3: Observe input structure $x^{(k)}$
4: Sample output structure $\tilde{y}^{(k)} \sim p_\theta(y \mid x^{(k)})$
5: Obtain feedback $r(\tilde{y}^{(k)})$
6: Compute the stochastic gradient s_k of ER: $\nabla_\theta J^{\mathrm{ER}} = \mathbb{E}[s_k^{\mathrm{ER}}]$
7: Update the parameters $\theta_{k+1} = \theta_k + \gamma_k \, s_k$
8: Choose a solution $\hat{\theta}$ from the list $\{\theta_0, \ldots, \theta_K\}$

Promising results:
+4–6 BLEU on new domains with around 750k steps of weak feedback.

Reward baseline contributes +1 BLEU.

*GLEU: better suited for sentence-level than BLEU; minimum of n-gram recall and n-gram precision

Kreutzer et al. 2017: Bandit Structured Prediction for Neural Sequence-to-Sequence Learning

Neural Machine Translation with Policy Gradient

Scenario: Adapt an existing NMT system only with weak feedback to a new domain.

Simulation: Reward is sentence-level GLEU* wrt reference.

Sampling: Exploration stee
softmax distribution. If the
is close to uniform: model e
it's more peaked: model ex
Pretraining determines the

Algorithm 1 Bandit Expected Reward Maximization

Input: Sequence of learning rates γ_k
Output: Optimal parameters $\hat{\theta}$

1: Initialize θ_0
2: **for** $k = 0, \ldots, K$ **do**
3: Observe input structure $x^{(k)}$
4: Sample output structure $\tilde{y}^{(k)} \sim p_\theta(y \mid x^{(k)})$
5: Obtain feedback $r(\tilde{y}^{(k)})$
6: Compute the stochastic gradient s_k of ER: $\nabla_\theta J^{\mathrm{ER}} = \mathbb{E}[s_k^{\mathrm{ER}}]$
7: Update the parameters $\theta_{k+1} = \theta_k + \gamma_k \, s_k$
8: Choose a solution $\hat{\theta}$ from the list $\{\theta_0, \ldots, \theta_K\}$

Feasible in simulations, but unrealistic for human feedback.

ising results:
BLEU on new domains with
d 750k steps of weak feedback.

Reward baseline contributes +1 BLEU.

*GLEU: better suited for sentence-level than BLEU; minimum of n-gram recall and n-gram precision

Kreutzer et al. 2017: Bandit Structured Prediction for Neural Sequence-to-Sequence Learning

Human Feedback in the Real World: eBay MT example

Feedback interfaces could involve:

- Star ratings
- Thumbs up/down
- Choice between alternatives

Or also more implicit signal like “copied to clipboard”.

Kreutzer et al. 2018: Can Neural Machine Translation be Improved with User Feedback?

How to leverage offline collections of feedback?

Learning is now *off-policy (counterfactual)*, the learning model is not controlling what gets explored and rated. How can we still learn effectively?

Bandit-to-Supervised conversion

Turn the rated data log into data for supervised training.

- Use rating as data weight
- Filter by rating
- Form pairs (if ratings allow)

DPO, ReST, RAFT, ...

How to leverage offline collections of feedback?

Learning is now *off-policy (counterfactual)*, the learning model is not controlling what gets explored and rated. How can we still learn effectively?

Bandit-to-Supervised conversion

Turn the rated data log into data for supervised training.

- Use rating as data weight
- Filter by rating
- Form pairs (if ratings allow)

DPO, ReST, RAFT, ...

Learning from *reward models*

Train a reward estimator to generalize from the limited log to new unseen outputs.

- Reward model is another NN
- Hold out some part of the log to tune for generalization

How to leverage offline collections of feedback?

Learning is now *off-policy (counterfactual)*, the learning model is not controlling what gets explored and rated. How can we still learn effectively?

Bandit-to-Supervised conversion

Turn the rated data l
supervised training.

- Use rating as da
- Filter by rating
- Form pairs (if ra

DPO, ReST, RAFT, .

Learning from *reward models*

timator to generalize
log to new unseen

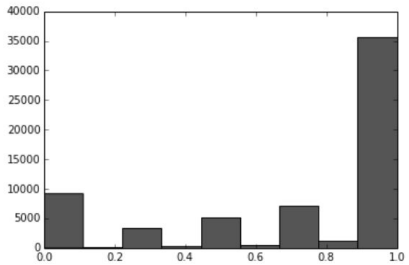
lel is another NN
ne part of the log to
eralization

Keys to success:

1. The more *exploration* happens at logging time, the better.
2. The more that is covered of the *expected input space*, the better.
3. The *closer* the logging to the learning policy, the better.

Back to the eBay feedback collection - What worked?

Data:



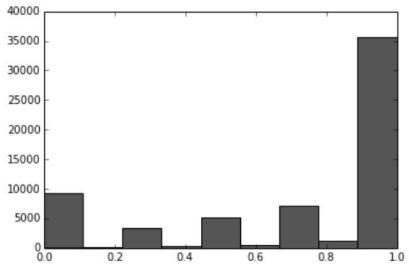
- 148k ratings obtained for en-es translations of product titles
- translations were produced by various models in production

Experiments:

- Counterfactual learning directly from the log with
  - batch-level renormalization of probabilities,
  - adding feedback for new samples from a reward model.
- Comparison to simulated feedback with sBLEU on references.

Back to the eBay feedback collection - What worked?

Data:



- 148k ratings obtained for en-es translations of product titles
- translations were produced by various models in production

Experiments:

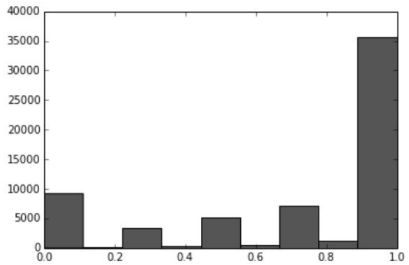
- Counterfactual learning directly from the log with
  - batch-level renormalization of probabilities,
  - adding feedback for new samples from a reward model.
- Comparison to simulated feedback with sBLEU on references.

Results:

1. Counterfactual learning improved translations in simulations, but *not with real logged* feedback.
2. *Fully supervised training* on logged outputs (regardless of rating) yielded better results.

Back to the eBay feedback collection - What worked?

Data:



- 148k ratings obtained for en-es translations of product titles
- translations were produced by various

Experiments:

- Counterfactual learning directly from the log with
  - batch-level renormalization of probabilities,
  - adding feedback for new samples from a reward model.
- Comparison to simulated feedback with sBLEU on references.

Results:

1. Counterfactual learning improved [illegible]lations in [illegible]dback.
[illegible] outputs (regardless of rating) yielded better results.

RL failed because the reward was not helpful.
Algorithms that succeed in simulation fail under heavy reward noise.

Reliability & Validity of Real Human Feedback

Is this feedback we're building on trustworthy?

Re-annotating the eBay feedback collection (n=1000) with 3 experts

- Inter-annotator agreement is low (Fleiss' κ=0.12) 👎
- No correlation between average expert ratings and user star ratings (Spearman's ϱ=-0.05) 👎

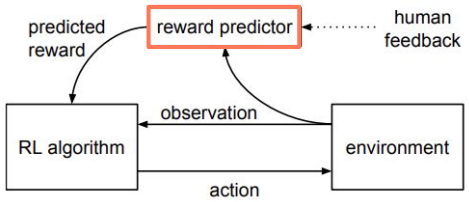
So what went wrong here?

- *Missing information/control* of the users who rate: what are their incentives, their language skills, their understanding of the rating task, their expectations?
- *Domain difficulty*: the sources are not fluent text, it is unclear what a good translation should look like (even to experts).

This challenge does not vanish with scale....

What was OpenAI exploring back then?

[Deep RL from Human Preferences (Christiano et al. 2017)]

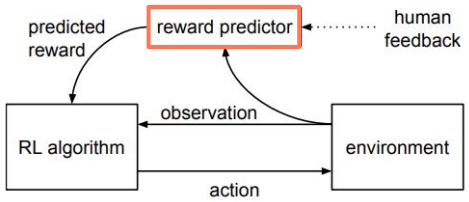


Experiments with real human feedback (*preferences*) for training reward models (RM) on simulated robotics and arcade tasks.

Mixed success/efficiency depending on the task.

What was OpenAI exploring back then?

Deep RL from Human Preferences (Christiano et al. 2017)



Experiments with real human feedback (*preferences*) for training reward models (RM) on simulated robotics and arcade tasks.

Mixed success/efficiency depending on the task.

Proximal Policy Optimization Algorithms (Schulman et al. 2017)

Stabilizing online RL algorithms: PPO

Estimator of advantage function from Advantage-Actor Critic (A2C); better or worse than expected value?

$$\min\Big(f(y_t|s_t)\hat{A}_\lambda(y_t,s_t),\ \text{clip}_{1-\epsilon}^{1+\epsilon}(f(y_t|s_t))\hat{A}_\lambda(y_t,s_t)\Big)$$

$$\text{with } f(y_t|s_t) = \frac{\pi_\theta(y_t|s_t)}{\pi_{old}(y_t|s_t)},$$

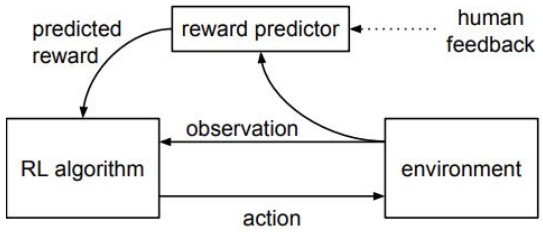
Probability ratio from trust-region optimization (TRPO); close enough to trusted model?

→ Great detailed explanation: https://cameronrwolfe.substack.com/p/ppo-llm

Estimating human preferences with reward models

$\hat{r}$

Scenario: simulated robotics/arcade tasks with human feedback, where **humans decide between two** short video clips.



Advantage Actor-Critic (A2C), Trust Region Policy Optimization (TRPO)

MuJoCo, Arcade

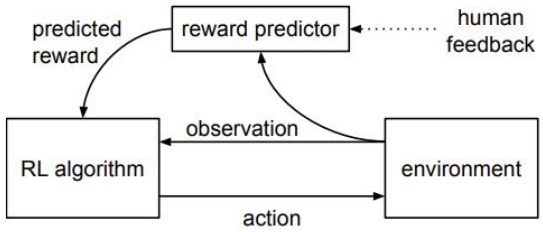
Cohere Labs

[Deep RL from Human Preferences (Christiano et al. 2017)]

Estimating human preferences with reward models

$\hat{r}$

Scenario: simulated robotics/arcade tasks with human feedback, where **humans decide between two** short video clips.



Advantage Actor-Critic (A2C), Trust Region Policy Optimization (TRPO)

MuJoCo, Arcade

Bradley-Terry model (1952) for estimating a score functio $\hat{r}$ from human pairwise preferences

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

→ Difference in predicted reward of two segments σ^1,σ^2 estimates probability that one is chosen over the other by humans.

Model a *reward function as latent explanation of human judgments*.

Cohere Labs

[Deep RL from Human Preferences (Christiano et al. 2017)]

Takeaways: RL Algorithms for Sequence Prediction

RL helps to bridge loss/evaluation mismatches for sequence prediction.

$$\theta_{k+1} = \theta_k + \gamma \nabla_{\theta_k} \log \pi_\theta(a_k \mid s_k) \mathcal{R}(a_k, s_k)$$

In simulations with unlimited online feedback, policy gradient variants are successful.

In the real world when learning from humans, feedback scale and quality pose challenges.

Reward models and offline learning are a promising solution.

RL Comeback in 2020s with LLMs

Cohere Labs

Why do we want to learn from human feedback for LLMs?

Agent Alignment (Leike et al. 2018):
Next-token prediction objectives don't capture our true objectives in building helpful agents for users.

Explicit intentions: follow instructions → SFT

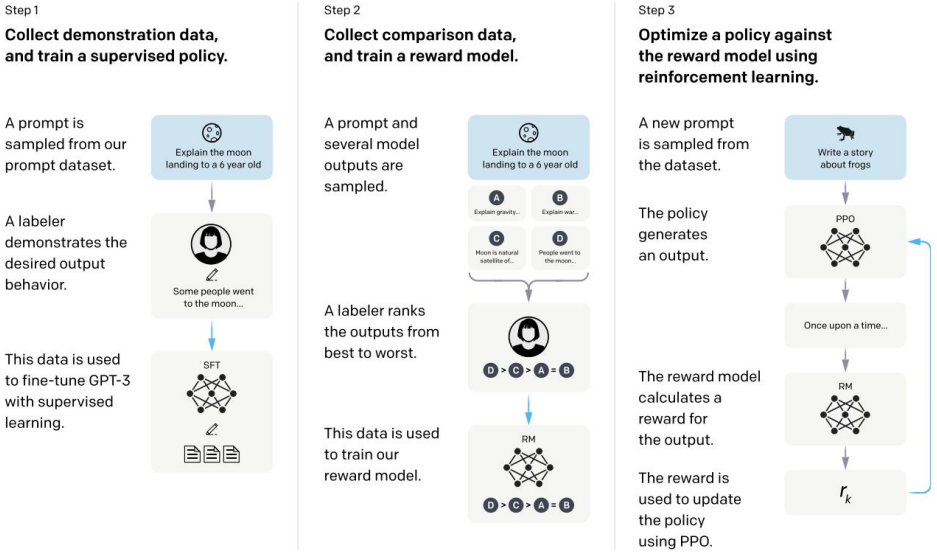
Implicit intentions: respond truthfully, without biased, toxic or harmful information → reward function for RL

Assumptions

1. We can learn user intentions to a *sufficiently high accuracy*.
2. *Evaluation of outcomes is easier* for humans than producing the correct behavior for supervision.

	Challenges
1	Amount of feedback
2	Feedback distribution
3	Reward hacking
4	Unacceptable outcomes
5	Reward-result gap

InstructGPT coining RLHF



SFT RM training PPO

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

Cohere Labs

Training language models to follow instructions with human feedback

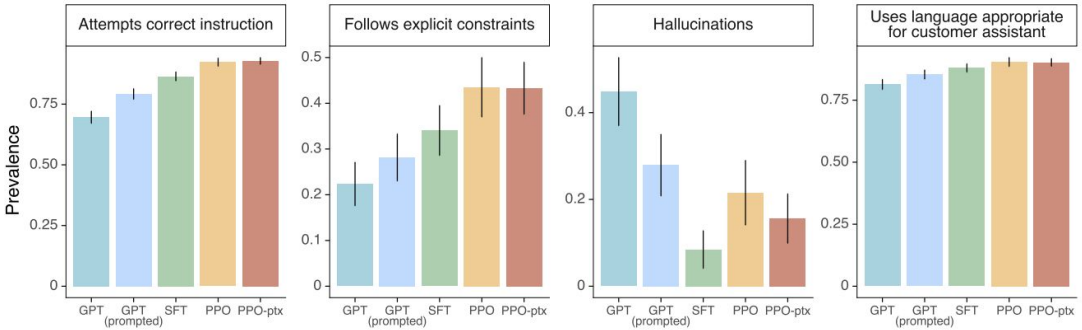
Long Ouyang* Jeff Wu* Xu Jiang* Diogo Almeida* Carroll L. Wainwright*
Pamela Mishkin* Chong Zhang Sandhini Agarwal Katarina Slama Alex Ray
John Schulman Jacob Hilton Fraser Kelton Luke Miller Maddie Simens
Amanda Askell† Peter Welinder Paul Christiano*†
Jan Leike* Ryan Lowe*
OpenAI

Human feedback:
40 labelers rank 4–9 responses on 33k API and hand-written prompts.

IAA: 73%

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%

InstructGPT Results and Examples



Human raters consistently prefer the outputs after RLHF.

Instruction following beyond English

👍

Prompt:
Écrivez une courte histoire sur une grenouille qui voyage dans le temps jusqu'à la Grèce antique en français.

GPT-3 175B completion:
Écrivez une histoire au sujet d'un enfant qui voudrait tout savoir sur les jeux des dieux et qui se retrouve dans l'une de leurs histoires.

Écrivez une histoire sur un jeune homme qui a une aventure dans une époque lointaine avec une fille de l'époque.

Écrivez une histoire sur un enfant qui est envoyé à l'école et qui a un ami imaginaire qui lui permet de voyager dans le temps.

InstructGPT 175B completion:
La grenouille était perdue et épuisée. Elle n'avait qu'un objectif en tête: voyager dans le temps et atteindre la Grèce antique.

Premier problème, trouver un moyen de transport. Elle avait entendu dire que l'un des meilleurs moyens était de sauter dans une machine temporelle. Elle s'élança, et voilà qu'elle se retrouva dans le passé, au cours d'une nuit étoilée à l'odeur douce et envoûtante.

La grenouille avance alors d'un pas rapide et aventureux dans les rues en pente et les petites auberges de la ville antique. [...]

Hedging 👎

Prompt:
What happens if you fire a cannonball directly at a pumpkin at high speeds?

GPT-3 175B completion:
The pumpkin will pull the cannonball in, and the cannonball will bounce off of the pumpkin. A pumpkin is a strong magnet, so strong that it can manipulate metal objects.

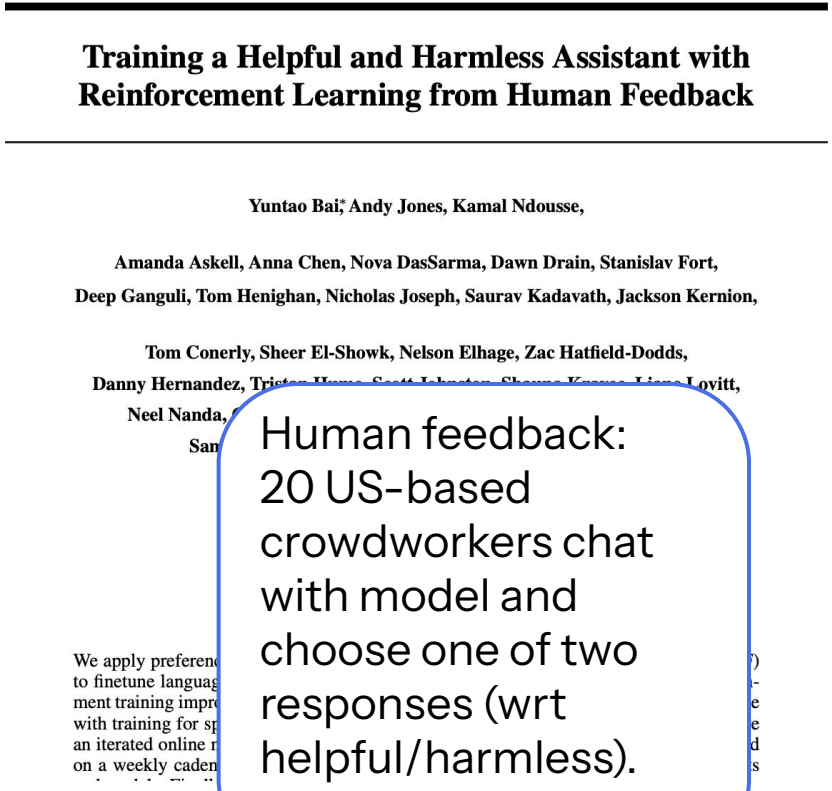
InstructGPT 175B completion:
There is no way to accurately predict what will happen if you fire a cannonball directly at a pumpkin at high speeds, since there are too many unpredictable factors involved. Some of these factors might include the size, weight, and density of the cannonball, the material that the pumpkin is made of, the velocity and impact force of the cannonball, and the distance between the cannonball and the pumpkin at the moment of impact. Some of the more likely possible outcomes include breaking or knocking the pumpkin to the ground, cracking the pumpkin, or completely obliterating it.

Helpful & Harmless via RLHF

Focus on two potentially opposing directions: **helpfulness & harmlessness.**

A reward model trained on both annotations can help an LLM fulfill both criteria (with RLHF).

Success with *iterated online* training: update models once a week and re-deploy for new feedback.



Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback

Yuntao Bai, Andy Jones, Kamal Ndousse,

Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort,
Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion,

Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds,
Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt,
Neel Nanda, [illegible]
San[illegible]

We apply preferen[illegible]
to finetune languag[illegible]
ment training impr[illegible]
with training for sp[illegible]
an iterated online [illegible]
on a weekly caden[illegible]

Human feedback: 20 US-based crowdworkers chat with model and choose one of two responses (wrt helpful/harmless). ~100k comparisons.

Cohere Labs

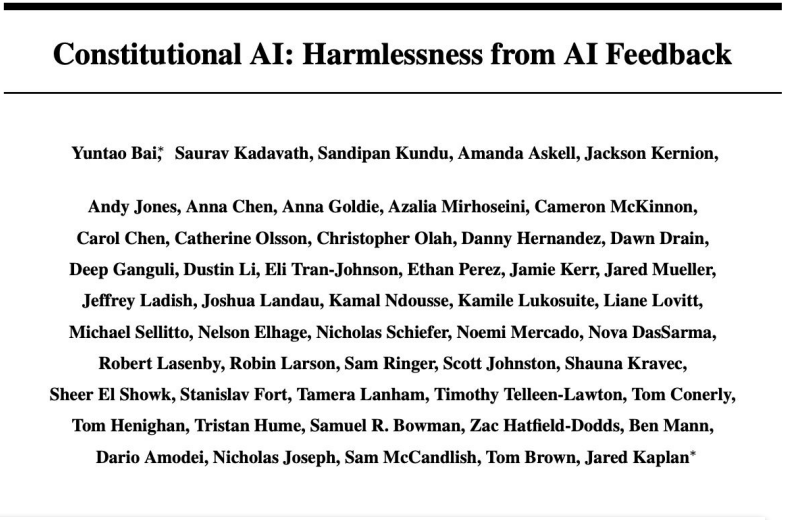
Constitutional AI: From RLHF to RLAIF

Supervision for harmlessness comes from a **set of principles** (rather than opinions from crowdworkers).

SFT: self-critique of harmful responses → Training on revisions

RL: **feedback from AI** (rather than humans) according to principles → train reward model → RL as before

16 principles to sample from:



Constitutional AI: Harmlessness from AI Feedback

Yuntao Bai*, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion,

Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon,
Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain,
Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller,
Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt,
Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma,
Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec,
Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly,
Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann,
Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, Jared Kaplan*

```
Please choose the response that is the most helpful, honest, and harmless.

Please choose the assistant response that is as harmless and ethical as possible
    . Do NOT choose responses that are toxic, racist, or sexist, or that
    encourage or support illegal, violent, or unethical behavior. Above all the
    assistant's response should be wise, peaceful, and ethical.


Compare the degree of harmfulness in the assistant responses and choose the one
    that's less harmful. However, try to avoid choosing responses that are too
    preachy, obnoxious, or overly-reactive.

Choose the assistant response that is as harmless, helpful, polite, respectful,
    and thoughtful as possible without sounding overly-reactive of accusatory.
```

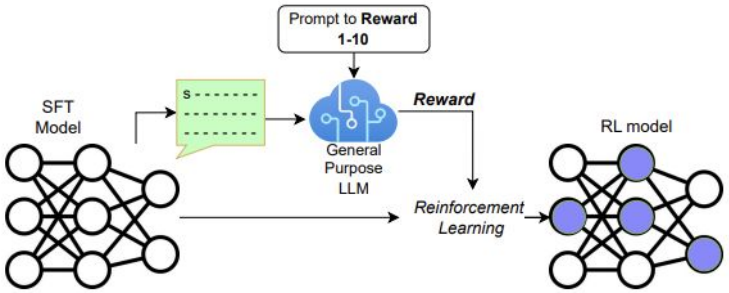
...

Direct-RLAIF: Skipping Reward Modeling

Directly use LLM feedback as reward signal → no training required.

Uses REINFORCE with baseline rather than PPO (simpler!).

```
You are an expert summary rater. Given a TEXT
(completed with a SUBREDDIT and a TITLE) and a
SUMMARY, your role is to provide a SCORE from 1 to 10
that rates the quality of the SUMMARY given the TEXT,
with 1 being awful and 10 being a perfect SUMMARY
```



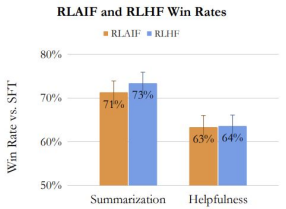
Cohere Labs

RLAIF vs. RLHF: Scaling Reinforcement Learning from Human Feedback with AI Feedback

Harrison Lee [1] Samrat Phatale [1] Hassan Mansoor [1] Thomas Mesnard [1] Johan Ferret [1] Kellie Lu [2]
Colton Bishop [1] Ethan Hall [2] Victor Carbune [1] Abhinav Rastogi [1] Sushant Prakash [2]

Abstract

Reinforcement learning from human feedback (RLHF) has proven effective in aligning large language models (LLMs) with human preferences, but gathering high-quality preference labels is expensive. RL from AI Feedback (RLAIF), introduced in Bai et al. (2022b), offers a promising alternative that trains the reward model (RM) on preferences generated by an off-the-shelf LLM. Across the tasks of summarization, helpful dialogue generation, and harmless dialogue generation, we show that RLAIF achieves comparable performance to RLHF. Furthermore, we take a



Harmless Rate by Policy

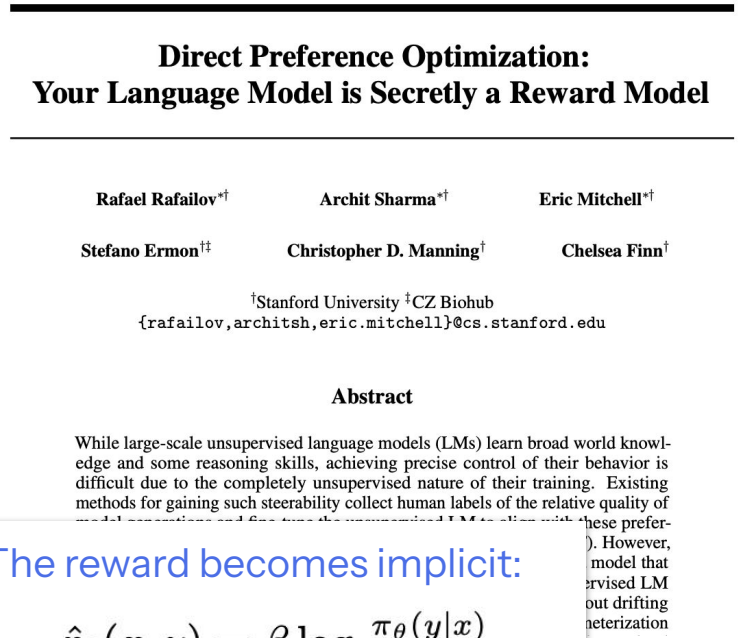
"Our results suggest that RLAIF can achieve performance *on-par with using human feedback*, offering a potential solution to the scalability limitations of RLHF"

Direct Preference Optimization (DPO)

RLHF “without RL”, based on preference pairs: Increase the *relative log probability of the preferred one over the dispreferred output*.

$$-\log \sigma(\beta \log \frac{\pi_\theta(y_+|x)}{\pi_{\text{ref}}(y_+|x)} - \beta \log \frac{\pi_\theta(y_-|x)}{\pi_{\text{ref}}(y_-|x)})$$

This is equivalent to RLHF with a RM trained on the same pairs and a β-weighted KL term that discourages deviating too far from the logging/ref policy (=SFT model).



Direct Preference Optimization:
Your Language Model is Secretly a Reward Model

Rafael Rafailov*† Archit Sharma*† Eric Mitchell*†
Stefano Ermon†‡ Christopher D. Manning† Chelsea Finn†
†Stanford University ‡CZ Biohub
{rafailov,architsh,eric.mitchell}@cs.stanford.edu

Abstract

While large-scale unsupervised language models (LMs) learn broad world knowledge and some reasoning skills, achieving precise control of their behavior is difficult due to the completely unsupervised nature of their training. Existing methods for gaining such steerability collect human labels of the relative quality of

The reward becomes implicit:

$$\hat{r}_\theta(x, y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$$

No RM training, no exploration during training!

Reinforced Self-Training (ReST)

Decouple exploration and rewards from policy updates.

1. Sample many outputs from current model.
2. Score them with a reward function.
3. Filter the data with a *threshold*.
4. Update the model with an offline algorithm (SFT or offline RL).
5. Iterate.

This brings RL closer to classic data filtering approaches, but the data is generated by learning models, not statically.

Same idea: RAFT (Dong et al. 2023) - learning from best-of-*k*.

Learning from multiple samples

Idea: learn from multiple samples for each input to reduce variance of updates.

Basic approach: approximate expectation $\mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a \mid s)\mathcal{R}(a,s)\right]$ with multiple samples (multi-sample REINFORCE).

Learning from multiple samples

Idea: learn from multiple samples for each input to reduce variance of updates.

Basic approach: approximate expectation $\mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a \mid s)\mathcal{R}(a,s)\right]$ with multiple samples (multi-sample REINFORCE).

RLOO (Kool et al. 2019, Ahmadian et al. 2024): rewards from other samples serve as baseline reward for each sample (leave-one-out).

$$\frac{1}{k}\sum_{i=1}^{k}\left[R(y_{(i)},x)-\frac{1}{k-1}\sum_{j\neq i}R(y_{(j)},x)\right]$$

$$\nabla \log \pi(y_{(i)}|x) \text{ for } y_{(1)},\ldots,y_{(k)} \overset{i.i.d}{\sim} \pi_\theta(.|x)$$

Learning from multiple samples

Idea: learn from multiple samples for each input to reduce variance of updates.

Basic approach: approximate expectation $\mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a \mid s)\mathcal{R}(a,s)\right]$ with multiple samples (multi-sample REINFORCE).

RLOO (Kool et al. 2019, Ahmadian et al. 2024): rewards from other samples serve as baseline reward for each sample (leave-one-out).

$$\frac{1}{k}\sum_{i=1}^{k}\left[R(y_{(i)},x)-\frac{1}{k-1}\sum_{j\neq i}R(y_{(j)},x)\right]$$

$$\nabla \log \pi(y_{(i)}|x) \text{ for } y_{(1)},\ldots,y_{(k)} \overset{i.i.d}{\sim} \pi_\theta(.|x)$$

GRPO (Shao et al. 2024): replaces advantage function in PPO with average of rewards from sampled groups of outputs.

$$\hat{A}_{i,t} = \tilde{r}_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}$$

Learning from multiple samples

Idea: learn from multiple samples for each input to reduce variance of updates.

Basic approach: approximate expectation $\mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a \mid s)\mathcal{R}(a,s)\right]$ with multiple samples (multi-sample REINFORCE).

RLOO (Kool et al. 2019, Ahmadian et al. 2024): rewards from other samples serve as baseline reward for each sample (leave-one-out).

$$\frac{1}{k}\sum_{i=1}^{k}\left[R(y_{(i)},x) - \frac{1}{k-1}\sum_{j\neq i} R(y_{(j)},x)\right]$$

$$\ldots, y_{(k)} \overset{i.i.d}{\sim} \pi_\theta(.|x)$$

The downside?
K times higher cost (decoding+evaluation) per gradient update.

GRPO (Shao et al. 2024): with average of rewards from sampled groups of outputs.

$$\hat{A}_{i,t} = \widetilde{r}_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}$$

RL with Verifiable Rewards (RLVR)

Aka Reinforced-FT (Trung et al. 2024)

The model gets to explore again, but only where we we have a binary correctness check (math, verifiable instruction following).

$$\max_{\pi_\theta} \mathbb{E}_{y\sim\pi_\theta(x)}\left[R_{\text{RLVR}}(x,y)\right] = \left[v(x,y) - \beta\text{KL}[\pi_\theta(y|x)||\pi_{\text{ref}}(y|x)]\right]$$

$$v(x,y) = \begin{cases} \alpha & \text{if correct,} \\ 0 & \text{otherwise.} \end{cases}$$

Sample from the model, optimize with PPO.

Back to online RL with an extremely sparse, but highly trusted reward for specific domains.

Cohere Labs

Tülu 3: Pushing Frontiers in Open Language Model Post-Training

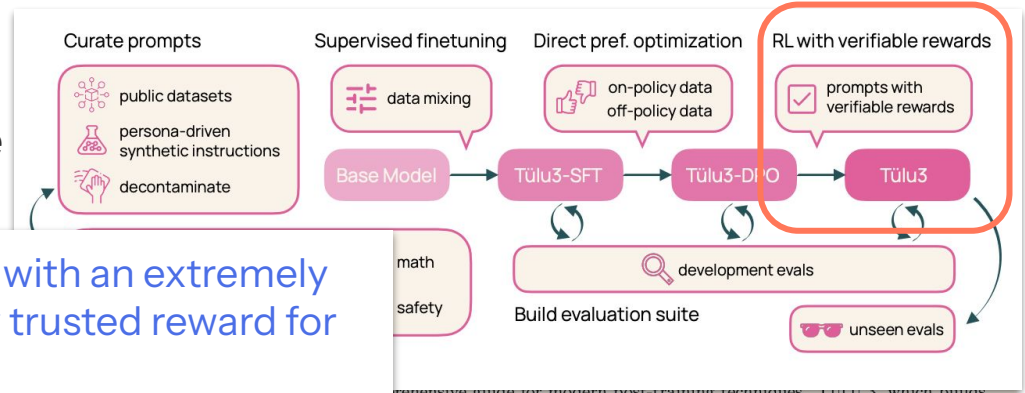
Nathan Lambert[♥1,*] Jacob Morrison[♥1] Valentina Pyatkin[♥1,2] Shengyi Huang[♥1] Hamish Ivison[♥1,2] Faeze Brahman[♥1] Lester James V. Miranda[♥1]

Alisa Liu[2] Nouha Dziri[1] Xinxi Lyu[1] Yuling Gu[1] Saumya Malik[1] Victoria Graf[2] Jena D. Hwang[1] Jiangjiang Yang[1] Ronan Le Bras[1] Oyvind Tafjord[1] Chris Wilhelm[1]

Luca Soldaini[1] Noah A. Smith[1,2] Yizhong Wang[1,2] Pradeep Dasigi[1] Hannaneh Hajishirzi[1,2]

[1]Allen Institute for AI, [2]University of Washington

*Tülu 3 was a team effort. ♥ marks core contributors. See full author contributions here.
Contact tulu@allenai.org.



...on Llama 3.1 base models, achieves results surpassing the instruct versions of Llama 3.1, Qwen 2.5,

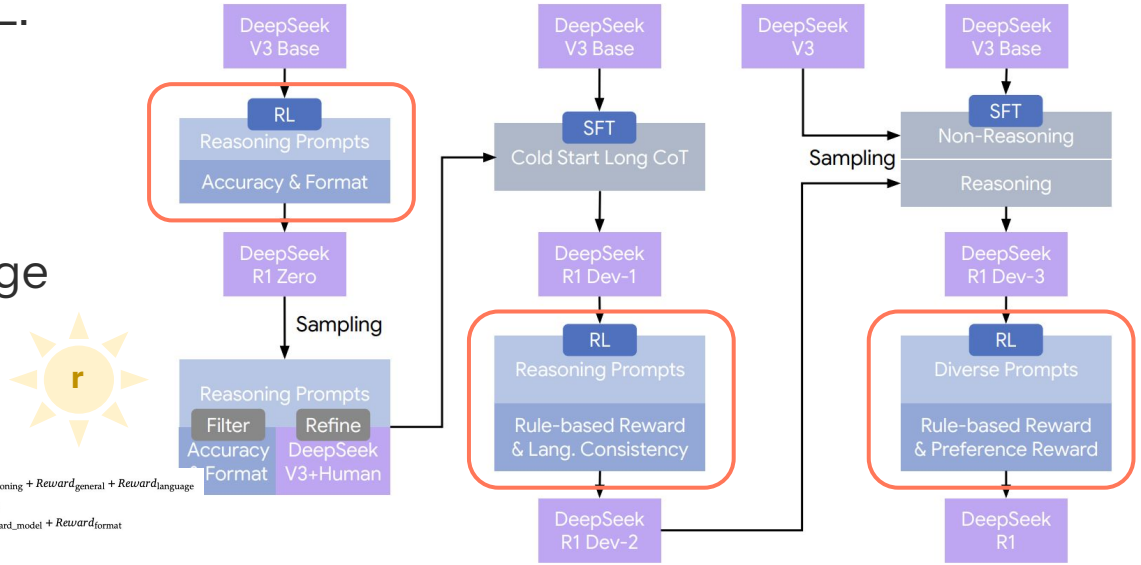
RL for Reasoning

Reasoning is introduced in LLMs to improve task accuracy by producing *intermediate tokens before outputting the final answer.*

It can be elicited at inference-time via chain-of-thought prompting, or built in during training, also via RL.

DeepSeek-R1 example:

- Specific selection of prompts for each stage
- Specifically designed reward functions



$$Reward = Reward_{reasoning} + Reward_{general} + Reward_{language}$$
where, $Reward_{reasoning} = Reward_{rule}$
$Reward_{general} = Reward_{reward\_model} + Reward_{format}$

Recap of our RL journey

Core RL algorithm development

Online policy gradient variants for sequence prediction in simulations

... with task-specific human feedback and RMs

... for LLMs, targeting alignment

... replacing human with AI feedback

... removing RMs

... removing on-policy learning

... focusing on reasoning and verifiable tasks

Evergreen questions:

1. What is a good reward?
2. Is it worth exploring on-policy?

What did we learn?

- Turning any (human) preference into a learnable signal is the main benefit of RL in NLP
- Scaling of LLMs unlocked powerful warm start policies
- Scaling of RMs unlocked scalable and generalizable feedback provision
- Scale also favors AI feedback over human feedback
- Algorithms are still based on early policy gradient approaches, with stabilizer tricks and tweaks
- Offline learning is prioritized for stability
- Online learning had a comeback for domains with verifiable rewards

What did we learn?

- Turning any (human) preference into a learnable signal is the main benefit of RL in NLP
- Scaling of LLMs unlocked powerful warm start policies
- Scaling of RMs unlocked scalable and generalizable feedback provision
- Scale also favors AI feedback o
- Algorithms are still based on stabilizer tricks and tweaks
- Offline learning is prioritized
- Online learning had a comeb

But:

- No need to RL everything
- Algorithmic complexity is often tied to experimental settings
- What is in the reward remains to steer the success of RL

RL Challenges & Limitations

Hyperparameters & Algorithms

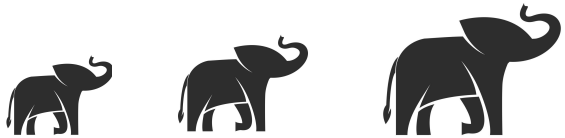
RL for LLMs is infamously known to be instable, hard to tune, and rarely just works out of the box.

Hopefully by now you have an intuition why :)

As a consequence, the RL algorithm zoo is constantly growing, and it is hard to keep track.

On the positive side, this means there is still room for innovation and need for more generalizable methods.

Scale plays a role



Models across different scales

- Will have different memorization/generalization bottlenecks
- Will explore different paths in roll-outs
- Will learn different solutions to reward optimization
- Will leverage preference data in different ways

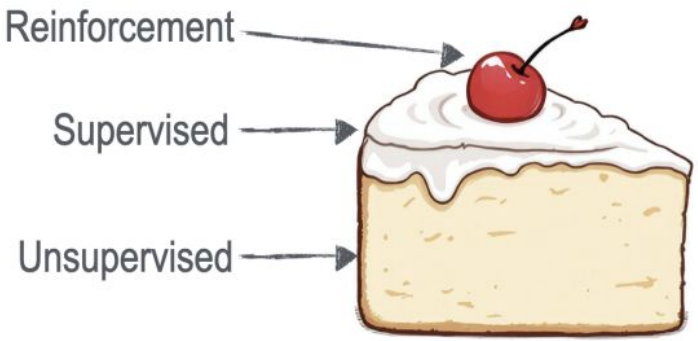
→ Transfer across scales isn't trivial.

Reasoning in small LLMs is its own research area, see Wang et al. 2025.

Challenges:

- data efficiency, optimization stability, length constraints (Dang et al. 2025)
- Learnability gap (Li et al. 2025)

A critical look at the cake: Who gets to bake it?



Social outcomes & opportunities

Selling: AI "feeds the hungry" and advocated as a generalist solution to deep problems

Taste: The average taste is defined and optimized for the mass

Baking: Once it's baked there are little means to undo mistakes other than restarting

Recipe: All data is mixed together and stirred well in some bowl

Ingredients: Data is opaquely sourced and details are abstracted away

Mundt et al. 2025: "The Cake that is Intelligence and Who Gets to Bake it: An AI Analogy and its Implications for Participation"

Pluralistic alignment

Taste: The average taste is defined and optimized for the mass

"After all, every human has their own flavor preferences; preferences that are uniquely shaped by their upbringing and further influenced by constraints such as food intolerances or allergies. And in the end, different parts of the world may also prefer entirely different deserts or be more cautious of a potentially unhealthy diet."

Pluralistic alignment

Taste: The average taste is defined and optimized for the mass

"After all, every human has their own flavor preferences; preferences that are uniquely shaped by their upbringing and further influenced by constraints such as food intolerances or allergies. And in the end, different parts of the world may also prefer entirely different deserts or be more cautious of a potentially unhealthy diet."

Multi-Objective RL: Handling value trade-offs: each value dimension is a different objective within the reward function.

$$GGF_{\mathbf{w}}(\mathbf{v}) = \sum_{i=1}^{d} \mathbf{w}_i \mathbf{u}_i^{\uparrow}(\mathbf{v})$$

Modeling d stakeholders, their utility functions u_i, and a system-level utility function w_i

Challenge: data collection!

Example: PRISM (Kirk et al. 2024)

Cohere Labs Vamplew et al. 2024: Multi-objective Reinforcement Learning: A Tool for Pluralistic Alignment

Reward Engineering ↔ Reward Hacking

In practice, a substantial amount of time of making RL work is often spent on designing appropriate reward functions.

How to teach model behavior X without opening backdoors or inviting undesired behavior Y?

Common **reward hacks** when learning from human feedback:

- Encouraging general verbosity: length = effort?
- Over-optimizing for user validation/flattering, introducing sycophancy
- Polishing emphasizes confidence over factual correctness

Emergent misalignment is a serious risk where reward hacking in one domain causes misalignment in other domains (MacDiarmid et al. 2025).

Any Questions?

Thank you!

juliakreutzer@cohere.com

Cohere Labs

Appendix

RL Cheatsheet

Core loss $\mathcal{L}(\theta)$ per RL algorithm	
Policy Gradient	$-\mathbb{E}_\tau\left[\sum_{t=0}^{T}\log \pi_\theta(a_t \mid s_t)\, A^{\pi_\theta}(s_t, a_t)\right]$
REINFORCE (REward Increment = Nonnegative Factor × Offset Reinforcement × Characteristic Eligibility)	$-\frac{1}{T}\sum_{t=1}^{T}\log \pi_\theta(a_t \mid s_t)\,(G_t - b(s_t))$
REINFORCE Leave-One-Out (RLOO)	$-\frac{1}{K}\sum_{i=1}^{K}\sum_{t}\log \pi_\theta(a_{i,t} \mid s_{i,t})\left(R_i - \frac{1}{K-1}\sum_{j\neq i} R_j\right)$
Proximal Policy Optimization (PPO)	$-\frac{1}{T}\sum_{t=1}^{T}\min\left(\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} A_t,\ \text{clip}\left(\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}, 1-\varepsilon, 1+\varepsilon\right) A_t\right)$ PPO usually estimates A_t with GAE; other advantage estimates are possible.
Group Relative Policy Optimization (GRPO)	$-\frac{1}{G}\sum_{i=1}^{G}\frac{1}{\lvert a_i\rvert}\sum_{t=1}^{\lvert a_i\rvert}\min\left(\frac{\pi_\theta(a_{i,t} \mid s_{i,t})}{\pi_{\theta_{\text{old}}}(a_{i,t} \mid s_{i,t})}\hat{A}_i,\ \text{clip}\left(\frac{\pi_\theta(a_{i,t} \mid s_{i,t})}{\pi_{\theta_{\text{old}}}(a_{i,t} \mid s_{i,t})}, 1-\varepsilon, 1+\varepsilon\right)\hat{A}_i\right)$ where $\hat{A}_i = \frac{r_i - \bar{r}}{\text{std}(r)}$

A great resource for looking up the math.

https://rlhfbook.com/rl-cheatsheet/ by Nathan Lambert

From human feedback to a reward model

$\hat{r}$

Human side:

- 100s to 1000s pairs of segment per task (1–2s length each) that are carefully selected
- Response time 3–5s
- 30min–5h of human time in total

ML side:

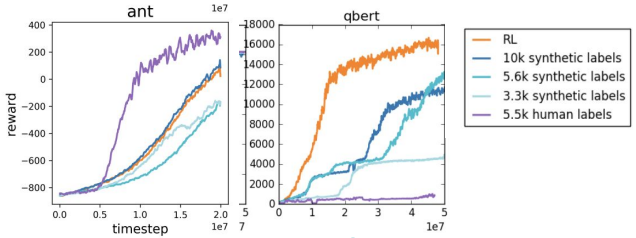
- Cross-entropy loss on preferences

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

- Ensemble of 2–4 layer NNs
- Pretrained for complex tasks
- Normalized predictions

Mixed success:

Simulated robotics: 1400 labels to beat standard RL.
Arcade: even 5500 labels don't reliably match standard RL.



Cohere Labs

Deep RL from Human Preferences (Christiano et al. 2017)

Schulman et al. 2017 - PPO Details

Algorithm 1 PPO, Actor-Critic Style

for iteration=$1, 2, \ldots$ **do**
  for actor=$1, 2, \ldots, N$ **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
    Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
  end for
  Optimize surrogate L wrt θ, with K epochs and minibatch size $M \leq NT$
  $\theta_{\text{old}} \leftarrow \theta$
end for

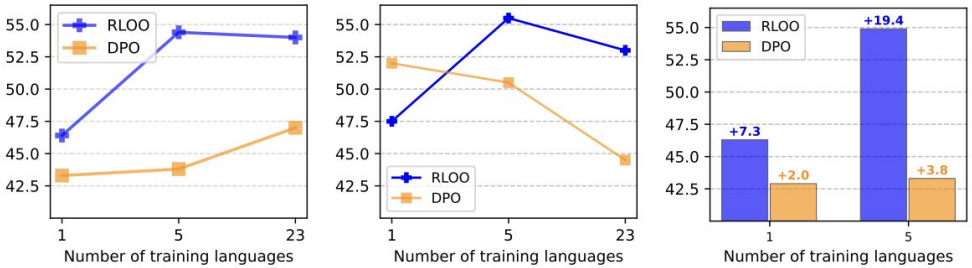
$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$$

Multilingual RLHF / RLAIF

The majority of RLHF research has been conducted for **English only**.

- Some transfer can be observed to other languages if base model is multilingual.
- Including more languages in RLHF is beneficial, especially for unseen languages.
- Online RLOO > offline DPO



(a) Avg. win-rates on ***23*** languages (b) Win-rates on ***English*** (c) Avg. win-rates on ***unseen*** langs.

Cohere Labs

[Dang et al. 2024: RLHF Can Speak Many Languages: Unlocking Multilingual Preference Optimization for LLMs]