Lecture 19: Model-Based RL and RL from Human Feedback CS486/686 Intro to Artificial Intelligence

2026-3-17

Pascal Poupart
David R. Cheriton School of Computer Science
CIFAR AI Chair at Vector Institute

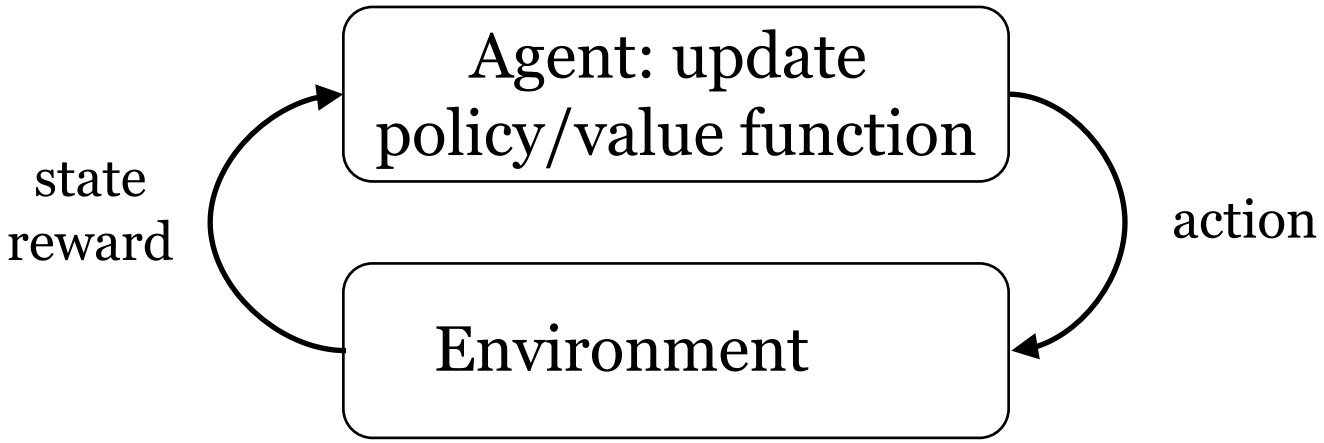
UNIVERSITY OF WATERLOO

Outline

- Model-based RL
- Dyna
- Reinforcement Learning from Human Feedback (RLHF)

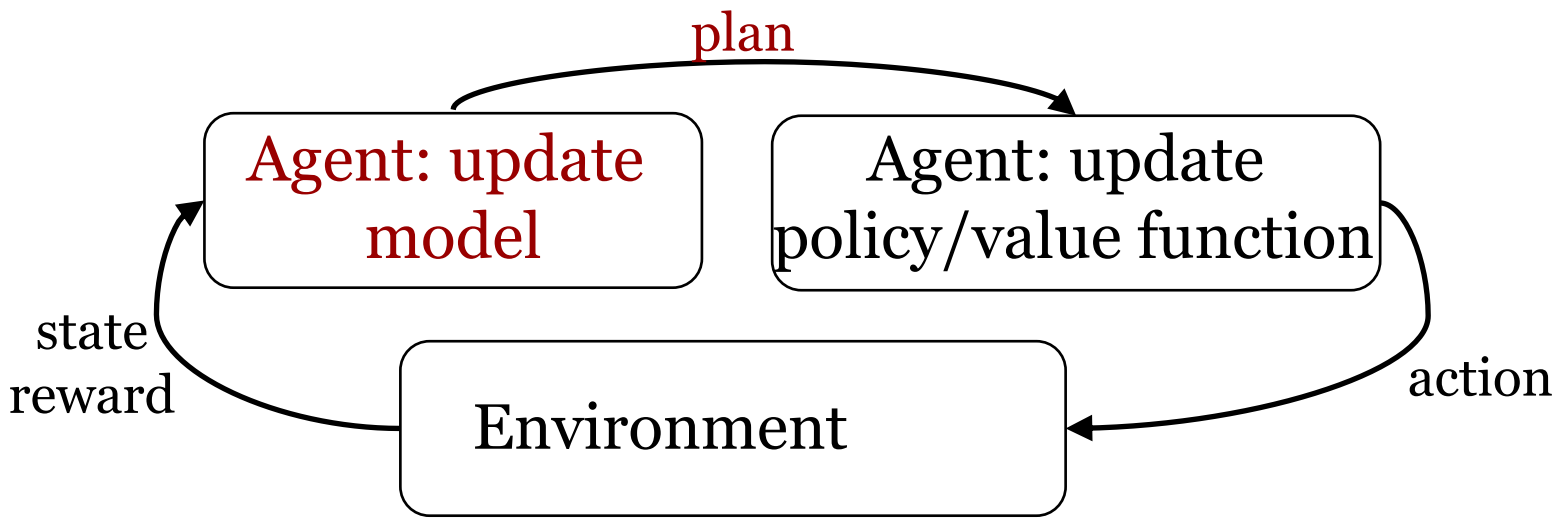
Model-free Online RL

- No explicit transition or reward models
  - Q-learning: **value-based method**
  - Policy gradient: **policy-based method**

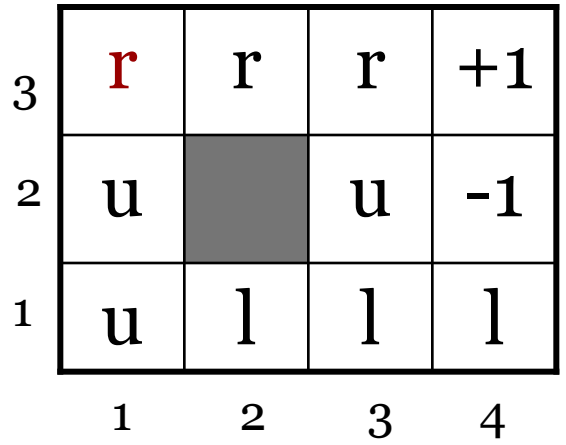


Model-based Online RL

- Learn explicit transition and/or reward model
  - Plan based on the model
  - Benefit: Increased sample efficiency
  - Drawback: Increased complexity



Maze Example



$\gamma = 1$

Reward is -0.04 for non-terminal states

We need to learn all the transition probabilities!

(1,1)→ (1,2)→ **(1,3)**→ (1,2)→ **(1,3)**→ (2,3)→ (3,3)→ $(4,3)_{+1}$
(1,1)→ (1,2)→ **(1,3)**→ (2,3)→ (3,3)→ (3,2)→ (3,3)→ $(4,3)_{+1}$
(1,1)→ (2,1)→ (3,1)→ (3,2)→ $(4,2)_{-1}$

$P((2,3)|(1,3),r) = 2/3$
$P((1,2)|(1,3),r) = 1/3$

} Use this information in

$$V^*(s) = \max_a R(s,a) + \gamma \sum_{s'} \Pr(s'|s,a)\, V^*(s')$$

Model-based RL

- Idea: at each step
  - Execute action
  - Observe resulting state and reward
  - Update transition and/or reward model
  - Update policy and/or value function

Model-based RL (with Value Iteration)

ModelBasedRL(s)

Repeat

Select and execute a

Observe s' and r

Update counts: $n(s,a) \leftarrow n(s,a) + 1$,

$n(s,a,s') \leftarrow n(s,a,s') + 1$

Update transition: $\Pr(s'|s,a) \leftarrow \frac{n(s,a,s')}{n(s,a)} \; \forall s'$

Update reward: $R(s,a) \leftarrow \frac{r + (n(s,a)-1)R(s,a)}{n(s,a)}$

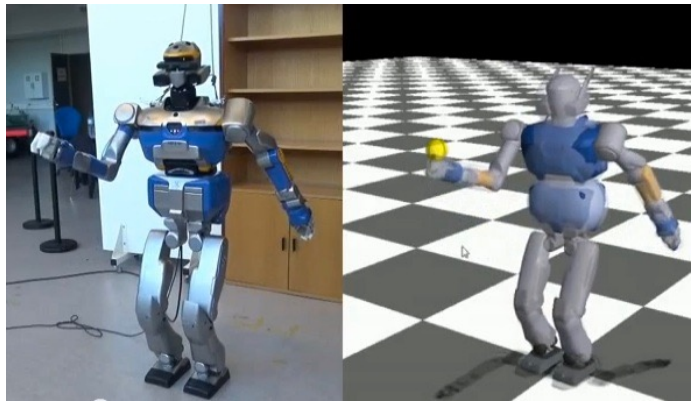
Solve: $V^*(s) = \max_a R(s,a) + \gamma \sum_{s'} \Pr(s'|s,a) V^*(s') \; \forall s$

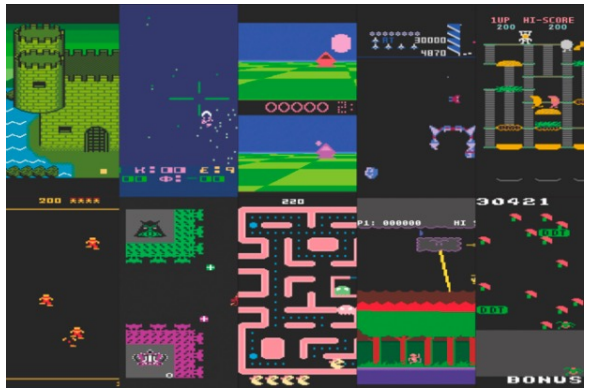
$s \leftarrow s'$

Until convergence of V^*

Return V^*

Complex Models





- Use function approximation for transition and reward models
  - Linear model: $pdf(s'|s,a) = N(s'|w^T \begin{bmatrix} S \\ a \end{bmatrix}, \sigma^2 I)$
  - Non-linear models:
    - Stochastic (e.g., Gaussian process): $pdf(s'|s,a) = GP(s|w^T \begin{bmatrix} S \\ a \end{bmatrix}, \sigma^2 I)$
    - Deterministic (e.g., neural network): $s' = T(s,a) = NN(s,a)$

Partial Planning

- In complex models, fully optimizing the policy or value function at each time step is intractable

- Consider partial planning
  - A few steps of Q-learning
  - A few steps of policy gradient

Model-based RL (with Q-learning)

ModelBasedRL(s)

Repeat

Select and execute a, observe s' and r

Update transition: $w_T \leftarrow w_T - \alpha_T(T(s,a) - s')\nabla_{w_T} T(s,a)$

Update reward: $w_R \leftarrow w_R - \alpha_R(R(s,a) - r)\nabla_{w_R} R(s,a)$

Repeat a few times:

sample $\hat{s}, \hat{a}$ arbitrarily

$\delta \leftarrow R(\hat{s},\hat{a}) + \gamma \max_{\hat{a}'} Q(T(\hat{s},\hat{a}),\hat{a}') - Q(\hat{s},\hat{a})$

Update Q: $w_Q \leftarrow w_Q - \alpha_Q \delta \nabla_{w_Q} Q(\hat{s},\hat{a})$

$s \leftarrow s'$

Until convergence of Q

Return Q

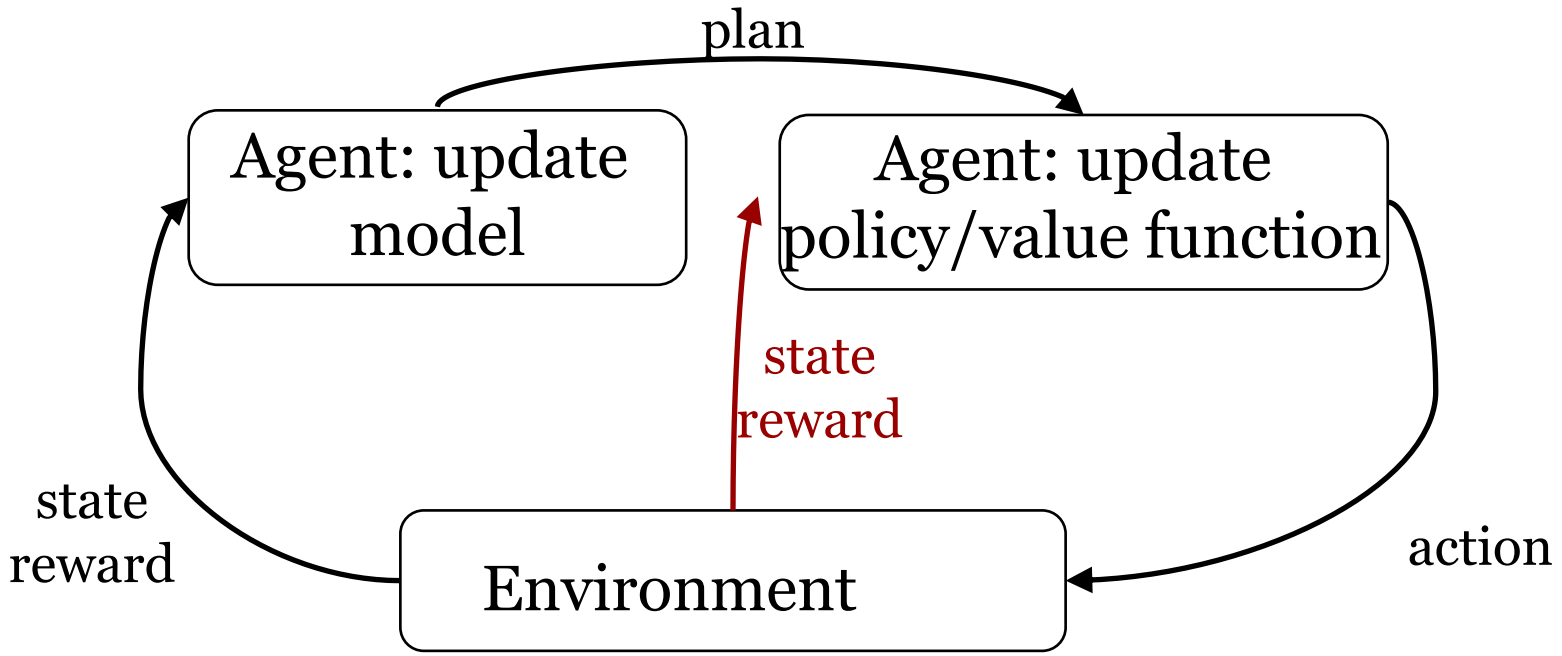
Partial Planning vs Replay Buffer

- Previous algorithm is very similar to Model-free Q-learning with a replay buffer
- Instead of updating Q-function based on samples from replay buffer, generate samples from model

- Replay buffer:
  - Simple, real samples, no generalization to other sate-action pairs
- Partial planning with a model
  - Complex, simulated samples, generalization to other state-action pairs (can help or hurt)

Dyna

- **Learn explicit transition and/or reward model**
  - Plan based on the model
- **Learn directly from real experience**



Dyna-Q

Dyna-Q(s)

Repeat

Select and execute a, observe s' and r

Update transition: $w_T \leftarrow w_T - \alpha_T(T(s,a) - s')\nabla_{w_T} T(s,a)$

Update reward: $w_R \leftarrow w_R - \alpha_R(R(s,a) - r)\nabla_{w_R} R(s,a)$

$\delta \leftarrow r + \gamma \max_{a'} Q(s',a') - Q(s,a)$

Update Q: $w_Q \leftarrow w_Q - \alpha_Q \delta \nabla_{w_Q} Q(s,a)$

Repeat a few times:

sample $\hat{s}, \hat{a}$ arbitrarily

$\delta \leftarrow R(\hat{s},\hat{a}) + \gamma \max_{\hat{a}'} Q(T(\hat{s},\hat{a}),\hat{a}') - Q(\hat{s},\hat{a})$

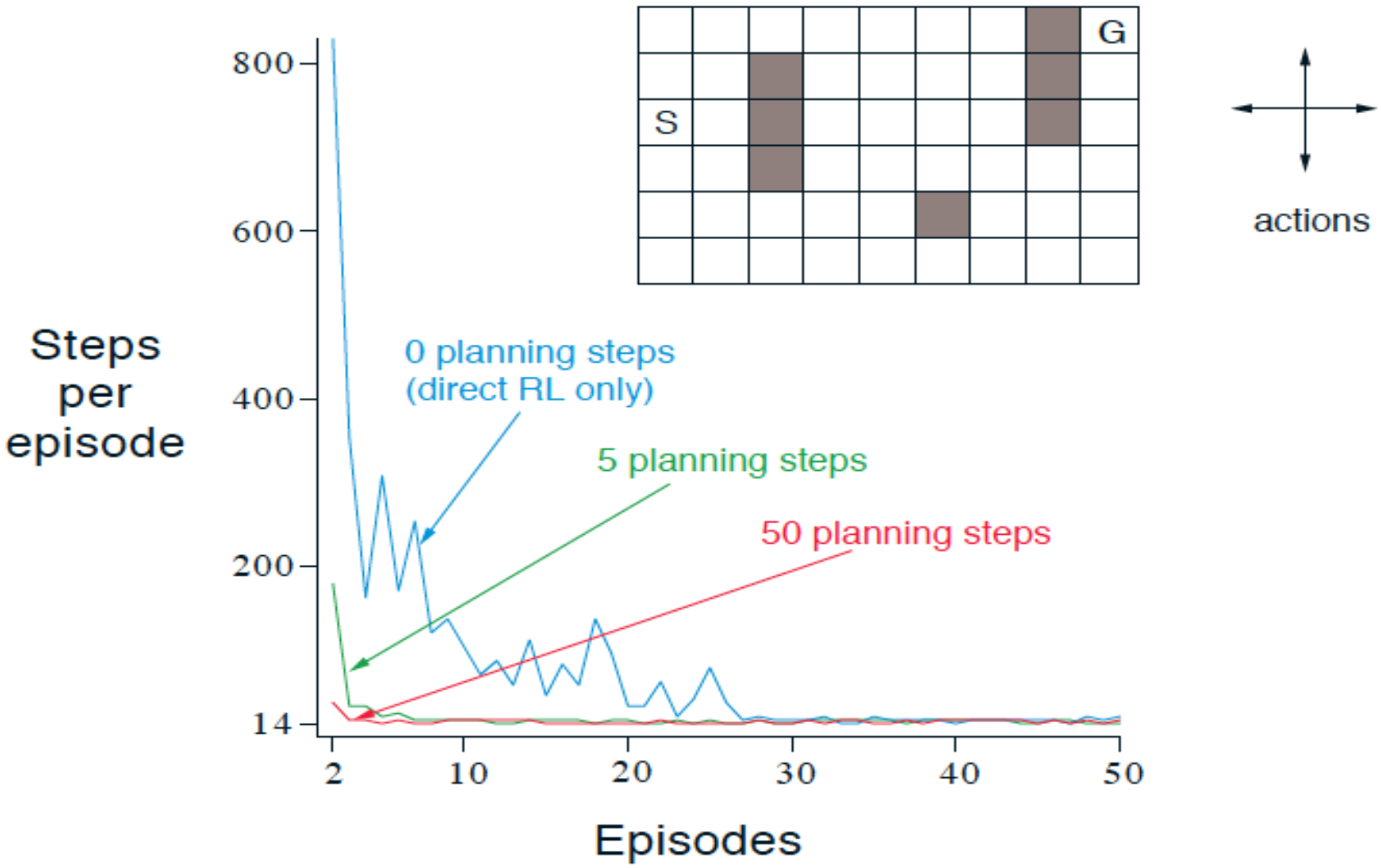
Update Q: $w_Q \leftarrow w_Q - \alpha_Q \delta \nabla_{w_Q} Q(\hat{s},\hat{a})$

$s \leftarrow s'$

Return Q

Dyna-Q

Task:
reach G
from S



Large Language Models

Source: Tie et al. (2025) A Survey on Post Training of Large Language Models

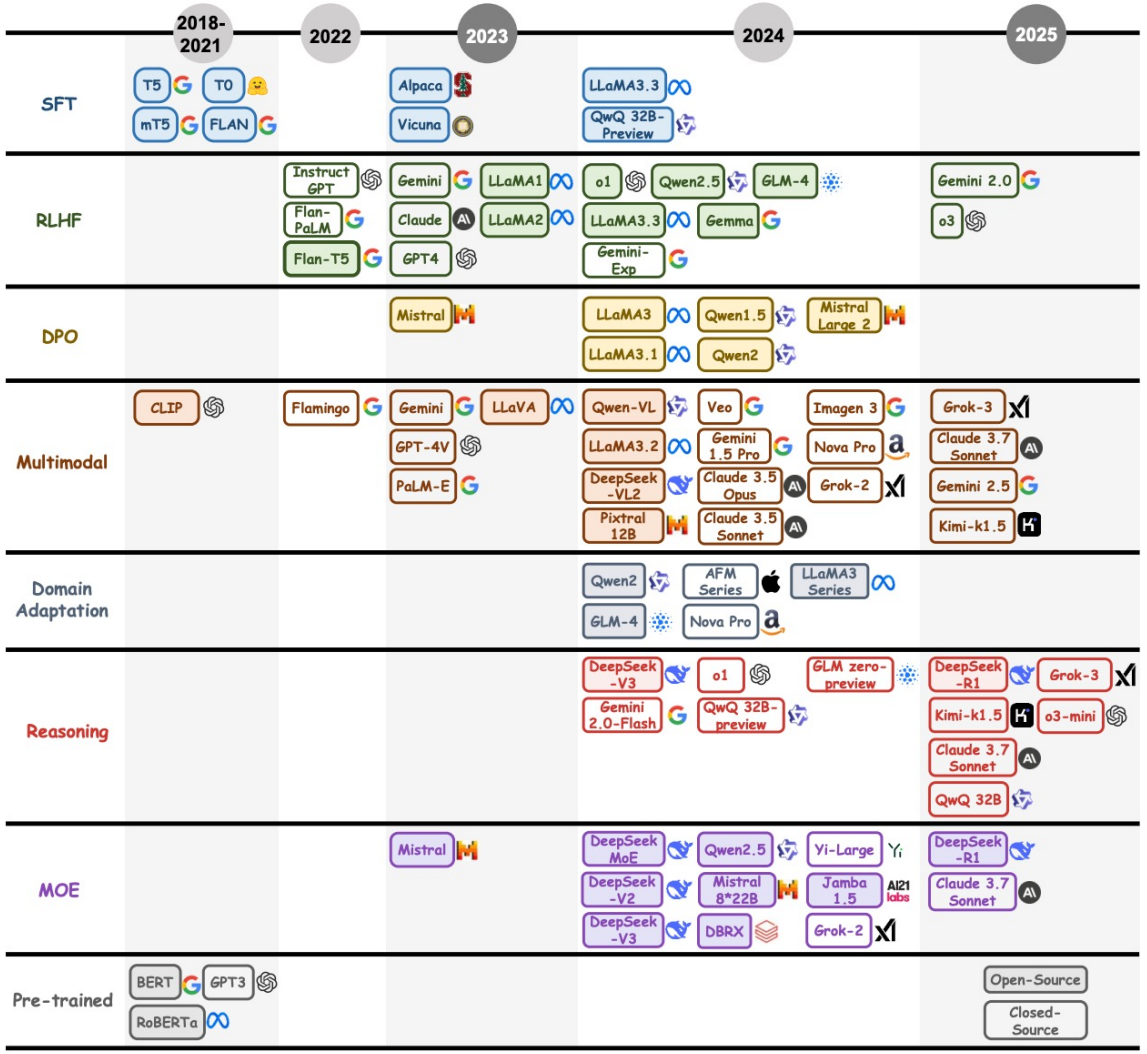
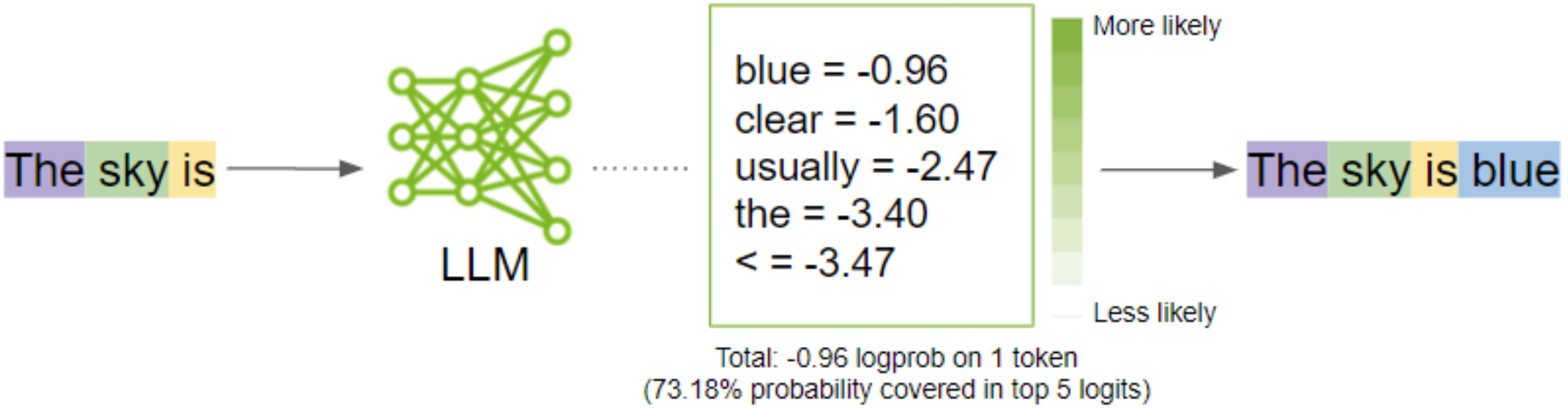


Figure 3: Timeline of post-training technique development for Large Language Models (2018–2025), delineating key milestones in their historical progression.

What is a Large Language Model (LLM)?

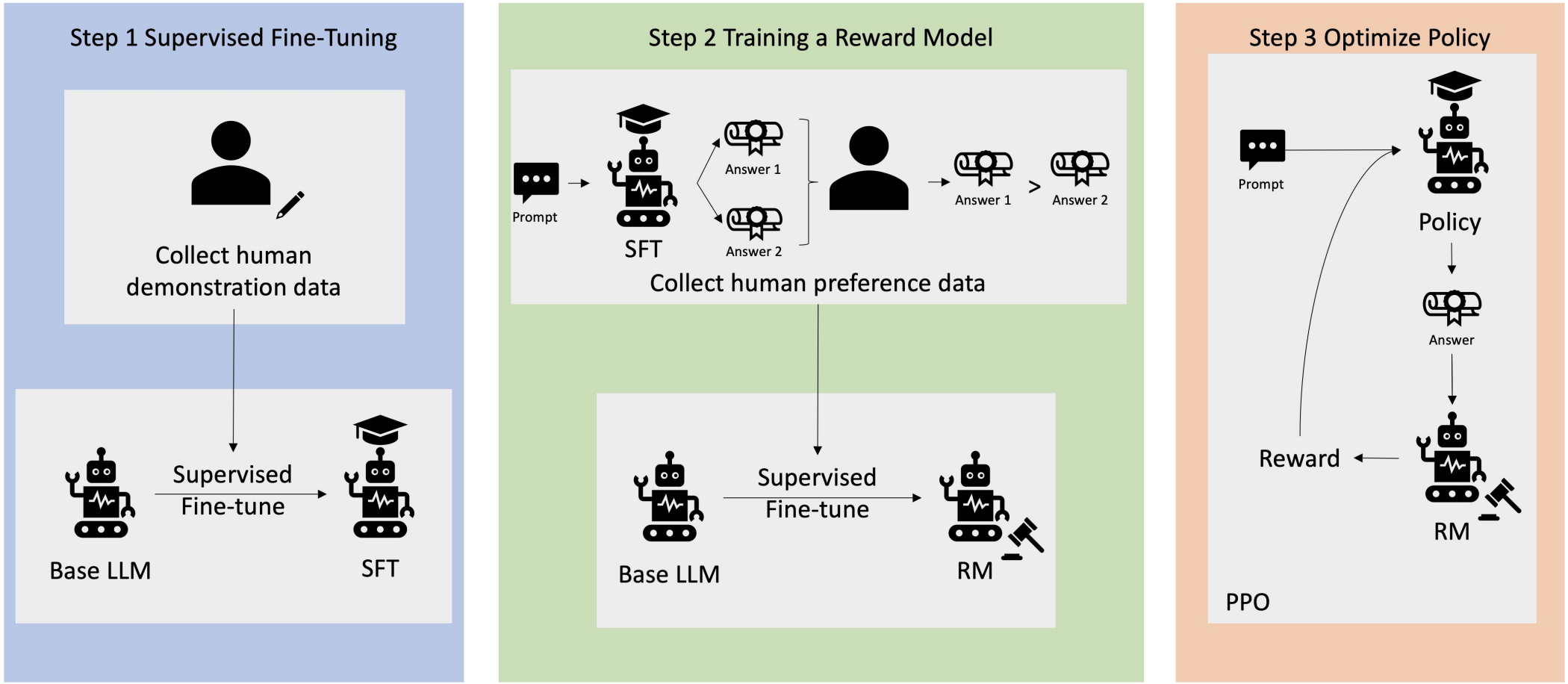
An LLM is a probabilistic model that

- takes as input a sequence of tokens and
- predicts the next token



Credit: https://developer.nvidia.com/blog/how-to-get-better-outputs-from-your-large-language-model/

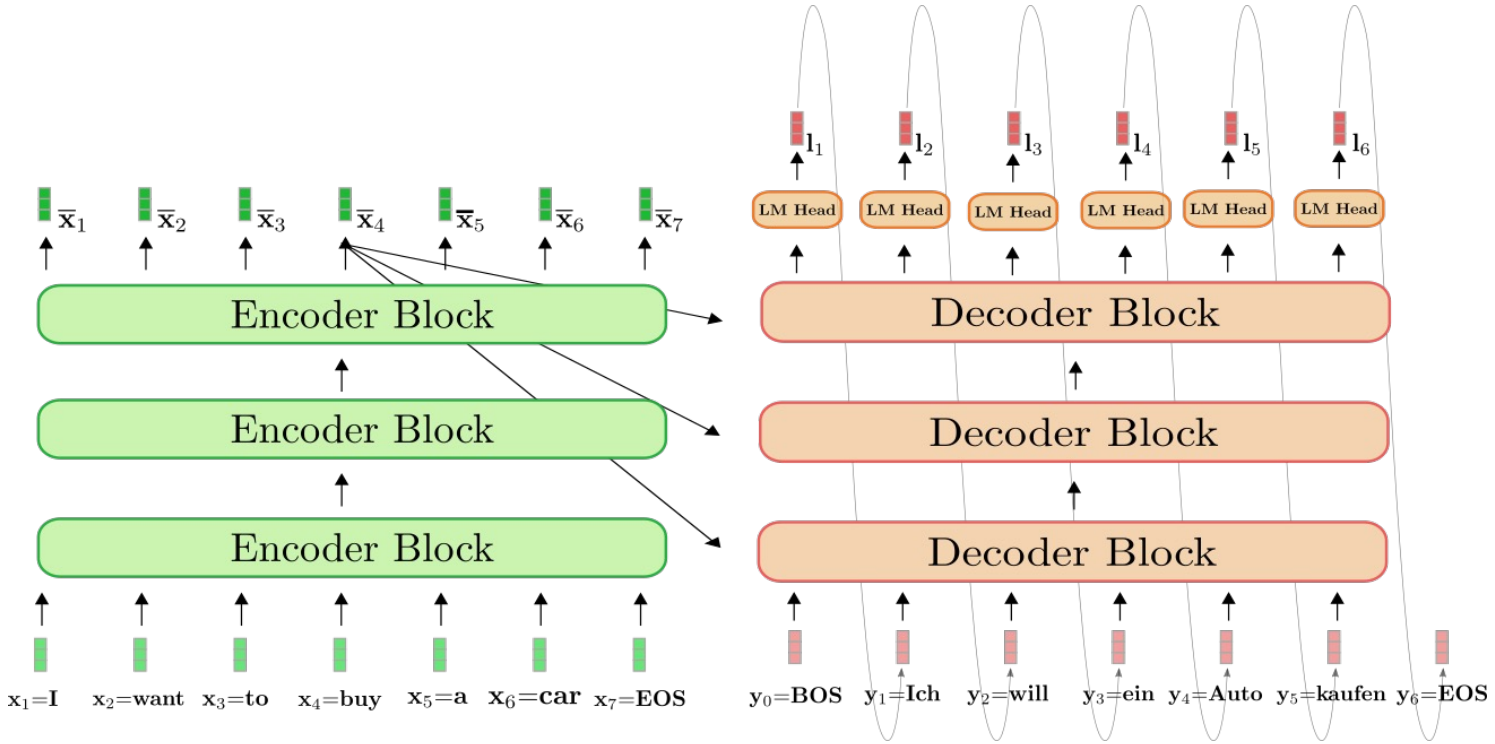
Recipe for Training LLMs



Credit: https://aws.amazon.com/what-is/reinforcement-learning-from-human-feedback/

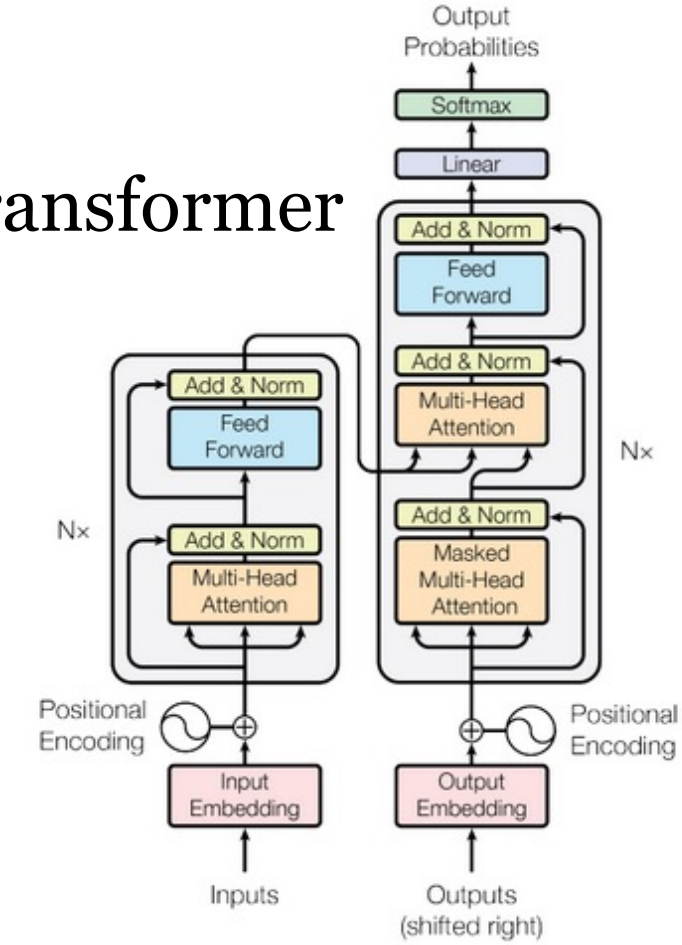
Self-Supervised Learning for LLMs

Encoder and Decoder



Credit: https://huggingface.co/blog/encoder-decoder

Transformer

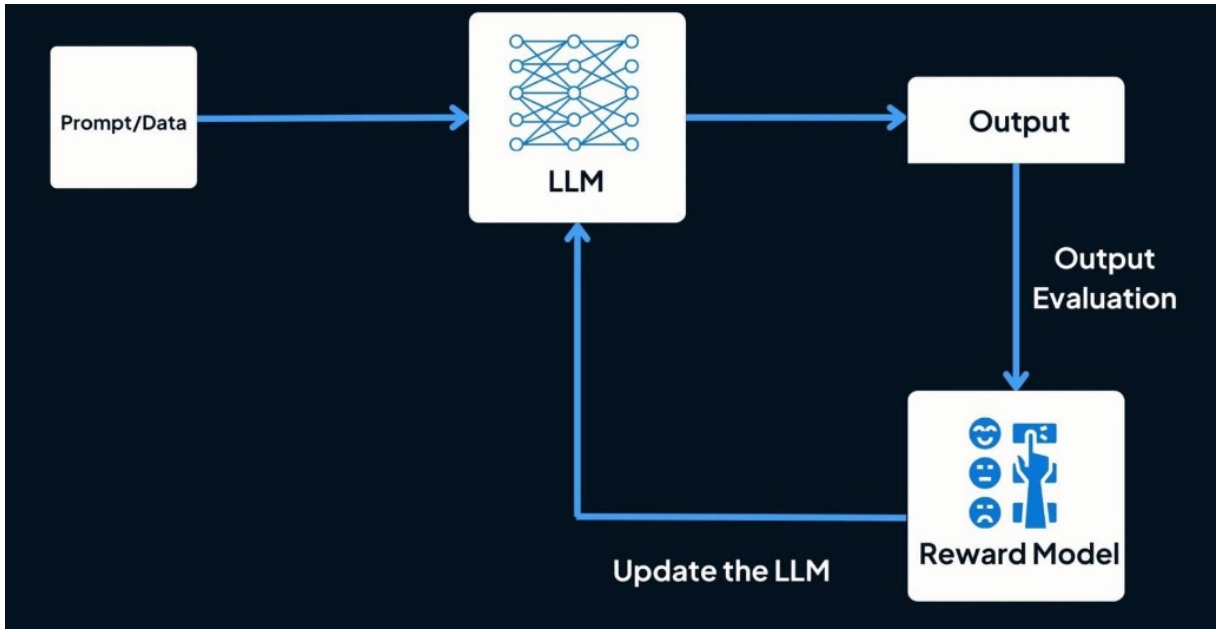


Credit: Vaswani et al., 2017

UNIVERSITY OF WATERLOO

Improving LLMs by RL

- **Agent:** system
- **Environment:** user
- **State:** history of past utterances
- **Action:** system utterance
- **Reward:** task completion, human feedback



Credit: https://www.twine.net/blog/what-is-reinforcement-learning-from-human-feedback-rlhf-and-how-does-it-work/

"*We posit that the superior writing abilities of LLMs, as manifested in surpassing human annotators in certain tasks, are fundamentally driven by RLHF, as documented in Gilardi et al. (2023) and Huang et al. (2023).*"
Llama 2 Technical Report (Touvron et. al 2023)

"*This behavior (re-evaluation) is not only a testament to the model's growing reasoning abilities but also a captivating example of how reinforcement learning can lead to unexpected and sophisticated outcomes*"
Deepseek-R1 Technical Report (Guo et. al 2025)

Response Ranking

PP

Explain reinforcement learning in three sentences.

1 / 2

Reinforcement learning is a type of machine learning where an agent learns to make decisions by receiving feedback in the form of rewards or punishments. The agent's goal is to maximize its cumulative reward over time by exploring different actions and learning which actions lead to the highest rewards. Reinforcement learning has been successfully applied to a variety of tasks, including game playing, robotics, and recommendation systems.

2 / 2

Reinforcement learning is a type of machine learning that focuses on training an agent to make decisions based on trial and error. The agent receives feedback in the form of rewards or penalties for each decision it makes. By learning from this feedback, the agent can improve its decision-making abilities over time.

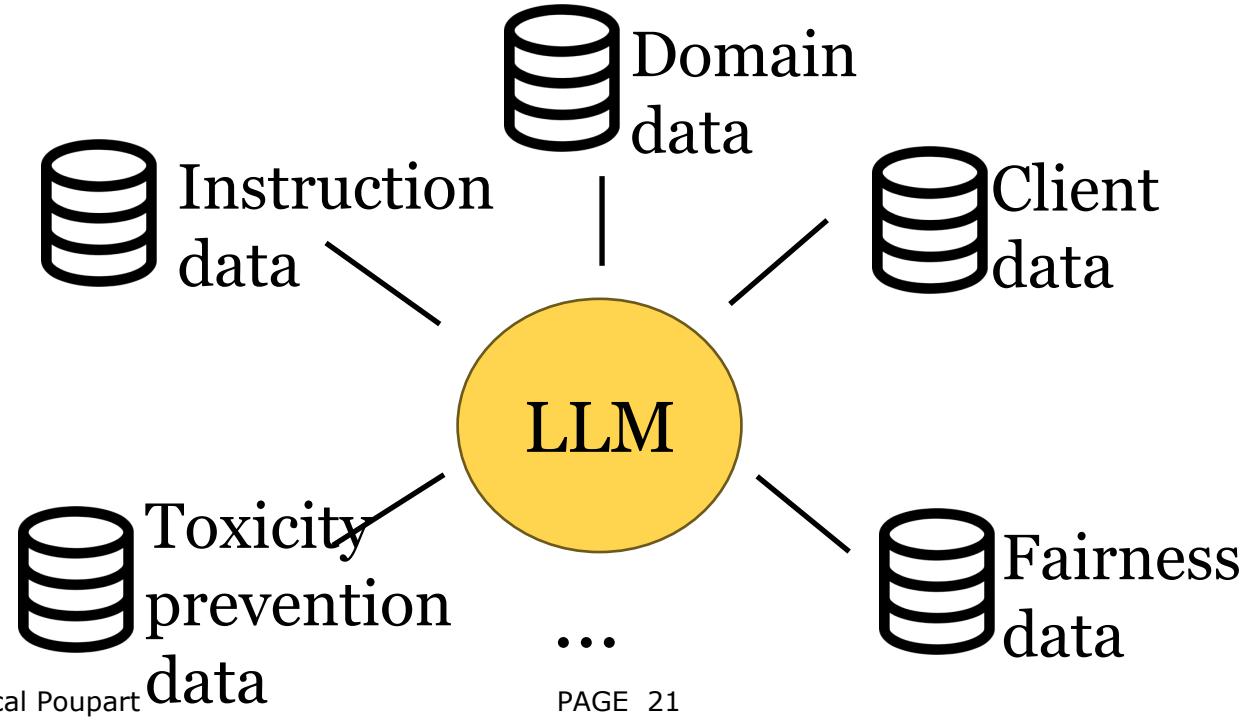
Was this response better or worse? Better Worse Same

LLM Alignment with Preference Data

- Collect preference data: $D = \{(\boldsymbol{s}, \boldsymbol{a}_+, \boldsymbol{a}_-)_k\}_{k=1}^{K}$

  where $\boldsymbol{s}$: user prompt $\boldsymbol{a}$: system response

  $\boldsymbol{a}_+$ is preferred to $\boldsymbol{a}_-$ (i.e., $\boldsymbol{a}_+ \succ \boldsymbol{a}_-$)



Reward Model

Stiennon, Ouyang, Wu, Ziegler, Lowe Voss, Radford, Amodei, Christiano (2020) **Learning to summarize from human feedback**, *NeurIPS.*

- Reward function: $\boldsymbol{r_\theta(s, a)} = real\ number$
- Consider several possible responses $\boldsymbol{a_1} \succcurlyeq \boldsymbol{a_2} \succcurlyeq \cdots \succcurlyeq \boldsymbol{a_k}$ ranked by annotator
- Training reward function to be consistent with the ranking:

$$Loss(\theta) = -\frac{1}{\binom{k}{2}} E_{(s,a_i,a_j)\in Dataset} \log \sigma\left(r_\theta(\boldsymbol{s}, \boldsymbol{a}_i) - r_\theta\left(\boldsymbol{s}, \boldsymbol{a}_j\right)\right)$$

Reinforcement Learning

Ouyang, Wu, Jiang, Wainwright, et al. (2022) **Training language models to follow instructions with human feedback**, *NeurIPS*.

- Pretrain language model (GPT-3)
- Fine-Tune GPT-3 by RL to obtain InstructGPT
  - Policy (language model): $\pi_\phi(a|s)$
  - Optimize $\pi_\phi(s)$ by Proximal Policy Iteration (PPO)

$$\max_{\phi} E_{s \in Dataset}\left[E_{a \sim \pi_\phi(a|s)}[r_\theta(s,a)] - \beta\, KL\left(\pi_\phi(\cdot\,|s)\,\middle|\,\pi_{ref}(\cdot\,|s)\right)\right]$$

Policy Optimization

Stochastic policy $\pi_\phi(a|s) = \Pr(a|s;\phi)$ parametrized by ϕ.

	Supervised Fine-Tuning	**Reinforcement Learning**
Data	$\{(s_1, a_1^*), (s_2, a_2^*), \dots\}$ (a^* denotes optimal action)	$\{(s_1, a_1, r_1), (s_2, a_2, r_2), \dots\}$ (r denotes reward for s,a pair)
Objective	Maximum likelihood $\max_\phi \sum_n \log \pi_\phi(a_n^*|s_n)$	Maximum expected rewards $\max_\phi \sum_n \gamma^n E_{\pi_\phi}[r_n|s_n, a_n]$
Policy update	$\phi \leftarrow \phi + \alpha \nabla_\phi \log \pi_\phi(a_n^*|s_n)$	$\phi \leftarrow \phi + \alpha \, \boldsymbol{G_n} \nabla_\phi \log \pi_\phi(a_n|s_n)$ where $G_n = \sum_{t=n}^{\infty} \gamma^t r_t$

PAGE 24

REINFORCE Algorithm

REINFORCE(s_0)

Initialize π_ϕ to anything

Loop forever (for each episode)

Generate episode $s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_T, a_T, r_T$ with π_ϕ

Loop for each step of the episode $n = 0, 1, \ldots, T$

$G_n \leftarrow \sum_{t=n}^{T} \gamma^t r_t$

Update policy: $\phi \leftarrow \phi + \alpha\, G_n \nabla_\theta \log \pi_\phi(a_n|s_n)$

Return π_ϕ

Proximal Policy Optimization (PPO)

Initialize π_ϕ and V_w to anything

Loop forever (for each episode)

Generate episode $s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_{N-1}, a_{N-1}, r_{N-1}$ with π_ϕ

Loop for each step of the episode $n = 0, 1, \dots, N-1$

$G_n \leftarrow \sum_{t=n}^{N} \gamma^t r_t$

$A(s_n, a_n) \leftarrow G_n - V_w(s_n)$

Update value function: $w \leftarrow w + \alpha_w A(s_n, a_n) \nabla_w V_w(s_n)$

Update π: optimize by stochastic gradient descent

$$\phi \leftarrow \underset{\tilde{\phi}}{\text{argmax}} \frac{1}{N} \sum_{n=0}^{N-1} \min \left\{ \frac{\pi_{\tilde{\phi}}(a_n|s_n)}{\pi_\phi(a_n|s_n)} A(s_n, a_n), \; clip\left(\frac{\pi_{\tilde{\phi}}(a_n|s_n)}{\pi_\phi(a_n|s_n)}, 1-\epsilon, 1+\epsilon\right) A(s_n, a_n) \right\}$$

Simplifying PPO

Source: Shao, Wang et al. (2024) DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models

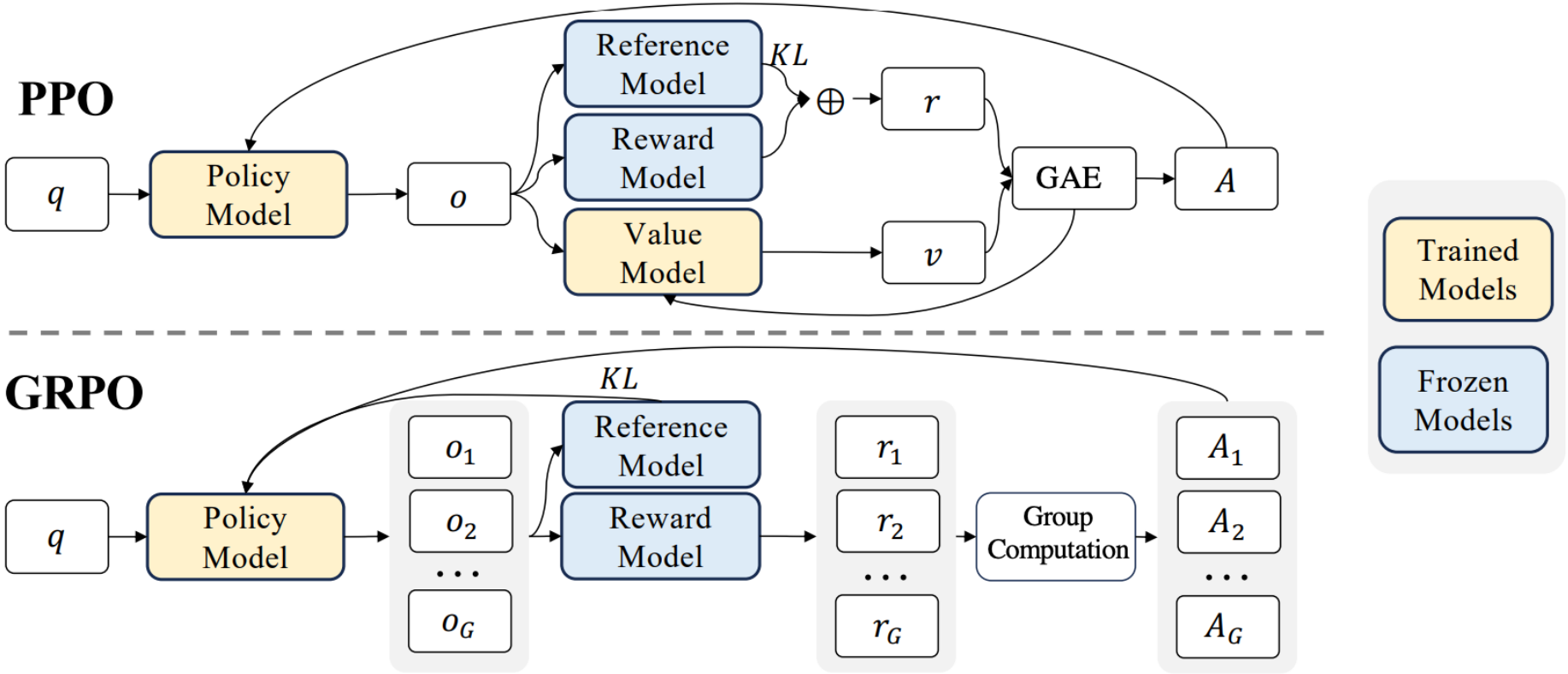


Figure 4 | Demonstration of PPO and our GRPO. GRPO foregoes the value model, instead estimating the baseline from group scores, significantly reducing training resources.

Group Relative Policy Optimization (GRPO)

Initialize π_ϕ and V_w to anything

Loop forever

Generate set of episodes $\{\tau_0, \dots, \tau_{G-1}\}$:

Sample $\tau_g = (s_0^g, a_0^g, r_0^g, s_1^g, a_1^g, r_1^g, \dots, s_{N-1}^g, a_{N-1}^g, r_{N-1}^g)$ with π_ϕ

Evaluate: $R_n^g \leftarrow \sum_{t=n}^{N} \gamma^t \, r\left(s_t^g, a_t^g\right) \; \forall n$

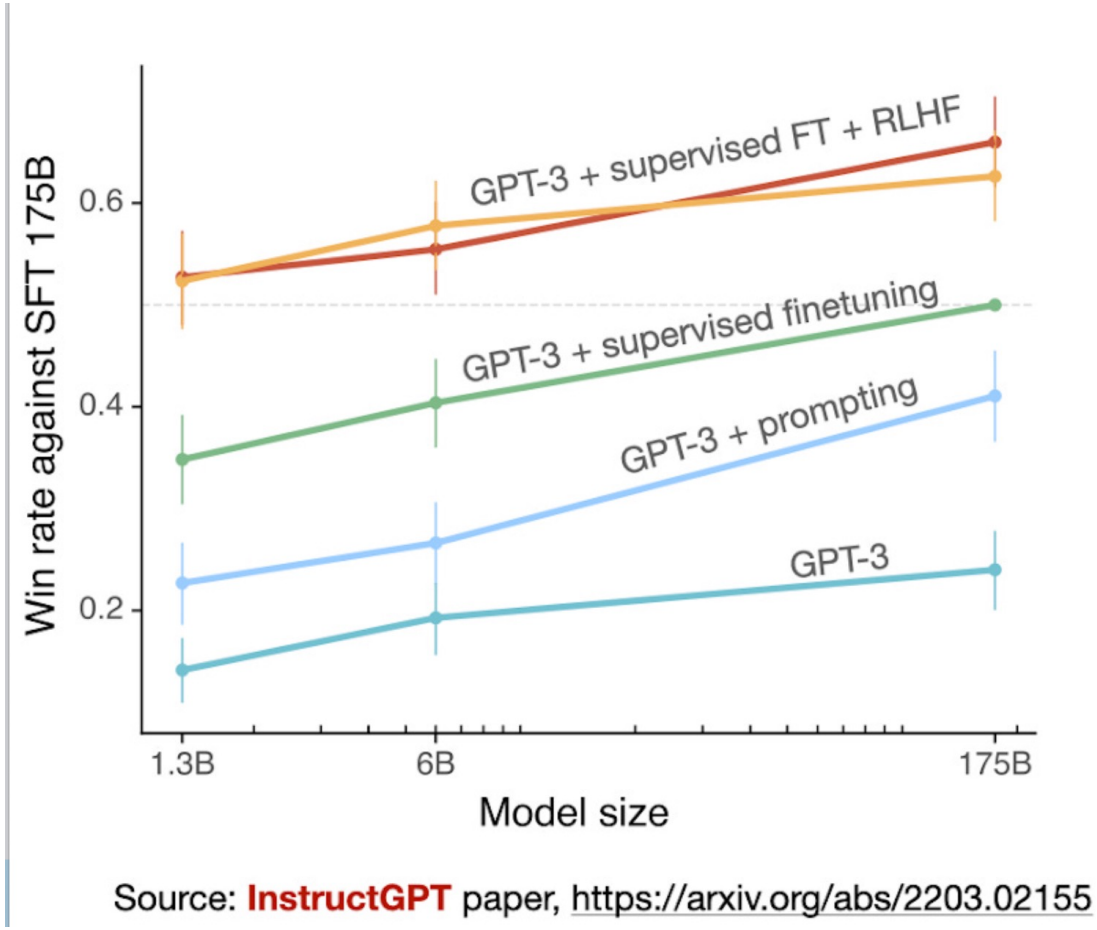
Loop for each episode g and step n

$A_n^g \leftarrow (R_n^g - mean(\{R_n^0, \dots, R_n^{G-1}\}))/std(\{R_n^0, \dots, R_n^{G-1}\})$

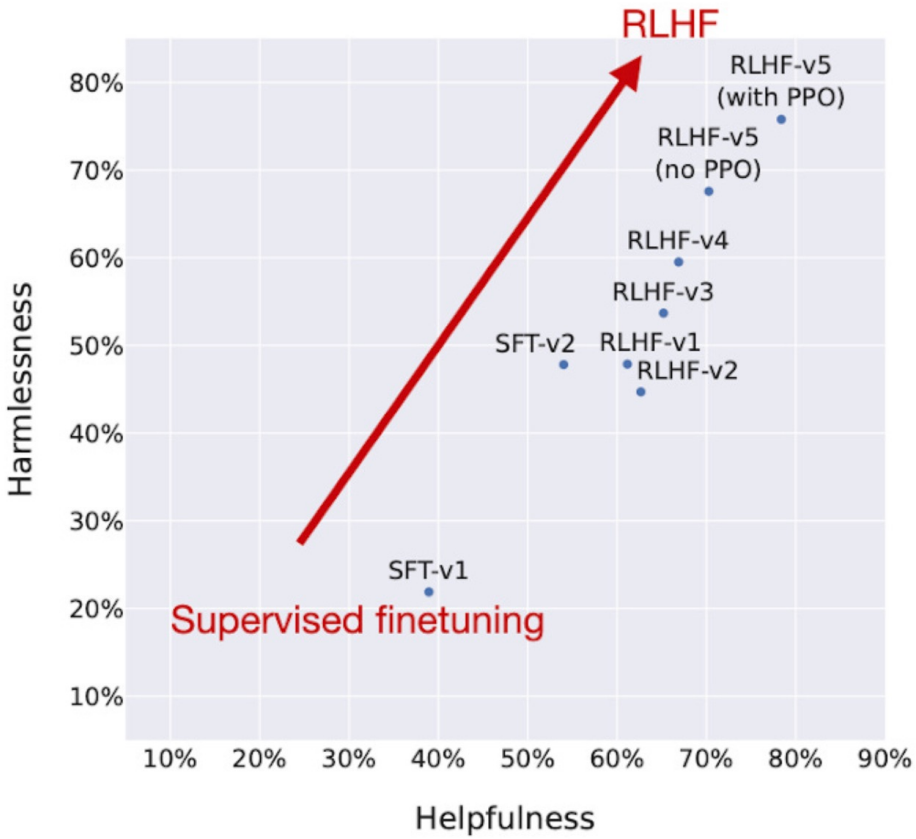
Update π:

$$\phi \leftarrow \underset{\tilde{\phi}}{\text{argmax}} \frac{1}{G} \sum_{g=0}^{G-1} \frac{1}{N} \sum_{n=0}^{N-1} \min \left\{ \begin{array}{c} \frac{\pi_{\tilde{\phi}}\left(a_n^g \middle| s_n^g\right)}{\pi_\phi\left(a_n^g \middle| s_n^g\right)} A_n^g \\ clip\left(\frac{\pi_{\tilde{\phi}}\left(a_n^g \middle| s_n^g\right)}{\pi_\phi\left(a_n^g \middle| s_n^g\right)}, 1-\epsilon, 1+\epsilon\right) A_n^g \end{array} \right\}$$

InstructGPT Results



Source: **InstructGPT** paper, https://arxiv.org/abs/2203.02155



Source: **Llama 2** paper, https://arxiv.org/abs/2307.09288

UNIVERSITY OF WATERLOO