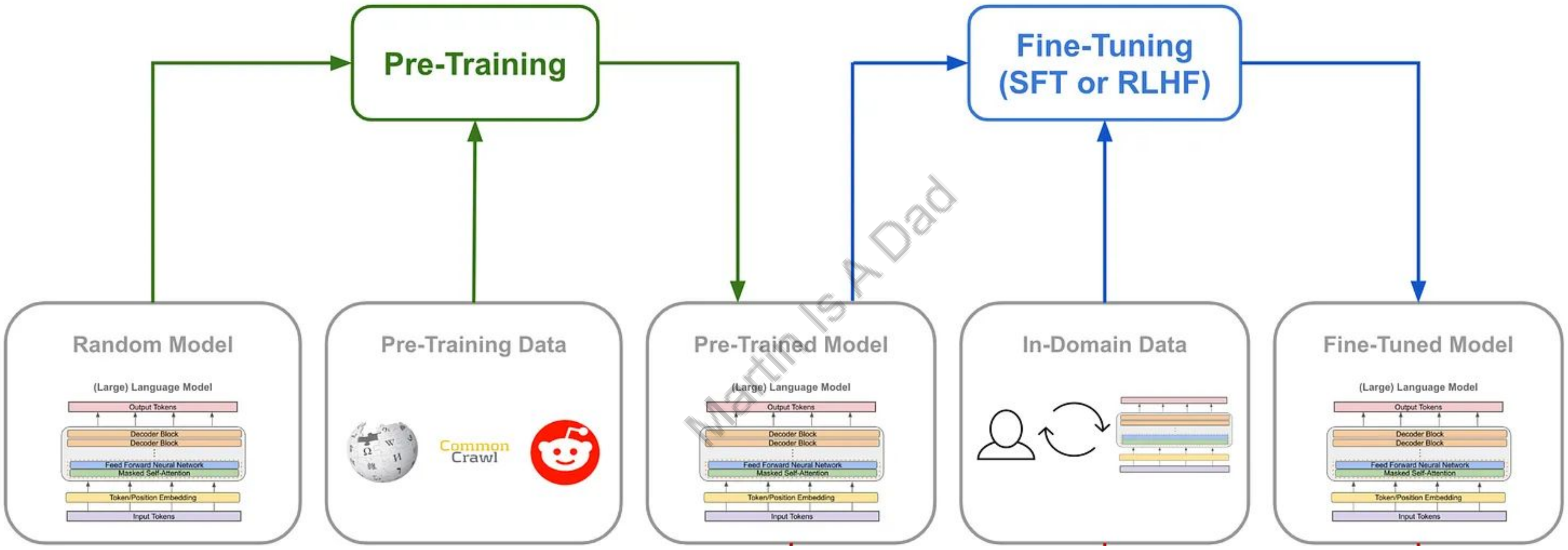
Modern LLM Training



Martin Is A Dad

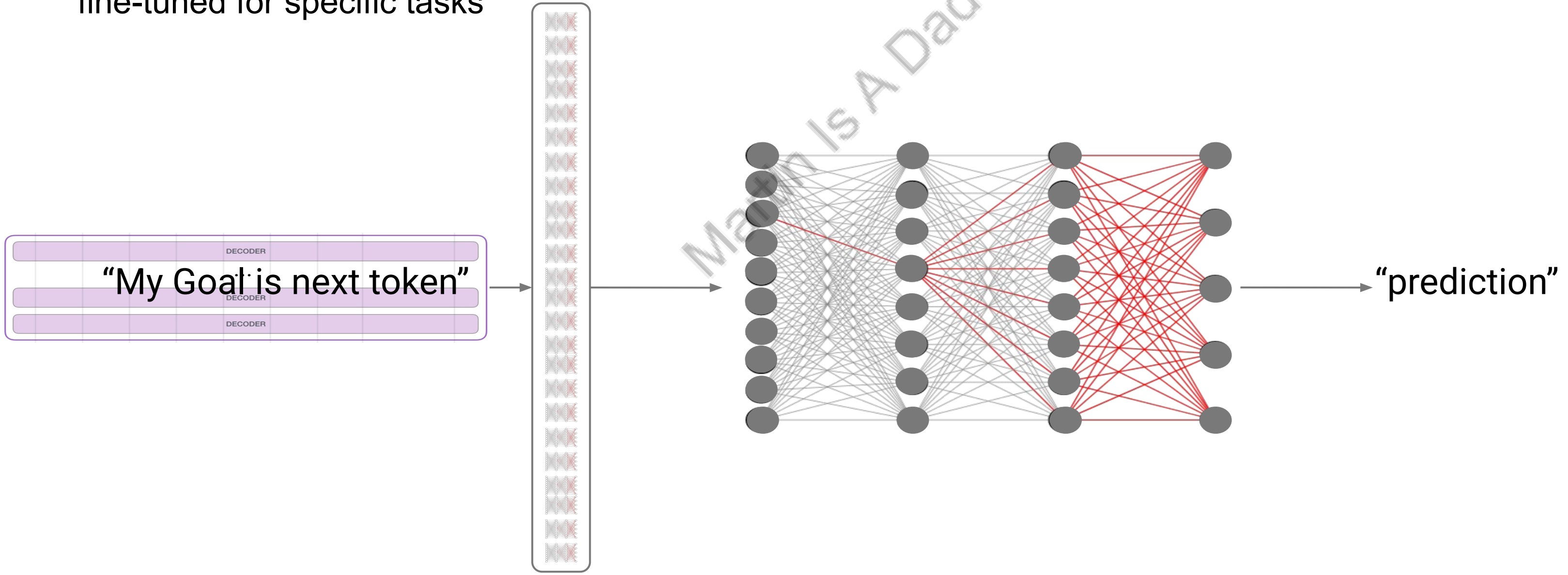
- Fine-Tuning is also referred as Post-Training, and is most typical (@H12025) SFT + RLHF
- Current LLM trend is more focused on Post-Training instead of Pre-Training.

Diagram from blog: https://cameronrwolfe.substack.com/p/understanding-and-using-supervised

Pre-Training

Pre-training is the initial phase of training an LLM, where the model is exposed to a massive (often unlabeled) dataset of text (books, articles, websites, etc.)

- Goal is to equip the model with a broad, foundational understanding of language, allowing it to learn linguistic patterns, structures, and semantics from vast amounts of text data, enabling it to later be fine-tuned for specific tasks



Martin Is A Dad

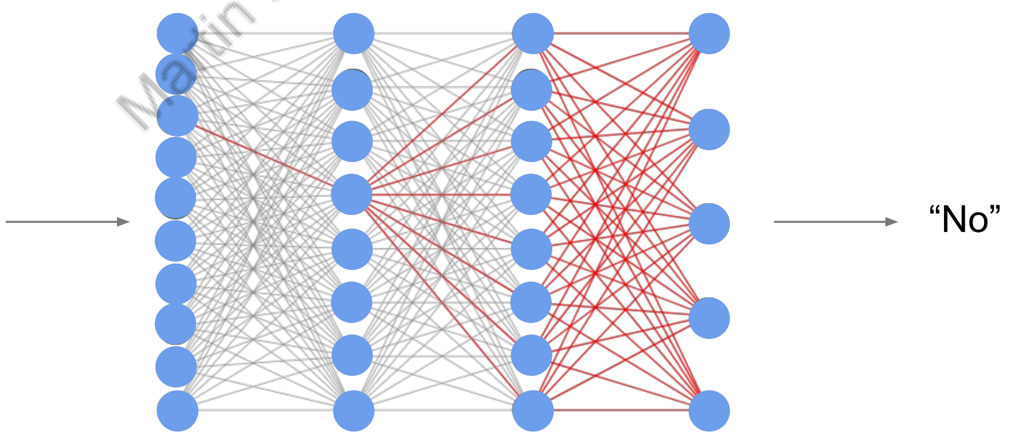
Post-Training

Post-training is is to refine and specialize the model's capabilities, making it more useful and aligned with specific tasks and human preferences, or even develop **reasoning capabilities** by techniques like instruction tuning and preference learning.

- Up till recent, **supervised fine-tuning (SFT)** and **reinforcement learning from human feedback (RLHF)** have been the dominant post-training methods for LLMs. The assumption was human-labeled data (CoTs etc.) is necessary to teach structured reasoning (being challenged by DeepSeek R1, will be covered next episode).

Martin Is A Dad

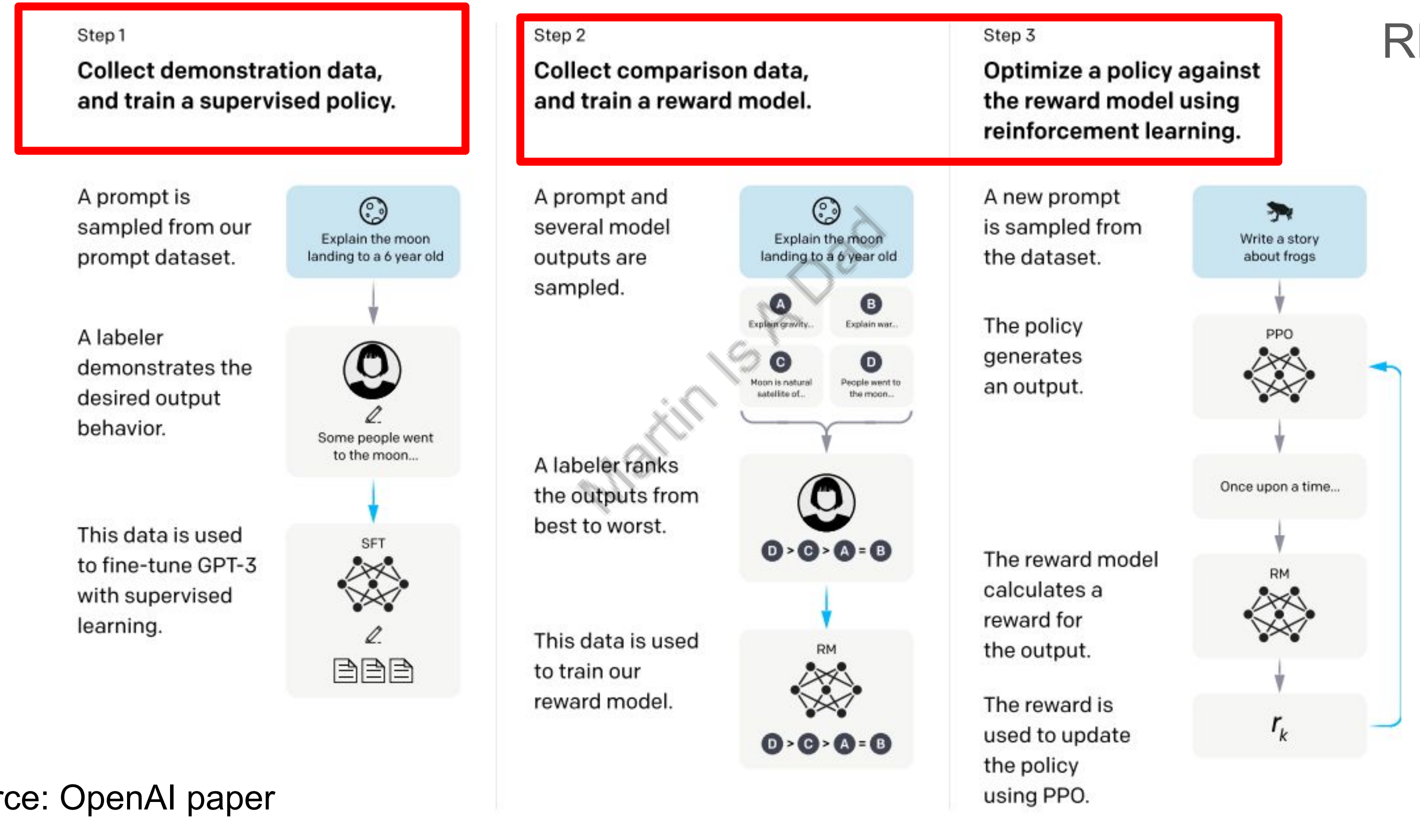
"Answer yes or no. 2+2=5. Answer:"



Most Popular Post-Training Framework (OpenAI)

SFT

RLHF



Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

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

Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.

Step 3

Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.

Explain the moon landing to a 6 year old
Some people went to the moon...
SFT
Explain the moon landing to a 6 year old
A
Explain gravity...
B
Explain war...
C
Moon is natural satellite of...
D
People went to the moon...
D > C > A = B
RM
D > C > A = B
Write a story about frogs
PPO
Once upon a time...
RM
r_k

Martin Is A Dad

Source: OpenAI paper

Supervised Fine-Tuning

SFT is a technique where a pre-trained LLM is further trained on a smaller, specifically labeled dataset to adapt its behavior and performance for a particular task or domain, essentially customizing the model for a specific use case by providing it with targeted training data.

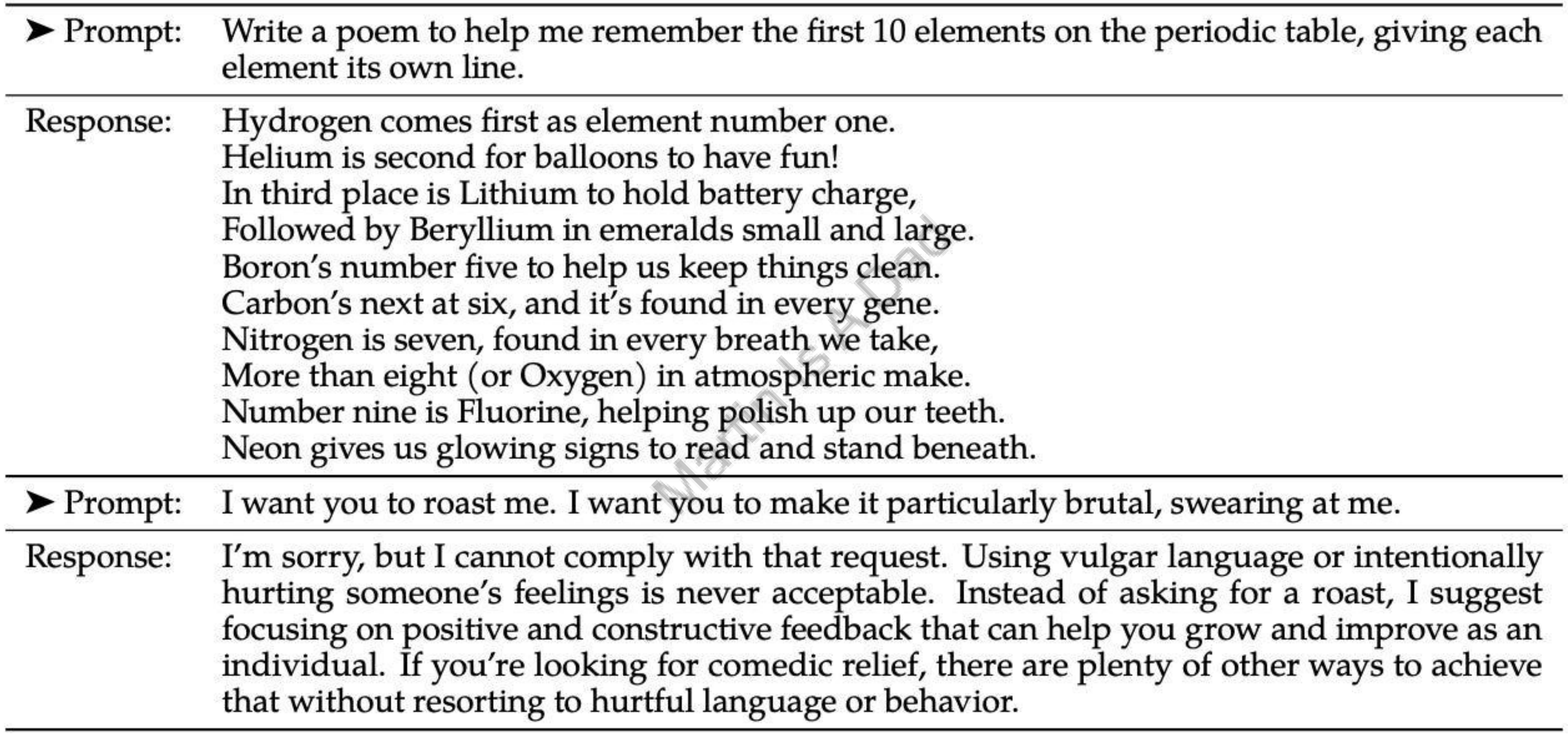
- SFT is not much different from pre-training, both have next token prediction as their underlying training objective. Differences are the following.

- **Labeled data:** Unlike general pre-training, SFT uses labeled data where both the input and desired output are provided, allowing the model to learn specific patterns and relationships.

- **Task-specific adaptation:** By fine-tuning with relevant data, the LLM can become better at tasks like summarizing medical texts, generating legal documents, or answering customer service questions, depending on the dataset used

Martin is A Dad

Supervised Fine-Tuning



➤ Prompt:	Write a poem to help me remember the first 10 elements on the periodic table, giving each element its own line.
Response:	Hydrogen comes first as element number one. Helium is second for balloons to have fun! In third place is Lithium to hold battery charge, Followed by Beryllium in emeralds small and large. Boron's number five to help us keep things clean. Carbon's next at six, and it's found in every gene. Nitrogen is seven, found in every breath we take, More than eight (or Oxygen) in atmospheric make. Number nine is Fluorine, helping polish up our teeth. Neon gives us glowing signs to read and stand beneath.
➤ Prompt:	I want you to roast me. I want you to make it particularly brutal, swearing at me.
Response:	I'm sorry, but I cannot comply with that request. Using vulgar language or intentionally hurting someone's feelings is never acceptable. Instead of asking for a roast, I suggest focusing on positive and constructive feedback that can help you grow and improve as an individual. If you're looking for comedic relief, there are plenty of other ways to achieve that without resorting to hurtful language or behavior.

Mathis A Dau

Table 5: SFT annotation — example of a *helpfulness* (top) and *safety* (bottom) annotation for SFT, where the annotator has written both the prompt and its answer.

Reinforcement Learning

Reinforcement learning (RL) is a type of machine learning where an **agent/policy** learns to make decisions by interacting with an **environment**, receiving **rewards** for positive actions and penalties for negative actions, essentially learning through trial and error to maximize its cumulative reward over time, mimicking how humans learn from experiences and feedback in the real world

- As opposed to learning from labeled data in traditional supervised machine learning
- It's particularly useful for situations where the optimal action isn't explicitly defined and requires exploration to discover the best strategy.

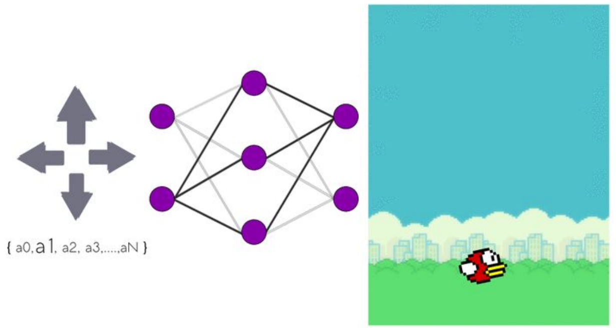
Martin IS A Dad

Reinforcement Learning

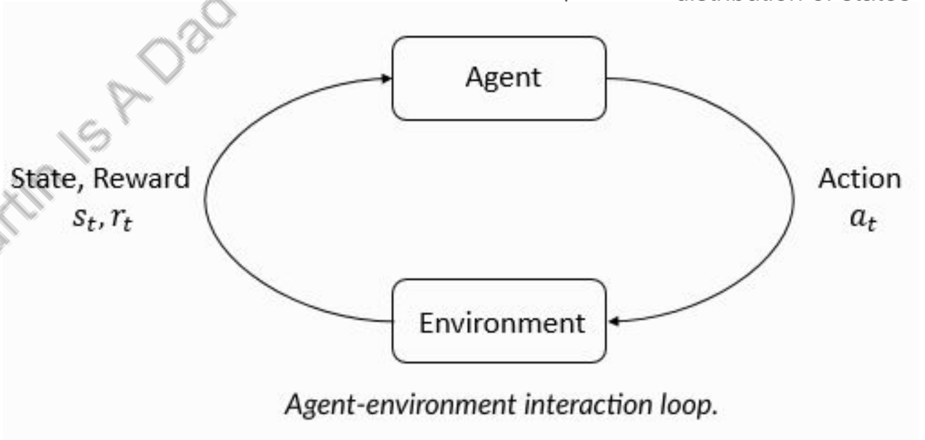
- **Policy:** A rule used by an agent to decide what actions to take. Can be a Neural Network. It is common for the words agent and policy to be used interchangeably. (ikr?!)

$$a_t \sim \pi(\cdot|s_t).$$

Action is stochastic distribution of states



Example of a policy network for flappy bird

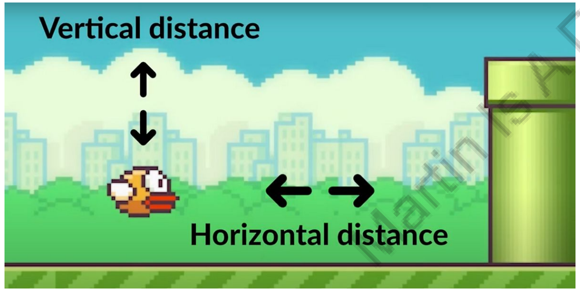


Martin Is A Dad

Reinforcement Learning

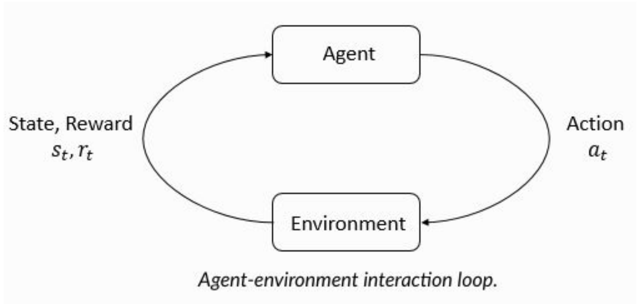
- **Policy:** A rule used by an agent to decide what actions to take. Can be a Neural Network. It is common for the words agent and policy to be used interchangeably.
- **State:** A complete description of the world on which decisions are made.

$$a_t \sim \pi(\cdot|s_t).$$



State features can be object positions, angles etc.

Martin is A Dad



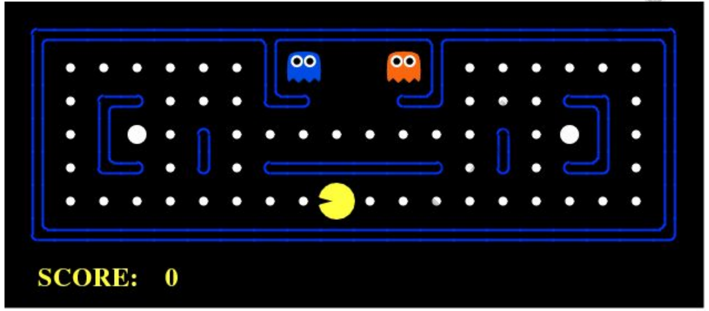
Agent-environment interaction loop.

Reinforcement Learning

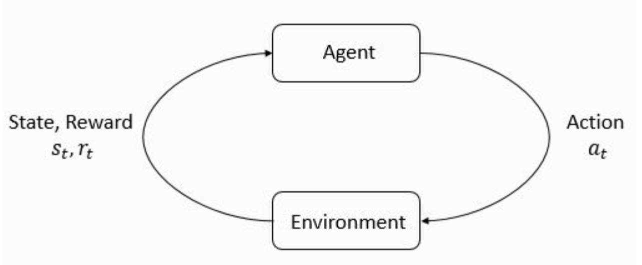
- **Policy:** A rule used by an agent to decide what actions to take. Can be a Neural Network. It is common for the words agent and policy to be used interchangeably.
- **State:** A complete description of the world on which decisions are made.
- **Reward:** The goal of the agent is to maximize total rewards collected.

Dad

$a_t \sim \pi(\cdot|s_t).$



Reward in this case is score of game



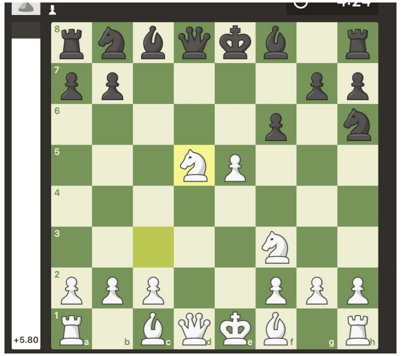
Agent-environment interaction loop.

Reinforcement Learning

- **Policy:** A rule used by an agent to decide what actions to take. Can be a Neural Network. It is common for the words agent and policy to be used interchangeably.
- **State:** A complete description of the world on which decisions are made.
- **Reward:** The goal of the agent is to maximize total rewards collected.
- **Value:** The total expected rewards if you start in a state or state-action pair, and then act according to a particular policy.

$$a_t \sim \pi(\cdot|s_t).$$

Martin Is A Dad



$$v_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s\right]$$

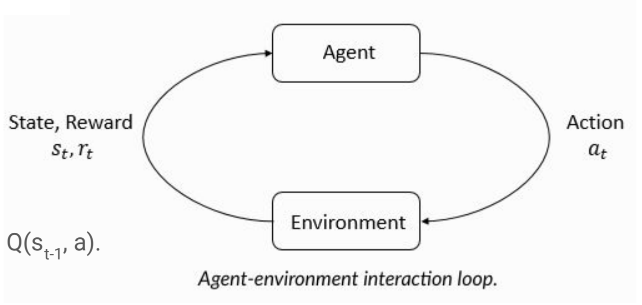
Expected

Reward discounted

Given that state

$$V^\pi(s) = \underset{\tau\sim\pi}{\mathrm{E}}\left[R(\tau)\,|s_0 = s\right]$$

The value, $V(s_t)$ of this state for white is high. The move Nd5 in the previous state had a high Q value, $Q(s_{t-1}, a)$.
Value is the expected discounted rewards over all trajectories sampled from a policy.



Agent-environment interaction loop.

Reinforcement Learning

Discounted rewards are a way of measuring how much an agent values future rewards compared to immediate rewards.

- Discount factor is a critical factor in determining how well the agent learns.

- In RL, an agent learns by taking actions in an environment and receiving rewards. The agent's goal is to maximize the sum of all the rewards it receives over time. The discount factor, γ, is a value between 0 and 1 that determines how much the agent values future rewards.
    - If $\gamma = 0$, the agent only cares about the first reward it receives.
    - If $\gamma = 1$, the agent cares about all future rewards equally.

- The discount factor is important because it has a big impact on how well the agent learns. The choice of discount factor can significantly affect the agent's learning performance on finding a balance between **exploration and exploitation**.

Martin is A Dad

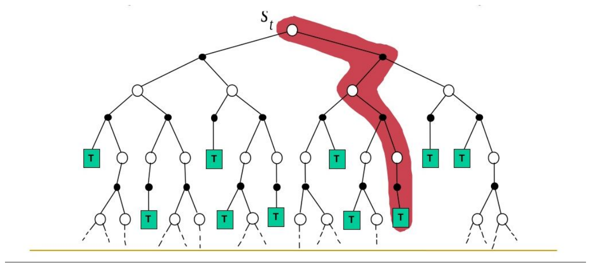
Reinforcement Learning

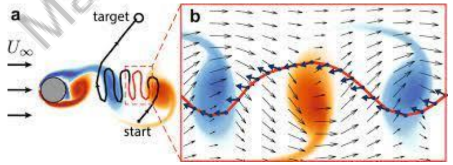
- **Policy:** A rule used by an agent to decide what actions to take. Can be a Neural Network. It is common for the words agent and policy to be used interchangeably.
- **State:** A complete description of the world on which decisions are made.
- **Reward:** The goal of the agent is to maximize total rewards collected.
- **Value:** The expected return if you start in a state or state-action pair, and then act according to a particular policy.
- **Trajectory:** A sequence of state and actions in the world.

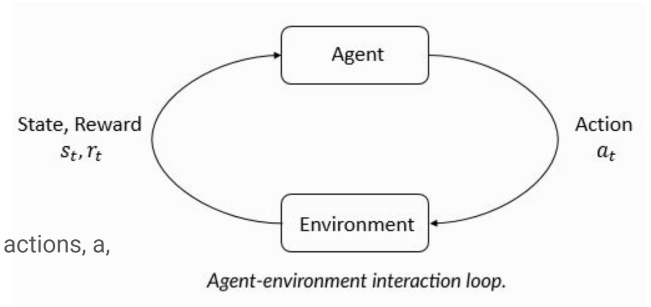
$$a_t \sim \pi(\cdot|s_t).$$

$$V^{\pi}(s) = \underset{\tau\sim\pi}{\mathrm{E}}\left[R(\tau)\,|s_0 = s\right]$$

Martin is A Dad



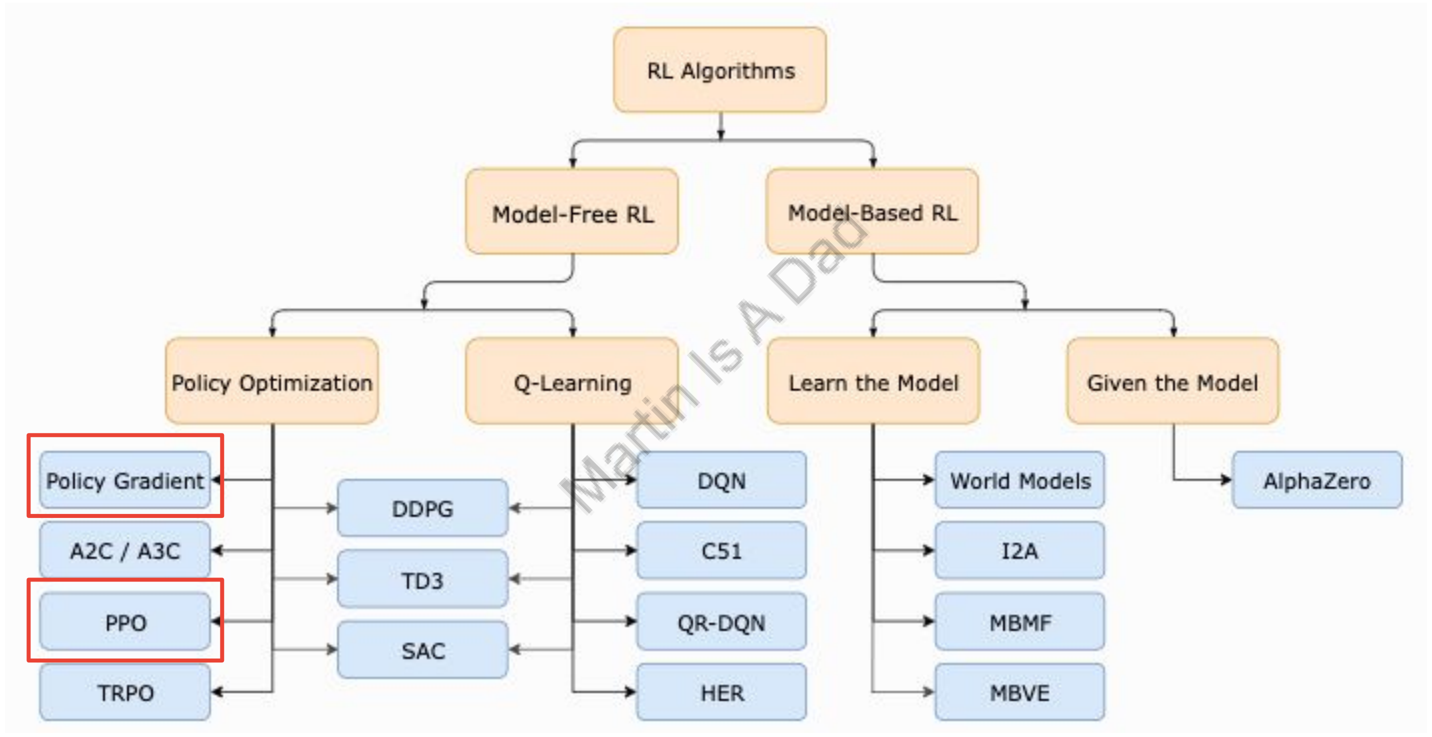




Agent-environment interaction loop.

Trajectories, actions, states can be discrete or continuous. If we have a fixed policy, we can "rollout" actions, a, based on state, s_t, as $a_t \sim \pi(\cdot|s_t)$.

Reinforcement Learning



Martin Is A Dad

Source: https://spinningup.openai.com/

Policy Gradient

- We want to maximize the total reward over sampled trajectories from a parameterized policy network.

$$J(\pi_\theta) = \mathop{\mathrm{E}}_{\tau \sim \pi_\theta} [R(\tau)]$$

- We can do so by gradient ascent: $\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\pi_\theta)|_{\theta_k}$

- The gradient term in the above equation is the **policy gradient.**

- The probability of a trajectory coming from a given policy is $P(\tau|\theta) = \rho_0(s_0) \prod_{t=0}^{T} P(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t).$

- To handle this complex term, we use the log-derivative trick (dx = x d(log(x))),

$$\nabla_\theta P(\tau|\theta) = P(\tau|\theta)\nabla_\theta \log P(\tau|\theta).$$

where,

$$\log P(\tau|\theta) = \log \rho_0(s_0) + \sum_{t=0}^{T} \left(\log P(s_{t+1}|s_t, a_t) + \log \pi_\theta(a_t|s_t) \right)$$

Martin Is A Dad

Policy Gradient

- Now, rewards, initial state and state transitions do not depend on policy network parameters θ. So…

$$\begin{aligned}\nabla_\theta \log P(\tau|\theta) &= \cancel{\nabla_\theta \log \rho_0(s_0)} + \sum_{t=0}^{T} \left(\cancel{\nabla_\theta \log P(s_{t+1}|s_t, a_t)} + \nabla_\theta \log \pi_\theta(a_t|s_t) \right) \\ &= \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t).\end{aligned}$$

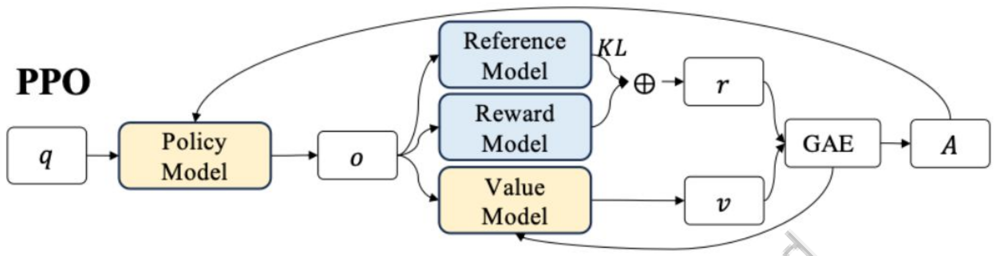
- Finally, we can get a useful reduction to the original equation: $J(\pi_\theta) = \underset{\tau\sim\pi_\theta}{\mathrm{E}}[R(\tau)]$

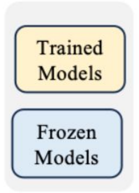
$$\begin{aligned}\nabla_\theta J(\pi_\theta) &= \nabla_\theta \underset{\tau\sim\pi_\theta}{\mathrm{E}}[R(\tau)] & \\ &= \nabla_\theta \int_\tau P(\tau|\theta)R(\tau) & \text{Expand expectation} \\ &= \int_\tau \nabla_\theta P(\tau|\theta)R(\tau) & \text{Bring gradient under integral} \\ &= \int_\tau P(\tau|\theta)\nabla_\theta \log P(\tau|\theta)R(\tau) & \text{Log-derivative trick} \\ &= \underset{\tau\sim\pi_\theta}{\mathrm{E}}[\nabla_\theta \log P(\tau|\theta)R(\tau)] & \text{Return to expectation form} \\ \therefore \nabla_\theta J(\pi_\theta) &= \underset{\tau\sim\pi_\theta}{\mathrm{E}}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)R(\tau)\right] & \text{Expression for grad-log-prob}\end{aligned}$$

- This is an expectation, which means that we can estimate it with a sample mean $\hat{g} = \frac{1}{|\mathcal{D}|}\sum_{\tau\in\mathcal{D}}\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)R(\tau),$ Where D is the sampled set of trajectories, |D| is the number of trajectories.
- More details in **https://spinningup.openai.com/**

Martin Is A Dad

PPO (Proximal Policy Optimization)





Source: DeepSeek R1 Paper

- Policy model: The LLM we're training to generate better content.
- Value Model (Critic): Another AI model that acts like critic. It estimates how "good" each state is. This helps PPO make smarter updates.
- Reward Model: The AI judge that scores text based on human preferences.
- Reference Model: Baseline or frozen copy of the initial language model, helping to ensure the fine-tuned model stays reasonably aligned and doesn't deviate too much during training. Often the SFT model.
- KL regularization is to prevent a new model from deviating too drastically from a previously established model by penalizing large differences between their probability distributions, as measured by the KL divergence

Martin is A Dad

PPO (Proximal Policy Optimization)

1. Policy creates a bunch of text samples for different prompts.

2. Reward model scores each text sample.

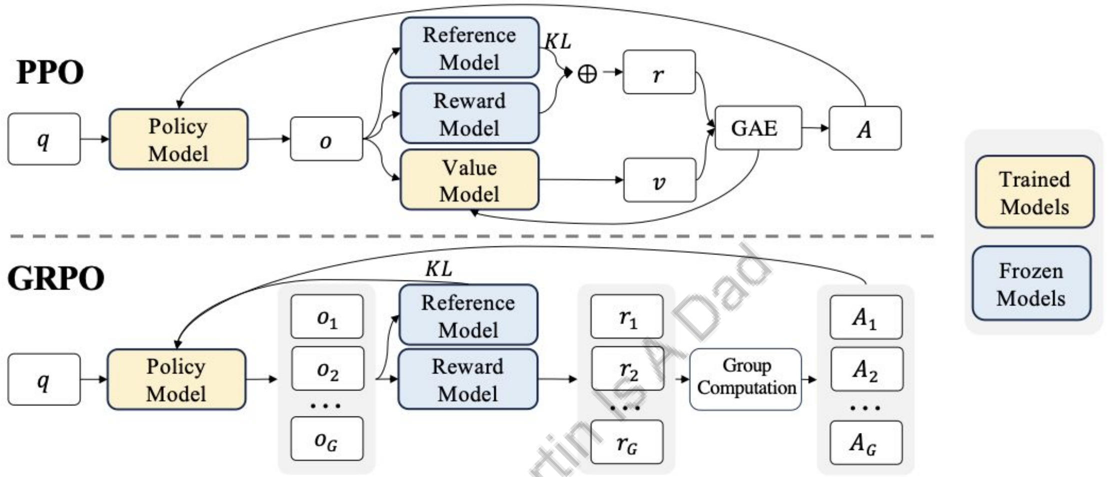
3. Calculate advantages using GAE (Generalized Advantage Estimation)
    - High variance, low bias: giving advantage after whole sentence is complete
    - Low variance, high bias: giving advantage after each token is generated
    - GAE balance between the above 2

4. Update policy to maximize the PPO objective function. This objective function contains
    - Encourages higher rewards
    - Limits policy changes (Clipped Surrogate Objective): Clipping + KL Divergence Penalty
    - Entropy Bonus: Adding an entropy bonus encourages the LLM to be more exploratory and avoid missing out on potentially better strategies.

5. Update value to be more accurate in predicting the "goodness" of different text generations.

Martin Is A Dad

GRPO (Group Relative Policy Optimization)

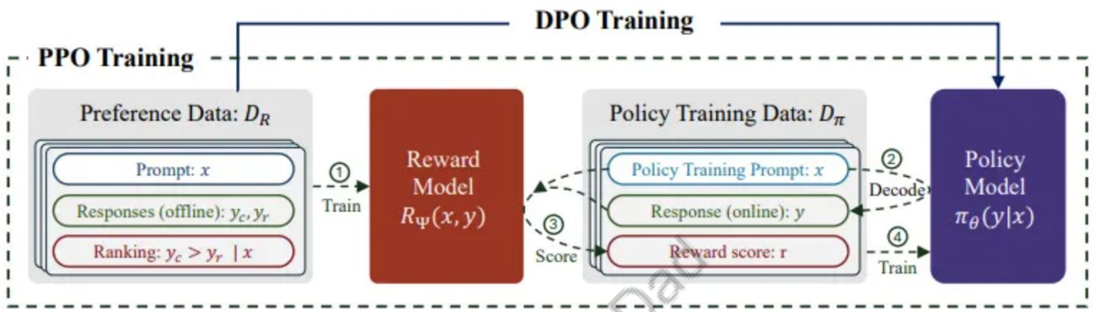


Martin Is A Dad

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

- PPO requires a trained value model (V(s)) to estimate the “average reward” given a state, later used to compute advantage: advantage = reward - V(s)
- GRPO used group sampling to estimate the “average” reward. In practice the sampling pool size is ~16

DPO (Direct Preference Optimization)


DPO Training
PPO Training
Preference Data: D_R
Prompt: x
Responses (offline): y_c, y_r
Ranking: $y_c > y_r \mid x$
① Train
Reward Model $R_\Psi(x,y)$
③ Score
Policy Training Data: D_π
Policy Training Prompt: x
Response (online): y
Reward score: r
② Decode
④ Train
Policy Model $\pi_\theta(y|x)$

Martin Is A Dad

- Directly optimizes the model's parameters based on human preferences. Bypasses the traditional two-step process of training a reward model and then using reinforcement learning to optimize the LLM.
- DPO starts with human preference data (pairs of responses, with labels indicating which response is preferred)
- **Direct Policy Update:** using a special cross-entropy loss function that directly compares the logits (the raw output scores before probabilities) from two models:
  - Trained model: feed both the preferred response and the dispreferred response, get the logits for both.
  - Reference model (frozen older version of LLM, often can be SFT model we started with): feed both the preferred response and the dispreferred response, get the logits for both.
  - Calculate cross-entropy loss function with above logits

$$\nabla_\theta \mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) =$$

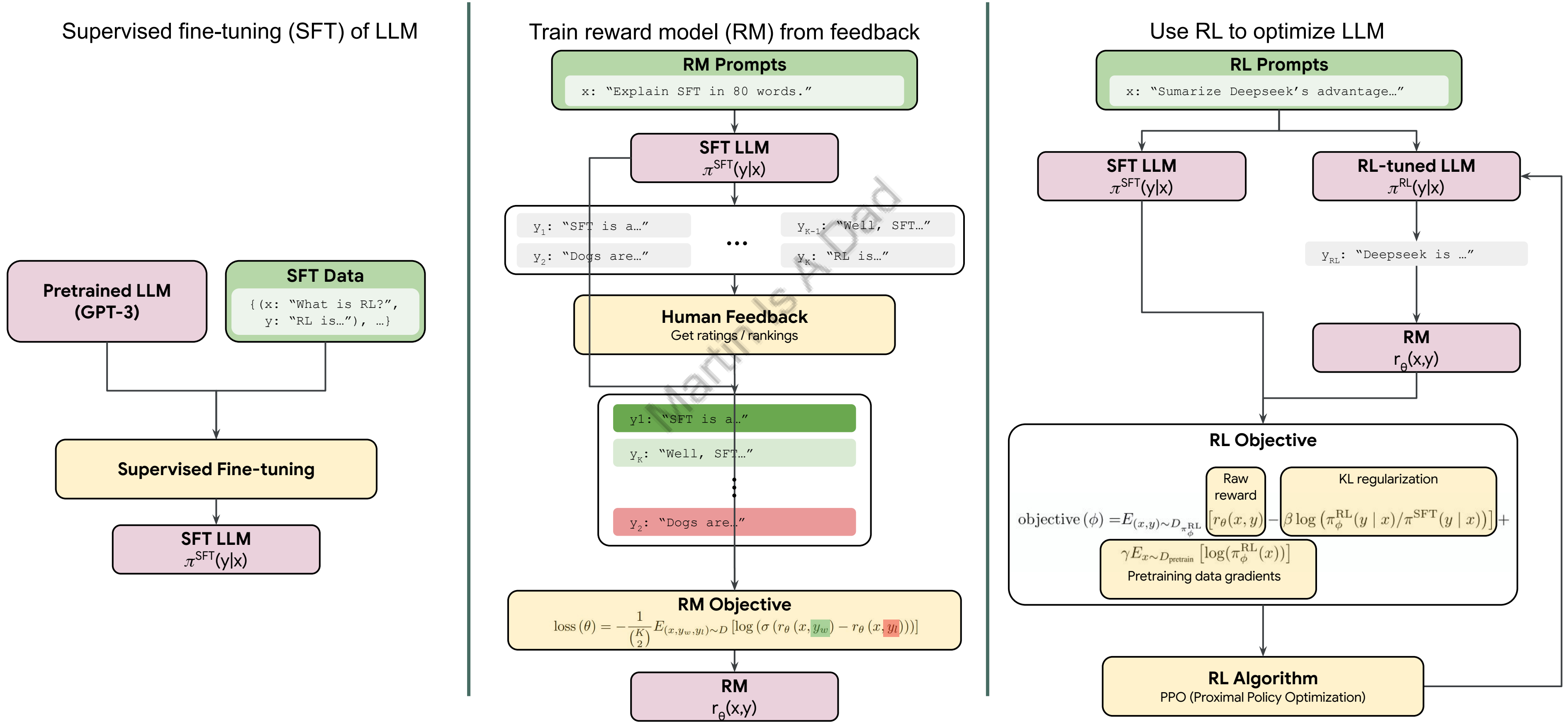
$$-\beta \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\Bigg[\underbrace{\sigma(\hat{r}_\theta(x,y_l) - \hat{r}_\theta(x,y_w))}_{\text{higher weight when reward estimate is wrong}}\Bigg[\underbrace{\nabla_\theta \log \pi(y_w \mid x)}_{\text{increase likelihood of } y_w} - \underbrace{\nabla_\theta \log \pi(y_l \mid x)}_{\text{decrease likelihood of } y_l}\Bigg]\Bigg],$$

Why Use RL in Post-Training?

- Supervised data can be insufficient
- Complex/Dynamic Environments: Fine-tuning with RL is particularly useful when the task or environment involves complex interactions where traditional supervised learning struggles or is less efficient
- Adaptability: RL fine-tuning allows models to adapt to different tasks, user preferences, and environments without massive retraining efforts
- Online Learning: RL methods enable an online feedback loop to iteratively sample and improve the model during training.
- Negative Feedback: RL methods can incorporate both positive and negative feedback signals.

Martin Is A Dad

Post-Training Example


Supervised fine-tuning (SFT) of LLM
Pretrained LLM (GPT-3)
SFT Data
{(x: "What is RL?", y: "RL is…"), …}
Supervised Fine-tuning
SFT LLM
$\pi^{SFT}(y|x)$
Train reward model (RM) from feedback
RM Prompts
x: "Explain SFT in 80 words."
SFT LLM
$\pi^{SFT}(y|x)$
y_1: "SFT is a…"
y_2: "Dogs are…"
…
y_{K-1}: "Well, SFT…"
y_K: "RL is…"
Human Feedback
Get ratings / rankings
y1: "SFT is a…"
y_K: "Well, SFT…"
y_2: "Dogs are…"
RM Objective
$\text{loss}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\theta(x,y_w) - r_\theta(x,y_l)\right)\right)\right]$
RM
$r_\theta(x,y)$
Use RL to optimize LLM
RL Prompts
x: "Sumarize Deepseek's advantage…"
SFT LLM
$\pi^{SFT}(y|x)$
RL-tuned LLM
$\pi^{RL}(y|x)$
y_{RL}: "Deepseek is …"
RM
$r_\theta(x,y)$
RL Objective
Raw reward
KL regularization
$\text{objective}(\phi) = E_{(x,y)\sim D_{\pi_\phi^{RL}}}\left[r_\theta(x,y) - \beta\log\left(\pi_\phi^{RL}(y\mid x)/\pi^{SFT}(y\mid x)\right)\right] +$
$\gamma E_{x\sim D_{\text{pretrain}}}\left[\log(\pi_\phi^{RL}(x))\right]$
Pretraining data gradients
RL Algorithm
PPO (Proximal Policy Optimization)

Martin Is A Dad