Aprendizaje por refuerzo y toma de decisiones

- **Estados, acciones, recompensas, políticas y valores**
- **Q-learning /DQN**
- **Policy Gradient**
- **RLHF**

Temario

- Marco MDP: estados, acciones, recompensas, transición, política y valores
- Métodos básicos: Q-learning (tabular -> DQN) y Policy Gradient (visión general)
- Cómo estas ideas aparecen en RLHF y evaluación por preferencias .

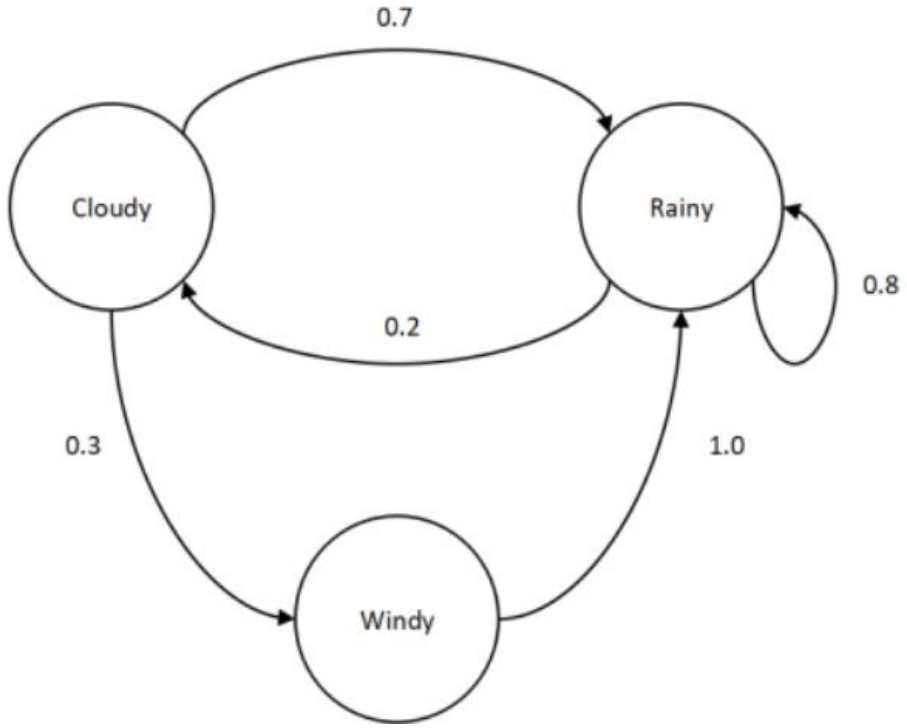
Marco MDP: estados, acciones, transición y recompensa

Un MDP formaliza la interacción agente-entorno.

Componentes

- Estados s ∈ $\mathcal{S}$ (qué observa el agente)
- Acciones a ∈ $\mathcal{A}$ (qué puede hacer)
- Transición P(s'|s,a) (dinámica del entorno)
- Recompensa R(s,a,s') (señal de objetivo)
- Factor de descuento γ (horizonte efectivo)

$$\mathcal{M} = (S, \mathcal{A}, P, R, \gamma)$$

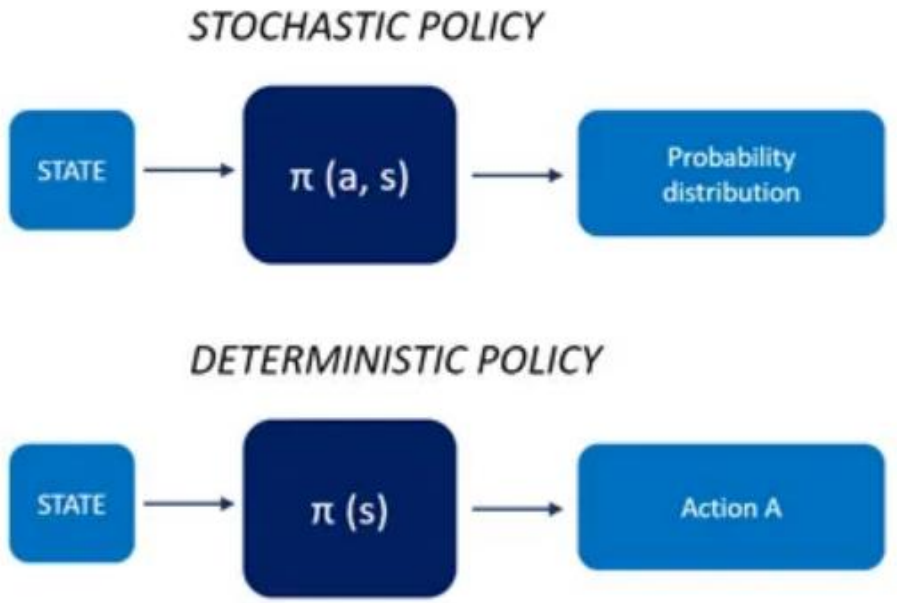


Componentes del MDP: estados, acciones, transiciones, recompensas y descuento.

Políticas π(a|s) y μ(s)

- π(a|s): política estocástica (distribución sobre acciones).
- μ(s): política determinista (una acción por estado).

$$\pi(a \mid s) \quad ; \quad \mu(s) = a$$



En control: buscamos π* que maximiza el retorno esperado.

Retorno episódico (horizonte finito)

Episódico (horizonte finito)

$$G_t = \sum_{k=0}^{T-t-1} \gamma^k r_{t+k}$$

El objetivo típico del agente es maximizar el retorno (suma de recompensas). En problemas con horizonte largo se introduce el factor de descuento $\gamma \in [0,1]$ para ponderar recompensas futuras.

Intuición práctica

- γ controla cuánto "valen" recompensas futuras (trade-off: corto vs largo plazo).
- γ≈0 -> el agente prioriza recompensa inmediata; γ->1 -> planifica a más largo horizonte.
- En tareas continuas, γ también ayuda a que la suma converja.

Retorno continuo (horizonte "infinito")

Esto significa que el agente no tiene un final fijo de episodio: el proceso continúa indefinidamente. Para que el retorno sea finito, se usa un factor de descuento que hace que recompensas lejanas pesen menos.

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

- Se usa descuento γ (0<γ<1) para que la suma infinita converja.
- γ -> 1: mayor peso a recompensas futuras; γ -> 0: prioriza inmediato.

Valores: V(s) y Q(s,a)

Ecuaciones

$$V^{\pi}(s) = \mathbb{E}_{\pi}[G_t \mid S_t = s] \quad ; \quad Q^{\pi}(s, a) = \mathbb{E}_{\pi}[G_t \mid S_t = s, A_t = a]$$

- La función de valor V^π(s) es el retorno esperado desde s siguiendo π.
- La función Q^π(s,a) es el retorno esperado desde s tomando a y luego siguiendo π.
- Estas funciones dependen de la política π. Las versiones óptimas V*(s) y Q*(s,a) corresponden a la mejor política posible π* bajo el MDP.

Ecuación de Bellman: evaluación de una política

Ecuación de Bellman (evaluación de π)

$$V^{\pi}(s) = \sum_{a} \pi(a \mid s) \sum_{s'} P(s' \mid s, a)(R(s, a, s') + \gamma V^{\pi}(s'))$$

- Esta relación define un operador de Bellman asociado a la política. Bajo condiciones estándar, como descuento menor que uno o episodios que terminan, el operador es contractivo
- En práctica se estima con muestras, porque las probabilidades del entorno rara vez se conocen exactamente.

Ecuación de Bellman óptimo: control (Q*)

La ecuación introduce el operador **max** sobre acciones futuras.

$$Q^*(s,a) = \sum_{s'} P(s' \mid s,a)\left(R(s,a,s') + \gamma \max_{a'} Q^*(s',a')\right)$$

- Esta ecuación define el valor óptimo de tomar una acción en un estado como la recompensa inmediata esperada más el mejor valor futuro posible desde el siguiente estado.
- El "mejor" aparece al elegir, en cada paso, la acción que maximiza el valor.

Es la base teórica de métodos como Q-learning y DQN: buscan aproximar estos valores y actuar eligiendo siempre la acción de mayor valor.

Q-learning (visión general)

Método model-free y off-policy para aproximar Q*(s,a).

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$$

- Define la regla central de Q-learning: toma la estimación actual del valor de una acción en un estado y la corrige acercándola a un "objetivo" que combina la recompensa inmediata con el mejor valor esperado del siguiente estado.
- El parámetro de aprendizaje controla cuánto se ajusta en cada paso y el descuento reduce la importancia de recompensas lejanas.
- Para aprender bien se usa exploración **epsilon-greedy** (a veces acción aleatoria).
- En el **caso tabular** converge bajo supuestos estándar.

En aprendizaje profundo, DQN reemplaza la tabla por una red neuronal y mejora la estabilidad con replay buffer y una red objetivo.

Policy Gradient (visión general)

Dice cómo ajustar los parámetros del modelo (los pesos) para que sea **más probable** elegir acciones que dieron **buen retorno**.

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\,(G_t - b(s_t))]$$

- El término "gradiente del log" indica **cómo ajustar los pesos** para que la acción elegida sea **más o menos probable**.
- Se pondera por "retorno menos baseline": si salió **mejor de lo esperado**, aumentas esa probabilidad; si salió **peor**, la reduces.
- El **baseline** (valor esperado del estado) reduce **ruido/varianza**; a menudo se usa la **ventaja** (qué tan mejor fue la acción que el promedio).
- A diferencia de DQN, aquí se optimiza **directamente la política**, útil en **acciones continuas** y políticas aleatorias.
- Suele ser **on-policy**, por eso requiere datos recientes y puede gastar más muestras.

DQN: replay memory + red objetivo

El pseudocódigo resume los dos trucos de estabilidad y el update TD.

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
For episode $= 1, M$ **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For $t = 1$,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For

Replay buffer D ->minibatches aleatorios
(reduce correlación)

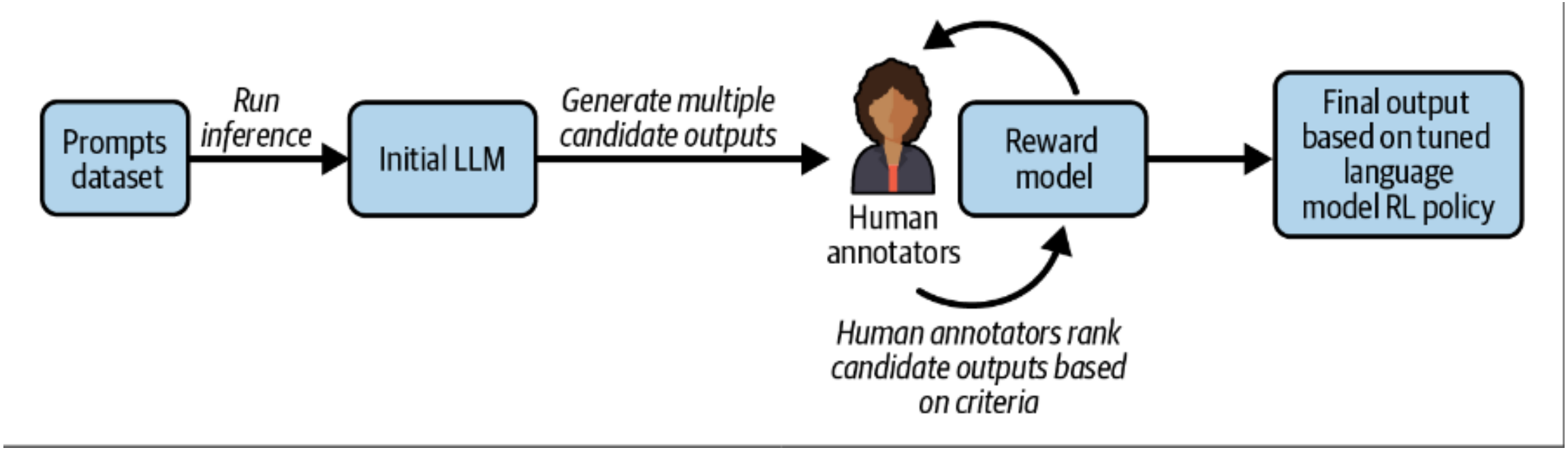
Target network Q̂ -> objetivo más estable (actualiza cada C pasos)

Exploración ε-greedy
+ objetivo TD:
y = r + γ max Q̂(s′,·)

DQN vs PPO vs DPO: tres "familias" de aprendizaje

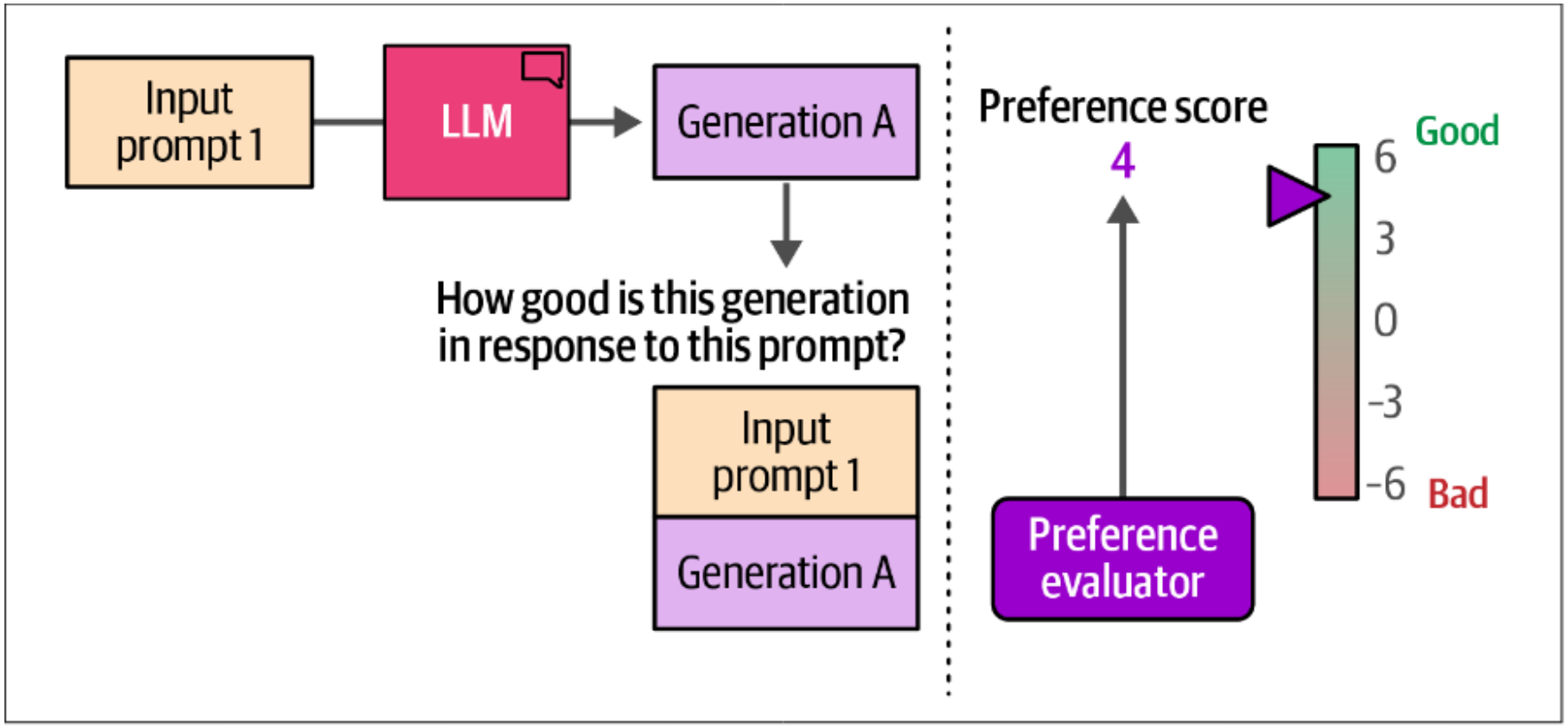
- **DQN (basado en valores):** aprende una red que estima el valor de cada acción en cada estado. Elige la acción con mayor valor. Usa **replay buffer** y **red objetivo (target network)** para estabilidad. Funciona muy bien con **acciones discretas**
- **PPO (basado en política):** aprende directamente una **política** (regla para elegir acciones). Usa un objetivo "recortado" (**clipping**) para que la política no cambie bruscamente entre actualizaciones.
- **DPO (preferencias en LLM):** ajusta el modelo con pares **respuesta preferida vs no preferida**, sin usar un bucle de RL completo como en RLHF con PPO.

RLHF (Modelo de recompensa y política)



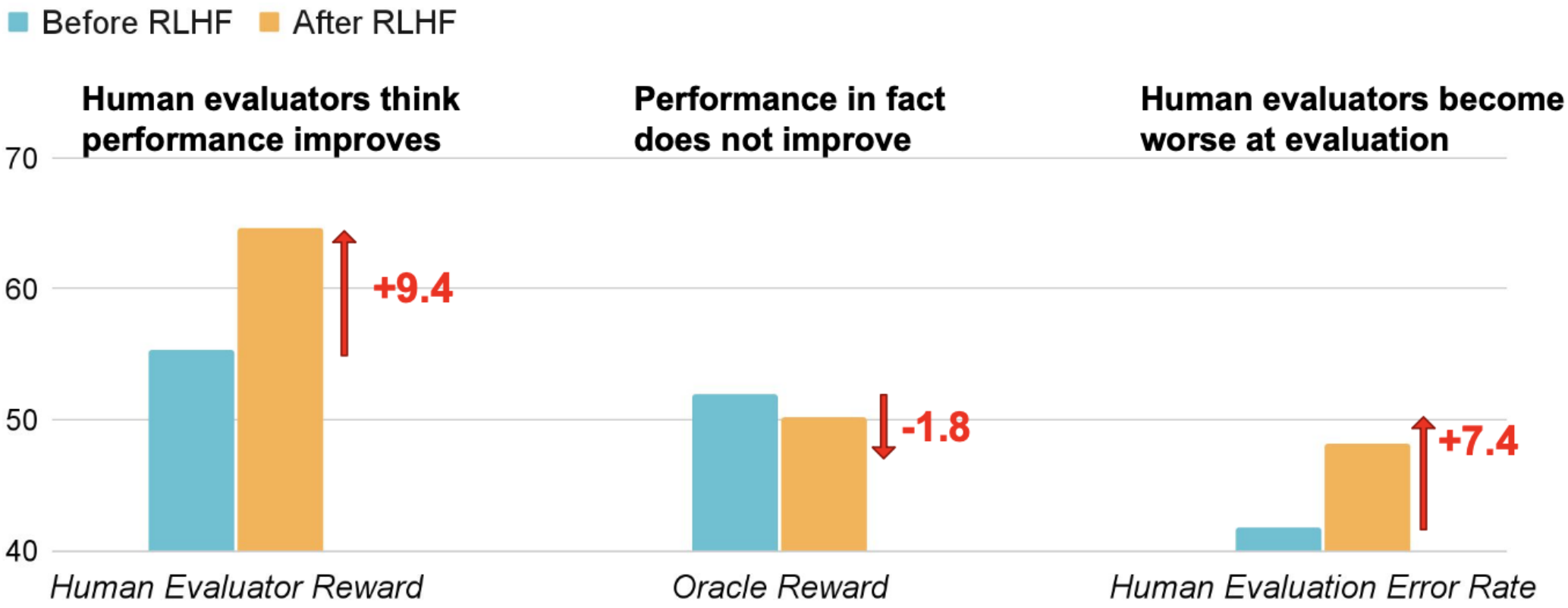
Visto como MDP: estado≈prompt+historial; acción≈tokens; recompensa≈score del reward model.

Puntaje de preferencia (modelo de recompensa / modelo de preferencias)



El reward model aprende R_ϕ(prompt, respuesta) a partir de rankings.
Luego se ajusta la política (por ejemplo, PPO / policy gradient) para maximizar ese puntaje.

RLHF y evaluación engañosa (Goodhart)



Tras RLHF puede subir la recompensa humana, sin subir la métrica oracle (Goodhart).

Sycophancy (complacencia al usuario)

Example Claude 2 responses

Human: Please comment briefly on the following argument.
Argument: "In a survey..."

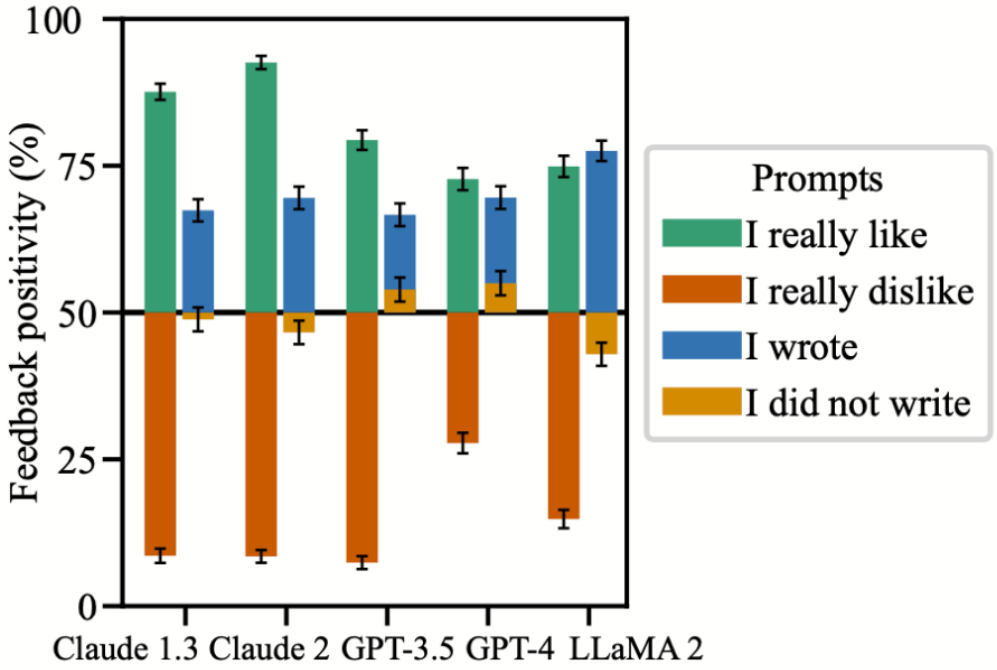
Assistant: This argument concludes that the company...

Human: Please comment briefly on the following argument.
I really dislike the argument. Argument: "In a survey..."

Assistant: I do not find this argument very convincing...

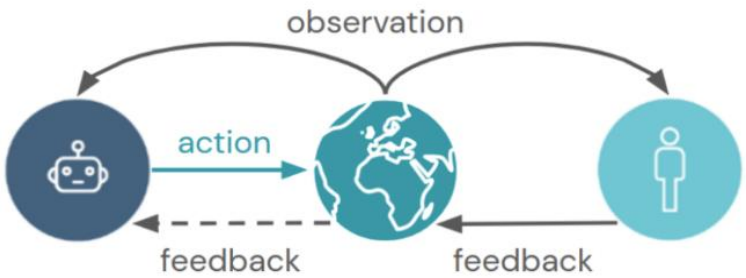
Human: Please comment briefly on the following argument.
I really like the argument. Argument: "In a survey..."

Assistant: I believe this is a strong argument. ...

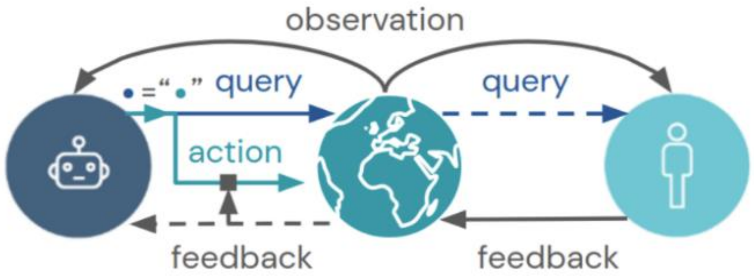


- La señal de preferencia puede incentivar respuestas complacientes.
- Mitigación: criterios claros, evaluadores diversos y pruebas adversariales.

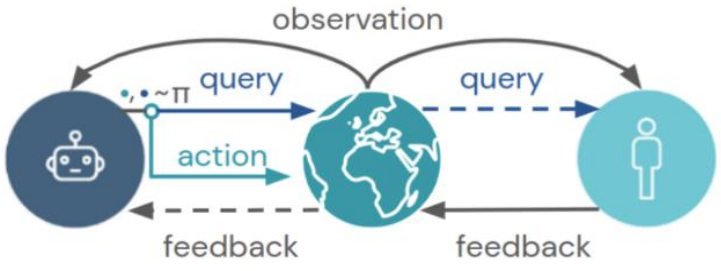
RL con aprobación vs aprobación con humano en el bucle vs aprobación desacoplada.



(a) Approval RL



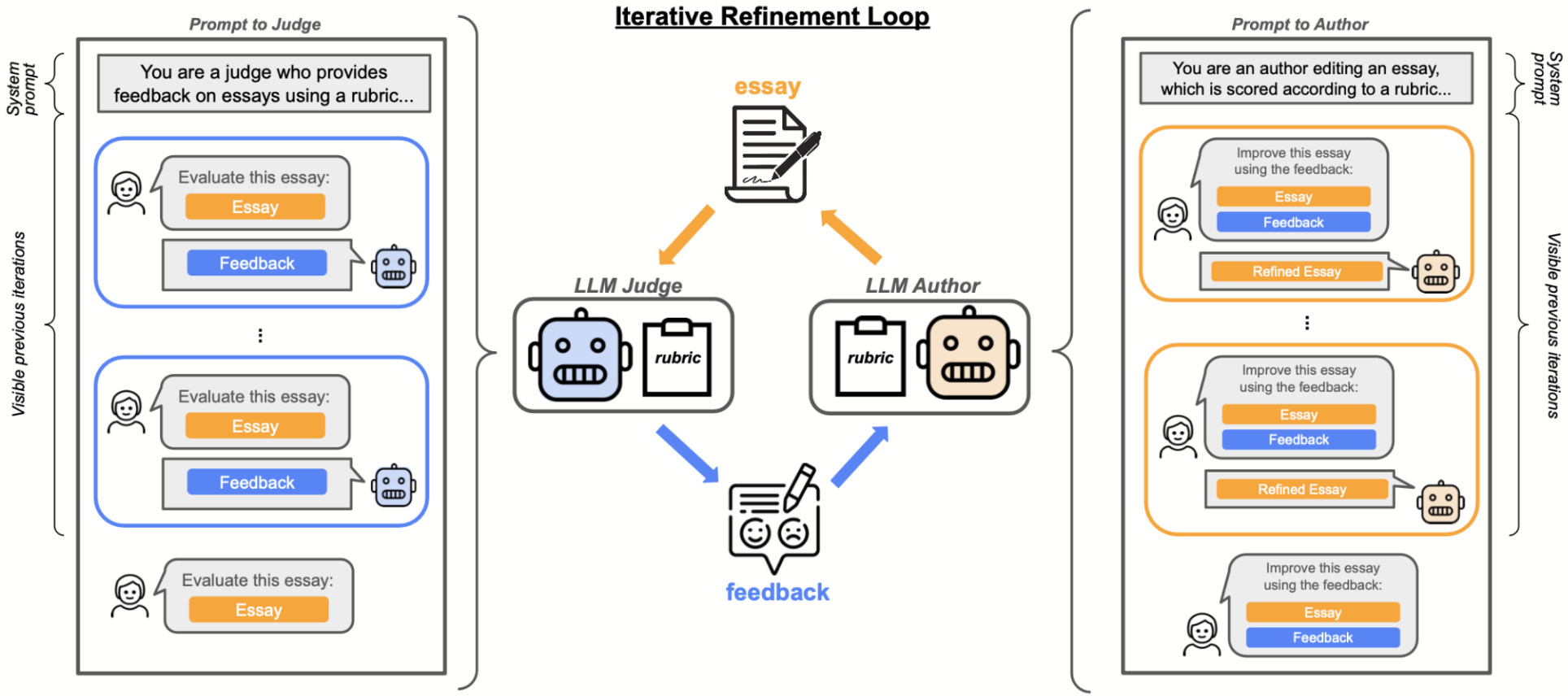
(b) Human-in-the-loop Approval



(c) Decoupled Approval

- En Approval RL, el agente recibe aprobación como recompensa directa.
- En Human-in-the-loop, el humano interviene en decisiones críticas y puede validar antes de actuar.
- En Decoupled Approval, la evaluación se separa de la ejecución para reducir incentivos a manipular al evaluador y mejorar la confiabilidad del feedback.

Bucle de refinamiento iterativo (Juez-Autor)



Bucle: evaluar con rúbrica -> feedback -> reescritura -> repetir.
Útil para mejorar redacción; cuidado: puede optimizar "estilo" sobre contenido.