Deep Learning (1470)

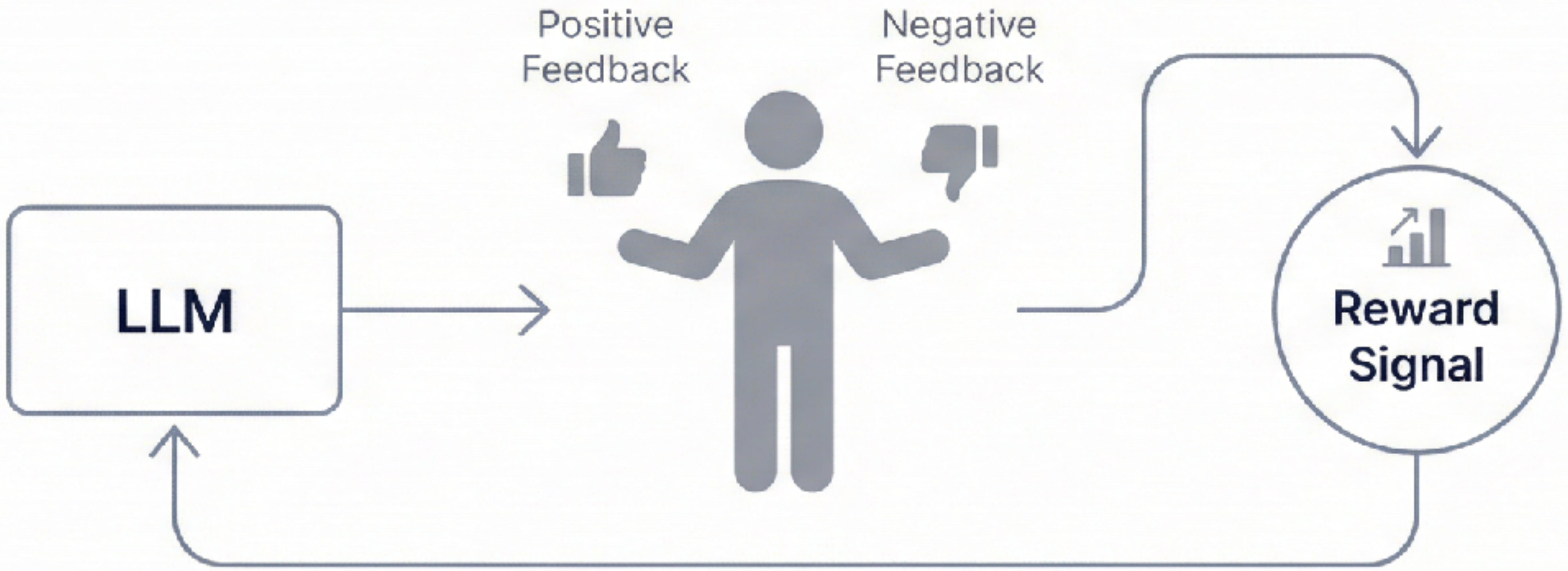
Randall Balestriero

Class 22: GRPO and Applications

Recap!

Reinforcement Learning from Human Feedback (RLHF) for Large Language Models

Bridging RL Theory and LLM Fine-tuning



Why?

Why RLHF? The Alignment Problem

Pre-trained LLMs CAN:	**But they MIGHT:**
✓ Generate fluent text	✗ Give harmful advice
✓ Complete patterns	✗ Hallucinate facts
✓ Store knowledge	✗ Be unhelpful

RLHF Goal: Align model outputs with human values and preferences

The RLHF Pipeline: Three Stages

STAGE 1

Supervised Fine-Tuning (SFT)

Train on high-quality demonstrations

STAGE 2

Reward Model Training

Learn to predict human preferences

STAGE 3

RL Fine-Tuning (PPO)

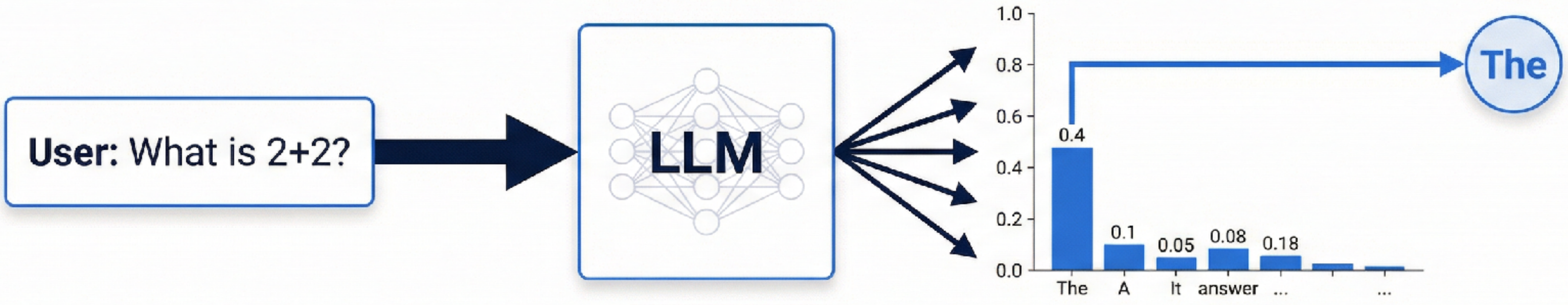
Optimize policy using reward model

From Toy RL to LLM RL: Key Differences

Toy RL (e.g., CartPole, Atari)	LLM RLHF
State: Low-dimensional vectors or pixels	State: Prompt text (high-dimensional embeddings)
Action: Discrete (left/right) or continuous	Action: Next token from vocabulary of 50K+ tokens
Episode: Hundreds of steps	Episode: One response generation (variable length)
Reward: Immediate, well-defined	Reward: Single score at end from reward model
Policy: Small neural network	Policy: Billions of parameters

SCALE-UP & TRANSITION

The Action Space: Tokens as Actions



Autoregressive Generation Process

Step 1: State = "What is 2+2?" → Action = "**The**" (sampled)

Step 2: State = "What is 2+2? The" → Action = "**answer**" (sampled)

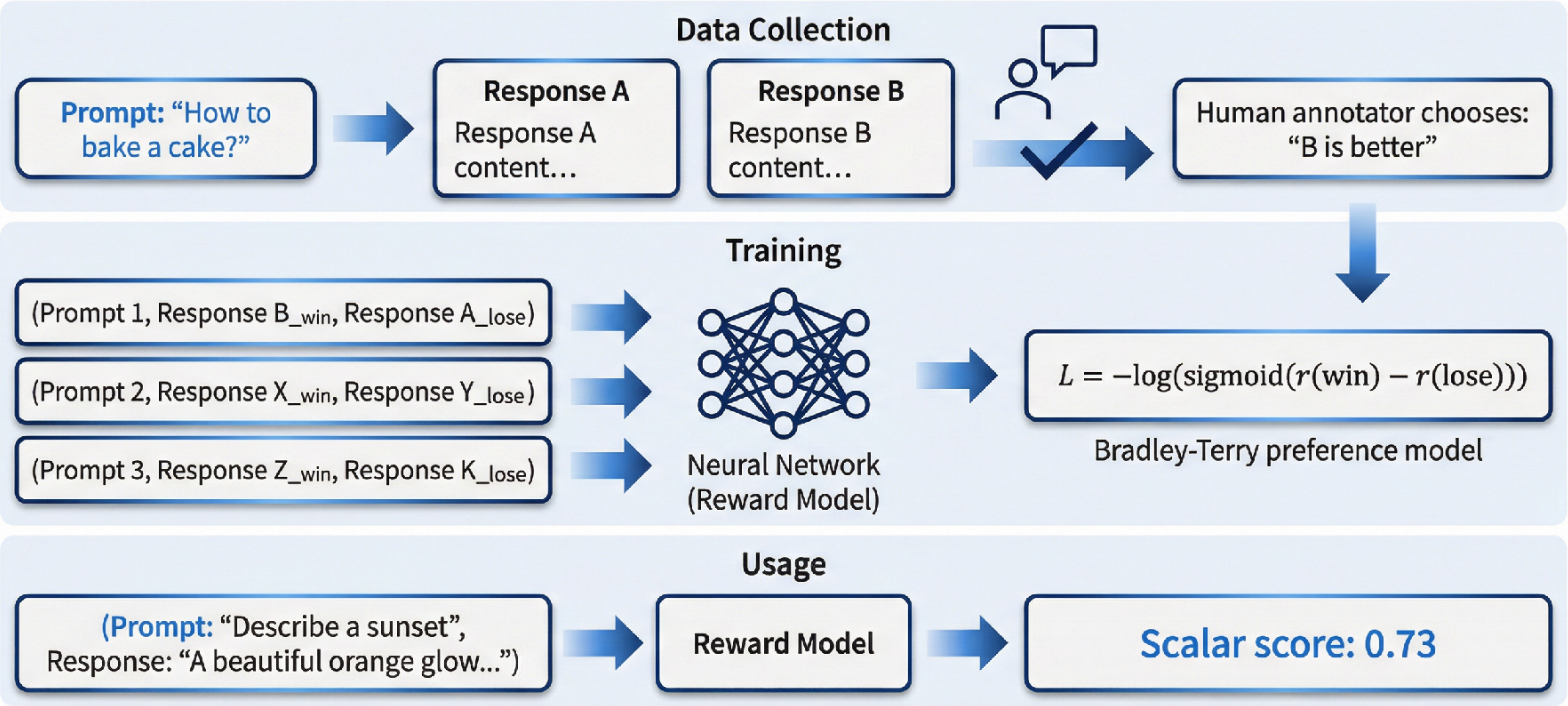
Step 3: State = "What is 2+2? The answer" → Action = "**is**" (sampled)

Step 4: ... → Action = "**4**"

Key insight: Each token selection is an ACTION. The policy is the LLM itself:

$$p_i(\text{action}|\text{state}) = P(\text{next\_token}|\text{context})$$

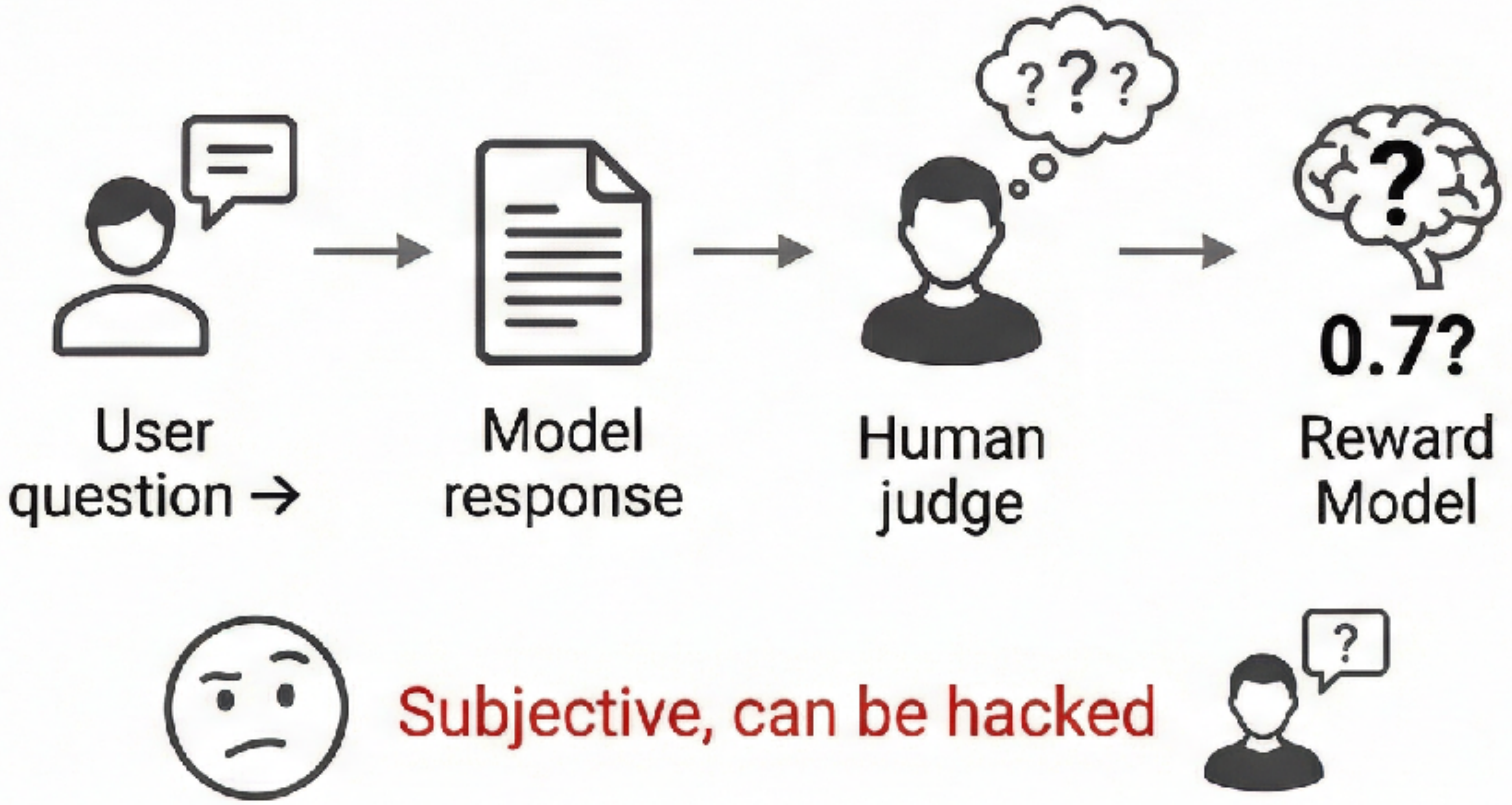
The Reward Model: Learning Human Preferences



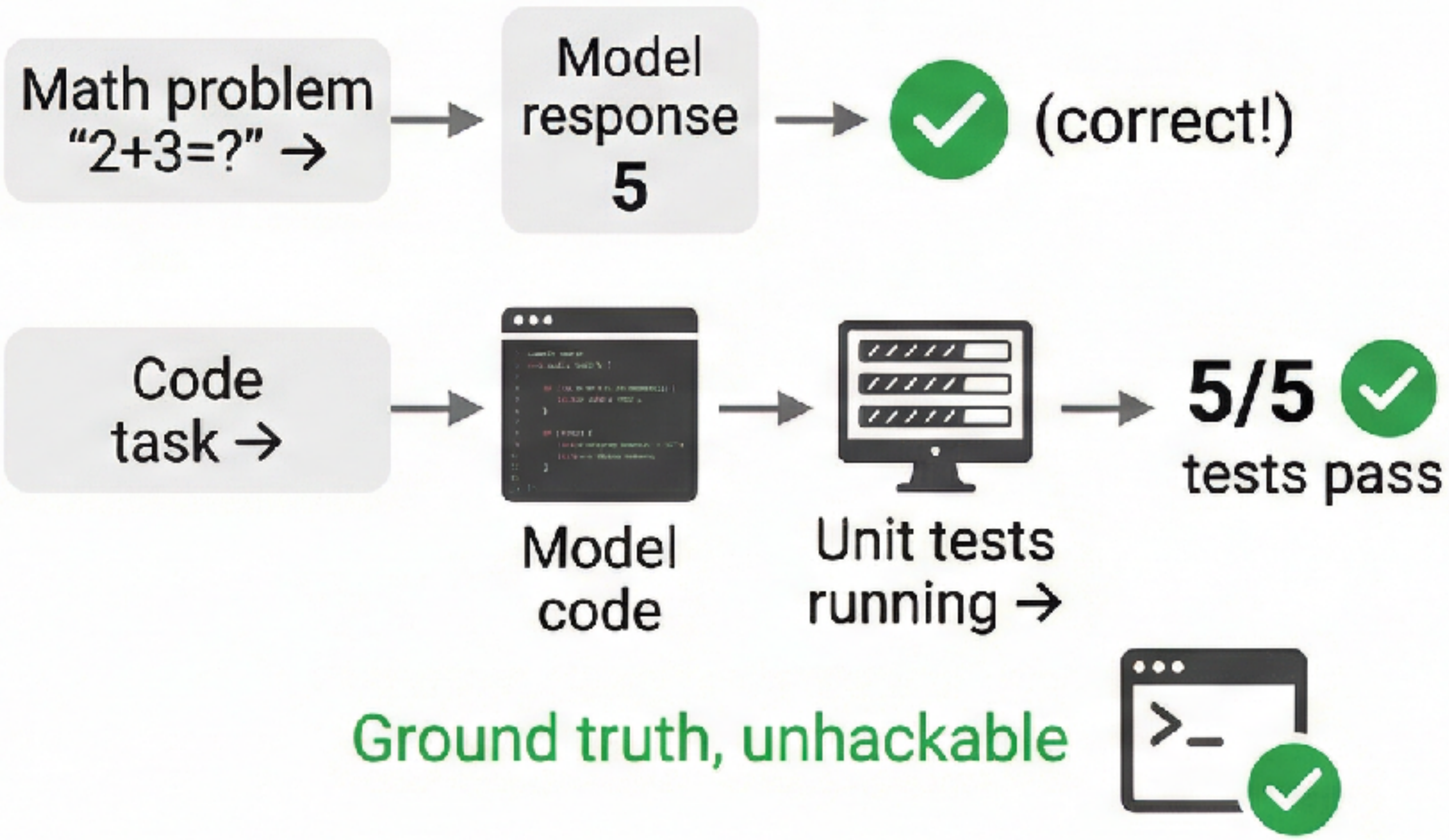
The reward model learns to predict which responses humans prefer, then provides reward signal during RL training

Verifiable Rewards: Math & Code

Standard RLHF (Chat)



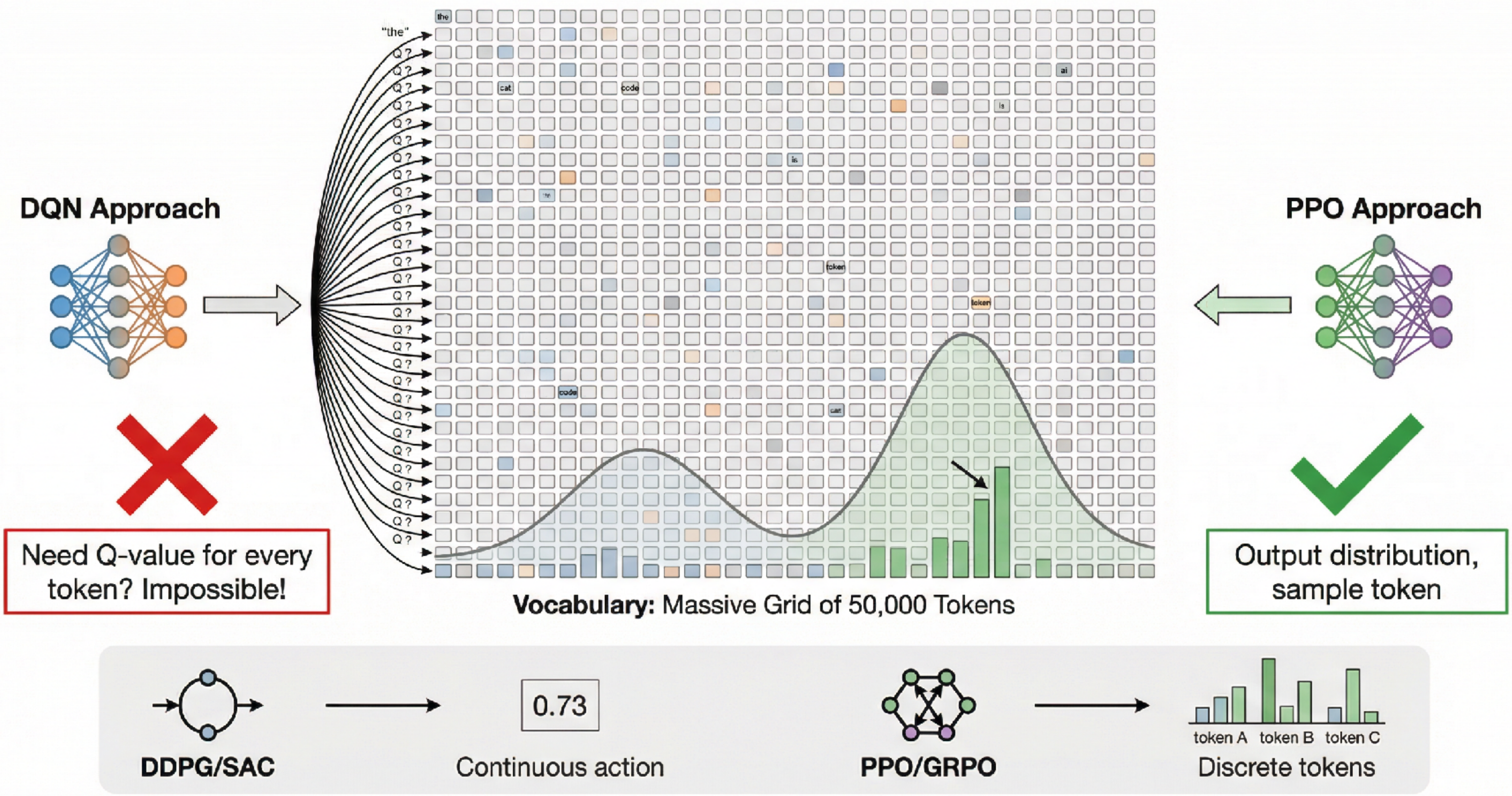
Verifiable Rewards



R1 Trained heavily on math & code with verifiable rewards → **Better reasoning on ALL tasks!**

The message: **verifiable domains** enable **massive scale RL.**

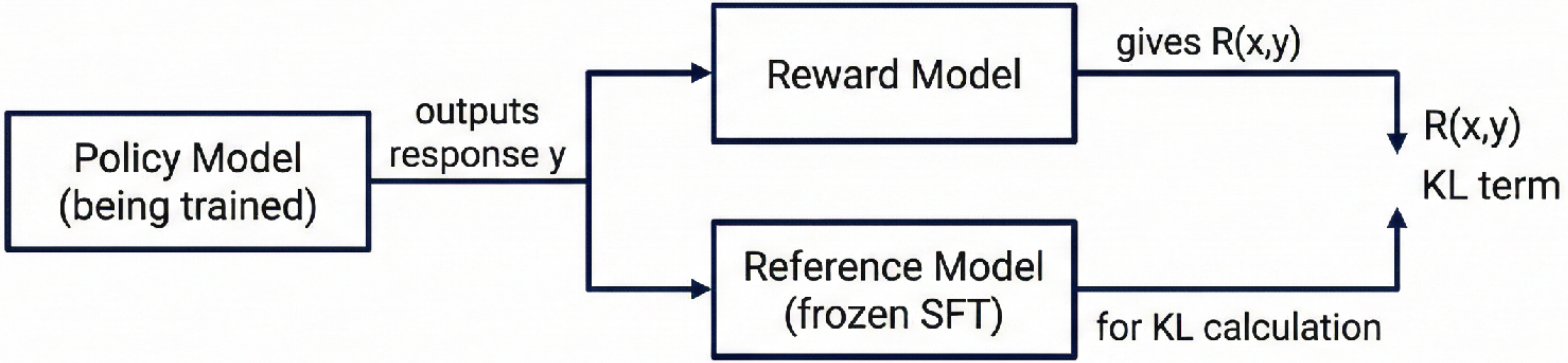
Why Not Other RL Methods?



PPO Objective for LLMs

$$L_{RLHF} = E[R(x,y) - \beta * KL(\pi_\theta \,||\, \pi_{ref})]$$

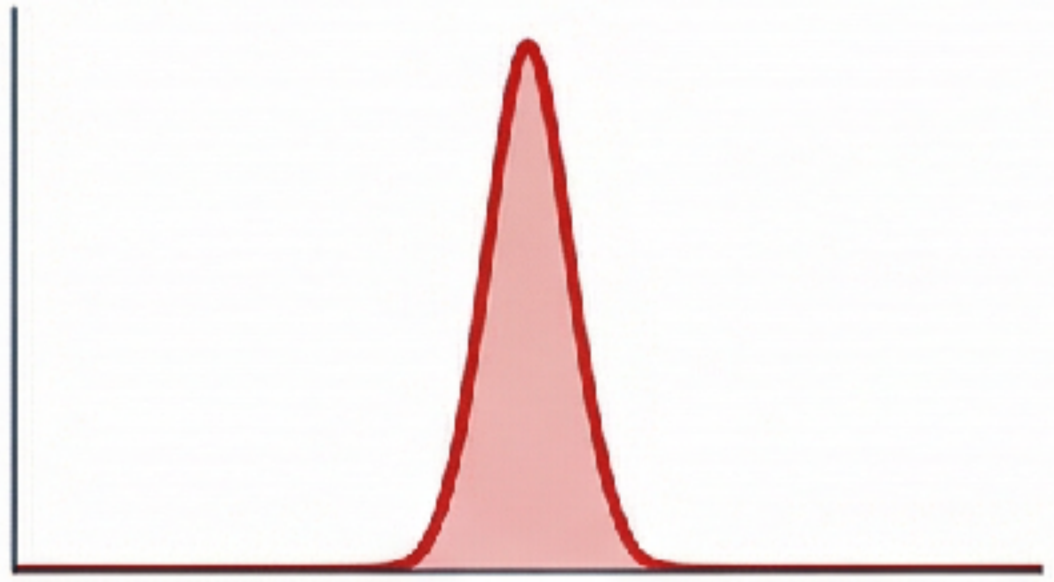
Where:
- $R(x,y)$ = reward model score for prompt x and response y
- KL term = divergence from reference policy (original SFT model)
- β = KL penalty coefficient (typically 0.01 to 0.1)



Key insight: The KL penalty prevents the policy from diverging too far from the original model, maintaining coherent language generation

The KL Penalty: Preventing Mode Collapse

Without KL Penalty ❌ (Bad)



Narrow distribution collapsing to single mode

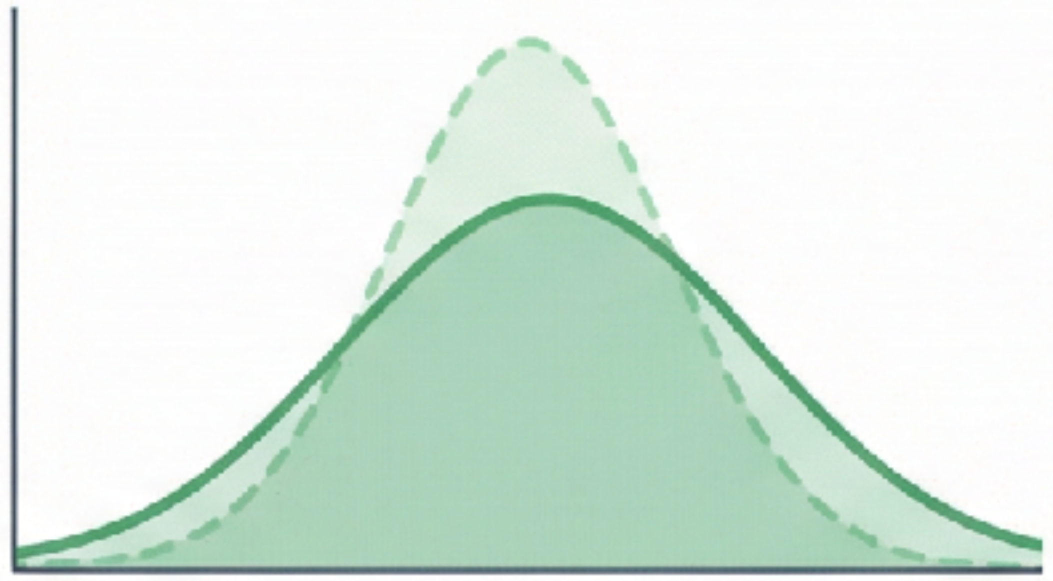
Example outputs all become similar:

- Sure! Here is...
- Sure! Here is...
- Sure! Here is...

⚠️ **Red warning:** "Model finds reward hacks, loses diversity"

The model exploits the reward model

With KL Penalty ✅ (Good)



Distribution staying broad, similar to reference

Example outputs remain **diverse and natural:**

- Sure, I can help with that...
- Absolutely, here's a thought...
- Okay, let's explore that option...

✅ **Green checkmark:** "Maintains language quality"

Stays close to pre-trained capabilities

Practical tip

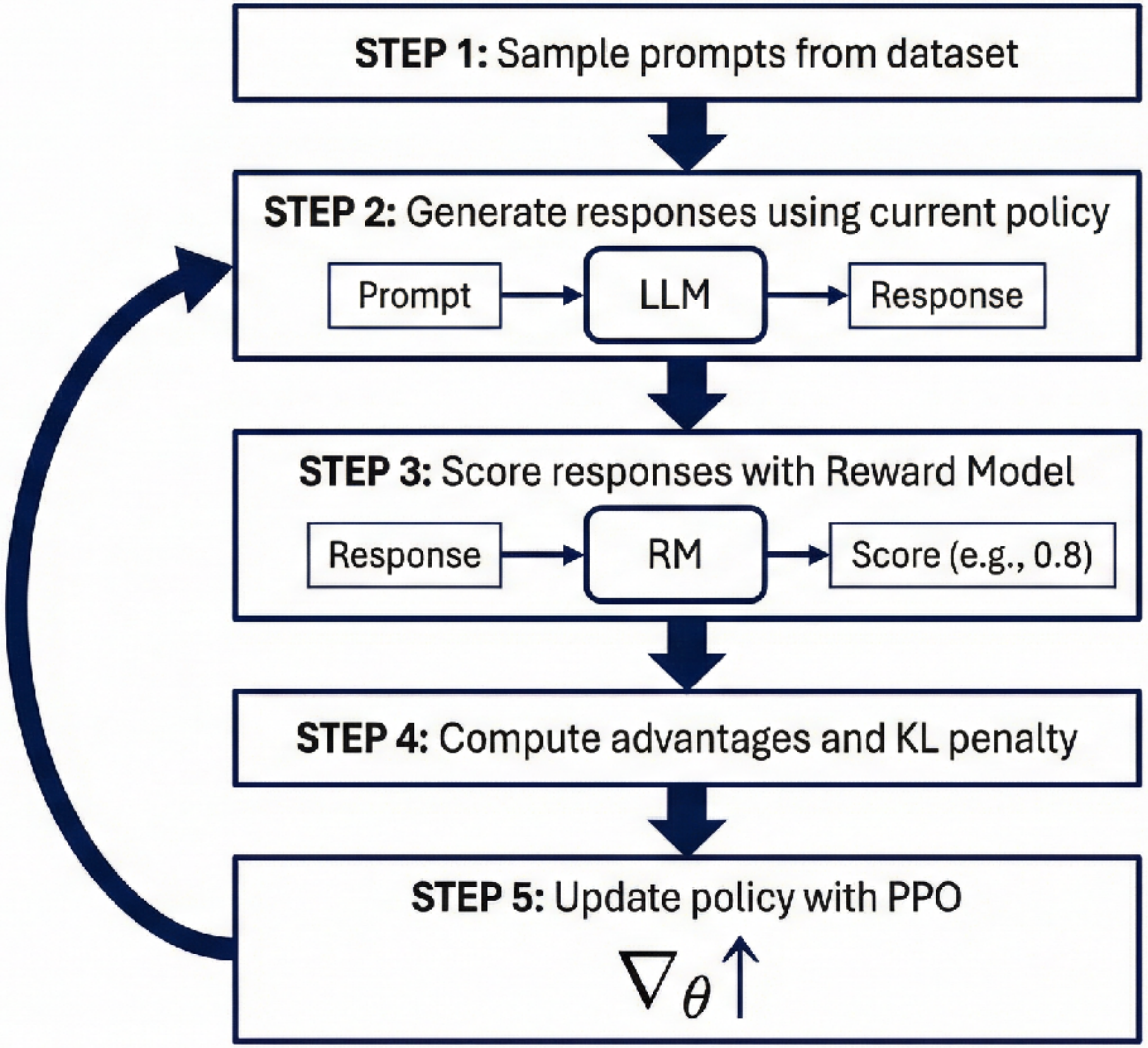
Start with beta=0.01, increase if model degrades in quality

How KL is computed:

$$\mathrm{KL} = \sum_{\text{tokens:}} \log(\pi_\theta(\text{token})) - \log(\pi_{\text{ref}}(\text{token}))$$

Penalizes when new policy assigns very different probabilities than reference

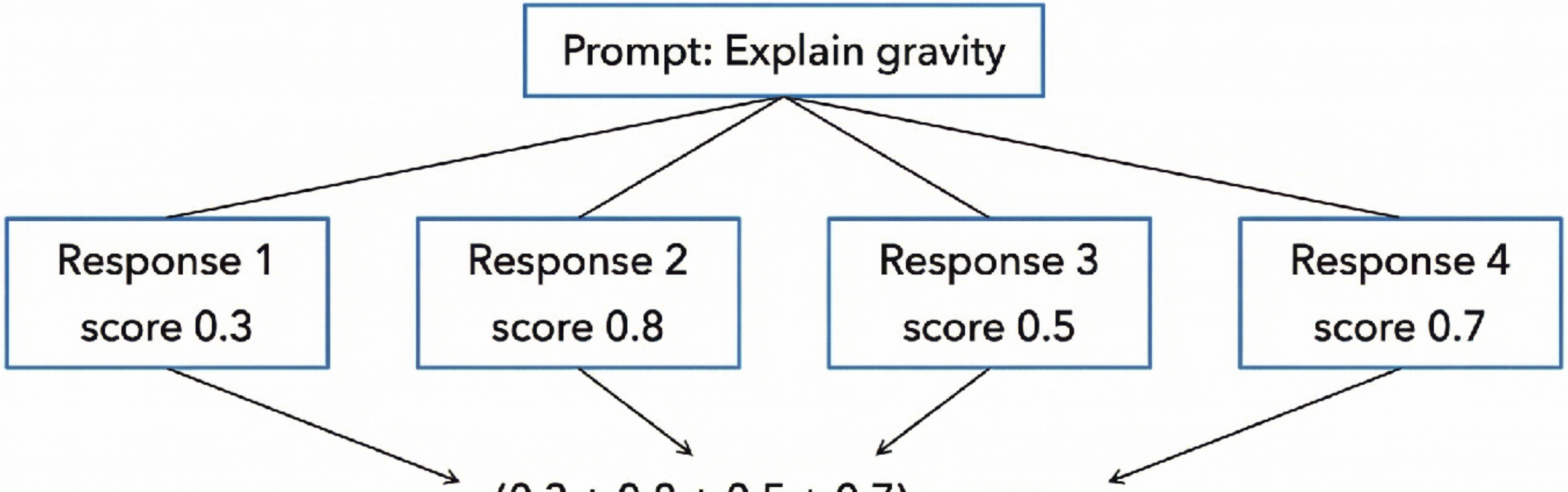
RLHF Training Loop: Step by Step



Batch Sizes:
- Batch of prompts: 32-128
- Responses per prompt: 1-4
- PPO epochs per batch: 1-4

But How Many Models We Have?

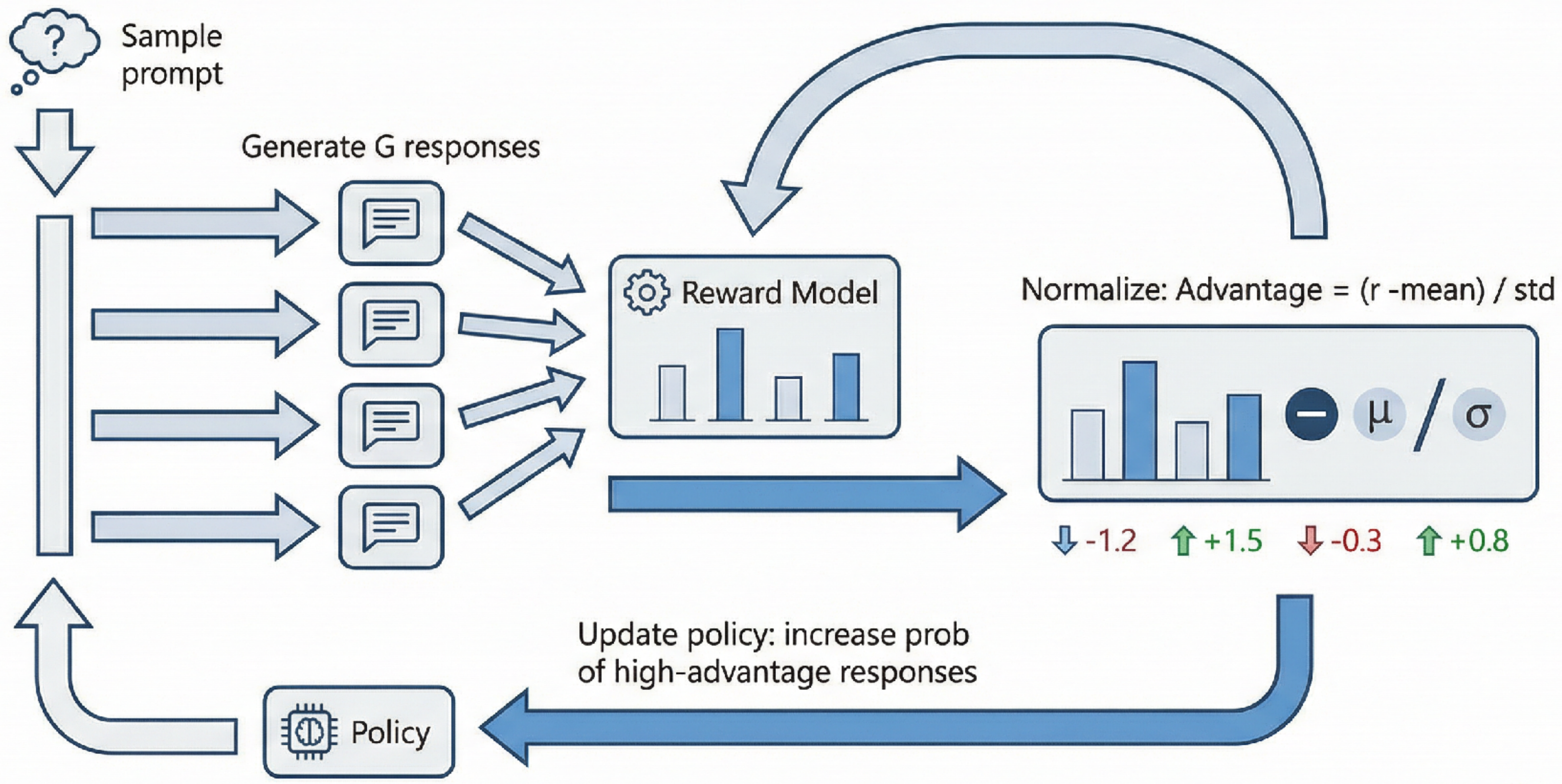
GRPO: No Critic Needed



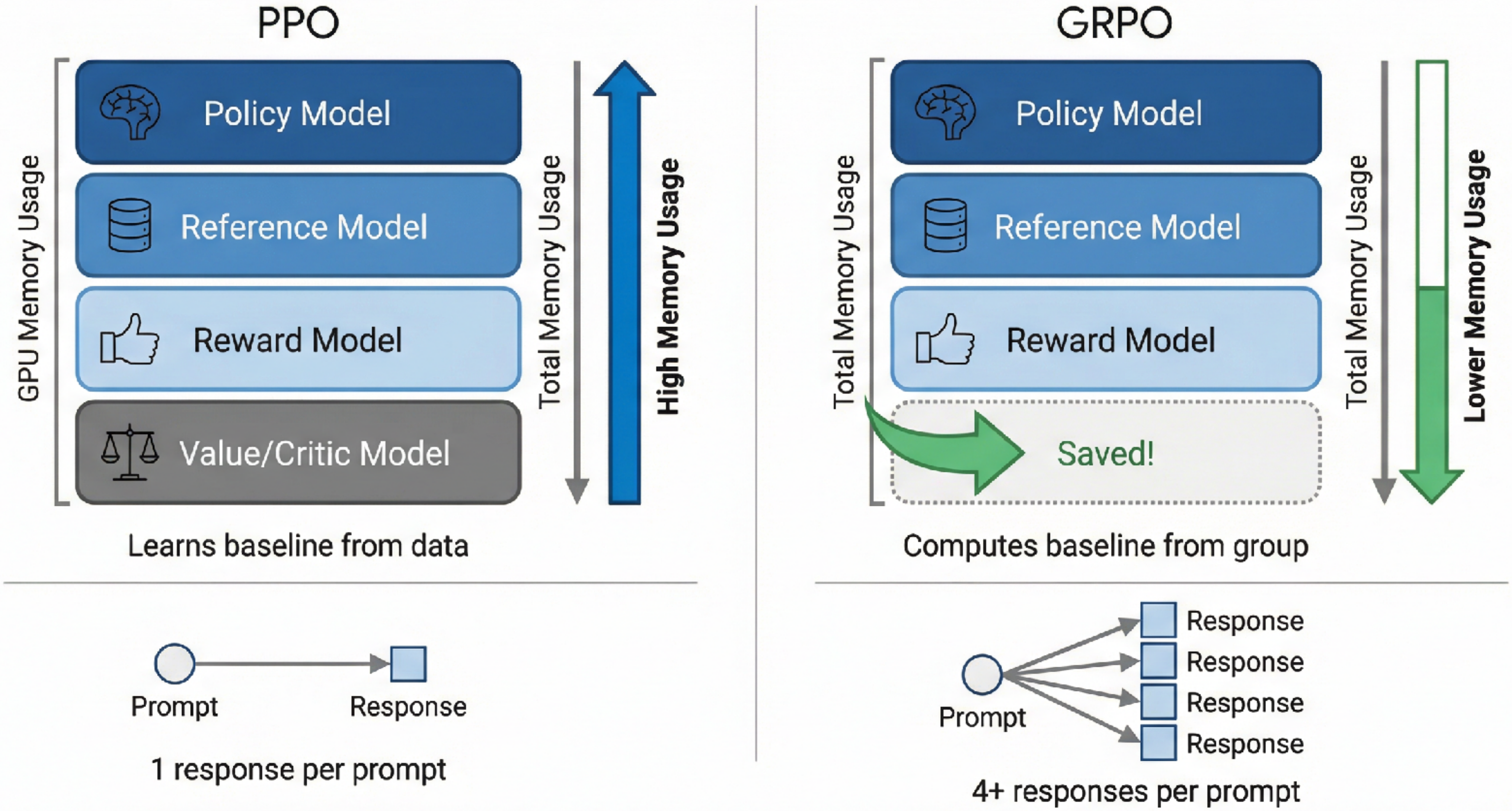
$$\frac{(0.3 + 0.8 + 0.5 + 0.7)}{4} = 0.575$$

Key visual insight: The group of responses provides the baseline naturally.

GRPO: How It Works



GRPO vs PPO: Visual Comparison

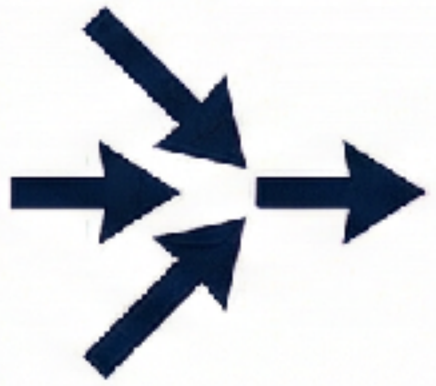


Common Pitfalls and Solutions

PITFALL 1: "Reward Hacking"

Problem: Model finds shortcuts to maximize reward (e.g., excessive flattery, repetition)

Solution: Increase KL penalty, improve reward model diversity



PITFALL 2: "Mode Collapse"

Problem: All outputs become similar, lose diversity

Solution: Lower learning rate, increase KL coefficient, use entropy bonus

PITFALL 3: "Forgetting / Capability Loss"

Problem: Model loses abilities it had after SFT

Solution: Mix in SFT data during RL, use smaller learning rate

PITFALL 4: "Unstable Training"

Problem: Loss spikes, reward oscillates wildly

Solution: Gradient clipping, warmup, smaller PPO epochs

See you on Monday!