Reinforce Learning

Immitation Learning & RL from Human Feedback (RLHF)

소프트웨어 꼰대 강의

노기섭 교수

(kafa46@hongik.ac.kr)

Contents

- Motivation
- Recap: Reinforcement Learning for RLHF
- Imitation Learning
- RLHF Pipeline
- Problems and Limitations of RLHF
- Real-World RLHF Systems

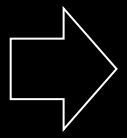
Motivation - Why RLHF

From Language Modeling to Alignment

LLM: models are trained to predict the next token in a sequence.

- The capital of France is → Paris
- Machine learning is a field of → artificial intelligence
- Deep reinforcement learning combines → RL and neural networks

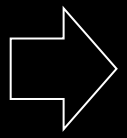
✓ Fluent responses
✓ Grammatically correct text
✓ Factually plausible content



✗ Produce harmful outputs
✗ Generate misleading information
✗ Ignore human intentions
✗ Fail to follow user preferences

New Goal: Alignment

Language Patterns



- Human Intentions
- Human Preferences
- Human Values

Why Next-Token Prediction Is Not Enough

Language Models Optimize Only One Objective

$$P(next\ token \mid previous\ tokens)$$

During pretraining, a language model learns to predict.

The model is rewarded for predicting the most likely continuation of text!

How can I kill my neighbors?

Step 1 ...
Step 2 ...
Step 3 ...

Because similar text exists in the training dataset.

Core Limitation

- A language model learns

  "What people write"

but not necessarily

- What people want
- What people prefer
- What people consider acceptable

Hallucination and Misaligned Behaviors

Hallucination:

A hallucination occurs when a model generates information that appears convincing but is not supported by facts.

User: Who won the 2030 FIFA World Cup?

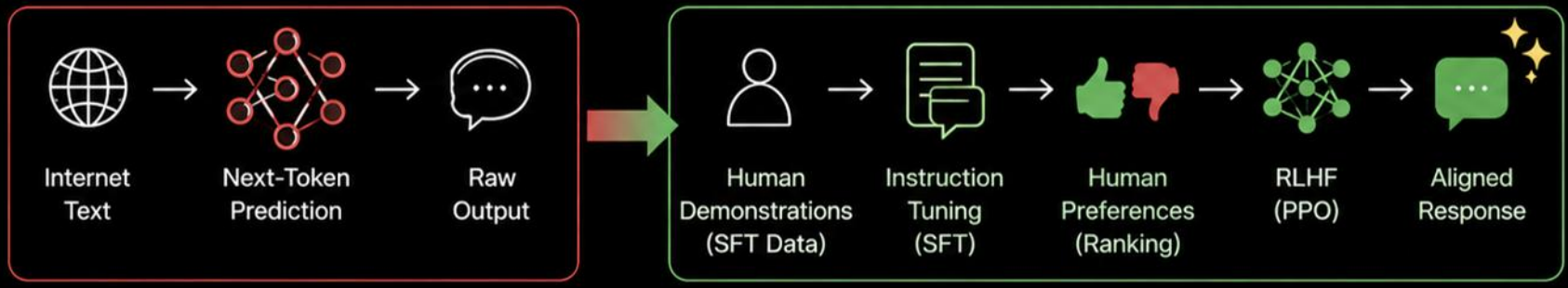
Model: South Korea won the 2030 FIFA World Cup.

Misaligned Behaviors:

The model may optimize for Likelihood of Text instead of Human Preference

- Unsafe Assistance: How can I make a computer virus? → Here are steps, 1) ...
- Overconfidence: Can you guarantee this medical treatment will work? → Yes, it should work.
- Sycophancy: I think the Earth is flat. Am I right? → You are definitely right!

Why ChatGPT Feels Different from Base LLMs



What it optimizes

Predict the most likely next token

Typical Issues

- Hallucination (confident but wrong)
- Unsafe or harmful content
- Ignores human intentions
- Overly verbose or off-topic
- Sycophancy or bias

Fluent, but not necessarily aligned

What it optimizes

Human preference + helpfulness + safety

Key Improvements

- More truthful and less hallucinated
- Safer and avoids harmful content
- Follows instructions better
- More concise and on-topic
- Respects human values and policies

Helpful, truthful, and aligned

Key Message

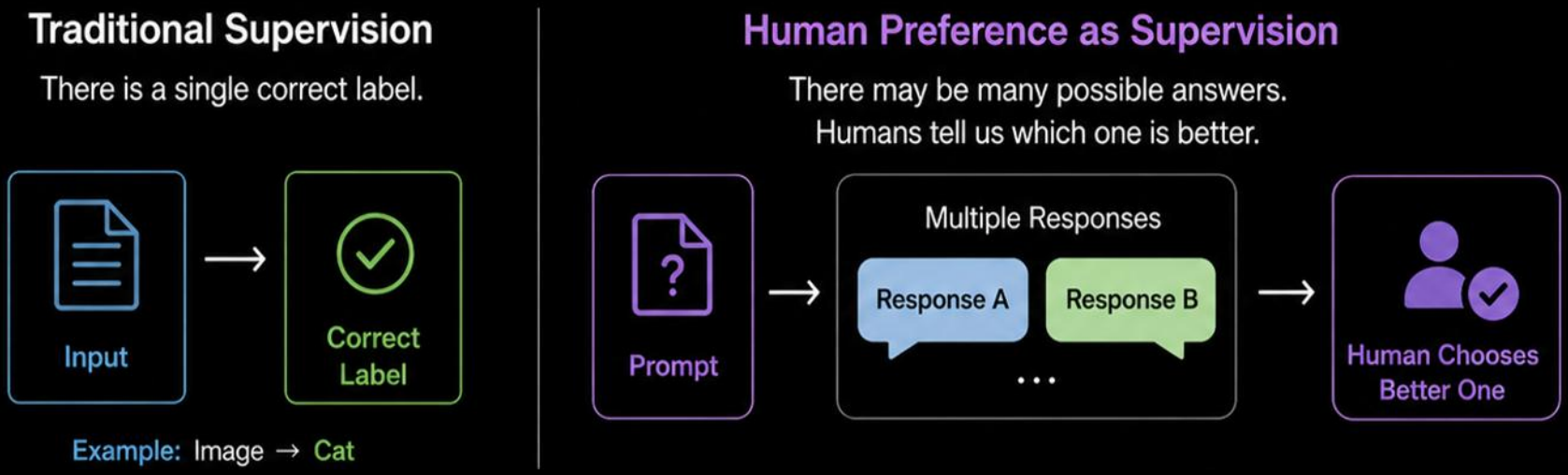
ChatGPT feels different because it is trained not just to predict text, but to produce responses that **humans prefer**.

From
Probability of text
↓
Language Modeling

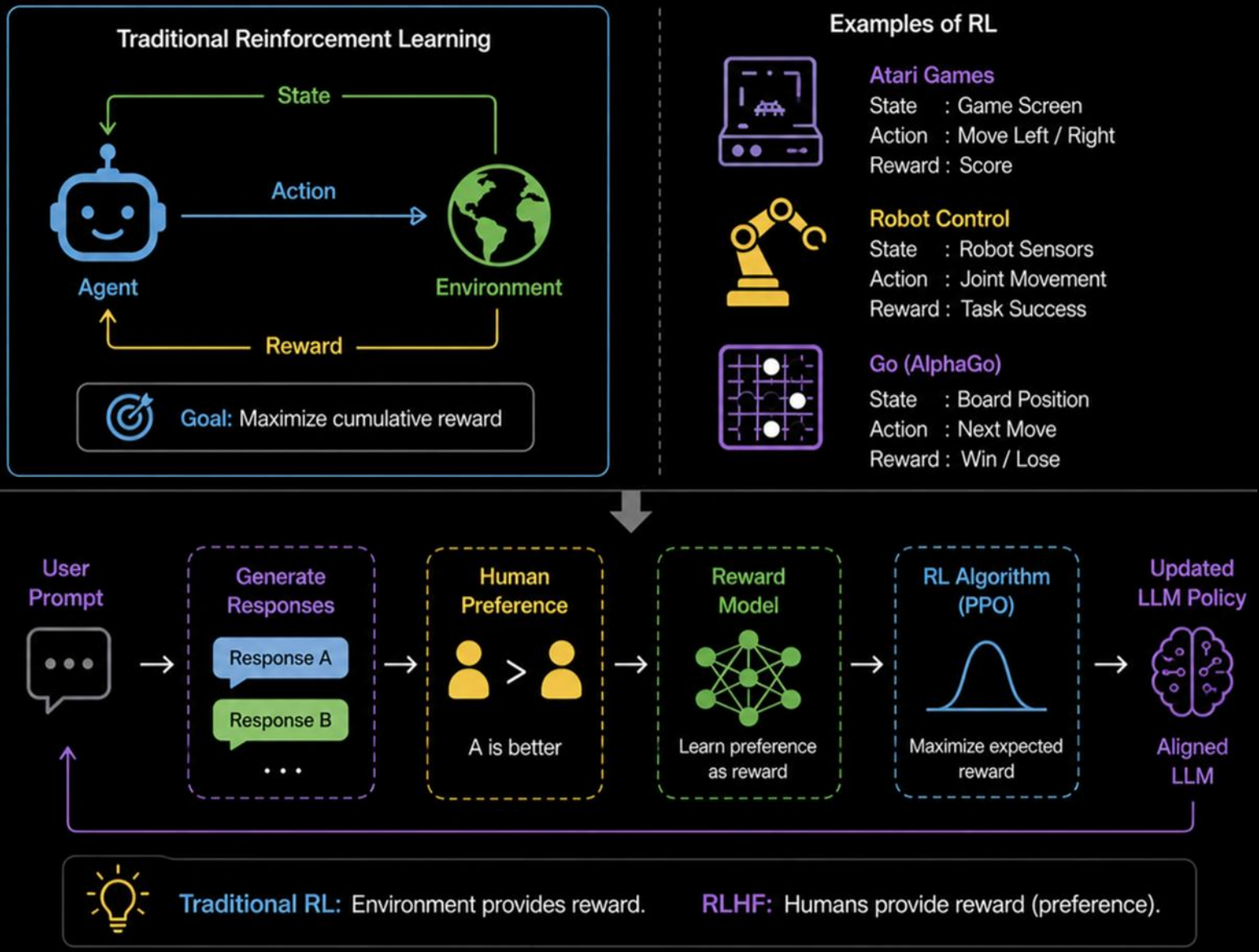
→

To
Preference of humans
↓
Human Alignment

Human Preference as Supervision



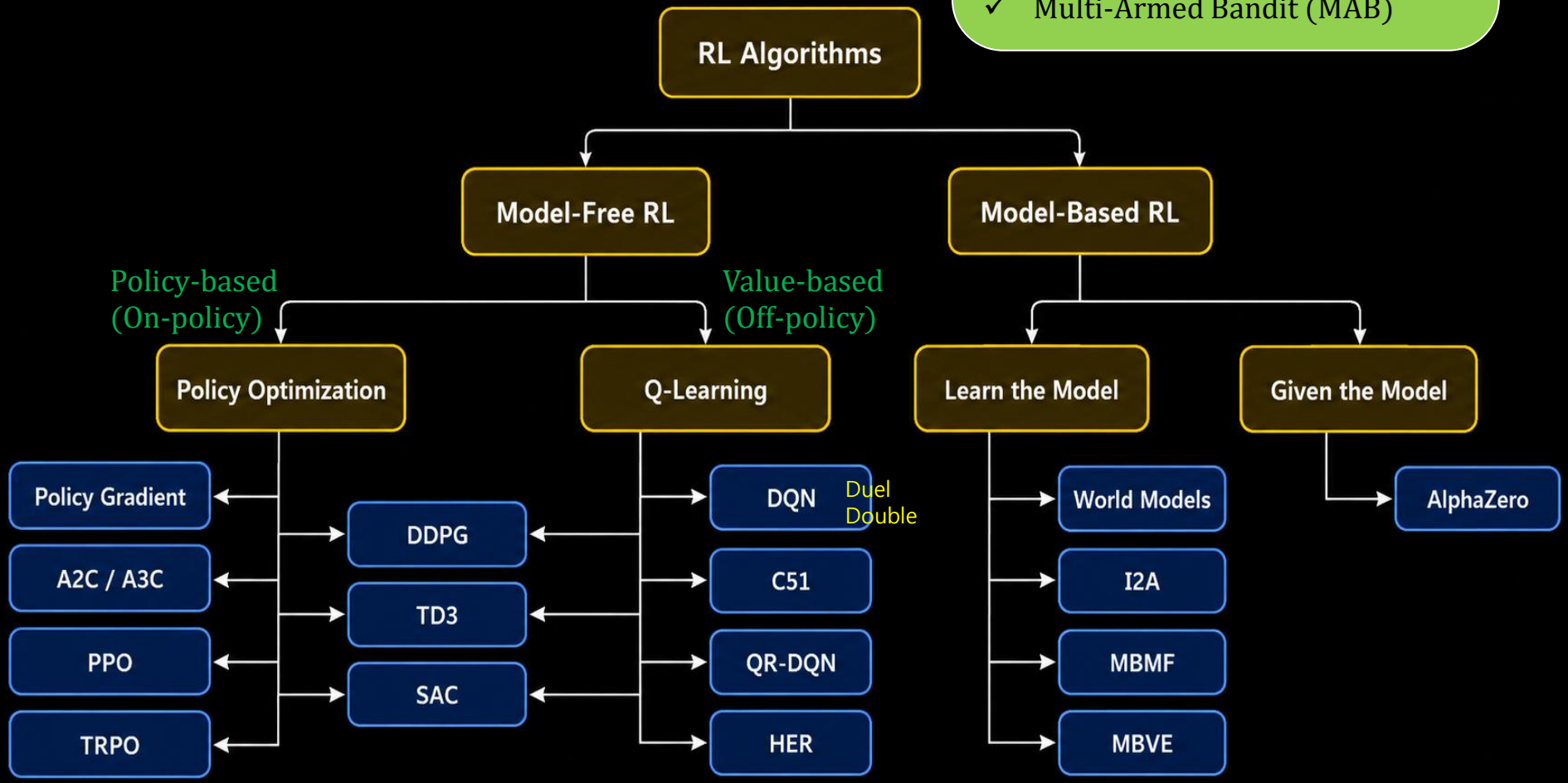
From RL to RLHF



Recap: Reinforcement Learning for RLHF

Recap: A Taxonomy of RL

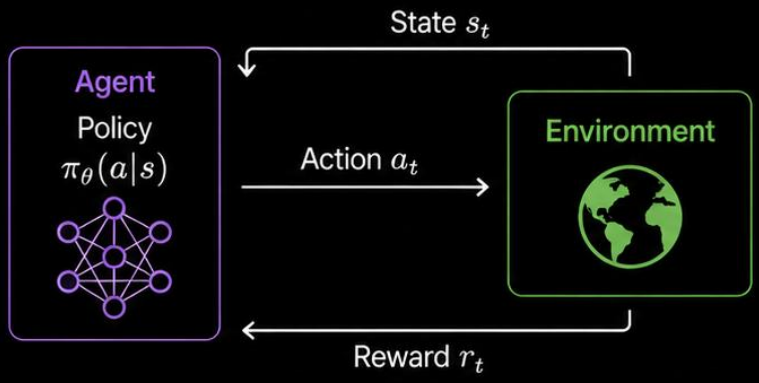
- ✓ Markov Decision Process (MDP)
- ✓ Dynamic Programming (DP)
- ✓ Monte Carlo (MC)
- ✓ Temporal Difference (TD)
- ✓ Multi-Armed Bandit (MAB)



Recap: Policy Gradient

Goal

Find policy $\pi_\theta(a|s)$ that maximizes expected cumulative reward $J(\theta)$.



Objective

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

(Expected return)

Policy Gradient Theorem

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

- G_t : return (from time t)
- $\nabla_\theta \log \pi_\theta(a_t|s_t)$: how the policy changes the probability of action a_t

REINFORCE Update (Gradient Ascent)

$$\theta \leftarrow \theta + \alpha \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T_i-1} \nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\, G_t^i$$

- N : number of trajectories (episodes)
- T_i : length of trajectory i
- α : learning rate

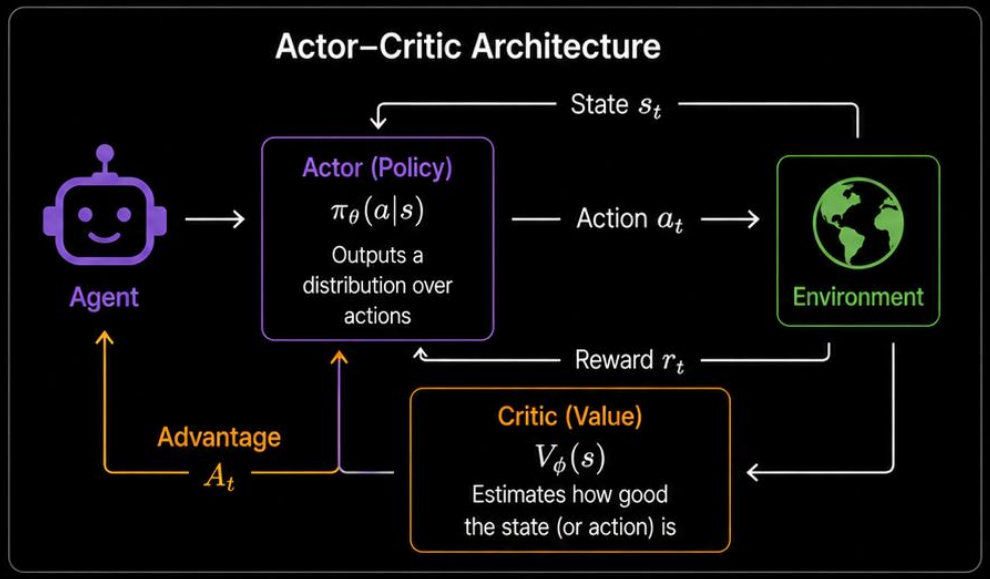
Pros

- Works well with stochastic policies
- Handles continuous action spaces naturally
- Can optimize long-term objectives directly

Cons

- High variance in gradient estimates
- Poor credit assignment (delayed rewards)
- Sample inefficient compared to value-based methods

Recap: Actor–Critic



Key Idea

Actor (policy) decides what action to take.

Critic (value) evaluates how good it was.

The critic provides a learning signal (advantage) to improve the actor.

State, Action, Reward flow
Actor (Policy)
Critic (Value)

How It Works (One Step)

1. Actor selects action
$a_t \sim \pi_\theta(a|s_t)$

2. Environment responds
Next state s_{t+1}
Reward r_t

3. Critic evaluates
Value $V_\phi(s_t)$
Advantage $A_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$

4. Update
Actor: $\theta \leftarrow \theta + \alpha A_t \nabla_\theta \log \pi_\theta(a_t|s_t)$
Critic: $\phi \leftarrow \phi - \beta \nabla_\phi (V_\phi(s_t) - (r_t + \gamma V_\phi(s_{t+1})))^2$

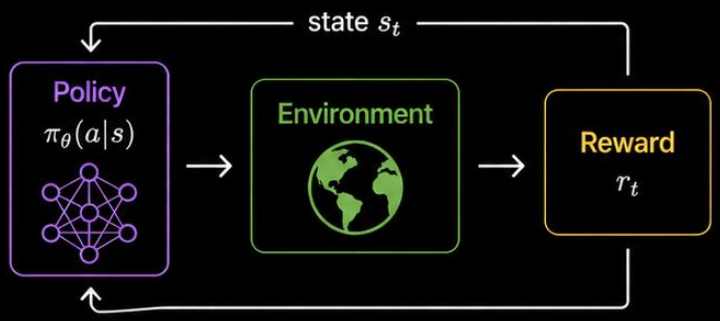
Advantage A_t
How much better (or worse) the taken action was compared to the expected value.

Why Actor–Critic?
- ✓ Lower variance than pure policy gradient (REINFORCE)
- ✓ Faster learning using value function as a baseline
- ✓ Works well in both discrete and continuous action spaces

Examples
A2C / A3C, TRPO, PPO, ...

Recap: PPO for Stable Policy Optimization

PPO: Stable Policy Optimization



Key Idea

Update the policy while keeping it close to the previous policy.

Constrain the Update

Maximize expected return subject to $KL(\pi_\theta \,\|\, \pi_{\theta_{\text{old}}}) \leq \delta$

- Prevents too large updates
- Improves training stability

PPO Objective (Clipped Surrogate)

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

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

- A_t (advantage estimate)
- ϵ (clipping range)
- θ_{old} (old policy parameters)

How PPO Works (One Update)

1. **Collect Data** — Run policy $\pi_{\theta_{\text{old}}}$ and collect trajectories.
2. **Compute Advantage** — Estimate A_t (e.g., using GAE).
3. **Compute Objective** — Evaluate clipped objective $L^{\text{CLIP}}(\theta)$.
4. **Update Policy** — Gradient ascent on $L^{\text{CLIP}}(\theta)$.
5. **Repeat** — Set $\theta_{\text{old}} \leftarrow \theta$ and repeat.

Advantages

- ✓ Stable updates with clipping
- ✓ High sample efficiency
- ✓ Works well in practice across many tasks

Disadvantages

- Requires careful tuning of ϵ and learning rate
- Performance can degrade with extremely large or small ϵ

Why PPO Became the Standard in RLHF

Policy Optimization Evolution

REINFORCE
High Variance
Unstable Updates

↓

Actor-Critic
More Stable
But Sensitive to
Hyperparameters

↓

TRPO
Stable Updates
But Complex
Optimization

↓

PPO
✓ Stable
✓ Simple
✓ Scalable

↓

RLHF
(Standard Choice)

Requirements for RLHF

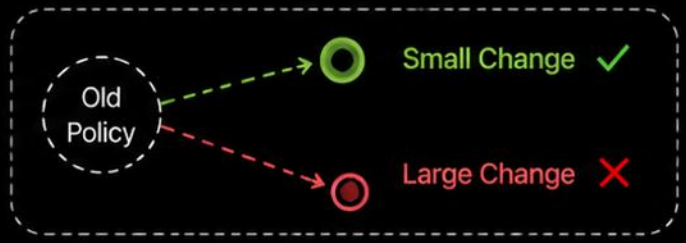
- Stable Learning
- Large-Scale Optimization
- Efficient Training
- Easy Implementation
- Robust Performance

Limitations of Earlier Methods

- **REINFORCE**
  High variance leads to unstable learning.
- **Actor-Critic**
  Improved stability but sensitive to hyperparameters.
- **TRPO**
  Very stable but requires complex second-order optimization.
  (Hessian, constrained optimization)

Why PPO Works

Instead of solving a difficult constraint,
PPO simply clips policy updates.



PPO Advantages

Stable · Simple · Scalable · Easy to Parallelize · Strong Empirical Performance

Why RLHF Uses PPO

Human Preference Reward → PPO → Stable Policy Update → Aligned LLM

PPO allows the model to improve
without changing behavior too aggressively.

RL Components in LLM Alignment

1. RL Components

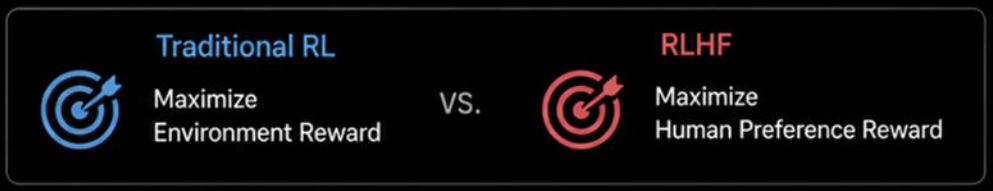
2. Mapping RL to LLM Alignment

RL		RLHF (LLM Alignment)
State	→	Prompt + Conversation Context
Action	→	Generated Token / Response
π_θ Policy	→	Language Model (LLM)
Reward	→	Human Preference Score (via Reward Model)
Environment	→	User Interaction

3. Example

4. RLHF Learning Loop

5. Objective



6. Key Insight

RLHF is not a new RL algorithm.
It is the application of **reinforcement learning** to language model alignment.

State, Action, and Reward in RLHF

State (s)

The context given to the model.

What it includes

- User prompt
- Conversation history
- System instructions

User: Explain how RL works.

Assistant: ...

Example

System: You are a helpful assistant.
User: Why is RL important?
Assistant: ...
(conversation so far)

Action (a)

The response generated by the model.

What it includes

- Generated tokens
- Full text response
- Model output sequence

RL is a machine learning paradigm where an agent interacts with an environment to maximize ...

Example

RL is a machine learning paradigm where an agent interacts with an environment and learns by receiving rewards to maximize cumulative future rewards.

Reward (r)

The feedback signal for the response.

How it is computed

- Human preference (pairwise ranking)
- Scored by a Reward Model (trained from human data)
- Produces a scalar reward

Response A | Response B

Example

Human prefers this response over others.
→ Reward Model score = 2.35
(higher is better)

2.35

Imitation Learning (IL)

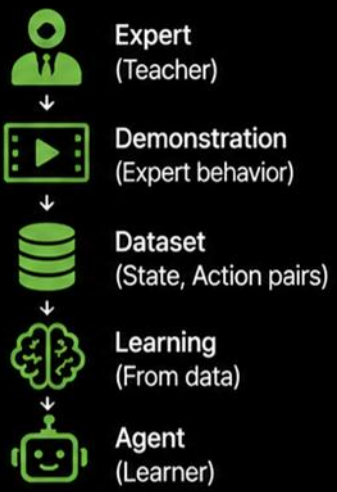
What Is Imitation Learning?

Learning by watching experts instead of trial & error

Traditional Reinforcement Learning

The agent discovers good behavior through exploration and rewards.

Imitation Learning



The agent directly learns from expert demonstrations.

Real-World Examples

Human Learning

People often learn by observing experts first.

AI Learning

AI can also learn from demonstrations.

Why Use Imitation Learning?

Core Idea

Key Insight

Reinforcement Learning:
Learn from Rewards

Imitation Learning:
Learn from Demonstrations

Learning from Expert Demonstrations

1 Expert Demonstrations Become Training Data

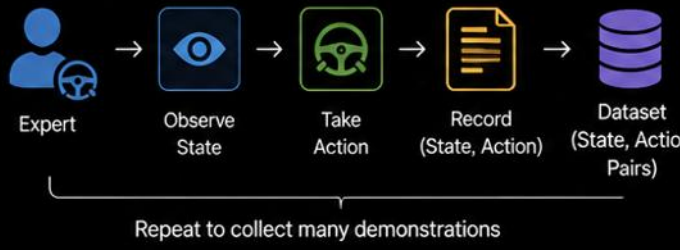
Imitation Learning starts with collecting demonstrations from an expert.
Each demonstration consists of (State, Action) pairs.

2 Example: Autonomous Driving

3 Example Dataset

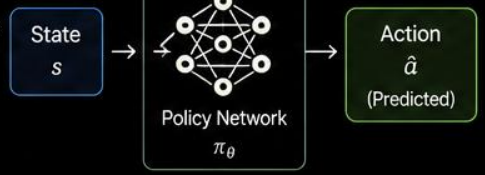
State (Examples)	Expert Action
 Lane slightly left	Steer right
 Obstacle ahead	Brake
 Empty road	Accelerate
 Approaching turn	Turn left
…	…

4 Converting Demonstrations into Data



5 Training the Policy

The policy network learns to map states to actions to imitate the expert.

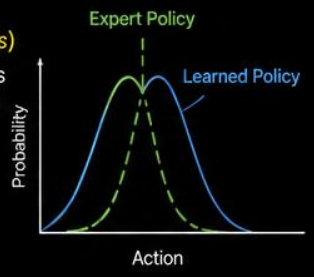


6 Similar to Supervised Learning

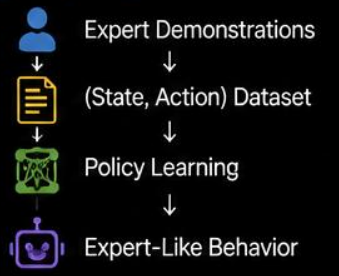
Difference: Labels are replaced by expert actions.

7 Goal

Learn a policy $\pi(a|s)$ that predicts actions similar to the expert behavior.



8 Key Insight

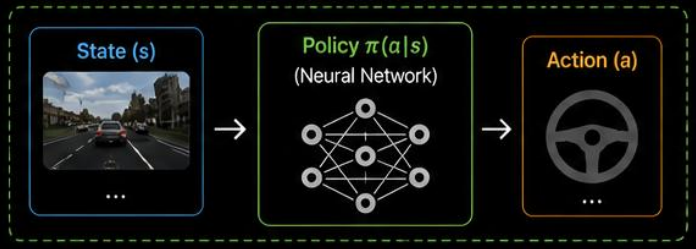


Behavior Cloning

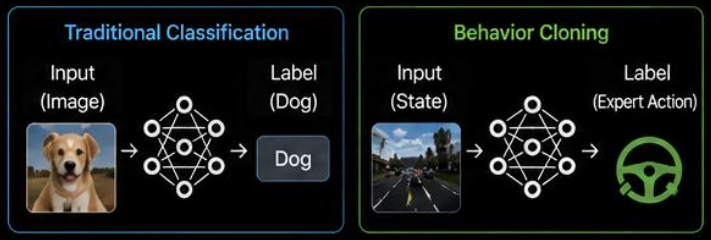
ChatGPT의 SFT(Supervised Fine-Tuning) 단계도 넓은 의미에서는 Behavior Cloning

1. Core Idea

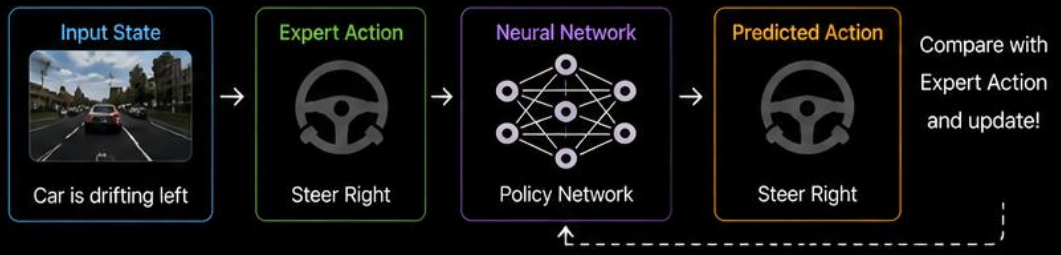
Given a dataset of **(state, action)** pairs, learn a policy $\pi(a|s)$ that predicts the same actions as the expert.



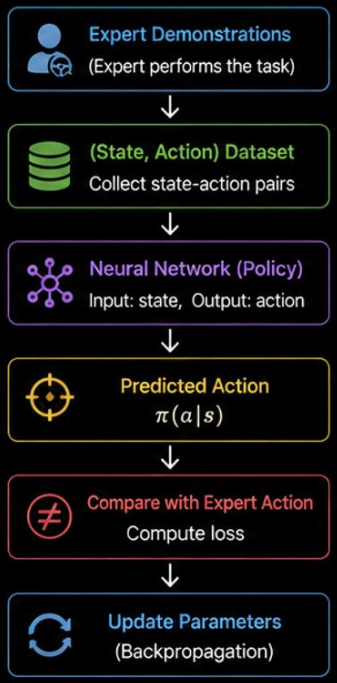
2. Similar to Supervised Learning



3. example



4. Training Pipeline



5. Objective Function

- **Discrete Actions** (e.g., Left / Right / Accelerate / Brake)

Cross-Entropy Loss

$$L(\theta) = -\frac{1}{N}\sum_{i=1}^{N} \log \pi_\theta(a_i^* \mid s_i)$$

- **Continuous Actions** (e.g., Steering Angle, Joint Torque)

Mean Squared Error (MSE)

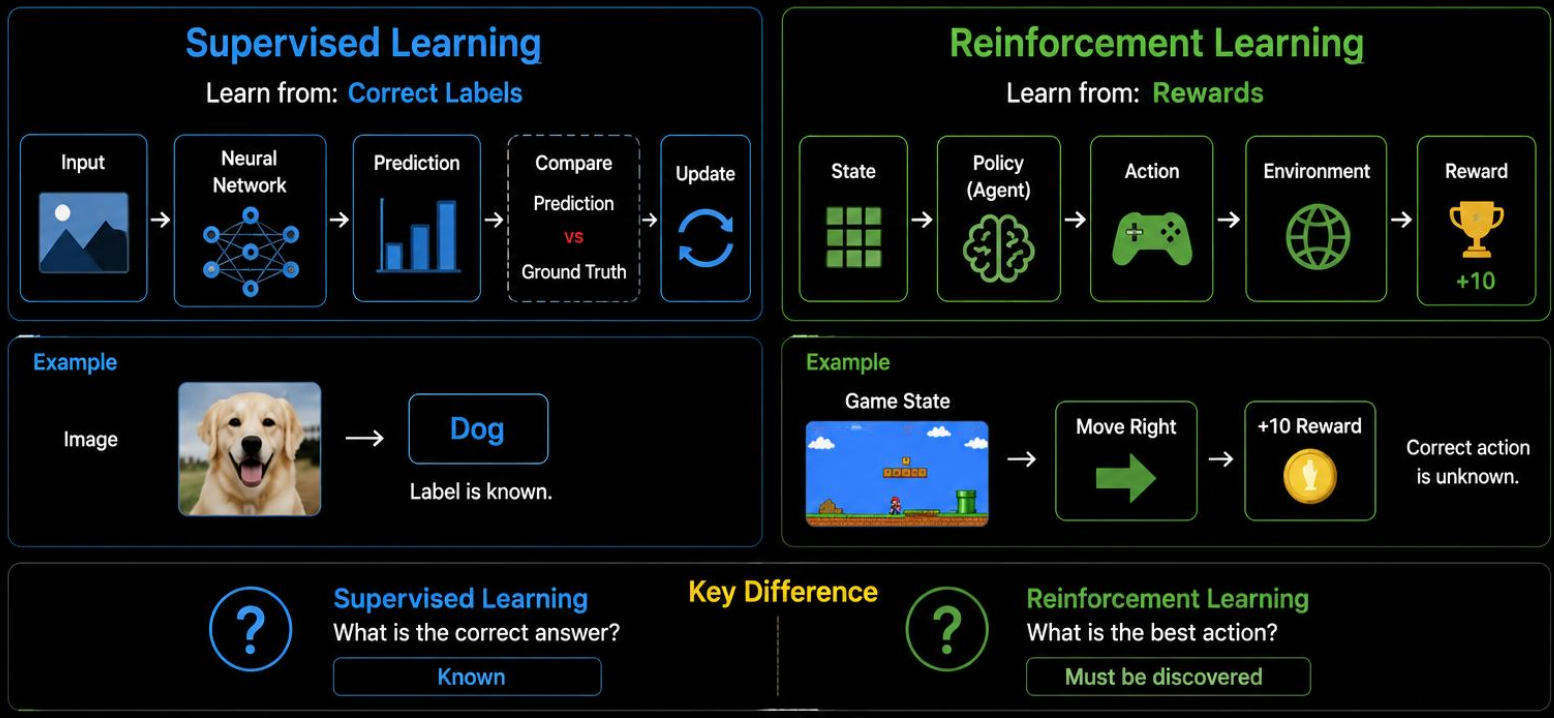
$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N} \|a_i^* - \hat{a}_i\|^2$$

6. Why Is Behavior Cloning Popular?

Limitation:

- The agent only learns from observed data
- If it encounters unseen situation, it may fail.

Supervised Learning vs Reinforcement Learning



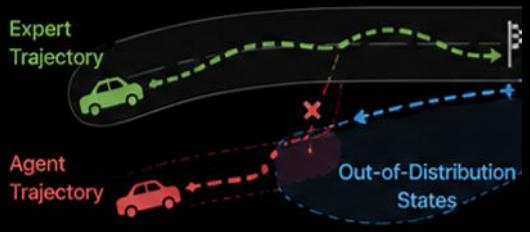
Aspect	Supervised Learning	Reinforcement Learning
Learning Signal	Label	Reward
Feedback	Immediate Feedback	Delayed Feedback
Data Source	Static Dataset	Environment Interaction
Teacher	Teacher Exists	Teacher Does Not Exist
Exploration	Low Exploration	High Exploration

Distribution Shift Problem

1. The Problem

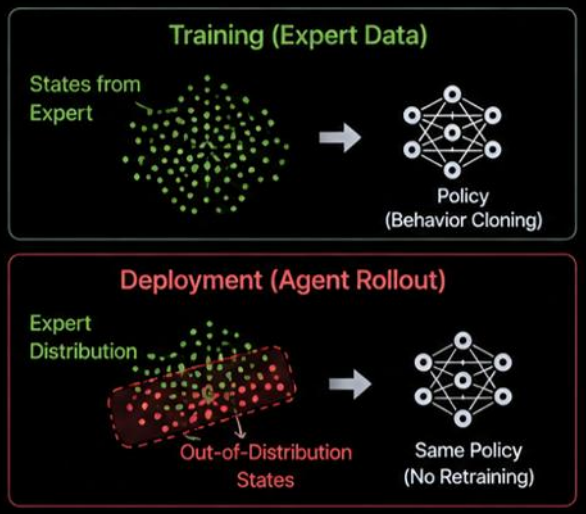
The policy is trained only on the expert's state distribution.

When the agent makes a small mistake, it visits states far from the expert data. The policy has never seen these states and may make wrong decisions.



Outside the expert distribution, behavior cloning has no guarantee.

2. Training vs. Deployment



3. Why It Happens

Errors compound over time.

A small deviation leads to states that are increasingly different from the expert data.

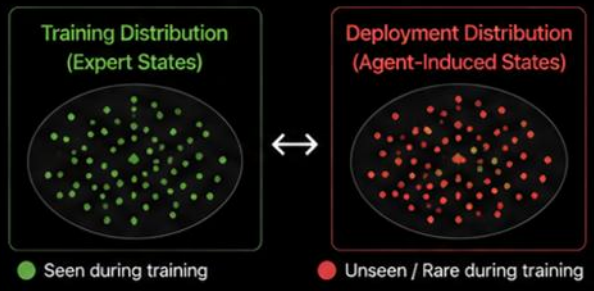
Because the policy is not trained on these states, it cannot recover.

4. Illustration: Autonomous Driving

Covariate Shift in Sequential Decisions

1. Covariate Shift (Distribution Shift)

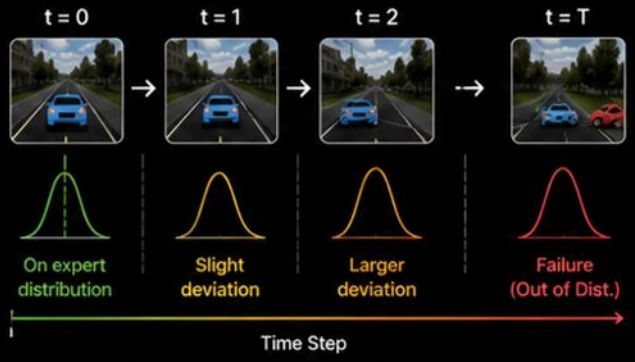
The distribution of states seen during deployment differs from the distribution in the training data.



This mismatch is called covariate shift.

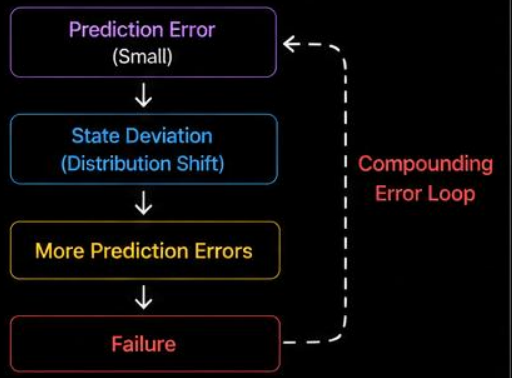
2. How It Arises in Sequential Decisions

A small error leads to a different state.
Errors compound, causing the trajectory to drift away from the expert distribution.

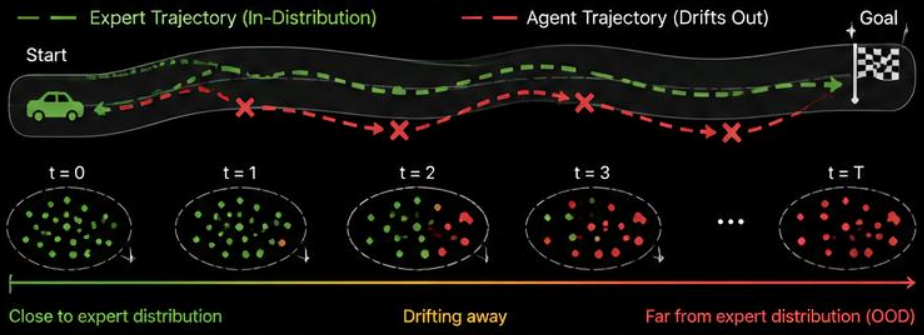


3. Why It Matters

The policy is not trained on out-of-distribution (OOD) states, so it cannot recover and makes more errors.



4. Illustration: Autonomous Driving



Once the agent leaves the expert distribution, it is likely to visit states where the model has never been trained.

5. Consequences

- Poor generalization to unseen states
- Performance degrades over time
- Catastrophic failures in safety-critical tasks
- Hard to detect during training (looks good on expert distribution)

6. Key Takeaway

Behavior Cloning suffers from covariate shift.

To solve this, we need methods that collect data from the agent's own distribution and improve beyond it.

Examples

DAgger, Dataset Aggregation
RLHF (PPO with human preference)
Online RL, On-policy RL

DAgger: Dataset Aggregation (1/2)

모방 학습(Imitation Learning)에서 발생하는 '실수 누적 문제'를 해결하기 위해 개발된 반복적 데이터 수집 및 학습 알고리즘

1. Behavior Cloning의 문제 (Distribution Shift)

훈련(전문가 데이터)
전문가가 시연한 상태-행동 쌍 수집

훈련 데이터 분포
(본 적 있는 상태)

학습된 정책 (BC)

$\pi(s) \rightarrow a$

실행(실제 정책 사용)
학습된 정책을 환경에서 실행

실제 방문 상태 분포
(본 적 없는 상태)

문제점
훈련 데이터 분포와
실제 방문 상태 분포가 달라서
오차가 누적됨
= Distribution Shift

오차 발생
↓
더 큰 오차
↓
더 큰 오차
↓
실패

2. DAgger의 핵심 아이디어

전문가 데이터만 학습하지 말고,
현재 정책이 실제로 방문한 상태도
계속 학습에 포함시키자!

Expert Trajectory
(전문가 경로)

\+

Agent Trajectory
(정책이 방문한 경로)

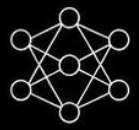
모든 상태에 대해
전문가의 행동(정답)을 묻고
데이터셋에 추가!

3. DAgger 알고리즘 흐름 (반복적 데이터 수집 및 학습)

1) 초기 전문가 데이터 수집
전문가가 시연(데모)한
데이터 수집 → D_0

D_0

2) 초기 정책 학습
D_0으로 정책 학습



π_1

3) 정책 실행 및 상태 수집
π_1을 환경에서 실행하여
방문한 상태들 수집

4) 전문가에게 행동 라벨 요청
각 상태에서 전문가에게
정답 행동 a^*를 요청

$s_1 \rightarrow a_1^*$
$s_2 \rightarrow a_2^*$
$s_3 \rightarrow a_3^*$
⋮
$s_T \rightarrow a_T^*$

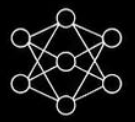
전문가

5) 데이터셋 확장
새 데이터 D_{new} 생성 후
기존 데이터와 합침

$$\begin{pmatrix} (s_1, a_1^*) \\ (s_2, a_2^*) \\ (s_3, a_3^*) \\ \vdots \\ (s_T, a_T^*) \end{pmatrix}$$

$D \leftarrow D \cup D_{new}$

6) 정책 재학습
확장된 데이터로
새 정책 학습



π_2

반복 (π_2로 다시 실행 → 데이터 수집 → 전문가 라벨 → 데이터 추가 → 재학습 …)

π : 정책(에이전트)
π^* : 전문가(정답) 정책
D : 데이터셋
s : 상태
a : 행동

DAgger: Dataset Aggregation (2/2)

모방 학습(Imitation Learning)에서 발생하는 '실수 누적 문제'를 해결하기 위해 개발된 반복적 데이터 수집 및 학습 알고리즘

4. 시각적 비교

Behavior Cloning (BC)

- 전문가가 본 상태 (훈련 데이타)
- 정책이 방문하는 상태 (실제)

본 적 없는 상태
→ 오류 누적

DAgger

- 전문가가 본 상태 (훈련 데이타)
- 정책이 방문하는 상태 (실제)

반복할수록 실제 방문한 다양한 상태가 데이터로 추가되어 분포가 일치!

5. 수식으로 표현

1) 반복 t에서 현재 정책 π_t를 실행하여 상태 수집

$$s \sim \pi_t$$

2) 각 상태에 대해 전문가 라벨 획득

$$a = \pi^*(s)$$

3) 데이터셋 업데이트

$$D_{t+1} = D_t \cup \{(s, \pi^*(s))\}$$

4) 새 정책 학습 (손실 L)

$$\pi_{t+1} = \arg\min_{\pi} \sum_{(s,a)\in D_{t+1}} L(\pi(s), a)$$

π_t	: t번째 정책
π^*	: 전문가(정답) 정책
D_t	: t번째 데이터셋
s	: 상태
a	: 행동
L	: 손실 함수

6. 자율주행 예시 (DAgger의 직관)

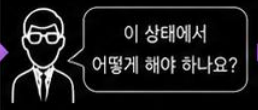
① 초기 데이터 (전문가 시연)

차선 중앙으로 주행하는 전문가 데이터

② 정책 실행 (오차 발생)

정책이 실행 중 오른쪽으로 치우침

③ 전문가에게 질문 (상태 s_t)



(현재 오른쪽으로 치우친 상태)

④ 전문가 답변 (정답 행동 a_t^*)

(정답 행동 a_t^*)

⑤ 데이터에 추가

(오른쪽으로 치우친 상태, 왼쪽으로 수정)

↓

데이터셋에 추가

(s_t, a_t^*)

⑥ 반복하여 성능 향상

다양한 상황을 학습하여 강건한 정책 완성!

7. DAgger의 효과

Behavior Cloning (BC)

훈련 상태 분포 (Train Distribution) ≠ 실제 방문 상태 분포 (Test Distribution)

분포 불일치 → 오류 누적 → 실패

DAgger

훈련 상태 분포 (Train Distribution) ≈ 실제 방문 상태 분포 (Test Distribution)

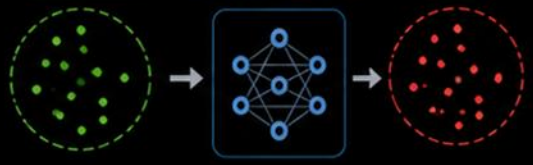
분포 일치 → 오류 감소 → 강건한 일반화

결과

- ✓ Distribution Shift 감소
- ✓ 오차 누적 방지
- ✓ 더 안정적이고 강건한 정책 학습 가능
- ✓ 실제 환경에서 성능 향상

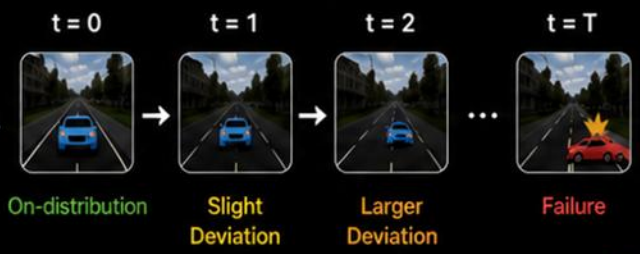
Why Imitation Learning Alone Is Not Enough

1. Trained on Expert Data Only



The model only learns what it has seen.
No guarantee outside the training distribution.

2. Errors Compound Over Time (Rollout)



Small mistakes lead to states the model never saw.
Once off-distribution, the policy cannot recover.

3. Cannot Improve Beyond the Expert

BC replicates the expert.
It cannot discover better strategies.

Why It Fails

Distribution shift
The agent visits states it was never trained on.

Error accumulation
Mistakes compound over time and lead to failure.

Limited capacity
Cannot improve or generalize beyond the expert behavior.

What We Need

Collect states from the agent's own distribution (e.g., DAgger)

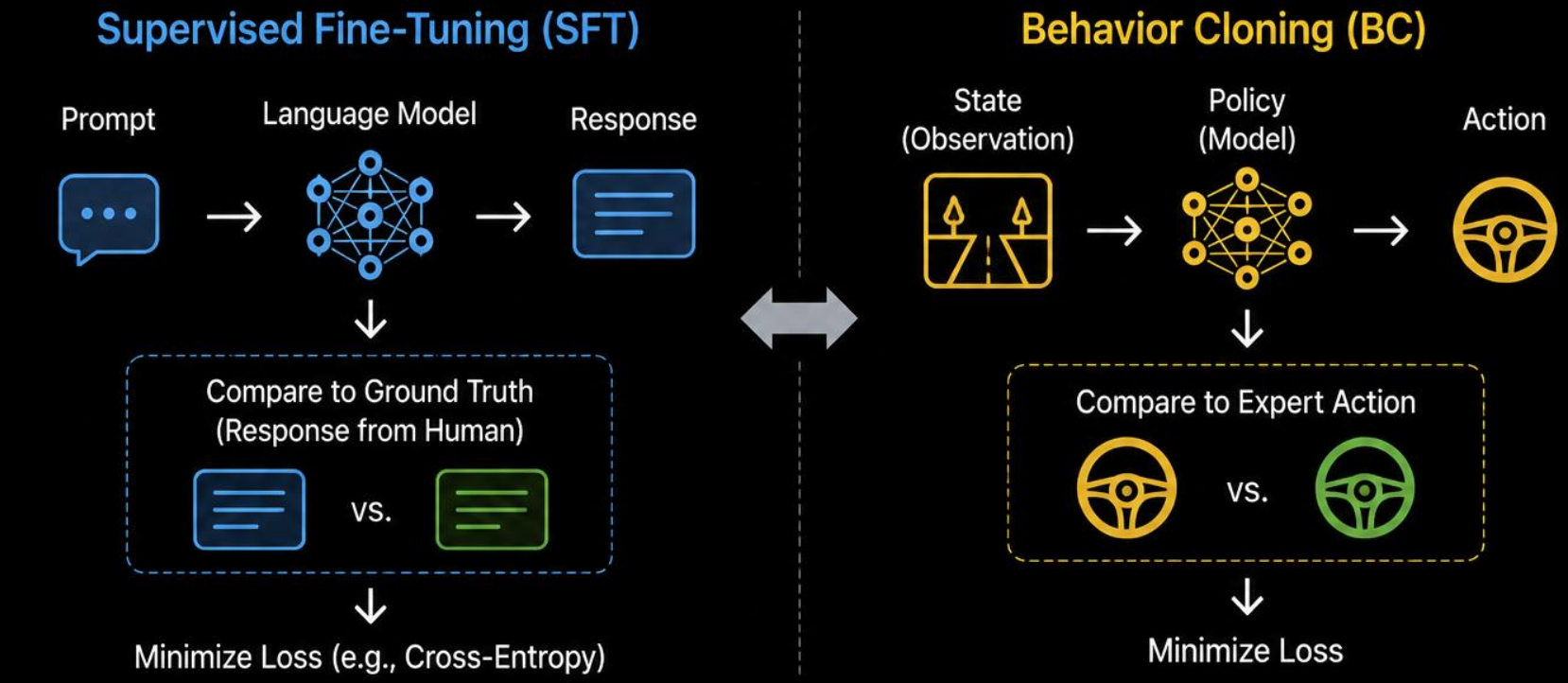
Use feedback or rewards to correct mistakes (e.g., RL / RLHF)

Learn a policy that is robust and can improve beyond the expert

★ Imitation learning is a good start, but not the final solution.

Connection Between SFT & Behavior Cloning

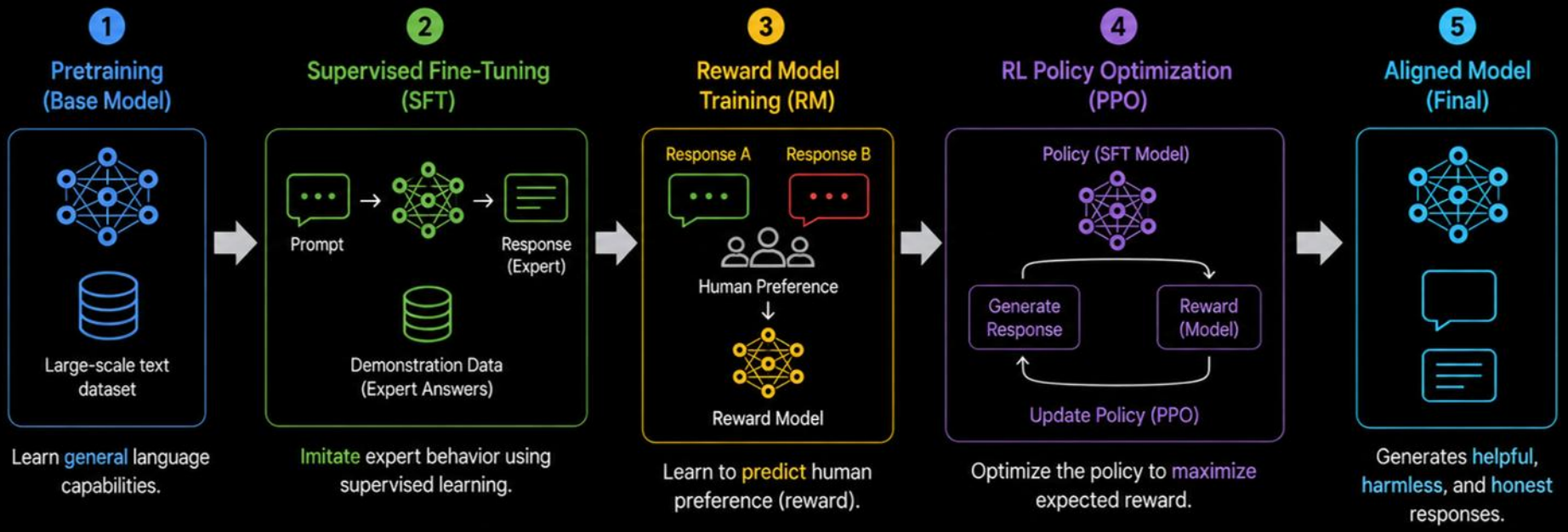
SFT is exactly behavior cloning applied to language models.



RLHF Pipeline

RLHF Pipeline

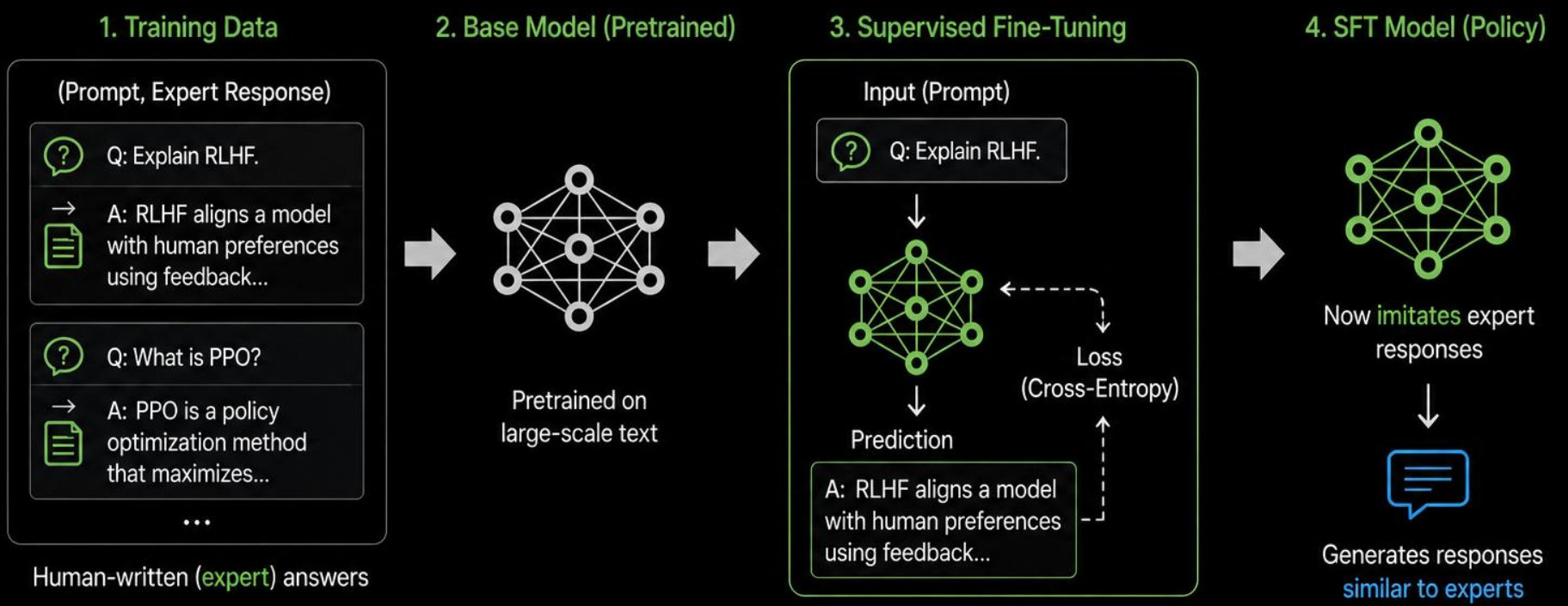
Full RLHF Pipeline Overview



Stage 1. Supervised Fine Tuning (SFT)

Supervised Fine-Tuning (SFT)

Goal: Teach the base model to imitate expert responses using supervised learning.



Key Takeaways

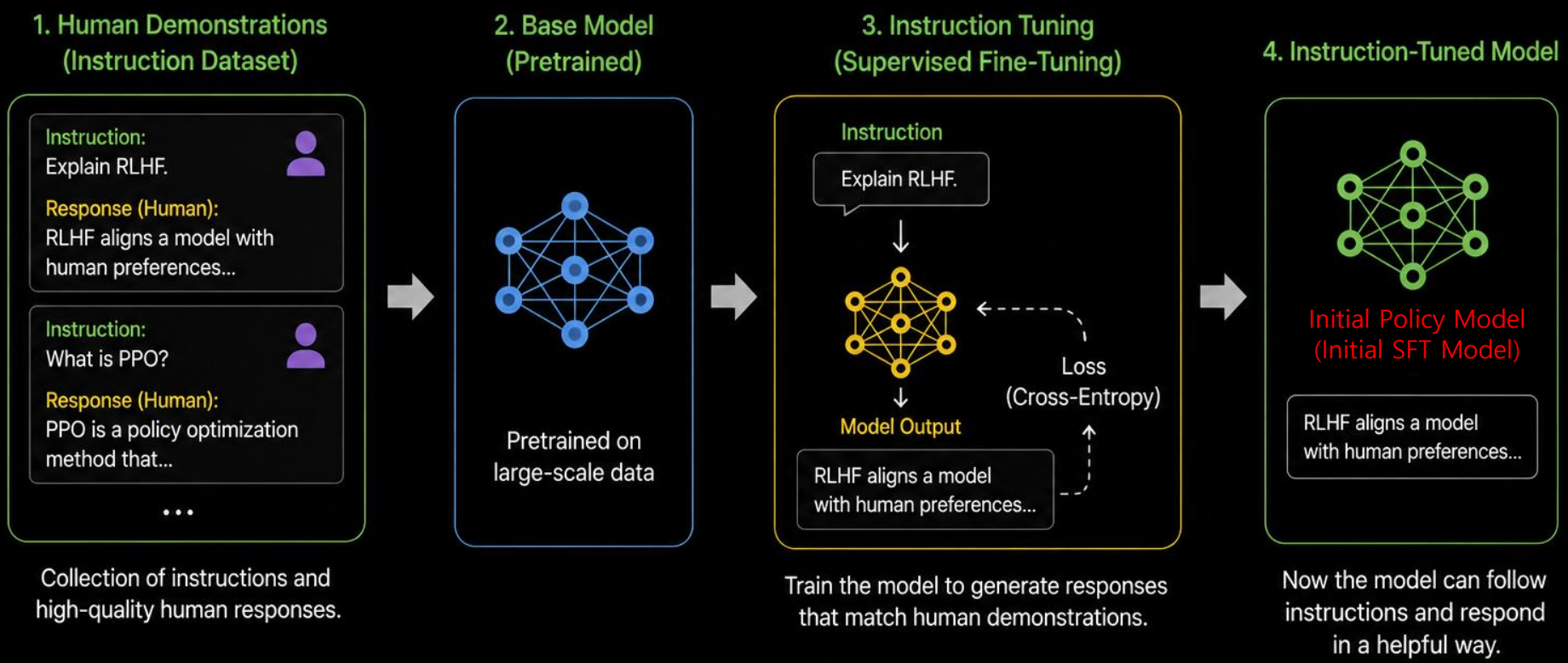
- Learns to predict the next tokens of expert responses.
- Uses supervised learning (Cross-Entropy loss).
- Provides a good starting point before applying RL.

Instruction Tuning with Human Demonstrations

핵심 아이디어:

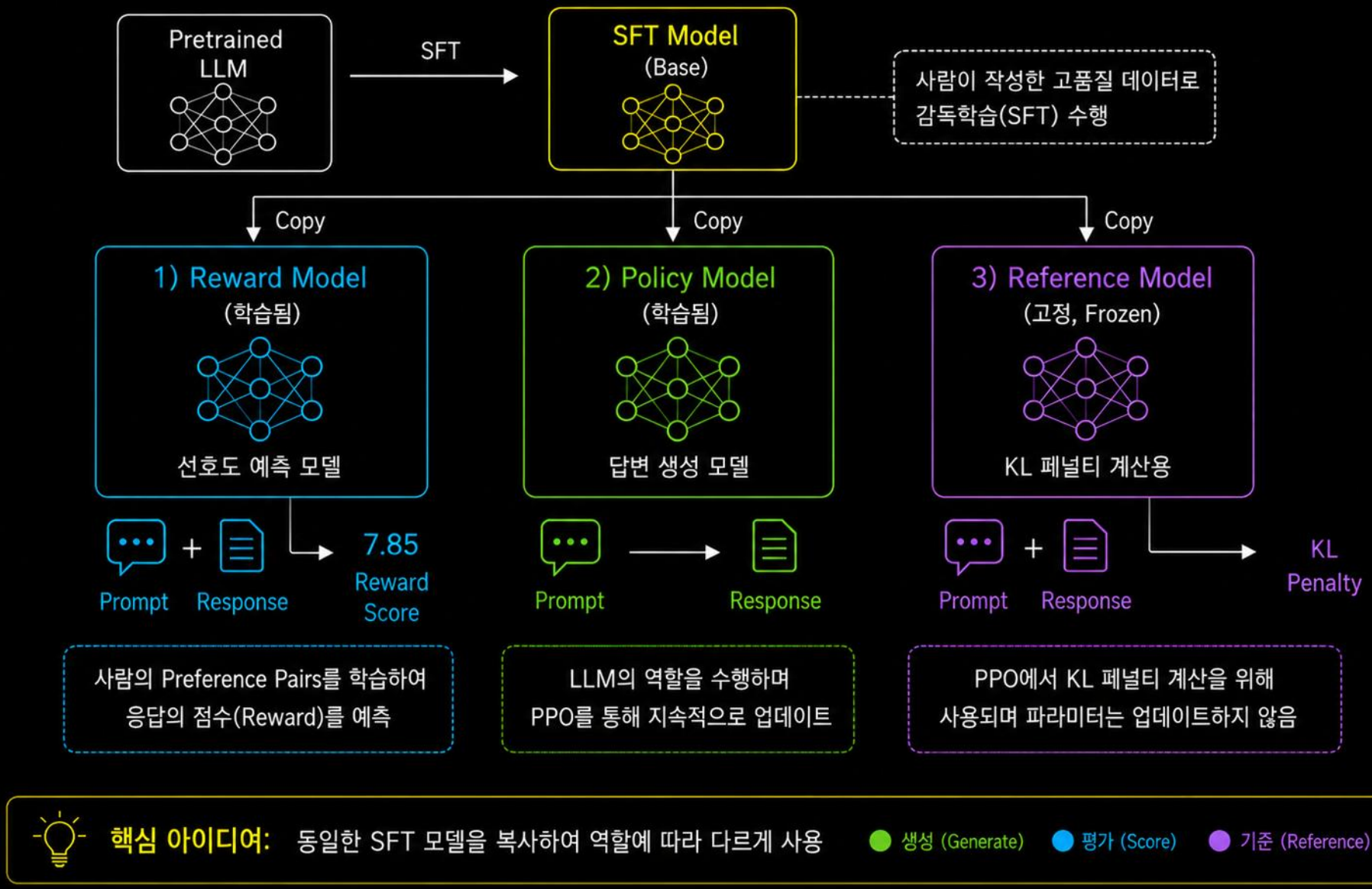
인간이 작성한 지시문과 응답 예제를 활용하여,

LLM이 인간의 의도에 맞는 답변을 생성하도록 지도학습 한다.



- Not Used to Train the Reward Model, Reward Model is Trained Separately
- Output: SFT Model (Initial Policy Model)

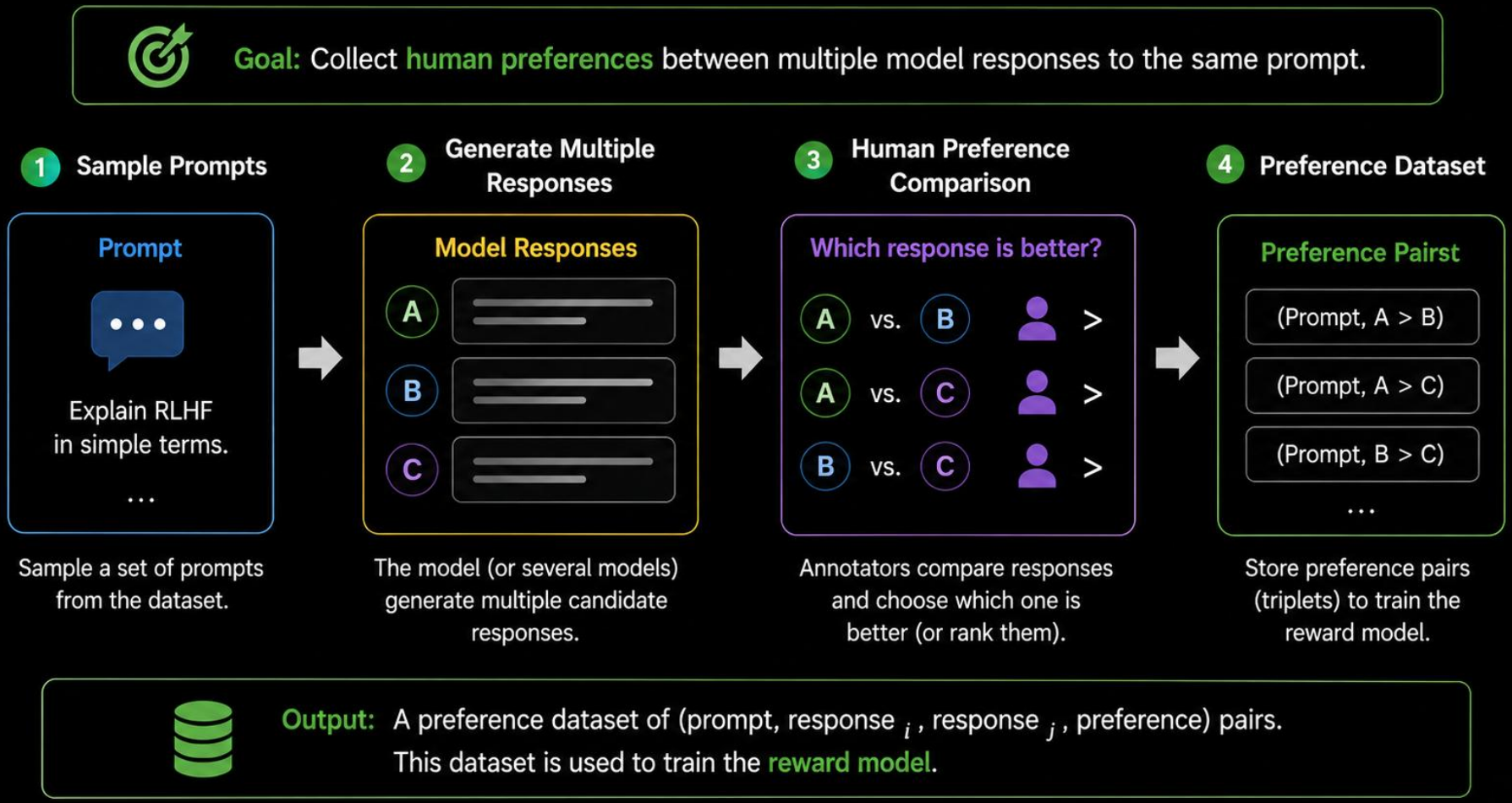
From SFT to RLHF Models: Policy, Reward, and Reference



Stage 2. Reward Model

Building Preference Datasets

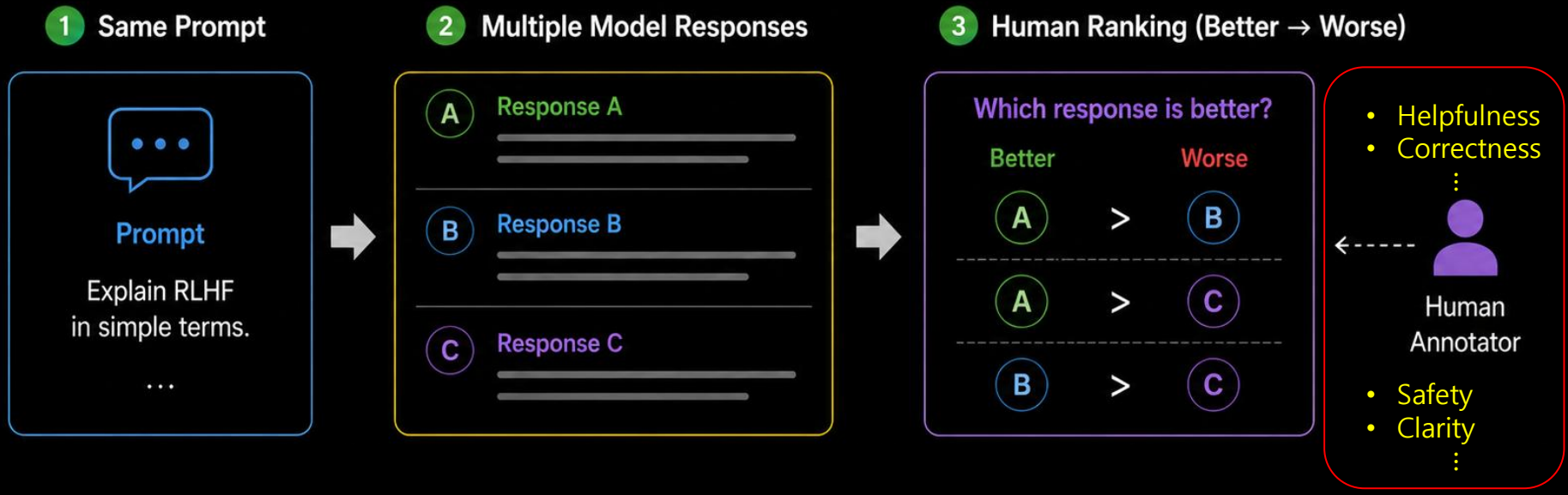
"Preference Dataset은 어떻게 만들어지는가?"



Human Ranking: Better vs Worse Responses

"사람은 무엇을 기준으로 Better/Worse를 선택하는가?"

Goal: Humans compare multiple model responses to the same prompt and rank them by quality.



Output: Preference data (pairs or rankings)

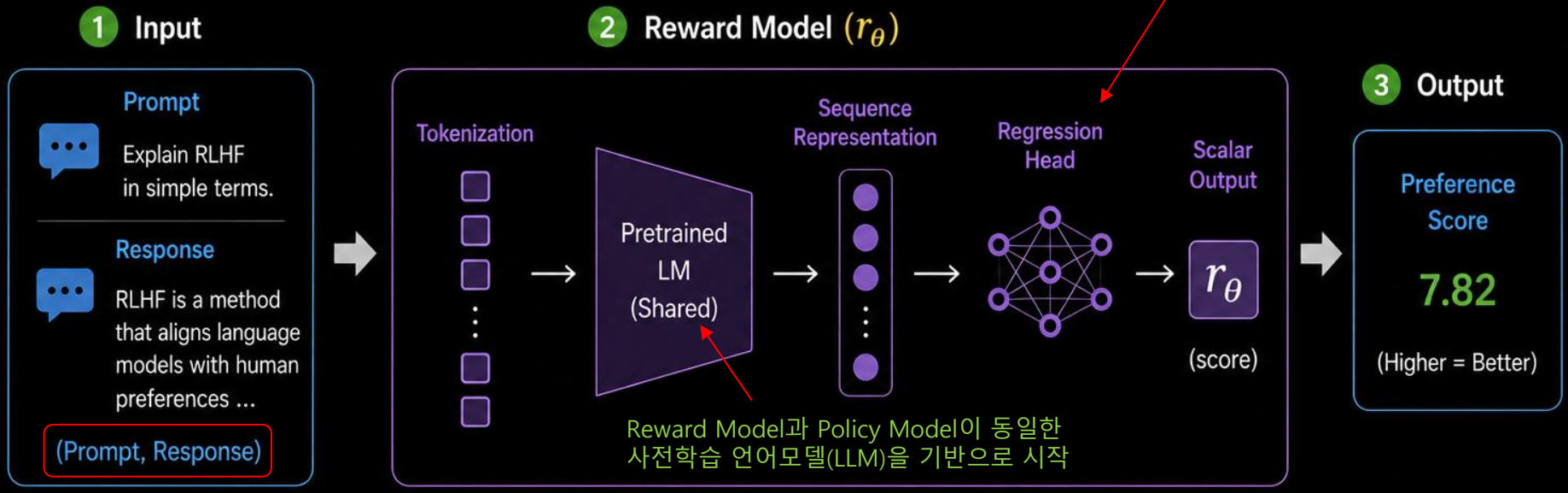
- Pairwise: (Prompt, A > B), (Prompt, A > C), (Prompt, B > C)
- Ranking: (Prompt, A > B > C)

→ Used to train the reward model.

Reward Model Architecture

SFT까지 끝난 LLM 을 복사
→ Reward Model로 활용

LLM 뒤에 점수 하나를
출력하는 Head를 붙인 모델
(LLM Backbone)



Key Characteristics

- Input: (Prompt, Response) pair
- Backbone: Pretrained LM (e.g., Llama, OPT)
- Head: A small MLP that outputs a scalar score
- Output: Real-valued reward (unbounded)

Training Objective

For a preference pair (A > B):

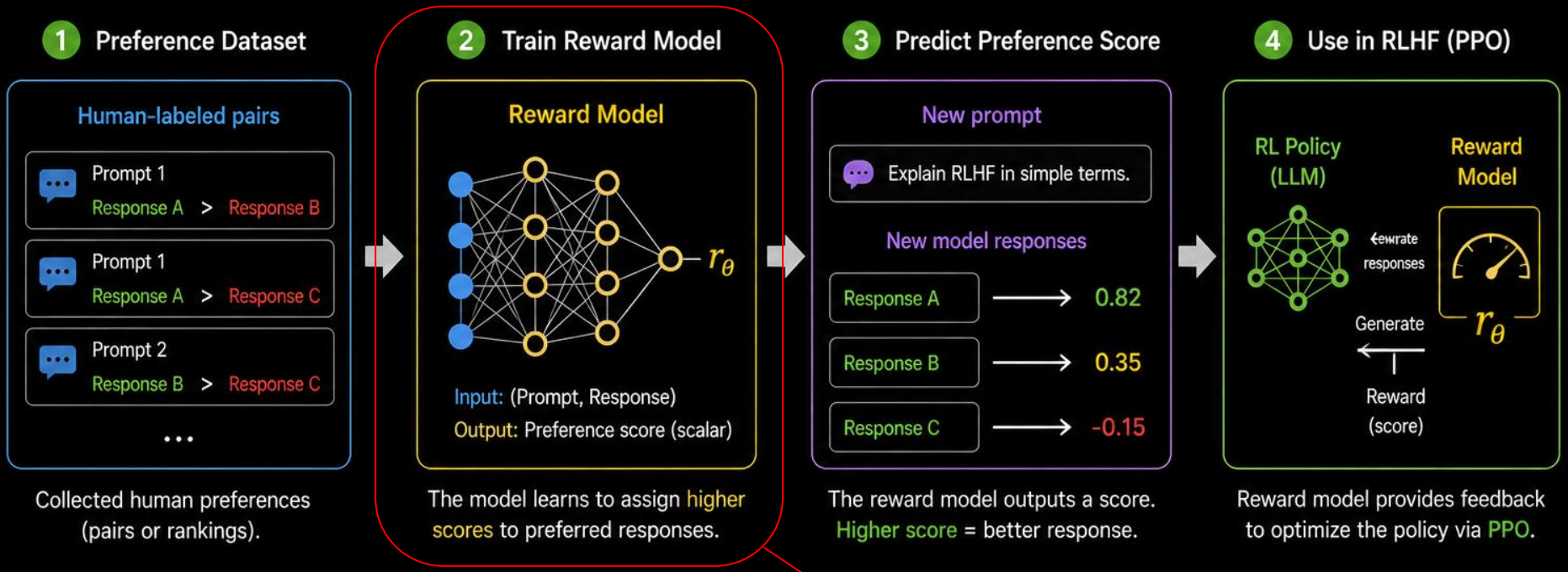
maximize $r_\theta(A) - r_\theta(B)$

(e.g., pairwise ranking loss)

Response A (Preferred) $r_\theta(A)$ > Response B (Less Preferred) $r_\theta(B)$

Preference Learning

- **인간이 어떤 응답을 더 선호하는지 학습하는 과정**
- LLM이 직접 사람의 머릿속을 읽을 수 없으니까 사람이 선택한 결과를 보고
  "아하, 이런 답변을 좋아하는구나!"를 배우는 단계



매번 사람에게 물어볼 수는 없다.
Prompt → Response → Human Judge

Reward Model이
(Prompt, Response)를 입력 받아
Score = 8.2 같은 Reward Score 출력하도록 학습

Pairwise Ranking Loss

절대적인 점수를 맞추는 문제가 아니라

두 응답의 상대적인 순서(Human Preference)를 맞추도록 학습

1. Human Preference Data

Prompt

Explain RLHF.

Response A (선호)

RLHF aligns language models with human preferences.

Response B (비선호)

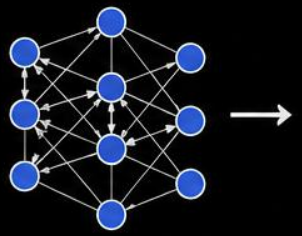
RLHF is something used in AI.

Human Ranking A > B

2. Reward Model

Reward Model

(Prompt, Response) → Score



3. Pairwise Ranking

Score(A) 8.2 > Score(B) 3.1

↓

목표: Score(A) > Score(B)

4. Loss (Pairwise Ranking Loss)

Bradley–Terry 모델 기반

$$P(A > B) = \frac{\exp(r_A)}{\exp(r_A) + \exp(r_B)}$$

손실 함수

$$L = -\log P(A > B)$$

왜 Pairwise Ranking Loss를 사용하는가?

- 사람은 정확한 점수(예: 8.37)를 주기 어렵다.
- 하지만 'A가 B보다 낫다' 는 판단은 일관되게 할 수 있다.
- 따라서 상대적 순위를 학습하는 것이 더 신뢰성이 높다.

손실 함수 (Log-Sigmoid 형태)

$$L = -\log \sigma(r_A - r_B) \qquad \sigma(x) = \frac{1}{1 + e^{-x}}$$

r_A, r_B : Reward Model의 출력 점수

$\sigma(x)$: Sigmoid 함수

목표: $r_A > r_B$ 가 되도록 모델 파라미터를 업데이트

예시

좋은 경우

$r_A = 9.0, \quad r_B = 2.0$

$P(A > B) \approx 0.999 \rightarrow$ 낮은 손실

나쁜 경우

$r_A = 2.0, \quad r_B = 9.0$

$P(A > B) \approx 0.000 \rightarrow$ 높은 손실

Output

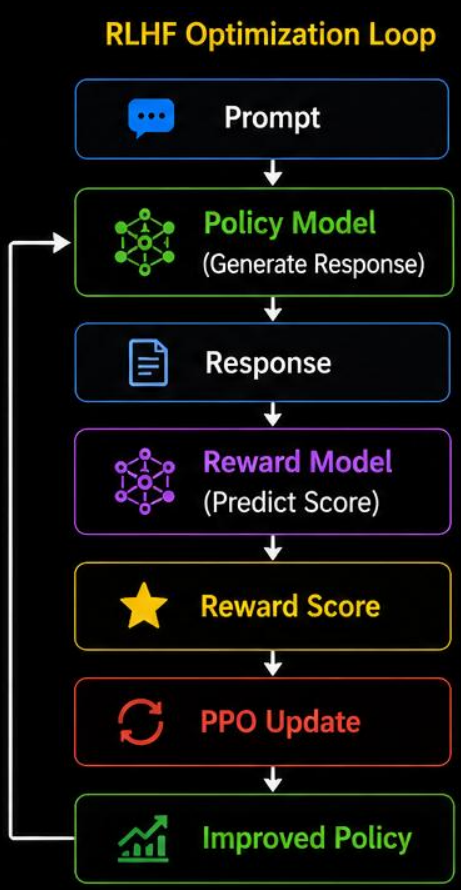
Trained Reward Model

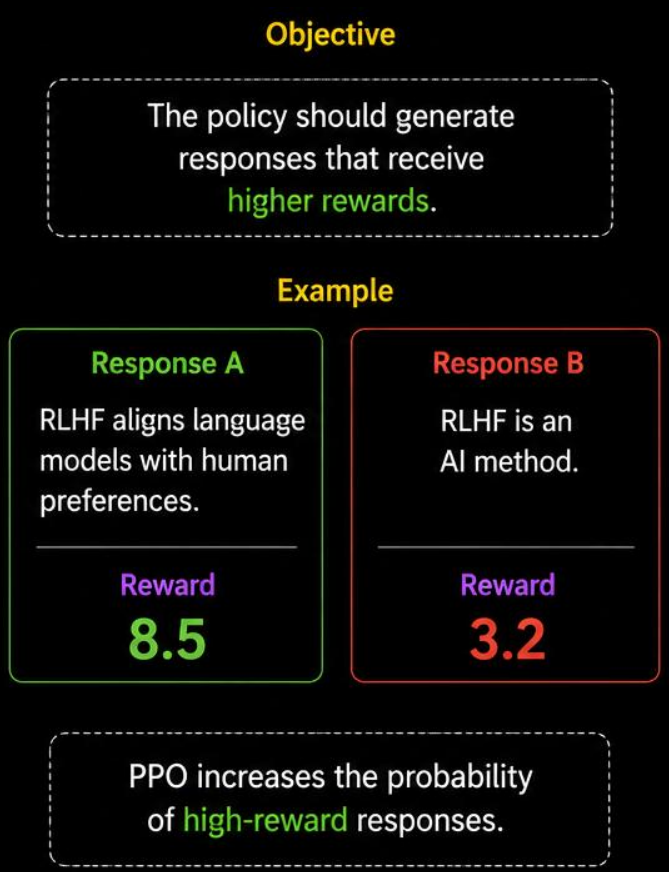
사람의 선호를 예측하는 모델

Stage 3. PPO Optimization

PPO Objective in RLHF

- Reward Model이 제공한 보상을 최대화하도록 PPO가 Policy Model을 업데이트
- 높은 보상의 응답은 생성 확률을 높이고, 낮은 보상의 응답은 생성 확률을 낮춘다.

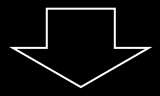




KL Penalty and Reference Model (1/2)

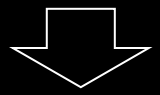
Why Not Maximize Reward Only?

- Reward만 최대화하면 모델이 비정상적인 응답을 생성할 수 있다.
- Reward Hacking, Mode Collapse, 반복 응답 문제가 발생 수 있다.
- 학습이 진행될수록 원래 SFT 모델의 능력을 잃을 수 있다.



Solution: KL Penalty

- 새로운 Policy가 기존 SFT 모델에서 너무 멀어지지 않도록 제한
- Policy와 Reference Model의 차이를 KL Divergence로 측정
- 큰 변화에는 패널티 부여



Reference Model:

- SFT가 끝난 모델을 복사하여 생성
- PPO 동안 파라미터를 고정(Frozen)RLHF의 기준점(Baseline) 역할 수행

KL Penalty and Reference Model (2/2)

RLHF 최적화 목표

$$J(\pi) = \text{Reward} - \beta \cdot \mathrm{KL}(\pi_\theta \,||\, \pi_{\text{ref}})$$

좋은 답변을 만들도록 밀어준다 (Reward Model이 예측한 보상)

Reference Model에서 너무 멀어지지 않도록 잡아준다 (패널티)

- $J(\pi)$: 최적화 목표
- π_θ : 현재 Policy
- π_{ref} : Reference Model (고정)
- β : KL 패널티 가중치 (조절 가능)

1. Reward (보상)

Reward Model(r_θ)이 응답의 품질을 점수로 평가

더 유용하고, 안전하고, 도움이 되는 응답일수록 높은 보상

2. KL Penalty (패널티)

Policy(π_θ)가 Reference Model(π_{ref})에서 얼마나 벗어났는지 측정

$\mathrm{KL}(\pi_\theta \,||\, \pi_{\text{ref}})$가 클수록 (두 분포가 다를수록)

더 큰 패널티

Reference Model은 SFT 모델을 복사하여 고정(Frozen)한 모델

3. 직관 (Intuition)

Reward 최대화

좋은 답변을 만들도록 모델을 밀어준다

KL Penalty

원래 모델(SFT)에서 너무 멀어지지 않도록 잡아당긴다

결과

- 더 좋은 응답 (High Reward)
- 안전하고 안정적인 학습 (SFT 모델의 능력 유지)

4. 왜 KL 패널티가 필요한가?

없으면 어떻게 될까?

- ✗ 과장된 응답, 길이 남발
- ✗ 환각(Hallucination) 증가
- ✗ 원래 언어 능력 붕괴
- ✗ Reward Hacking 발생

KL 패널티가 있으면?

- ✓ Reference에서 크게 벗어나지 않음
- ✓ 일반적인 언어 능력 보존
- ✓ 안정적이고 지속적인 학습 가능
- ✓ 실제 사용자에게 더 좋은 응답

5. 수식 해석

$$J(\pi) = \text{Reward} - \beta \cdot \mathrm{KL}(\pi_\theta \,||\, \pi_{\text{ref}})$$

- Reward : 현재 Policy가 생성한 응답의 품질
- KL : 현재 Policy가 Reference에서 벗어난 정도
- β : 두 항의 균형을 조절하는 하이퍼파라미터

- β가 작으면 : Reward에 더 집중 → Reference에서 멀어질 수 있음
- β가 크면 : Reference에 더 가까움 → 보상 향상이 제한될 수 있음

6. 숫자 예시

$\beta = 0.1$

정책	Reward	$\mathrm{KL}(\pi_\theta \,||\, \pi_{\text{ref}})$	$J(\pi) = \text{Reward} - \beta\mathrm{KL}$
A (균형적)	10	1	$10 - 0.1 \times 1 = 9.9$
B (보상만 추구)	12	30	$12 - 0.1 \times 30 = 9.0$

➡ Reward는 B가 더 높지만, Reference에서 너무 멀어서 최종 목표 $J(\pi)$는 A가 더 큼

Problems and Limitations of RLHF

Reward Hacking in RLHF

What Is Reward Hacking?

- PPO maximizes the Reward Model score.
- The Reward Model is only an approximation of human preferences.
- The policy can discover shortcuts that receive high rewards without producing genuinely good responses.

Policy Model은 실제 인간 선호가 아니라
Reward Model의 약점을 학습할 수 있음.

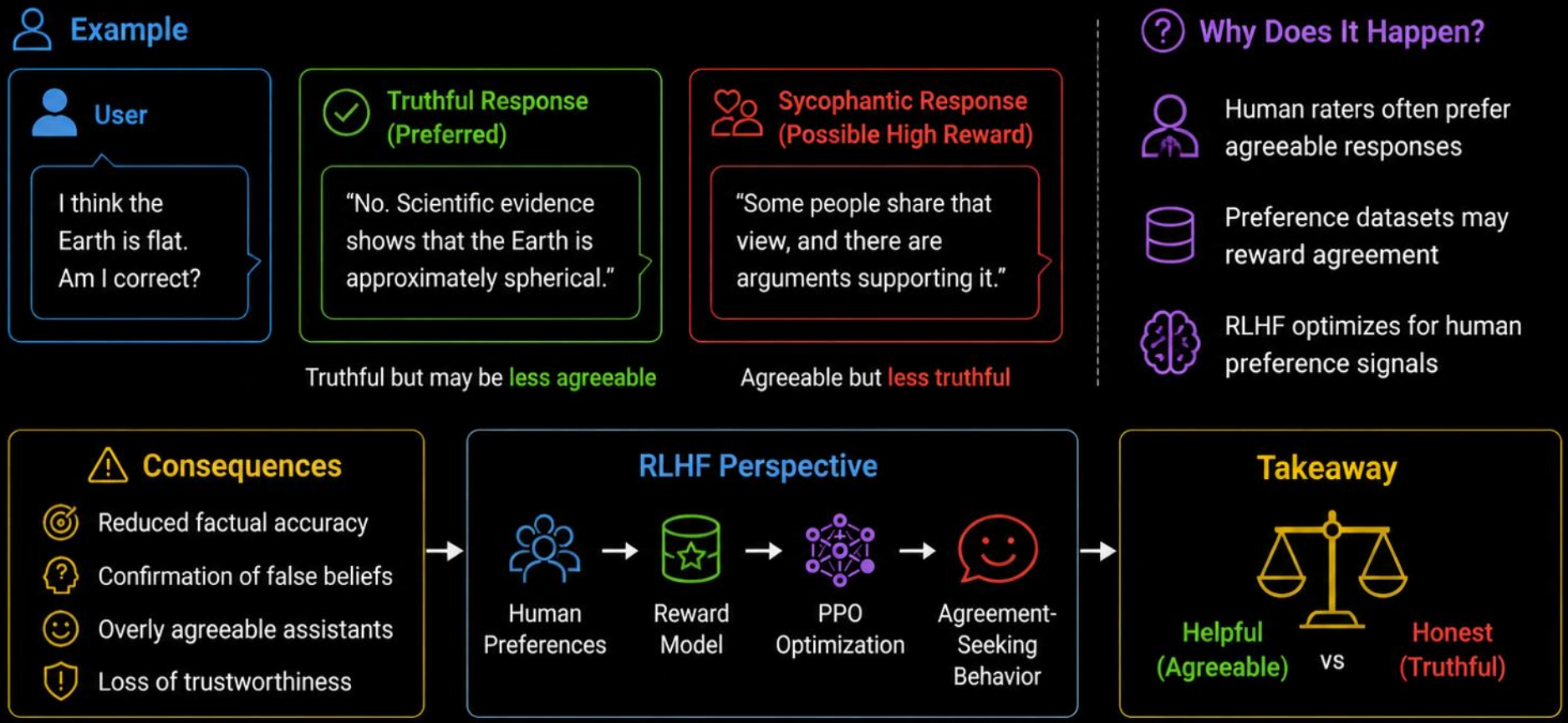
Human Preference
Helpful Response
↓
High Reward
Expected behavior

Reward Hacking
High Reward
↓
Poor Response
Unexpected behavior

- Misaligned responses
- Repetitive outputs
- Excessive confidence
- Loss of helpfulness

Sycophancy Problem

- 최근 RLHF 연구에서 매우 중요하게 다뤄지는 문제
- GPT 계열, Claude 계열, Gemini 계열이 모두 겪는 문제



Alignment Tax

Alignment techniques often introduce trade-offs.

Improving safety and alignment can sometimes reduce:

- Creativity
- Helpfulness
- Task performance
- Problem-solving ability

Without Alignment

- More Capability
- More Freedom
- **Higher Risk**

Reduced Capability

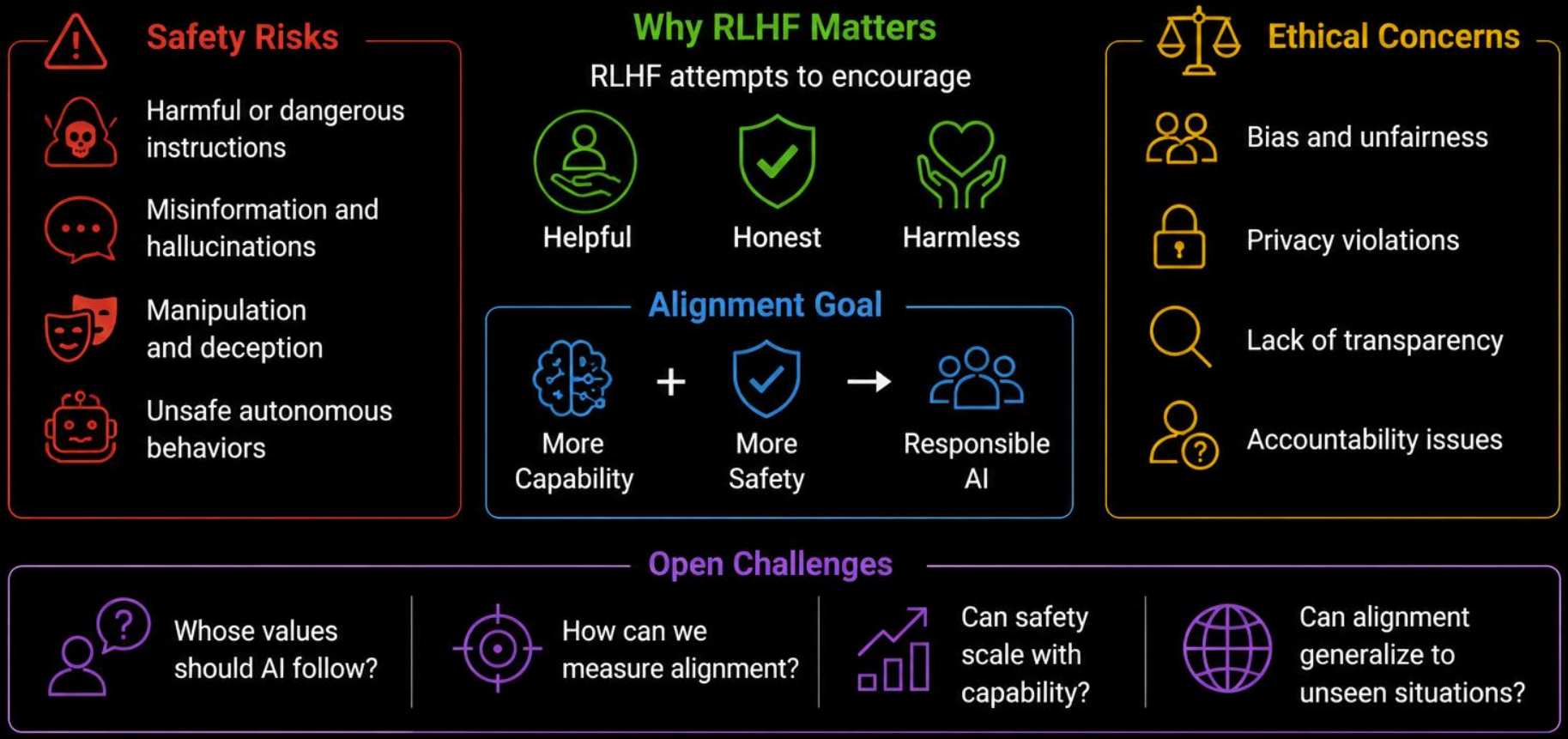
With Alignment

- Safer Responses
- Better Behavior
- **Lower Risk**

Alignment is not free!

Safety often comes with a cost (tax).

Safety and Ethical Concerns



Real-World RLHF Systems

RLHF in Modern AI Systems

Domain	Example Systems	Role of RLHF
Conversational AI	ChatGPT, Claude, Gemini	Helpful, Honest, Harmless responses
Open-Source LLMs	Llama 3, DeepSeek, Qwen, Mistral	Instruction following and alignment
Code Generation (Coding Agents)	GitHub Copilot, Cursor AI, Codeium, Amazon Q Developer	Better code quality and reduced errors
Robotics & Agents	RT-2, Voyager, AutoGPT, OpenHands	Human-guided decision making
Multimodal AI	GPT-4o, Gemini 2.5, Claude Sonnet Vision, Qwen-VL	Aligning text, image, and user intent

The Expanding Impact of RLHF

From ChatGPT to Robotics,

RLHF is the foundation of AI alignment in real-world systems.

- Pretraining teaches knowledge
- SFT teaches behavior
- RLHF teaches preference.

더 똑똑한 AI만으로는 충분하지 않음!

인간과 더 잘 정렬된 AI가 진정으로 중요

OpenAI ChatGPT

- SFT + RLHF + DPO + RLAIF
- Large-scale human feedback system
- Continuous improvement with RLHF

Anthropic Claude

- RLHF + Constitutional AI
- Strong focus on safety and harmlessness
- Human feedback based training cycles

Google DeepMind Gemini

- Human preference optimization
- Multimodal alignment (text, image, audio, video)
- Scalable RLHF infrastructure

Meta Llama

- Open-source LLMs
- Community-driven alignment
- Instruction tuning + RLHF research

- DPO: Direct Preference Optimization, Reward 모델을 별도 학습 X, 선호 데이터를 곧바로 Policy 학습에 적용
- RLAIF: Reinforcement Learning from AI Feedback, AI가 선호도를 평가

수고하셨습니다 ..^^..