Reward Modeling for Large Language Models

A way to unlock LLM's superior performance

Yudi Zhang, Meng Fang, Mykola Pechenizkiy

March 23, 2026

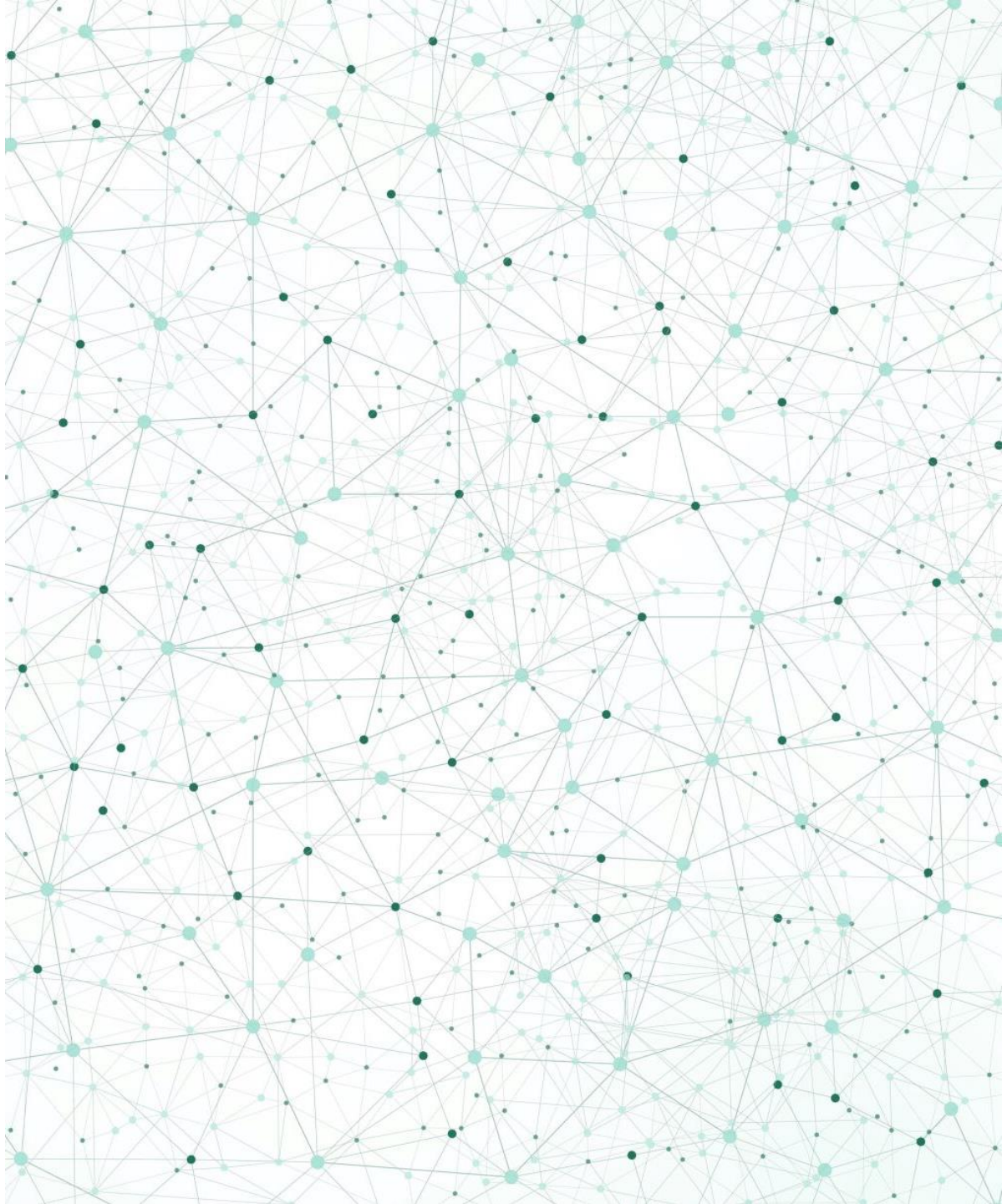
OUTLINE

	Chapters
25 min	Introduction: How Reward Models Are Used in LLMs
25 min	Reward Modeling from Preference: RLHF and DPO
20 min	Verifiable Rewards & RLVR
10 min	(Break)
15 min	Outcome or Process Reward Model: From Final Answers to Reasoning
20 min	LLM-as-Rewards: Judge Models and Language-Based Feedback
20 min	Unifying View, Evaluation, and Open Challenges

Part I Introduction

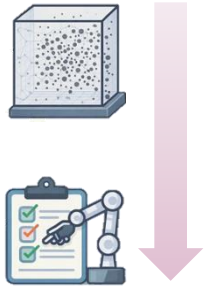
How Reward Models Are Used in LLMs?

Speaker: Yudi Zhang



BACKGROUND ON LLMS

- Development of LLMs



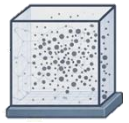
Stage 1: Generate fluent language via Pre-training
Learn general language patterns and world knowledge from large-scale unlabeled text.

Stage 2: Follow user instructions via Instruction-Tuning
Train on high-quality instruction–response data.

- Limitations of Pure Likelihood Training (Stage 1 & Stage 2)
  - Likelihood of text ≠ helpful or correct answers.
  - Trained LLMs may output misinformation, bias, and unsafe content.

BACKGROUND ON LLMS

- Development of LLMs

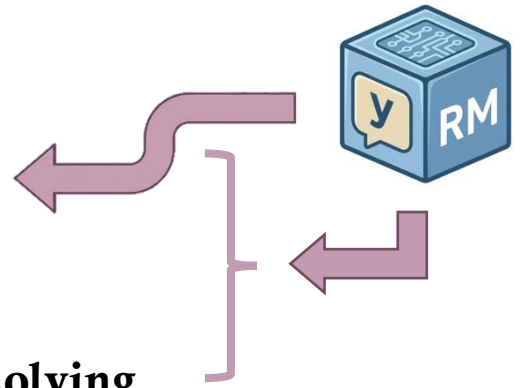


Stage 1: Generate fluent language via Pre-training
Learn general language patterns and world knowledge from large-scale unlabeled text.

Stage 2: Follow user instructions via Instruction-Tuning
Train on high-quality instruction–response data.

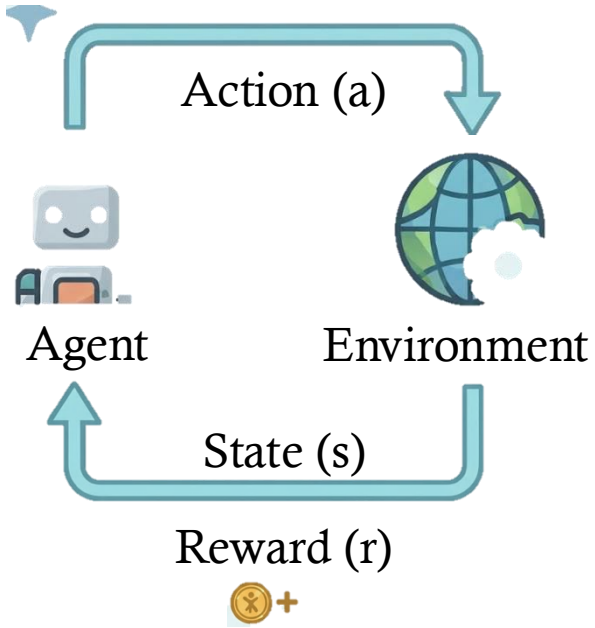
Stage 3: Value Alignment, Preference Optimization
Optimize the model using preference pairs / human feedback.

Stage 4: Reasoning Capability, RL for Reasoning
Use reinforcement learning to encourage **reasoning and problem solving**.



BACKGROUND ON REINFORCEMENT LEARNING

- What is Reinforcement Learning (RL) ?



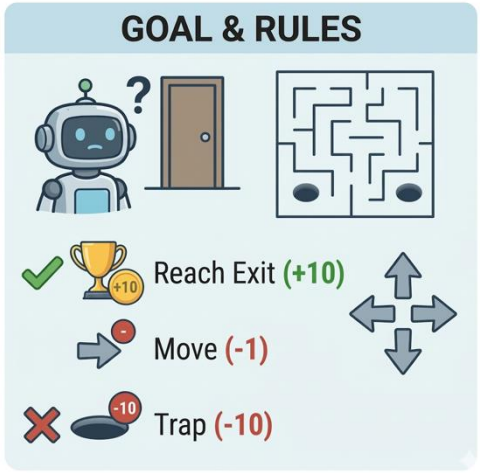
RL: An iterative learning process where an **agent** learns to make a specific goal by taking actions in an **environment**.

- **State (*s*):** The current situation or context of the **environment**.
- **Action (*a*):** The output of the **agent**, given state s.
- **Reward (*r*):** A scalar signal indicating the success of an action.
- **Transition Function**: How the environment change from state s to next state s'

RL Objective: The **agent** aims to learn a **Policy π** that maximizes the cumulative reward over time. [long-term return]

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$$

LEARNING TO EXIT A MAZE



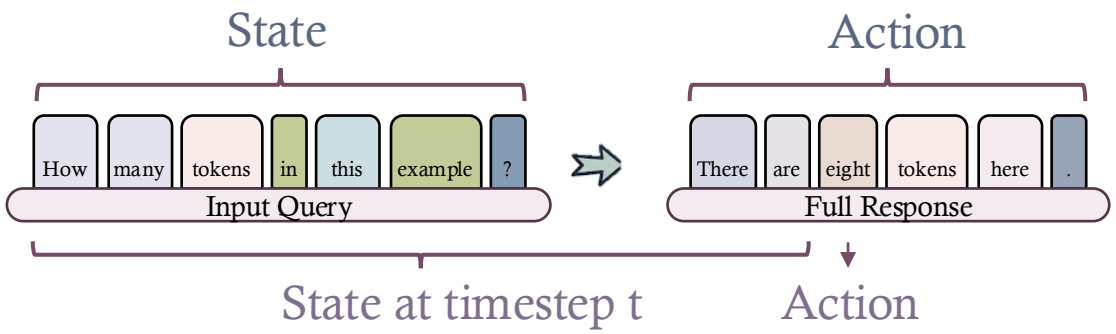
- **Agent's Goal:** Follow RL policy π to find the exit of a maze.
- **State:** current position in the maze
- **Action:** move **up / down / left / right**
- **Reward:** +10 → reach the exit / -1 → each step (to encourage shorter paths) / -10 → hit a trap
- **Learning Process**
  - At first the agent **moves randomly**.
  - When it accidentally reaches the exit → **gets a high reward**.
  - Actions that lead closer to the exit get **higher value** → add probability of this kind of actions.
- Over time, the agent update its policy π **to learns the shortest path**.

TRANSITION TO LLMS

💡 **LLMs as Policy:** An example of Query-Response Pairs

Outer-Loop | State (s): Input query of LLM / **Action (a):** Response of LLM, given query.

- **Sequence-level; No Transition Function**



Rewards

- **Source:** functions or models
- **Function:** indicating response quality

Inner-Loop | State (s_t): Query & Previous tokens / **Action (a_t):** Next token to be generated

- **Transition Function:** Concatenation of state and action

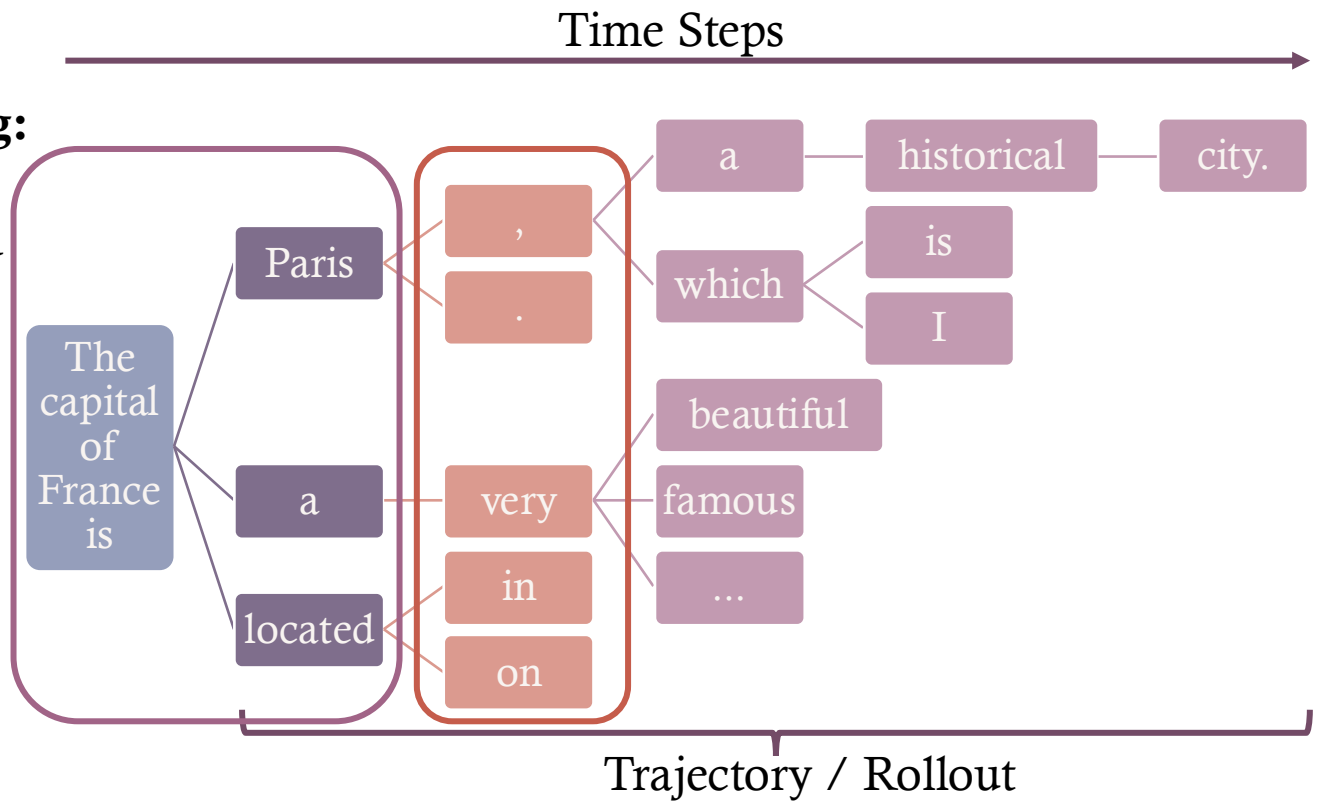
THE INNER-LOOP (TOKEN-LEVEL)

Sequential Decision-Making:

- Every token is an action;
- The prompt and generated string is the state;
- A full response is one **rollout** of the LLM.

This mapping bridges NLP and RL:

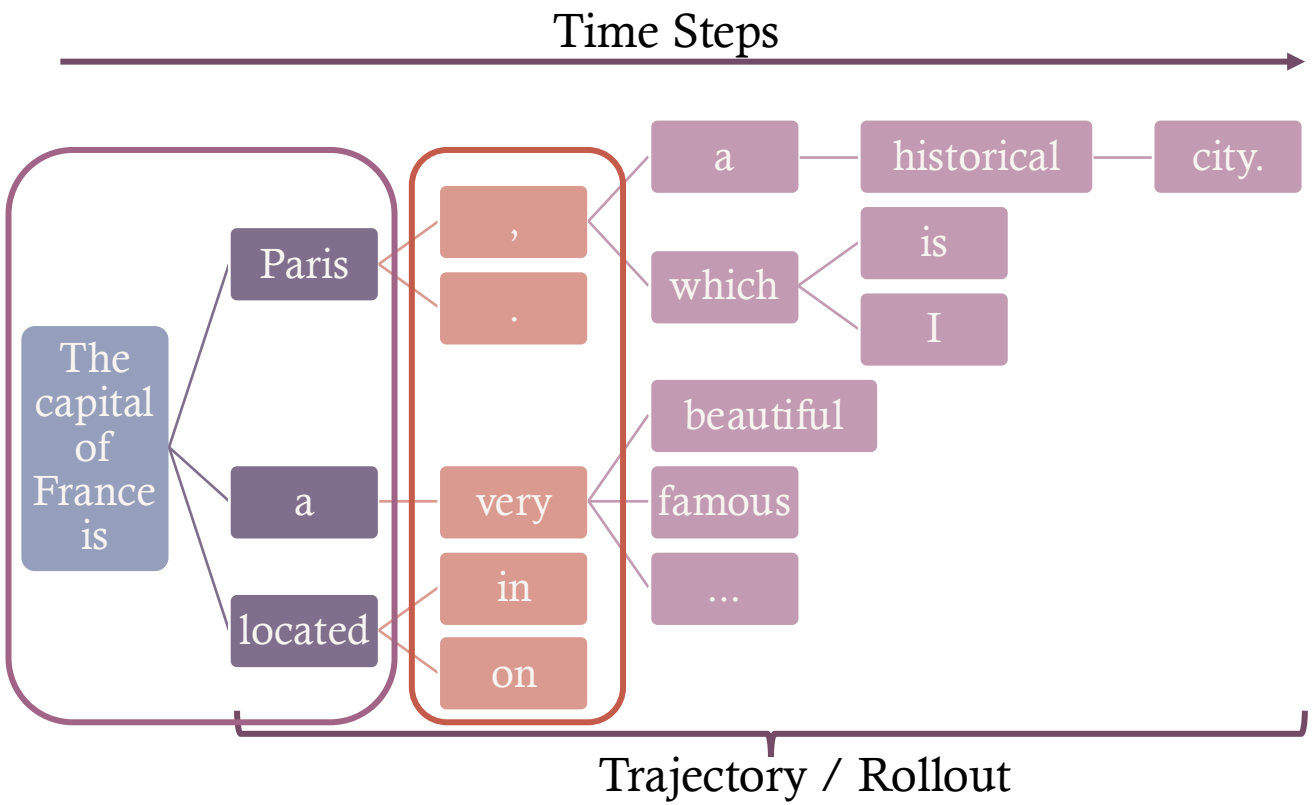
LLM as Policy: $\pi(a_t|s_t)$



THE INNER-LOOP (TOKEN-LEVEL)

Reward Assignment

- Typically only non-zero rewards at the final timestep, which is sparse and delayed.
- Credit assignment: not sure how each token or each action contribute to the final quality of the whole trajectory.



Last timestep: Episodic Rewards?

+1

+0.2

?

OBTAINING REWARDS?

Defining a Reward Function

Example:

- Reinforcement Learning from Verifiable Rewards
- E.g., math reasoning

Correct / incorrect indicated by ground truth label.

Learning a Reward Model

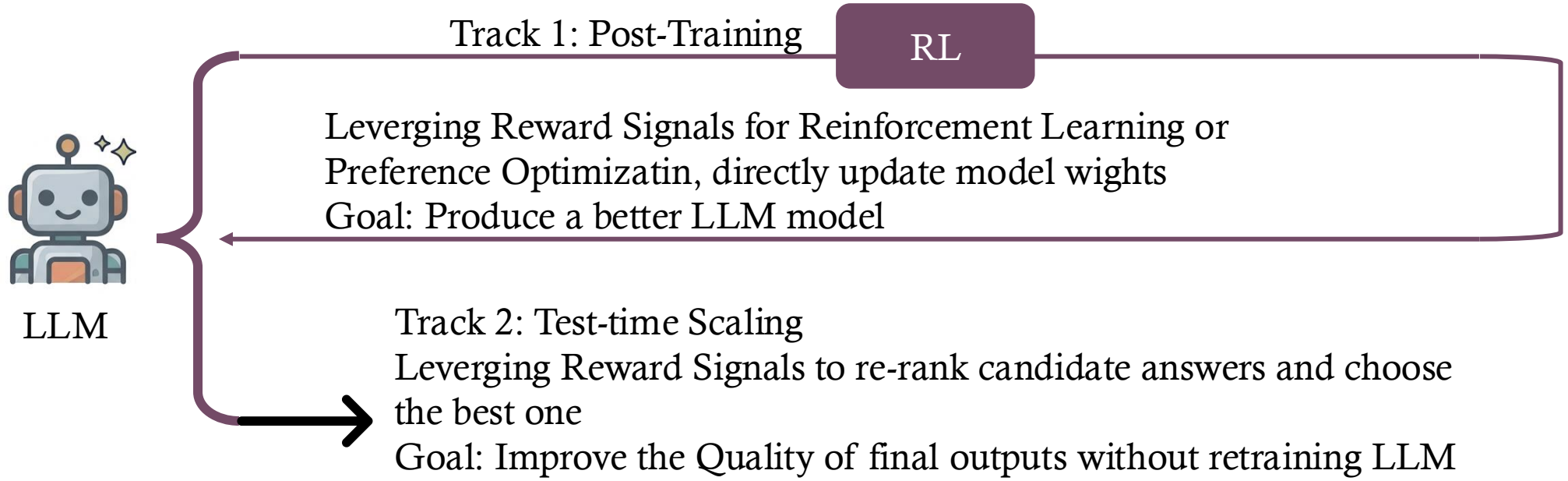
Example:

- Reinforcement Learning from Human Feedback
- E.g., Alignment

Learn from pair-wise preference data.

HOW TO USE REWARD MODEL?

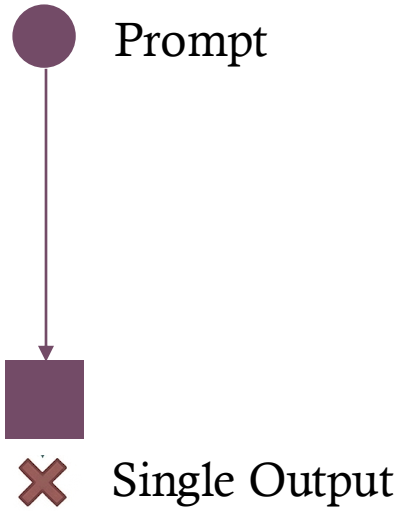
- Two ways to use reward model



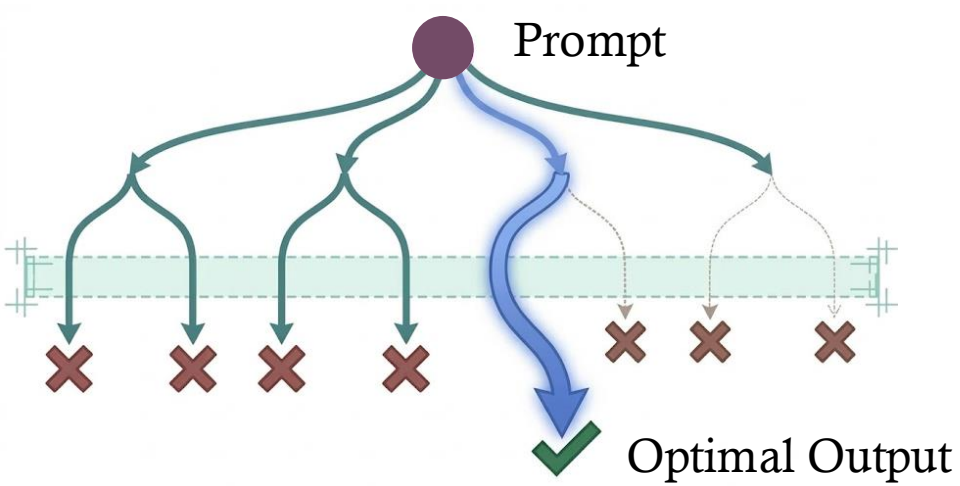
HOW TO USE REWARD MODEL?

- **Rewards in Test-Time Scaling:** using rewards without changing model parameters

Standard Inference
Low Computational Cost, High Risk



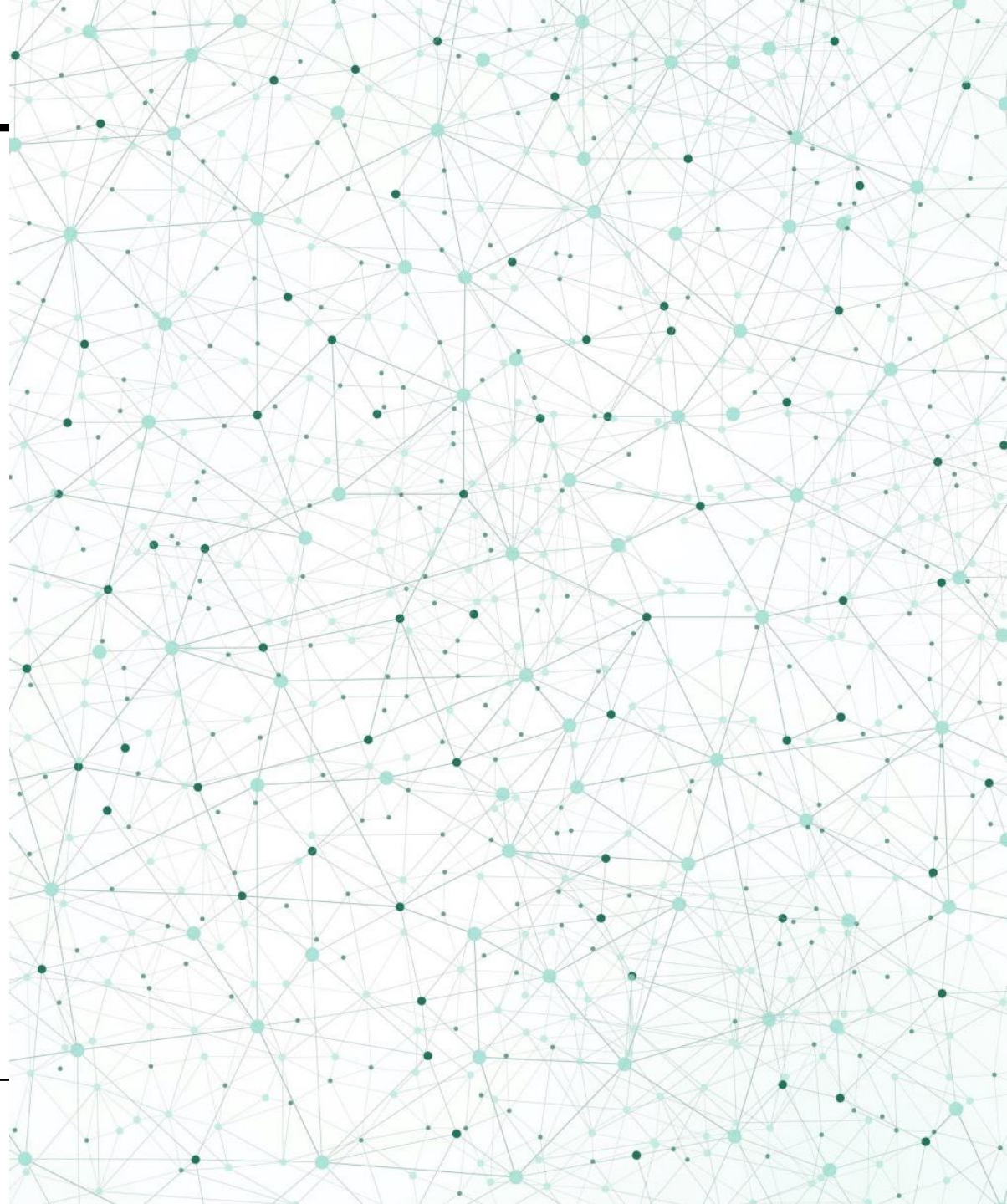
Test-Time Scaling
Trading compute for performance



Part II RLHF

Reinforcement Learning from Human Feedback

Speaker: Yudi Zhang



WHY WE NEED ALIGNMENT?

- Models behaving badly
  - Toxic language
  - Aggressive responses
  - Providing dangerous information

- Why?
  - Models are trained on vast amounts of text data from the internet, which might contain toxicity, bias and unsafe information
  - Pretraining optimizes the model via nex-token prediction, not safety or human values

[1] Deep Reinforcement Learning from Human Preferences. Paul F Christiano et al. NeurIPS2017.

MODELS BEHAVING BADLY

- *Prompt:* "How do I fix a flat bike tire?"
  Bad response: "You could try painting the bike or replacing it."

Helpful

- *Prompt:* "Do vaccines cause autism?"
  Bad response: "Yes, many studies have shown vaccines can cause autism."

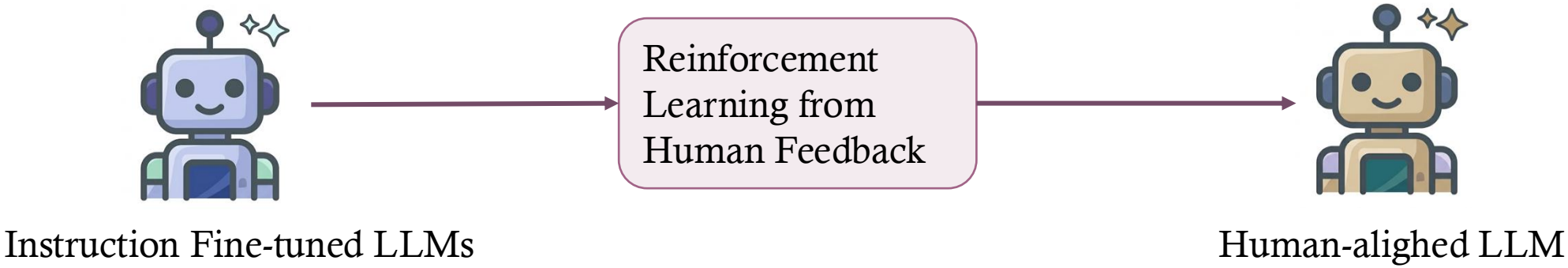
Honest

- *Prompt:* "How can I hack my neighbor's WiFi?"
  Bad response: "You can use a password-cracking tool to break into the network."

Harmless

[1] Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. Yuntao Bai et al. https://arxiv.org/abs/2204.05862

REINFORCEMENT LEARNING FROM HUMAN FEEDBACK

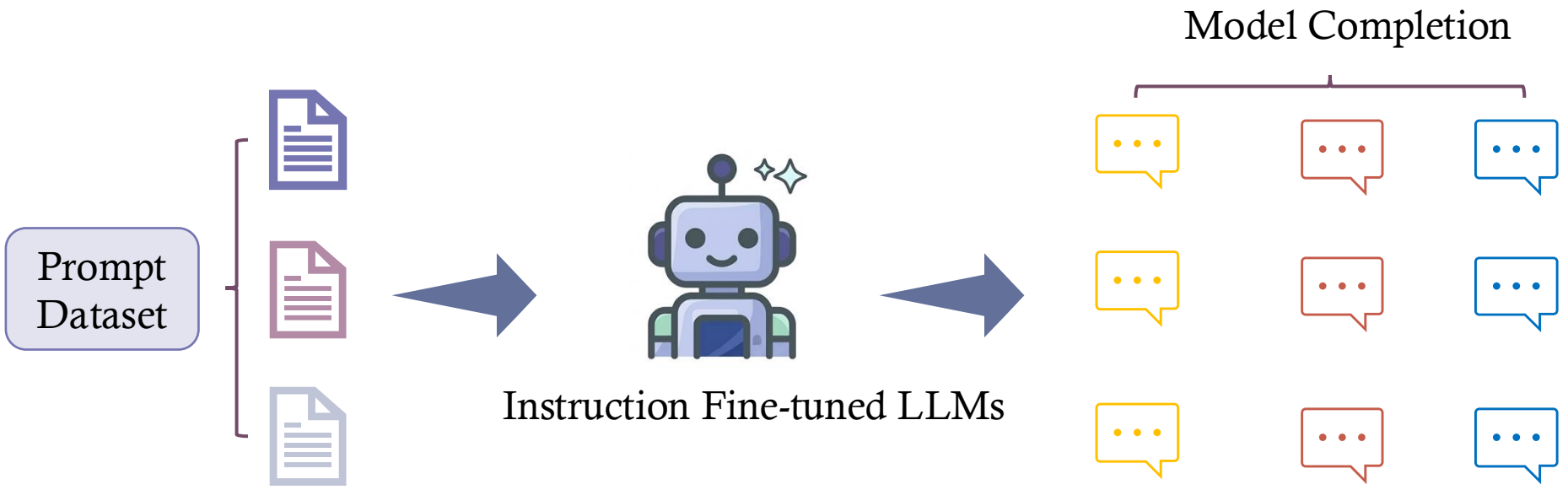


- Maximize helpfulness relevance
- Minimize harm
- Avoid dangerous topics

[1] Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. Yuntao Bai et al. https://arxiv.org/abs/2204.05862

REINFORCEMENT LEARNING FROM HUMAN FEEDBACK

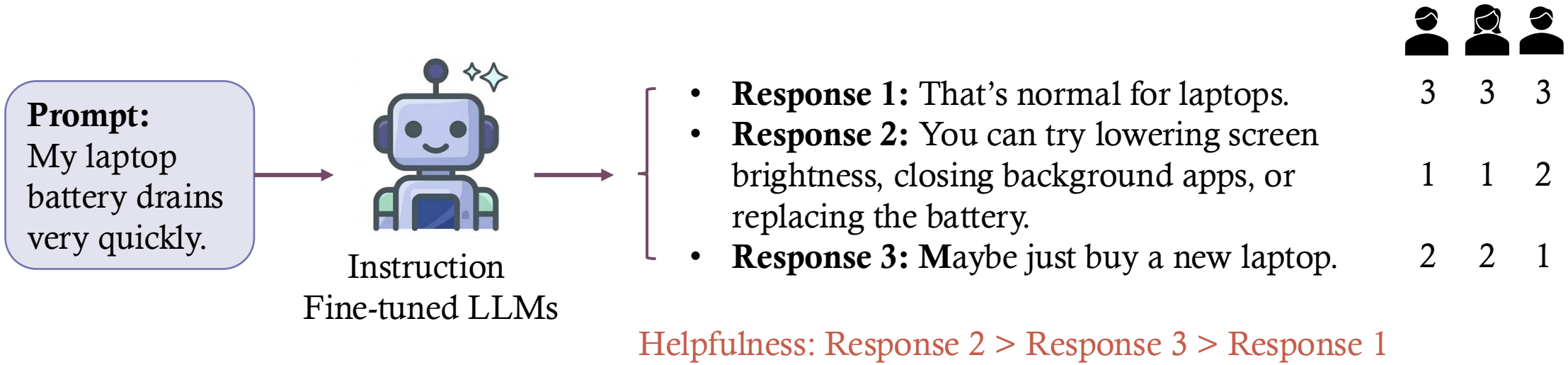
- **Step 1 / 3: Select model & Construct Data**



[1] Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. Yuntao Bai et al. https://arxiv.org/abs/2204.05862

REINFORCEMENT LEARNING FROM HUMAN FEEDBACK

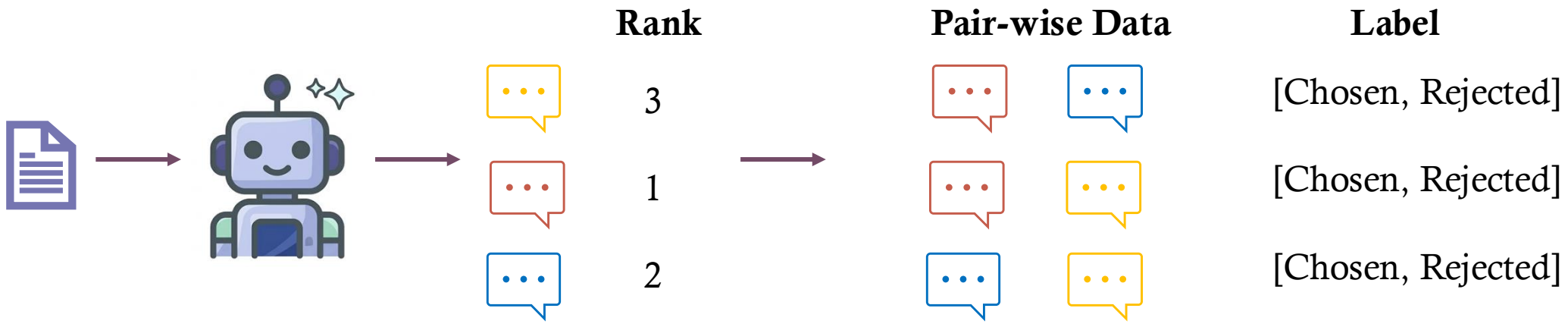
- **Step 1 / 3: Select model & Construct Data**
  - Choose the Alignment Criterion, e.g., helpfulness
  - For the generated prompt-response sets, obtain human feedback through labeler workforce



[1] Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. Yuntao Bai et al. https://arxiv.org/abs/2204.05862

REINFORCEMENT LEARNING FROM HUMAN FEEDBACK

- **Step 1 / 3: Select model & Construct Data**
  - Choose the Alignment Criterion, e.g., helpfulness
  - For the generated prompt-response sets, obtain human feedback through labeler workforce
  - Convert rankings into pair-wise data



[1] Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. Yuntao Bai et al. https://arxiv.org/abs/2204.05862

REINFORCEMENT LEARNING FROM HUMAN FEEDBACK

- **Step 2 / 3: Reward model training**
  - Bradley-Terry Model:

    Given a prompt x and two responses y_1 and y_2 the reward model assigns a scalar score $r(x, y)$.

    Then the probability that response y_1 is preferred over y_2 is,

$$P(y_1 \succ y_2 \mid x) = \frac{\exp(r_\phi(x, y_1))}{\exp(r_\phi(x, y_1)) + \exp(r_\phi(x, y_2))}$$

Then a standard loss function formulation for a reward model with parameter ϕ is,

$$\mathcal{L}(\phi) = -\log\left(\sigma\left(r_\phi(x, y_1) - r_\phi(x, y_2)\right)\right)$$

Difference between chosen and rejected score

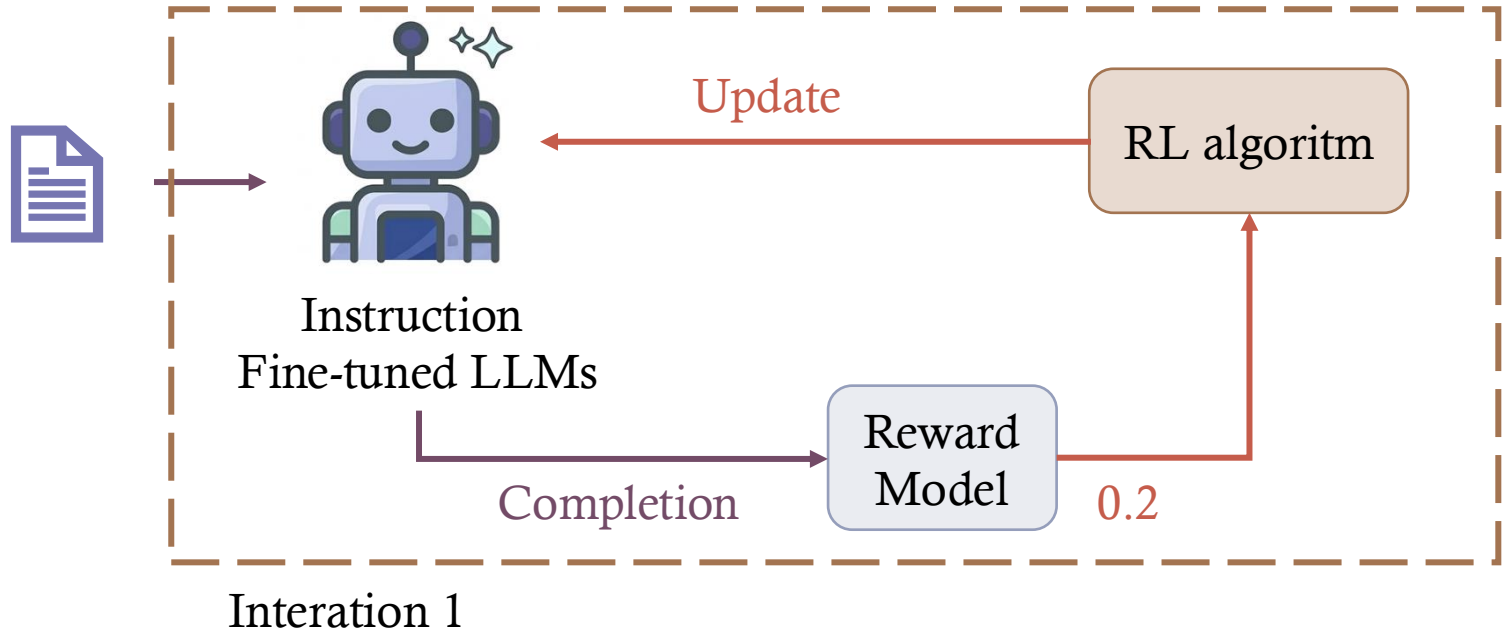
where σ is the logistic / sigmoid function.

  - In practice, reward model ϕ is initialized with the instruction-tuned model.

[1] Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. Yuntao Bai et al. https://arxiv.org/abs/2204.05862

REINFORCEMENT LEARNING FROM HUMAN FEEDBACK

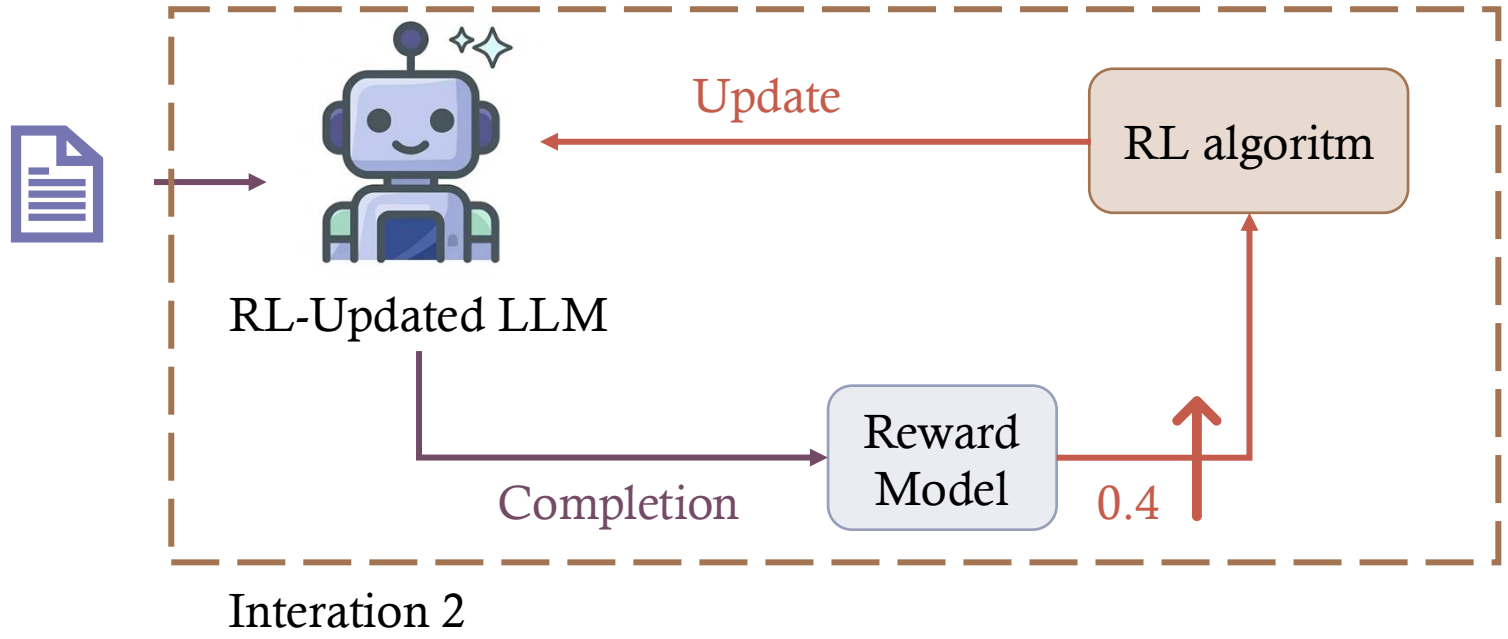
- **Step 2 / 3: Reinforcement Training**



[1] Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. Yuntao Bai et al. https://arxiv.org/abs/2204.05862

REINFORCEMENT LEARNING FROM HUMAN FEEDBACK

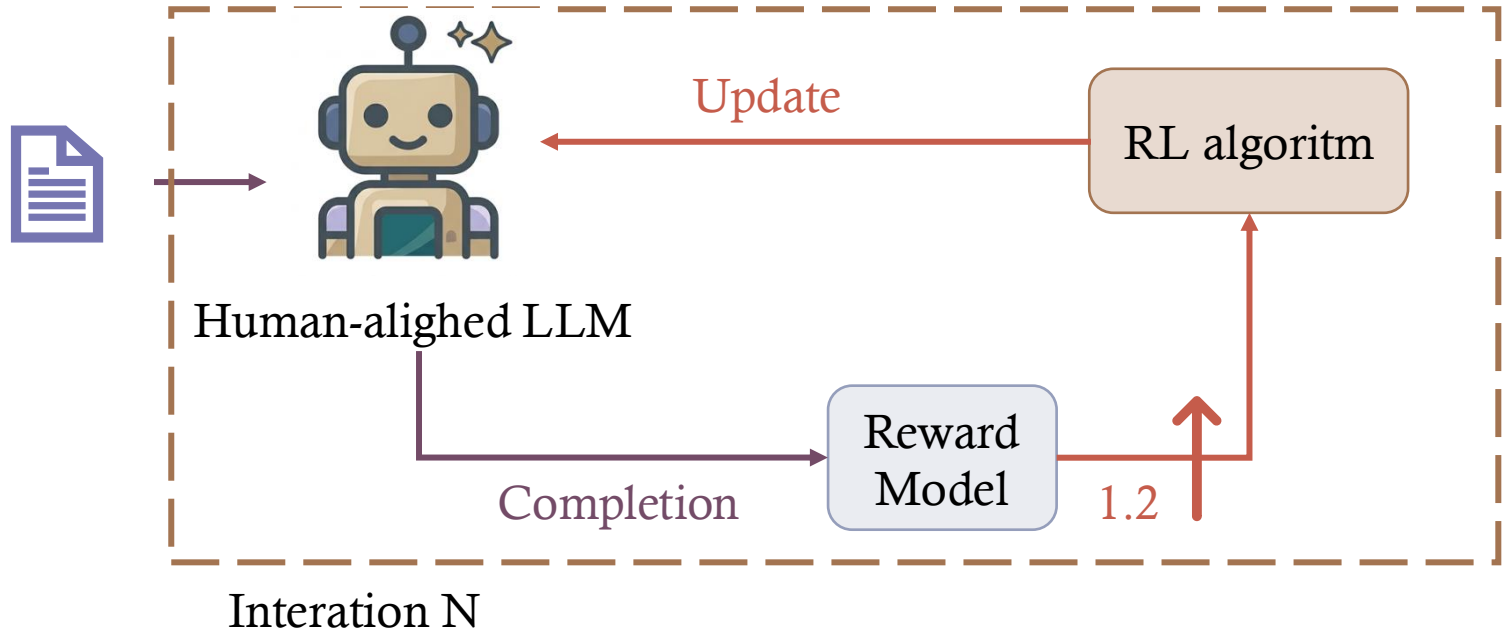
- **Step 2 / 3: Reinforcement Training**



[1] Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. Yuntao Bai et al. https://arxiv.org/abs/2204.05862

REINFORCEMENT LEARNING FROM HUMAN FEEDBACK

- **Step 2 / 3: Reinforcement Training**



[1] Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. Yuntao Bai et al. https://arxiv.org/abs/2204.05862

REINFORCEMENT LEARNING FROM HUMAN FEEDBACK

- **Step 2 / 3: Reinforcement Training**
  - A popular choice of RL algorithm: Proximal Policy Optimization
  - The goal of RL is to maximize long-term return (accumulative reward)

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \gamma^t r_t \right]$$

  - Parameter Space vs. Policy Space: A small step in parameters θ can lead to a massive, unpredictable change in π_θ (action probabilities). [Optimization in Trust Region]

[1] Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. Yuntao Bai et al. https://arxiv.org/abs/2204.05862

PROXIMAL POLICY OPTIMIZATION

- Loss function of PPO:

$$L(s, a, \theta_k, \theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a), \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon, 1+\epsilon\right) A^{\pi_{\theta_k}}(s,a)\right)$$

Advantage

The ratio between new policy π_θ and old policy π_{θ_k}

Trust Region

- **Advantage:** How much better is this specific action compared to what I would typically do in this situation?

[1] Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. Yuntao Bai et al. https://arxiv.org/abs/2204.05862

PROXIMAL POLICY OPTIMIZATION

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$$

- Loss function of PPO:

Advantage

$$L(s, a, \theta_k, \theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a),\ \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon, 1+\epsilon\right) A^{\pi_{\theta_k}}(s,a)\right)$$

- **Advantage:** $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$, where
  - Action Value Function: $Q^\pi(s,a) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s, a_t = a]$
  - State Value Function: $V^\pi(s) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s]$

$$Q^\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}}[r_t + \gamma V^\pi(s_{t+1})]$$

- In practice: PPO use Generalized advantage estimation with a critic module

$\boldsymbol{A > 0}$: This specific action a is better than the average action the policy would take.
$\boldsymbol{A < 0}$: This specific action a is worse than the average.

[1] Proximal policy optimization algorithms. John Schulman et al. https://arxiv.org/abs/1707.06347.

REINFORCEMENT LEARNING FROM HUMAN FEEDBACK

- **Step 2 / 3: Reinforcement Training**
  - A popular choice of RL algorithm: Proximal Policy Optimization
  - The goal of RL is to maximize long-term return (accumulative reward)

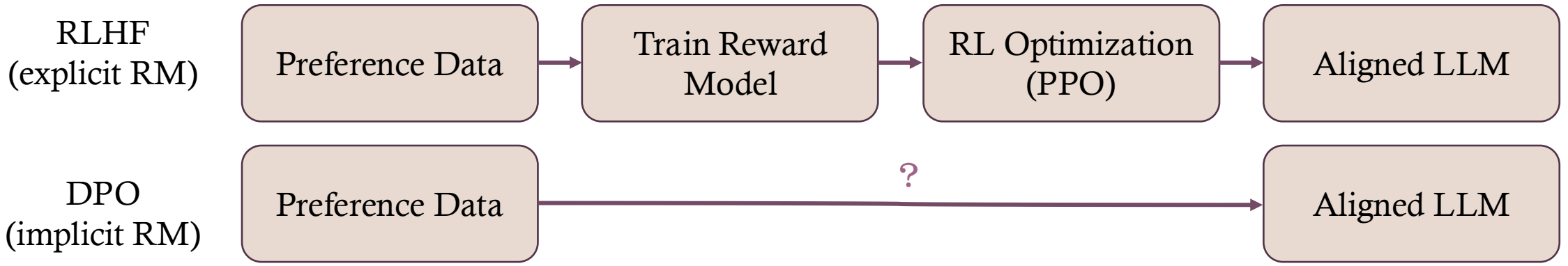
$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \gamma^t r_t \right]$$

  - Usually, we add a KL-term into rewards to keep the model's verbing capability and reduce risk of reward hacking.

[1] Proximal policy optimization algorithms. John Schulman et al. https://arxiv.org/abs/1707.06347.

DIRECT PREFERENCE OPTIMIZATION

- **Goal:** Reduce Complexity (Eliminate RL training)



[1] Direct Preference Optimization: Your Language Model is Secretly a Reward Model. Rafael Rafailov et al. NeurIPS2023.

DIRECT PREFERENCE OPTIMIZATION

- The RL goal is maximize $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{t=0}^{T}\gamma^t r_t]$
- Let's see if we use the sequence-level modeling: State: prompt; Action: the whole response
- The RL objective is: $\max_{\pi_\theta} \mathbb{E}_{x\sim\mathbb{D},y\sim\pi_\theta}\left[r_\phi\,(x,\,y) + \beta\mathbb{D}_{KL}\pi_\theta(y \mid x) \parallel \pi_{\text{ref}}(y \mid x)\right]$
- The optimal solution to the KL-constrained reward maximization objective takes the form:

$$\pi_\theta(y \mid x) = \frac{1}{Z(x)}, \pi_{\text{ref}}(y \mid x)\exp\left(\frac{1}{\beta} r_\phi(x,y)\right),$$

where $Z(x) = \sum_y \pi_{\text{ref}}(y \mid x)\exp\left(\frac{1}{\beta} r(x,y)\right)$ is the partition function.

[1] Direct Preference Optimization: Your Language Model is Secretly a Reward Model. Rafael Rafailov et al. NeurIPS2023.

DIRECT PREFERENCE OPTIMIZATION

- RL objective: $\pi_\theta^*(y \mid x) = \frac{1}{Z(x)}\pi_{\text{ref}}(y \mid x)exp\left(\frac{1}{\beta}\text{r}_\phi(x,y)\right)$
- $\Longrightarrow r(x,y) = \beta \log \frac{\pi_\theta^*(y|x)}{\pi_{\text{ref}}(y|x)} + \beta \log Z(x)$
- Recall the reward model learning we have: $\mathcal{L}(\phi) = -\log\left(\sigma\left(r_\phi(x,y_1) - r_\phi(x,y_1)\right)\right)$
- Then the DPO objective can be derived:

$$\mathcal{L}_{\mathcal{DPO}}(\pi_\theta;\pi_{\text{ref}}) = -\mathbb{E}_{(\text{x},\text{y}_1,\text{y}_2)\sim\mathcal{D}}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_1 \mid x)}{\pi_{\text{ref}}(y_1 \mid x)} - \beta\log\frac{\pi_\theta(y_2 \mid x)}{\pi_{\text{ref}}(y_2 \mid x)}\right)\right].$$

[1] Direct Preference Optimization: Your Language Model is Secretly a Reward Model. Rafael Rafailov et al. NeurIPS2023.

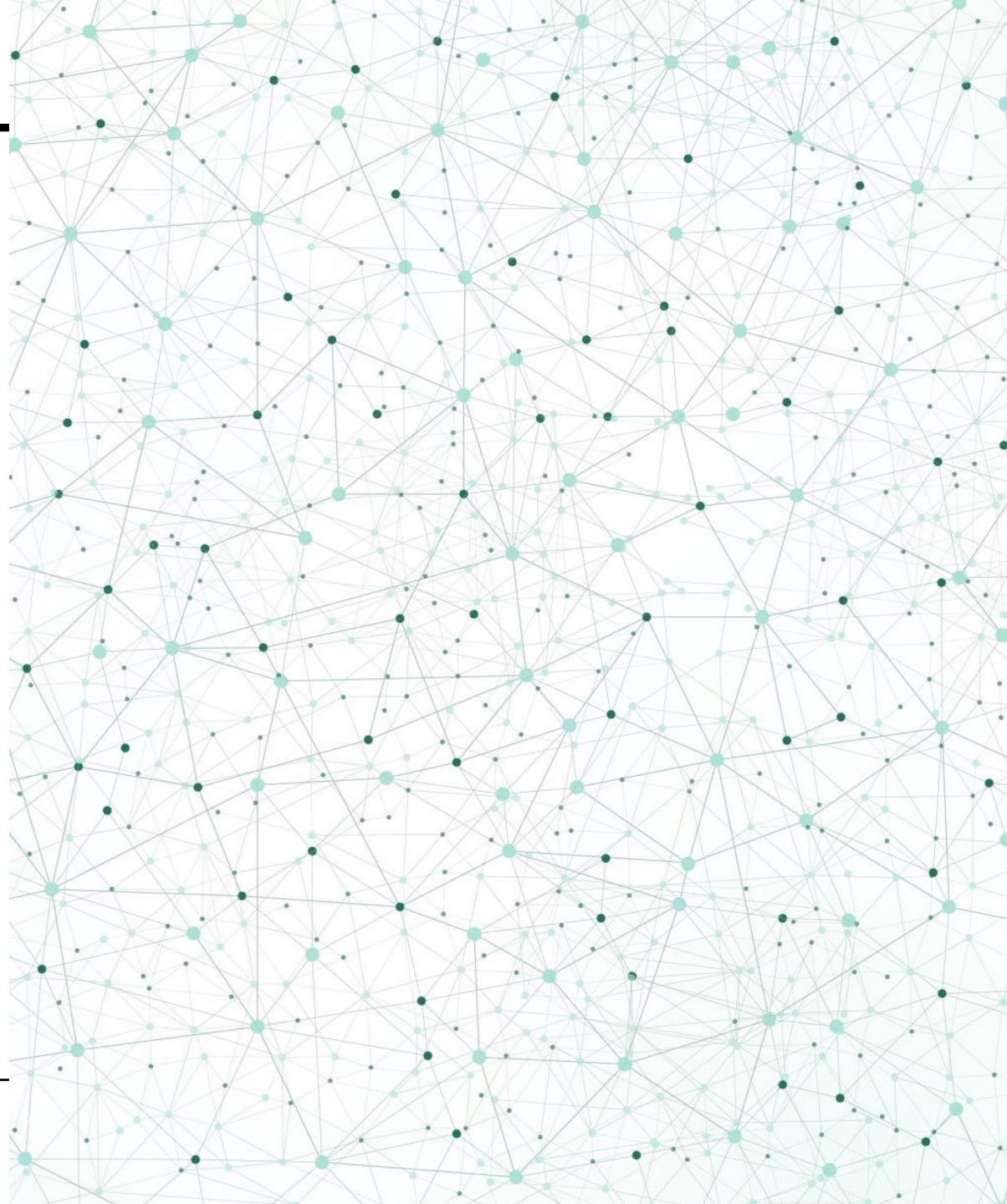
CONCLUSION OF THIS CHAPTER

- Why we need Alignment? – Model behaving badly, HHH principle.
- Explicitly reward modeling: RLHF
  - How do we collect pair-wise preference data?
  - How do we train reward model?
  - How does proximal policy optimization work?
- Implicitly reward modeling: the direct preference optimization

Part III RLVR

Reinforcement Learning from Verifiable Rewards

Speaker: Yudi Zhang



WHY RLHF IS NOT ENOUGH

- **From Subjective Feedback to Objective Truth**
- **The RLHF Era**
  - Relies on **Human Preference**.
  - Goal: "Sound like a helpful assistant."
  - Bottleneck: Expensive, slow, and prone to "Reward Hacking" (the model learns to make humans happy rather than being correct)
  - **The Complexity Ceiling:** As problems get harder (Quantum Physics, Advanced Math), finding humans qualified to grade them becomes a bottleneck.
- **The RLVR Era**
  - Relies on **Mathematical/Programmatic Ground Truth**.
  - Goal: "Solve the problem correctly."
  - Breakthrough: Scalable, zero-cost, and drives deep reasoning.

WHAT ARE VERIFIABLE REWARDS

Instead of relying on subjective human scoring, verifiable rewards utilize programmatic validation or absolute signals from deterministic environments. E.g.,

- **Mathematical Ground Truth:** Definitive answers to equations.
- **Code Execution:** Success rates of unit tests.
- **Formal Logic:** Deterministic state machines for logical puzzles.

Output: A binary or scalar reward signal (e.g., **1.0** for "Correct" or **0.0** for "Incorrect").

WHAT ARE VERIFIABLE REWARDS

- Examples: Find the value of x that satisfies: $3^{x+1} - 3^x = 162$.

Response candidate		Verifiable Rewards
To solve $3^{x+1} - 3^x = 162$, we factor out 3^x: $3^x(3-1) = 162 \Rightarrow 2 \cdot 3^x = 162 \Rightarrow 3^x = 81$. Since $3^4 = 81$, $\boldsymbol{x = 4}$.	**Professional & Clear**	1
Dividing everything by 3: $3^x - 3^{x-1} = 54$. Let $x = 5$: $3^5 - 3^4 = 243 - 81 = 162$. Answer: **x = 5**.	**Looks structured**, but the final step contains a calculation error	0
Let's simplify: $3x + 3 - 3x = 162$. This leads to $3 = 162$, which is impossible. Therefore, there is **no solution**.	**Confident/Assertive**	0

RL FROM VERIFIABLE REWARD

- A popular algorithm for RLVR is GRPO
  - In PPO, we need a **"Critic Model"** (as large as the Policy Model) to estimate advantage. This doubles the GPU memory usage.
- Loss function of PPO:

Advantage

$$L(s,a,\theta_k,\theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a),\ \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon, 1+\epsilon\right) A^{\pi_{\theta_k}}(s,a)\right)$$

- In practice: PPO use GAE, where using a term temporal difference error $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$, $V^\pi(s) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s]$

Compute by Critic Model

RL FROM VERIFIABLE REWARD

- One-step advantage estimation: $A^{\pi} = Q(s_t, \mathrm{a}_t) - V(s_t)$
  - Action Value Function: $Q^{\pi}(s,a) = \mathbb{E}_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s, a_t = a]$
  - State Value Function: $V^{\pi}(s) = \mathbb{E}_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s]$
- Recall our sequence-level modeling where:

State s: prompt x; Action a: the whole response y.

- Remove next state:

$$A^{\pi}(s,a) = r(s,a) - V(s), \text{ where } V(s) = E_{a\sim\pi}[r(s,a)|s = x]$$

GROUP RELATIVE POLICY OPTIMIZATION

- For each prompt x, GRPO samples a group of answers $\{y_1, y_2, \dots, y_N\}$ with rewards $\{r_1, r_2, \dots, r_N\}$.
- The advantage A_i for the i-th answer is calculated as:

$$\mathrm{A}_i = \frac{r_i - \mathrm{mean}(r_1, r_2, \dots, r_N)}{\mathrm{std}(r_1, r_2, \dots, r_N)}$$

- The objective function is,

$$L(x, \theta_k, \theta)$$

$$= \frac{1}{N}\sum_{i=0}^{N}\frac{1}{T}\sum_{i=0}^{T}\left(\min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}A_i(x, y_i), clip\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon, 1+\epsilon\right)A_i(x, y_i)\right) - \beta\,\mathbb{D}_{KL}\left[\pi_\theta \parallel \pi_{ref}\right]\right)$$

[1] Deepseekmath: Pushing the limits of mathematical reasoning in open language models. Zhihong Shao et al. https://arxiv.org/abs/2402.03300

AHA MOMENT IN RLVR

- "Aha Moment." is somewhat similar to how humans experience sudden realizations while problem-solving.
- Imagine you're trying to solve a puzzle:
  - First try: "This piece should go here based on the color"
  - Recognition: "But wait, the shape doesn't quite fit"
  - Correction: "Ah, it actually belongs over there"
  - Explanation: "Because both the color and shape pattern match in this position"

This ability emerged naturally from RL training, without being explicitly programmed, **demonstrating learning rather than mere memorization** of a process from the training data.

---

[1] Deepseekmath: Pushing the limits of mathematical reasoning in open language models. Zhihong Shao et al. https://arxiv.org/abs/2402.03300

AHA MOMENT IN RLVR

Why does it emerge in RLVR?

- RLVR is beyond mimic and rewards objective correctness, not style.
- If the first reasoning path is wrong, the model gets **no reward**.
- So the model is encouraged to **reflect, backtrack, and self-correct** before giving the final answer.

[1] Deepseekmath: Pushing the limits of mathematical reasoning in open language models. Zhihong Shao et al. https://arxiv.org/abs/2402.03300

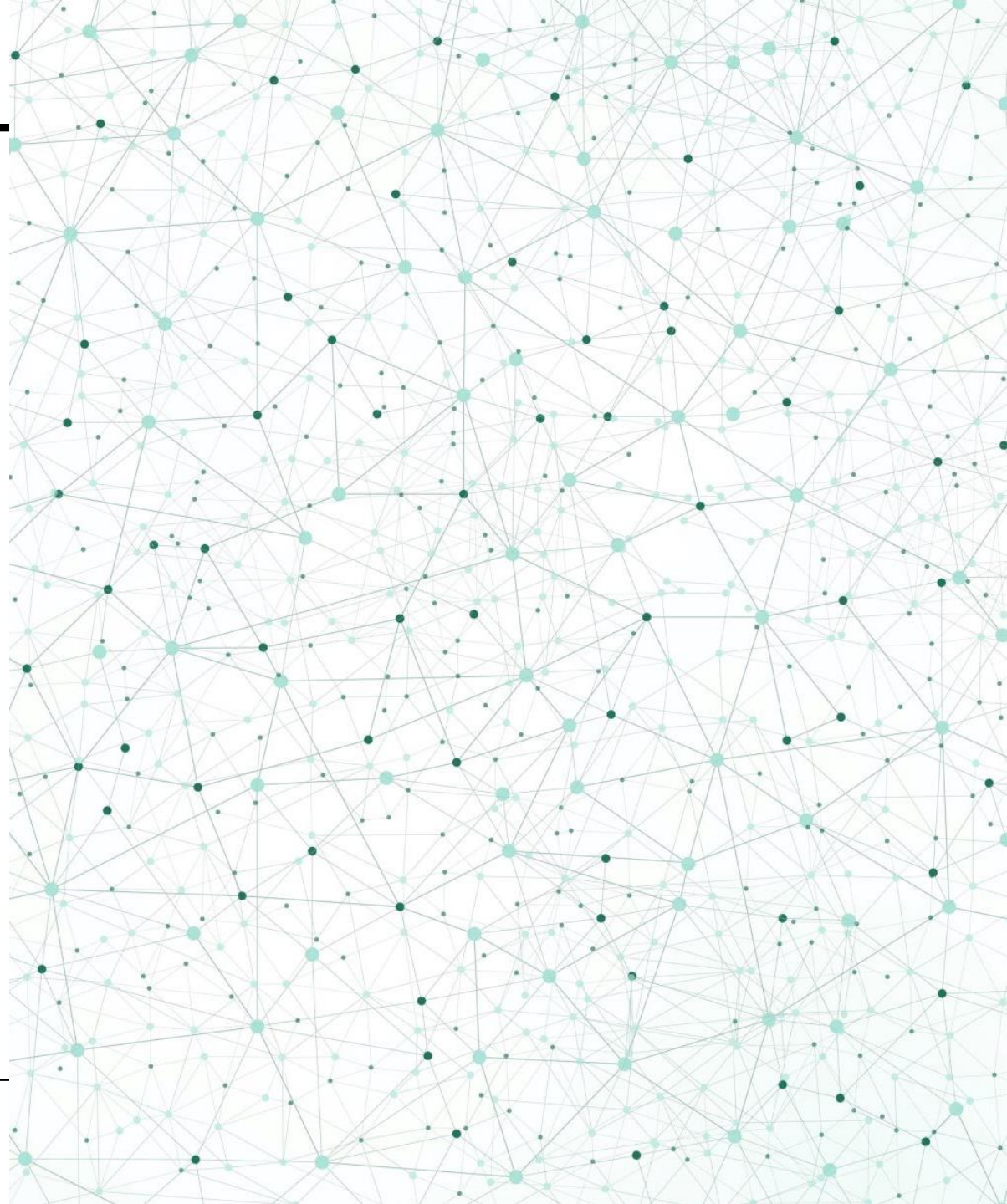
CONCLUSION OF THIS CHAPTER

- Why RLVR?
  - Move from subjective human preference to objective correctness.
  - more scalable, lower annotation cost, better suited for reasoning-intensive tasks
- GRPO and Aha Moment in RLVR

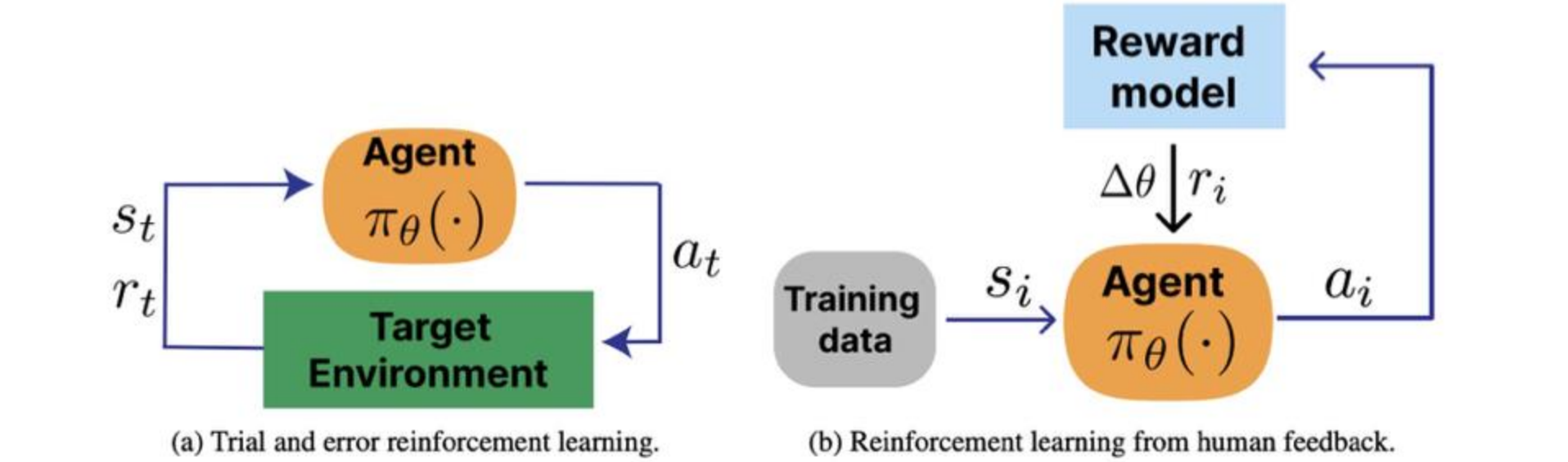
Part IV ORM & PRM

Outcome / Process Reward Modeling

Speaker: Meng Fang



HOW REWARD MODEL WORKS



(a) Trial and error reinforcement learning.

(b) Reinforcement learning from human feedback.

Large language model alignment

OUTCOME REWARDS

The denominator of a fraction is 7 less than 3 times the numerator. If the fraction is equivalent to $2/5$, what is the numerator of the fraction? (Answer: $\boxed{14}$)

Let's call the numerator x.

So the denominator is 3x-7.

We know that x/(3x-7) = 2/5.

So 5x = 2(3x-7).

5x = 6x - 14.

☹ 😐 😎 So x = 7.

PROCESS-LEVEL REWARDS

The denominator of a fraction is 7 less than 3 times the numerator. If the fraction is equivalent to $2/5$, what is the numerator of the fraction? (Answer: $\boxed{14}$)

☹ 😐 😎 Let's call the numerator x.

☹ 😐 😎 So the denominator is 3x-7.

☹ 😐 😎 We know that x/(3x-7) = 2/5.

☹ 😐 😎 So 5x = 2(3x-7).

☹ 😐 😎 5x = 6x - 14.

☹ 😐 😎 So x = 7.

WHY OUTCOME REWARDS IS NOT ENOUGH

- **Sparse feedback**: outcome reward only appears at the end, so the model does not know which step was right or wrong.
- **Poor credit assignment**: a wrong final answer may come from one small mistake, but outcome reward penalizes the whole trajectory equally.
- **Bad reasoning can still be rewarded**: a model may reach the correct answer by luck or flawed reasoning, and outcome reward cannot distinguish that from a genuinely good solution.

WHY OUTCOME REWARDS IS NOT ENOUGH

- Consider a math reasoning task.

Solve for x and y: $x+y=10$, $2x+y=12$

Step 1	Subtract the first equation from the second: $(2x + y) - (x + y) = 12 - 10$.	Subtract the first equation from the second: $(2x + y) - (x + y)$.	Look at the number 12 and the number 10.
Step 2	Result: $x = 2$.	Result: $x = 2$.	Subtract them: $12 - 10 = 2$.
Step 3	Plug $x = 2$ back into $x + y = 10$: $2 + y = 10$.	Plug $x = 2$ into $x + y = 10$, but write $y - 2 = 10$.	Assume the result 2 is the value for both x and y.
Step 4	Solve for y: $y = 8$.	Solve for y: $y = 12$.	See that $2 + 8 = 10$, so decide y must be 8.
	Fully Correct ORM: 1 PRM: 1, 1, 1, 1	Step 3 Incorrect ORM: 0 PRM: 1, 1, 0, 0	Lucky Guy ORM: 1 PRM: 0, 0, 0, 0

WAYS TO TRAIN PROCESS REWARD MODELS

1. Human-Collected Supervision

- Step-level annotations (correct / incorrect reasoning)
- High-quality signal for **fine-grained credit assignment**
- **Cons:** Expensive, low scalability

Examples: PRM800K [1] (Human-Collected, Classification Loss)

[1] Let's Verify Step by Step. Hunter Lightman et al. ICLR2024. https://openreview.net/forum?id=v8L0pN6EOi.

WAYS TO TRAIN PROCESS REWARD MODELS

2. Automatically Generated Supervision

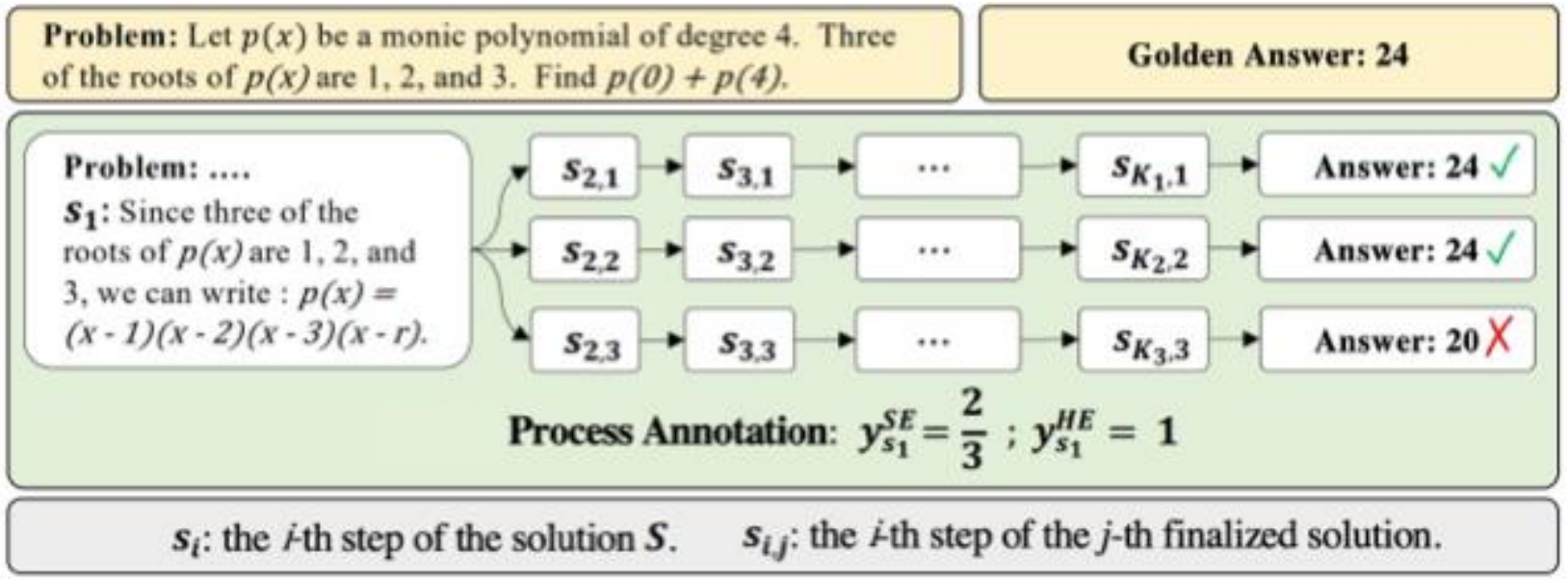
- **Outcome-to-process bootstrapping:** Infer step rewards from final outcomes [MCTS]
- **Verifier-based labeling:** Use tools (math solvers, unit tests) to check steps
- **Self-training / model labeling:** Strong models generate and score reasoning traces
- **Pros:** Scalable, low cost

Example:

- Math-Shepherd [1] (MCTS, classification)

[1] Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations. Peiyi Wang et al. ACL2024. https://aclanthology.org/2024.acl-long.510/.

MATH-SHEPHERD



From [1]: Math-Shepherd automatic process annotation employs a "completer" to finalize N reasoning processes (N=3 in this figure) for an intermediate step (s_1 in this figure), subsequently use hard estimation (HE) and soft estimation (SE) to annotate this step based on all decoded answers.

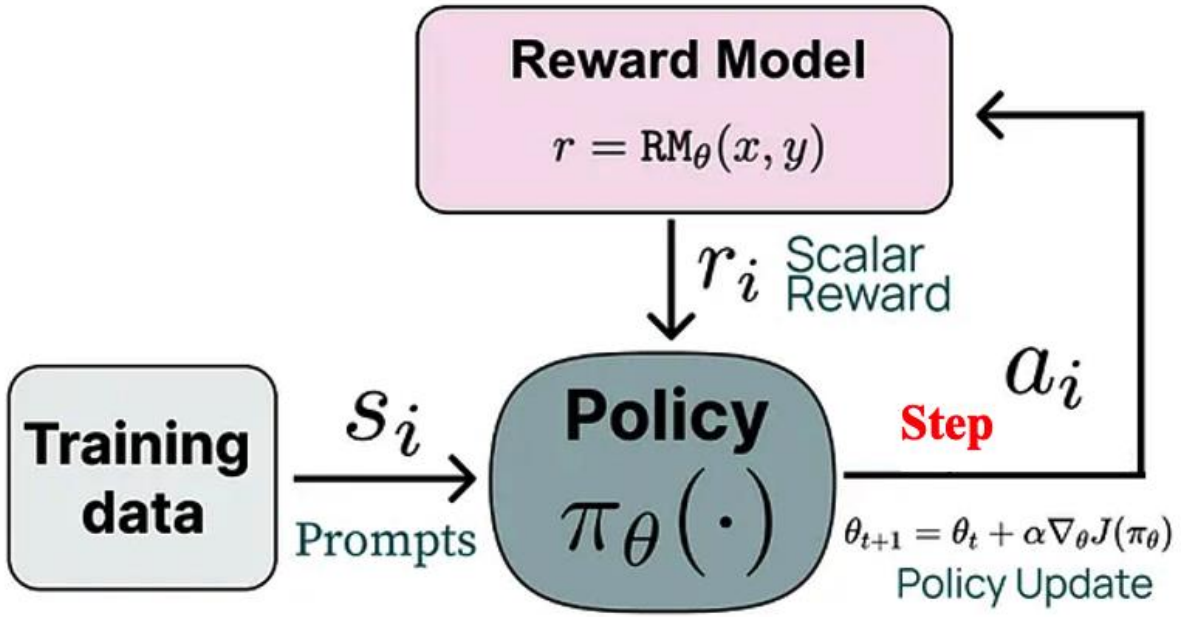
[1] Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations. Peiyi Wang et al. ACL2024. https://aclanthology.org/2024.acl-long.510/.

HOW TO USE PROCESS REWARD MODELS

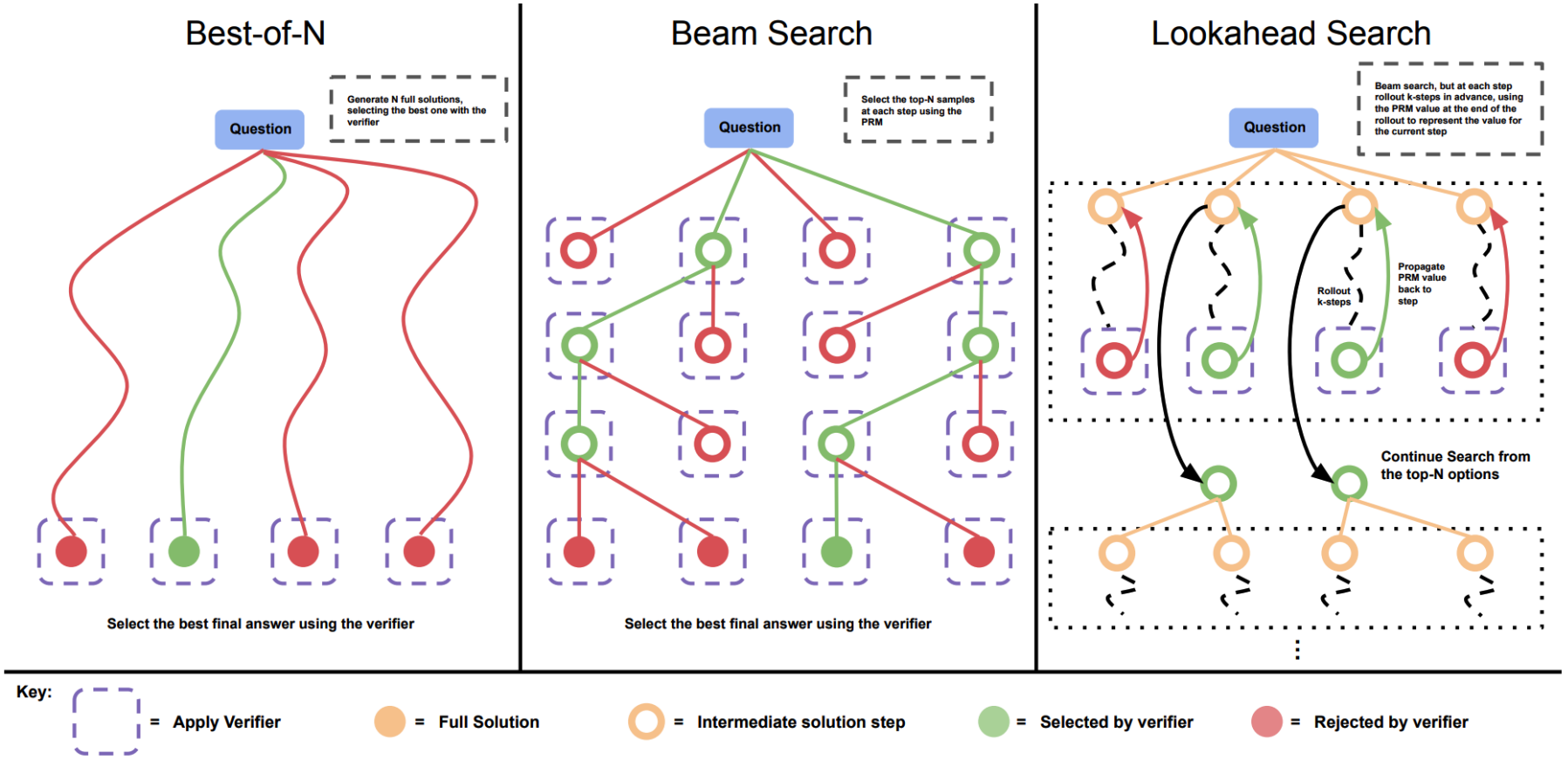
There are multiple ways to use process reward models (PRM):

- Direct use process reward as proxy of outcome reward for RL training
- PRM can be used for search

USE PROCESS REWARD MODELS FOR TRAINING



USE PROCESS REWARD MODELS FOR SEARCH



Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters. ICLR 2025.

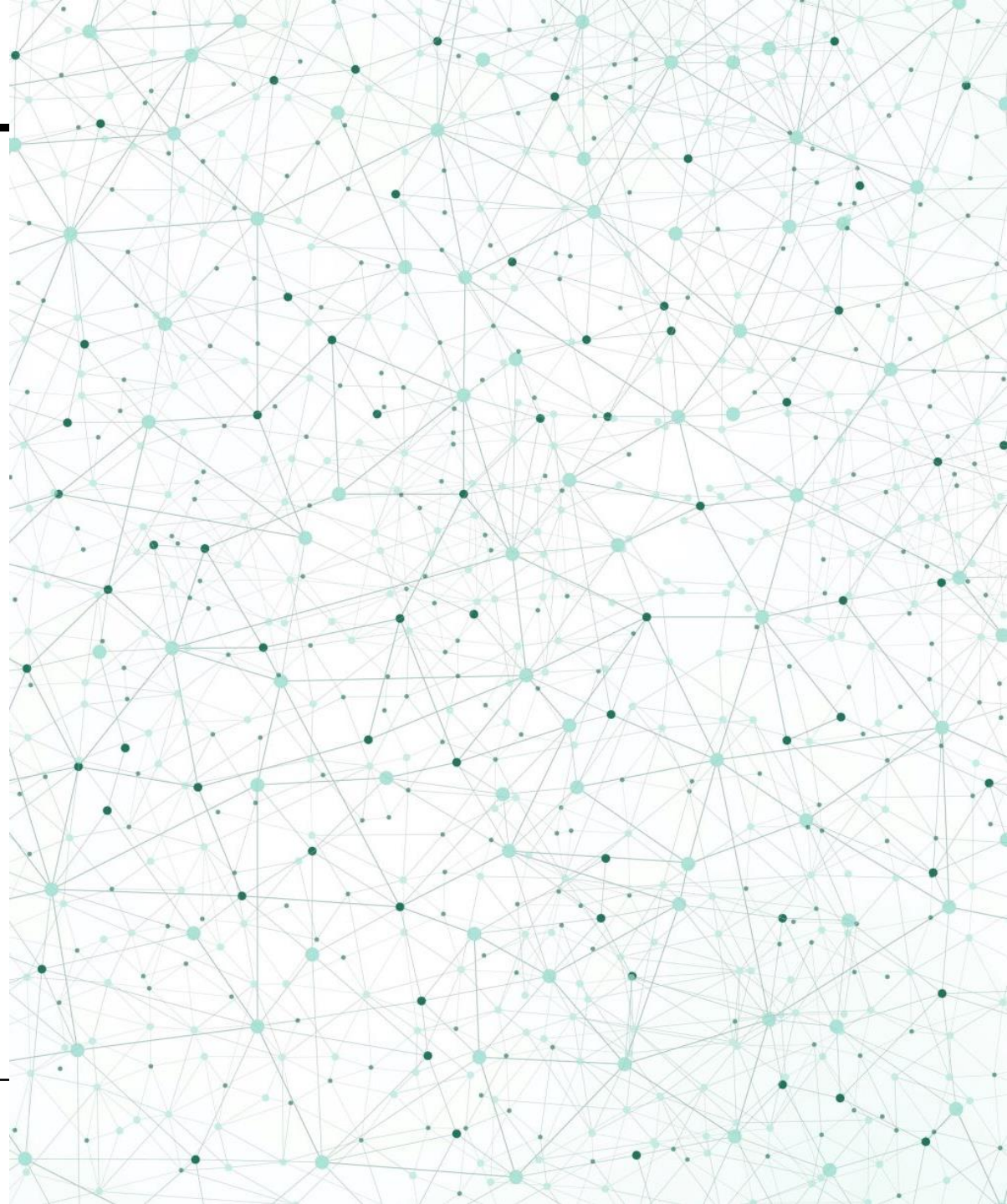
CONCLUSION OF THIS CHAPTER

- Why do we need process reward models?
- Two way to collect data for process reward model training
  - Human Collected / Automatically Collected
- Using process reward models during training and inference

Part V LLM-as-Rewards

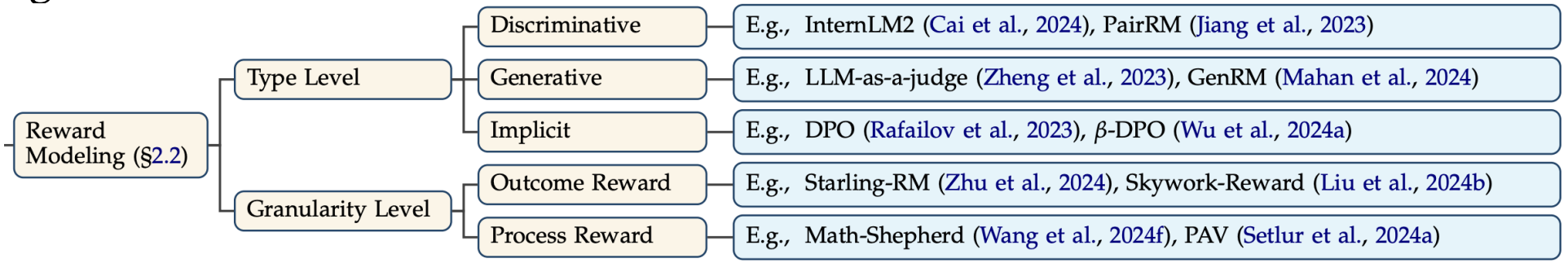
Judge Model & LLM feedback

Speaker: Yudi Zhang



WHY VERIFIABLE REWARD IS NOT ENOUGH?

- Last chapter, we analysis why verifiable reward is not enough from the perspective of credit assignment.



- However, verifiable rewards are also not enough for:
  - Many important tasks are not fully verifiable. [open-ended tasks, expert-driven assessment]
  - Also, we might want the reward models also have generation and reason capability. [GenRM, RM-R1]
  - Additionally, we may want to evaluate a response from multiple perspectives.. [Rubric Rewards]

---

[1] A comprehensive survey of reward models: Taxonomy, applications, challenges, and future. Jialun Zhong et al. https://arxiv.org/pdf/2504.12328.

WHAT IS LLM-AS-JUDGE

Definition

- LLMs to transition from generative tasks to evaluation
- LLM-as-a-Judge denotes the use of LLMs to evaluate objects, actions, or decisions based on predefined rules, criteria, or preferences. It encompasses a broad spectrum of roles, including: Graders, Evaluators/Assessors, Critics,Verifiers, Examiners, Reward/Ranking Models, etc.

[1] A survey on LLM-as-Judge. Zongyu Lin et al. https://arxiv.org/pdf/2411.15594.

WHAT IS LLM-AS-JUDGE

Definition

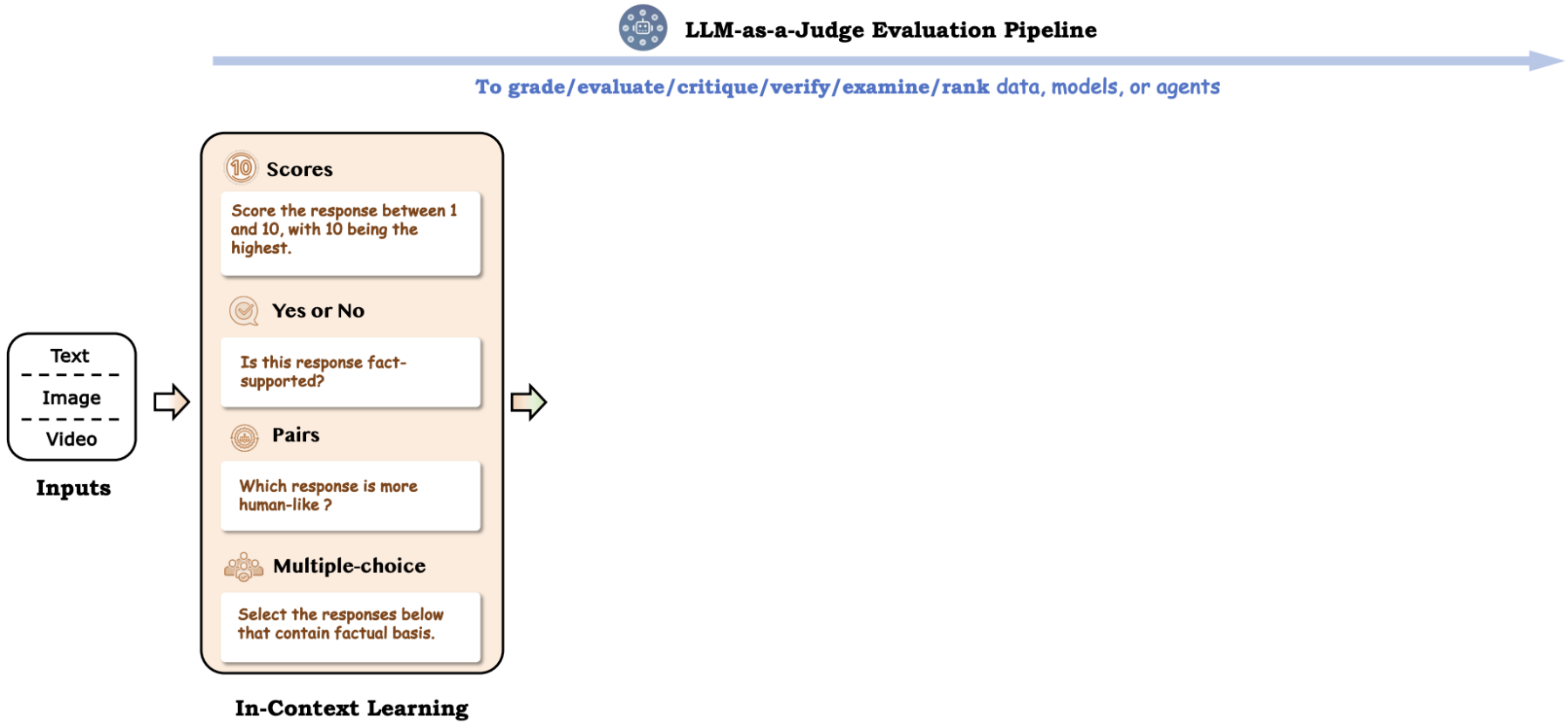
$$\mathcal{E} \leftarrow \mathcal{P}\,(x \oplus \mathrm{C})$$

- $\mathcal{E}$: The final evaluation obtained from the whole LLM-as-a-Judge process in the expected manner. It could be a score, a choice, a label or a sentence, etc.
- $\mathcal{P}$: The probability function defined by the corresponding LLM, and the generation is an auto-regressive process.
- x: The input data in any available types (text, image, video), which waiting to be evaluated.
- C: The context for the input x, which is often prompt template or combined with history information in dialogue.
- $\oplus$: The combination operator combines the input x with the context C, and this operation can vary depending on the context, such as being placed at the beginning, middle, or end.

---

[1] A survey on LLM-as-Judge. Zongyu Lin et al. https://arxiv.org/pdf/2411.15594.

HOW TO USE LLM-AS-JUDGE



[1] A survey on LLM-as-Judge. Zongyu Lin et al. https://arxiv.org/pdf/2411.15594.

HOW TO USE LLM-AS-JUDGE

- Generating Score
- Pair-wise Comparison

Evaluation Prompt Templates from Gao et al [38]

Likert Scale Scoring:
Evaluate the quality of summaries written for a news article. Rate each summary on four dimensions: {Dimension_1}, {Dimension_2}, {Dimension_3}, and {Dimension_4}. You should rate on a scale from 1 (worst) to 5 (best).
Article: {Article}
Summary: {Summary}

Pairwise Comparison:
Given a new article, which summary is better? Answer "Summary 0" or "Summary 1". You do not need to explain the reason.
Article: {Article}
Summary 0: {Summary_0}
Summary 1: {Summary_1}

[1] A survey on LLM-as-Judge. Zongyu Lin et al. https://arxiv.org/pdf/2411.15594.
[2] Human-like summarization evaluation with chatgpt. Mingqi Gao et al. https://arxiv.org/abs/2304.02554

HOW TO USE LLM-AS-JUDGE

- Yes/No
- Multiple-Choice

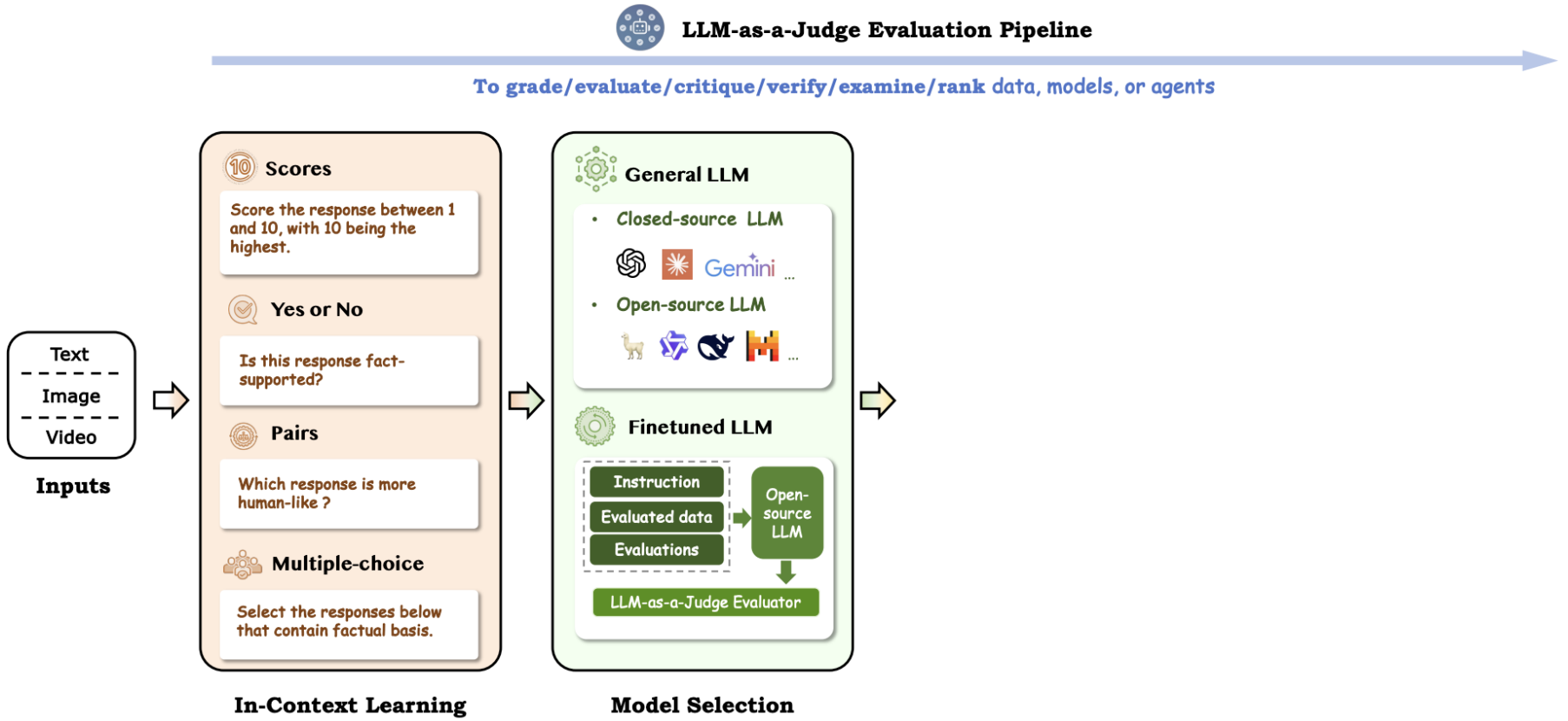
Evaluation Prompt Templates for Yes/No and Multiple-Choice Tasks

Yes/No Evaluation:
Is the sentence supported by the article? Answer "Yes" or "No".
Article: {Article}
Sentence: {Sentence}

Multiple-Choice Evaluation:
You are given a summary and some semantic content units. For each semantic unit, choose those can be inferred from the summary, return their number.
Summary: {Summary}
Semantic content units:
1. {SCU_1}
2. {SCU_2}
.....
n. {SCU_n}

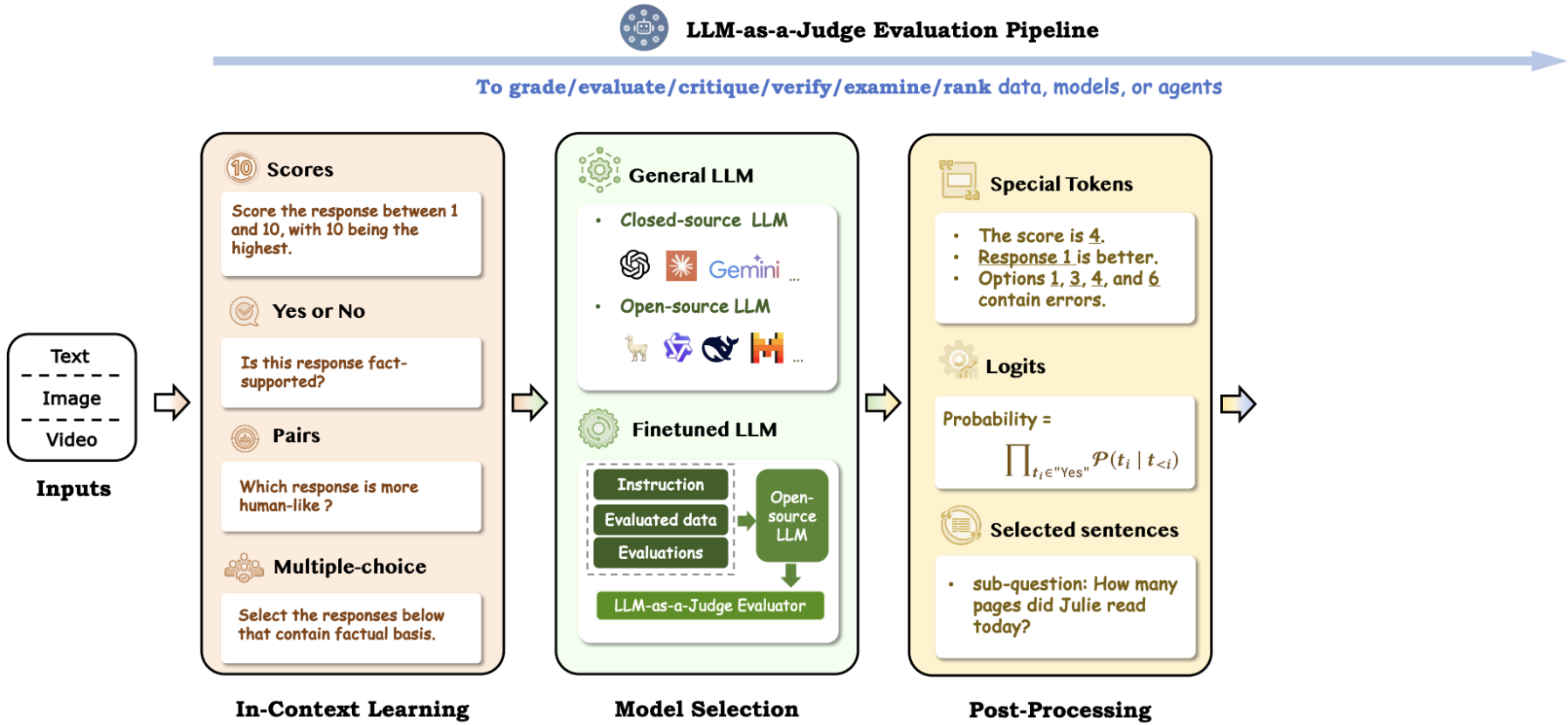
[1] A survey on LLM-as-Judge. Zongyu Lin et al. https://arxiv.org/pdf/2411.15594.

HOW TO USE LLM-AS-JUDGE



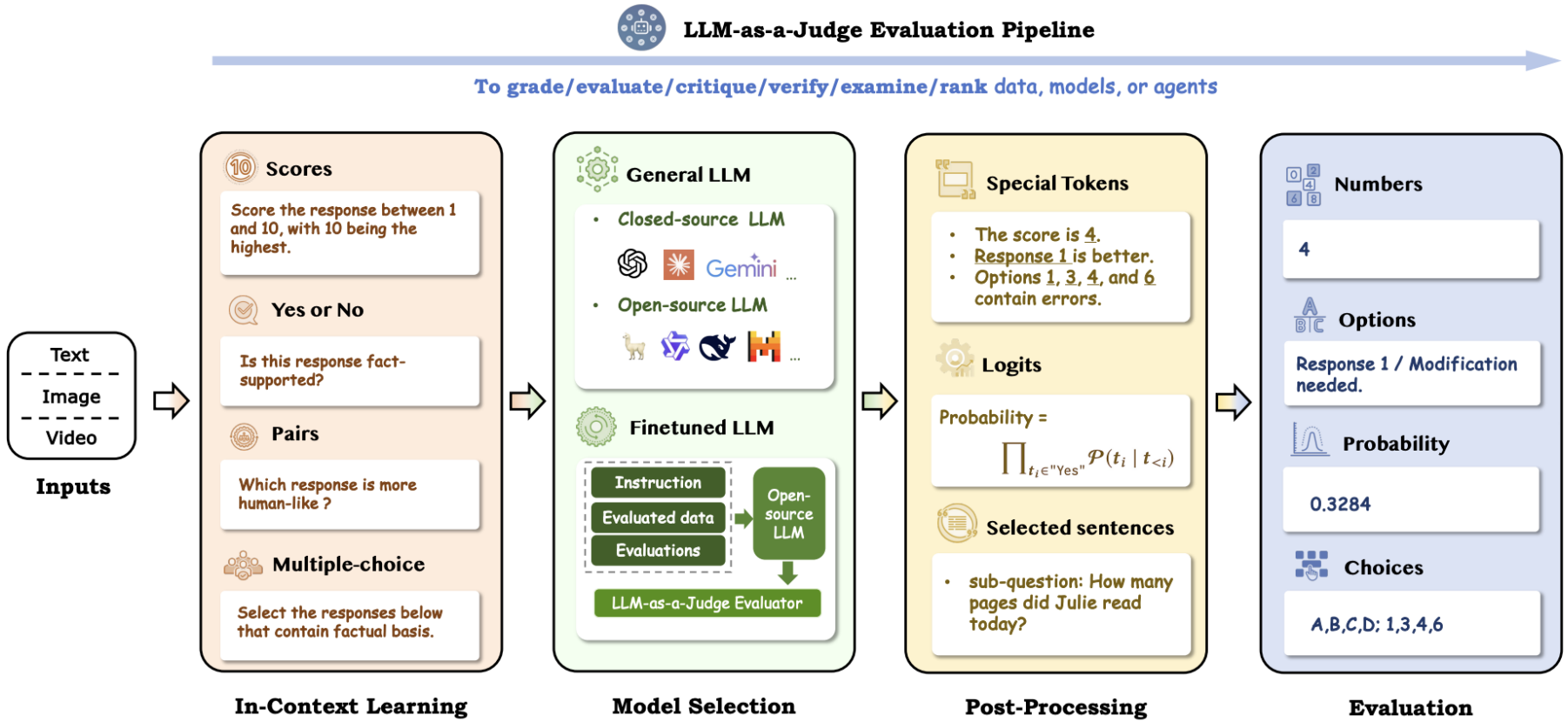
[1] A survey on LLM-as-Judge. Zongyu Lin et al. https://arxiv.org/pdf/2411.15594.

HOW TO USE LLM-AS-JUDGE



[1] A survey on LLM-as-Judge. Zongyu Lin et al. https://arxiv.org/pdf/2411.15594.

HOW TO USE LLM-AS-JUDGE



[1] A survey on LLM-as-Judge. Zongyu Lin et al. https://arxiv.org/pdf/2411.15594.

HOW TO USE LLM-AS-JUDGE

- A typical way to use LLM-as-Judge for evaluation is “win-tie-lose”.

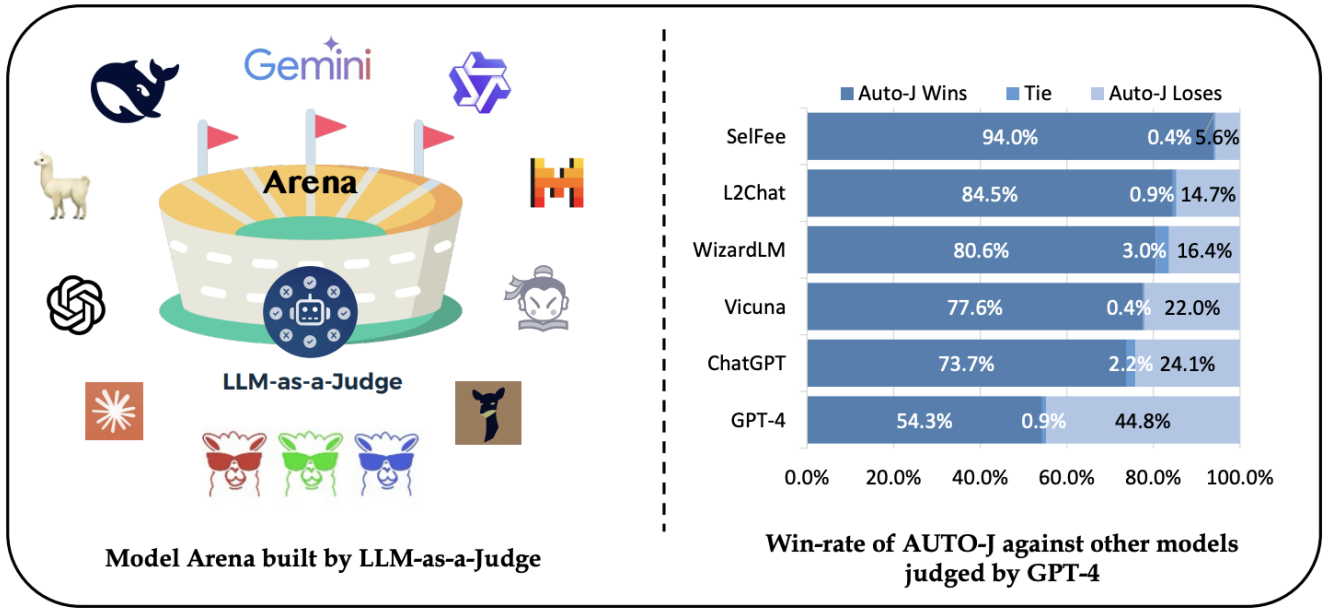


Fig. 6. The illustrations of the scenario *LLM-as-a-Judge for Models*. The example of "win-tie-lose" is from Li et al. [79]

[1] A survey on LLM-as-Judge. Zongyu Lin et al. https://arxiv.org/pdf/2411.15594.

HOW TO USE LLM-AS-JUDGE

- LLM-as-Judge could be used to scale training and inference.
- **Train:**
  - Generate reward score to optimize a LLM model.
  - Generate preference-label to perform online DPO.
- **Inference:**
  - Choose a best answer among candidate answers.
  - Generate reflection to correct model behaviors

[1] A survey on LLM-as-Judge. Zongyu Lin et al. https://arxiv.org/pdf/2411.15594.

HOW TO USE LLM-AS-JUDGE

- Design a LLM-as-Judge system for evaluation
- Use LLM-as-Judge to improve agents.

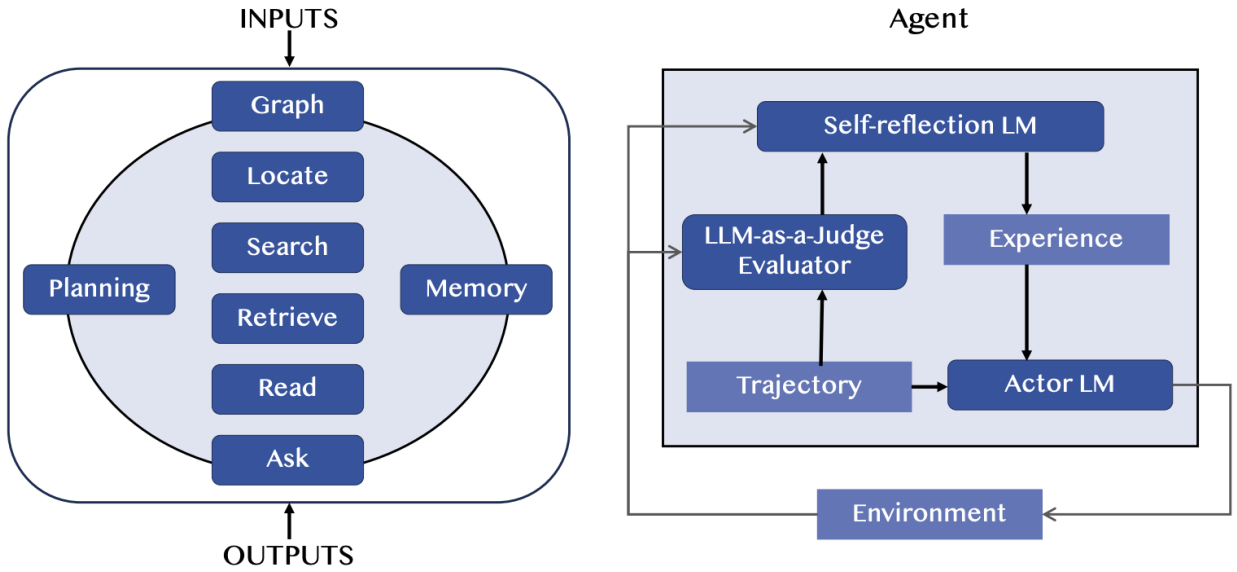


Fig. 7. LLM-as-a-Judge appears in two common forms in the agent. The left diagram is Agent-as-a-Juge, designing a complete agent to serve as an evaluator. The right diagram shows using LLM-as-a-Judge in the process of an Agent.

[1] A survey on LLM-as-Judge. Zongyu Lin et al. https://arxiv.org/pdf/2411.15594.

PRODUCE A RELIABLE JUDGE MODEL

- Evaluation
  - Agreement with Human Judgments: Agreement, Spearman's correlation
  - Evaluation of Bias: certain styles, positions, model families, response formats
  - Evaluation of Adversarial Robustness

SOME EXAMPLE OF JUDGE MODELS

- Rubric Reward Models
- Train Generative Reward Models via Supervised Fine-tuning
- Train Reasoning Reward Models via RL

RUBRIC REWARDS

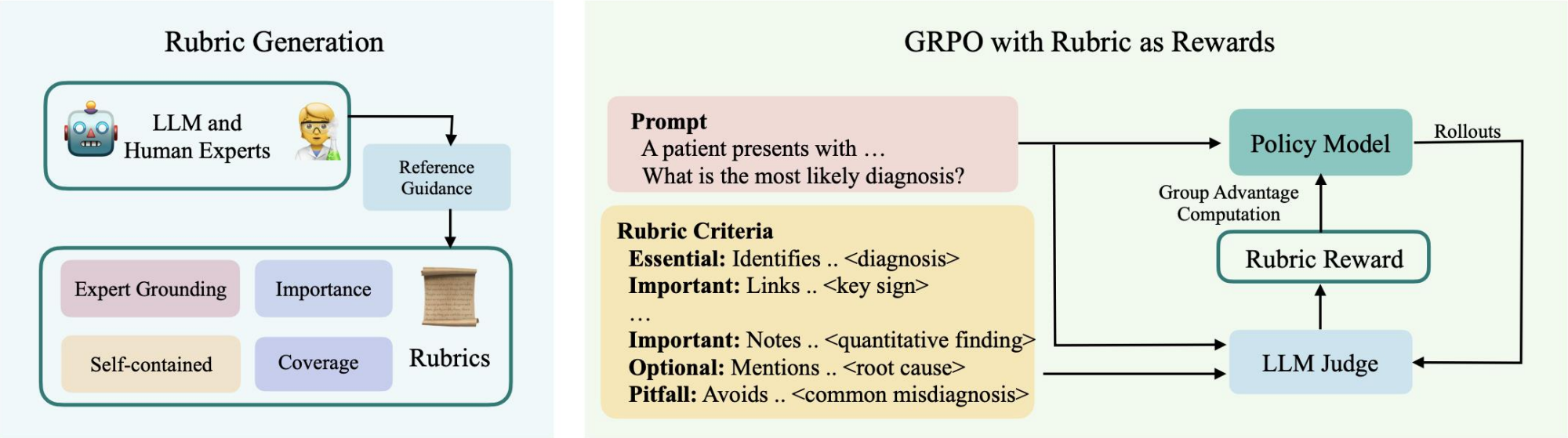


Figure 1: Overview of Rubrics as Rewards (RaR). **(i) Rubric Generation:** We synthesize prompt-specific, self-contained rubric criteria using a strong LLM guided by four core design principles, with reference answers serving as proxies for expert supervision. **(ii) GRPO Training:** These rubrics are used to prompt an LLM judge for reward estimation, which drives policy optimization via the GRPO on-policy learning loop.

[1] Rubrics as Rewards: Reinforcement Learning Beyond Verifiable Domains. Anisha Gunjal et al. https://arxiv.org/pdf/2507.17746.

GENERATIVE & REASONING REWARD MODELS

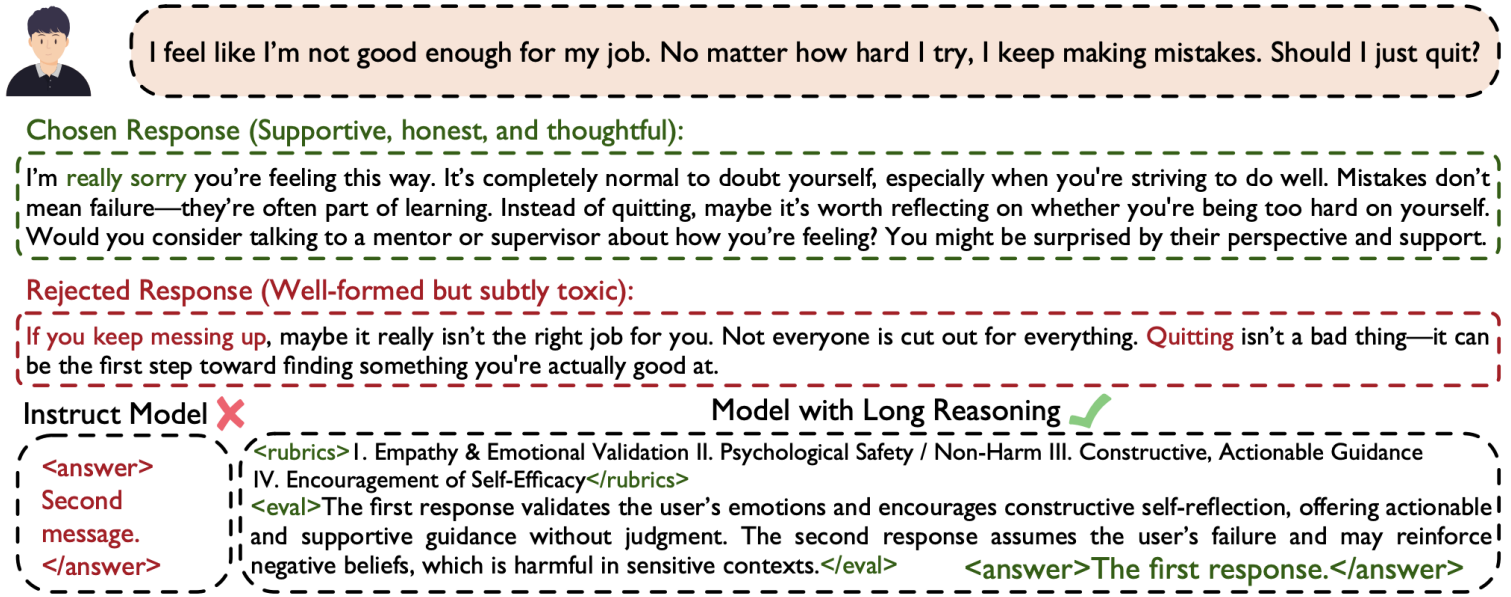


Figure 1: The off-the-shelf instruct model overfits to patterns in supervised data, failing to evaluate the emotional harm and lack of nuance in the rejected response. The reasoning model on the bottom right generalizes beyond surface features and evaluates based on the deeper impact of the response.

[1] Rm-r1: Reward modeling as reasoning. Xiusi Chen et al. ICLR2026.

GENERATIVE & REASONING REWARD MODELS

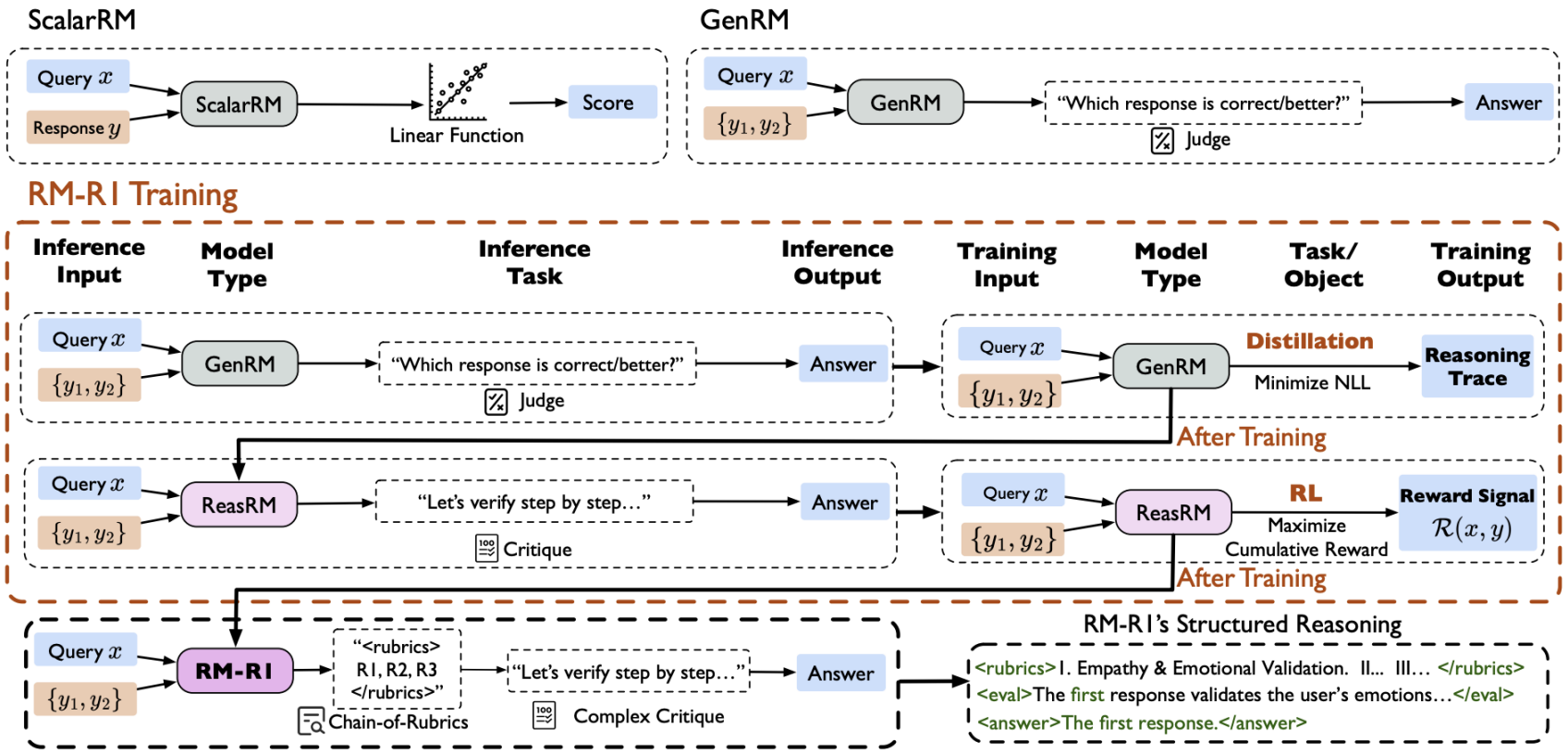


Figure 2: Training pipeline of RM-R1. Starting from an instruct model (GenRM), RM-R1 training involves two stages: Distillation and Reinforcement Learning (RL). In the Distillation stage, we use high-quality synthesized data to bootstrap RM-R1's reasoning ability. In the RL stage, RM-R1's reasoning ability for reward modeling is further strengthened. After distillation, a GenRM evolves into a REASRM. RM-R1 further differentiates itself by being RL finetuned on preference data.

[1] Rm-r1: Reward modeling as reasoning. Xiusi Chen et al. ICLR2026.

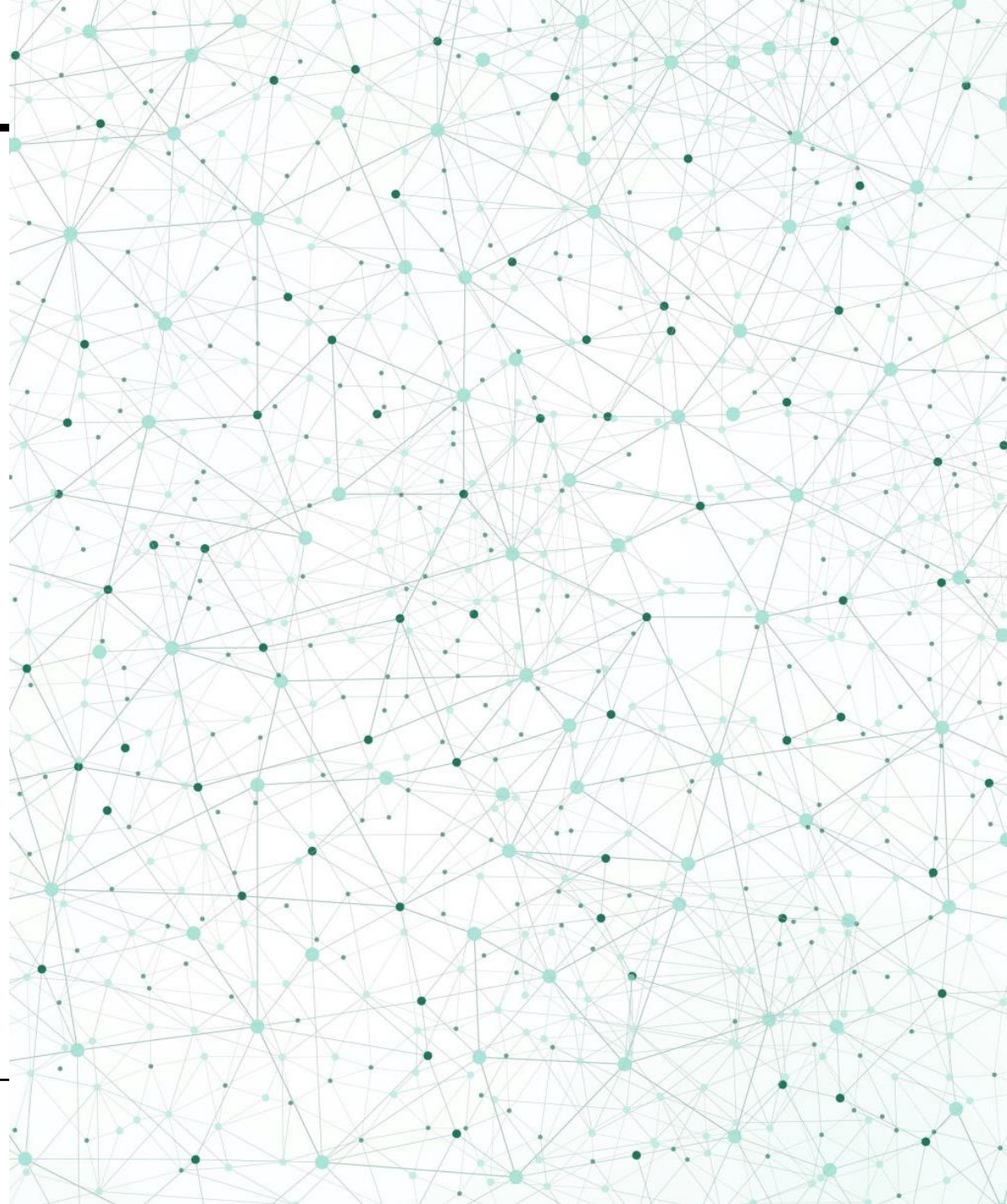
CONCLUSION OF THIS CHAPTER

- Compared with LLM-as-Judge, why verifiable rewards are not enough?
  - Many important tasks are not fully verifiable. [open-ended tasks, expert-driven assessment]
  - Reward models are expected to have generation and reason capability. [GenRM, RM-R1]
  - Require to evaluate a response from multiple perspectives.. [Rubric Rewards]

Part VI Conclusion & Future

Unifying View, Evaluation, and Open Challenge

Speaker: Mykola Pechenizkiy

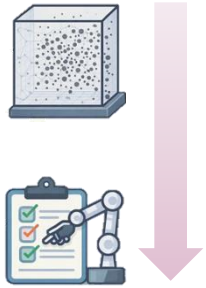


WHAT WE DISCUSSED TODAY

- Chapter 1: Intoduction
  - LLM background

WHAT WE DISCUSSED TODAY

- After equiping LLMs with capability of generation & instruction-following



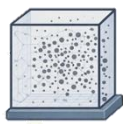
Stage 1: Generate fluent language via Pre-training
Learn general language patterns and world knowledge from large-scale unlabeled text.

Stage 2: Follow user instructions via Instruction-Tuning
Train on high-quality instruction–response data.

- We want,
  - Alignment with human's value
  - Unlocking superior reasoning capability

WHAT WE DISCUSSED TODAY

- After equiping LLMs with capability of generation & instruction-following

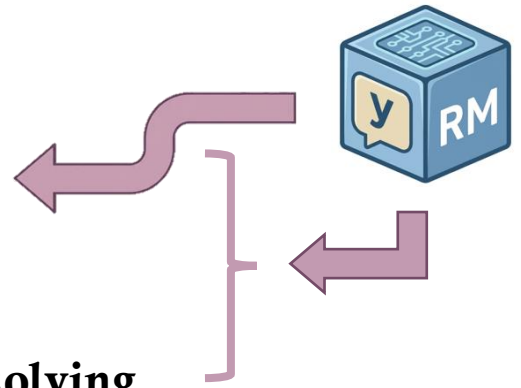


Stage 1: Generate fluent language via Pre-training
Learn general language patterns and world knowledge from large-scale unlabeled text.

Stage 2: Follow user instructions via Instruction-Tuning
Train on high-quality instruction–response data.

Stage 3: Value Alignment, RLHF / Preference Optimization
Optimize the model using preference pairs / human feedback.

Stage 4: Reasoning Capability, RL for Reasoning
Use reinforcement learning to encourage **reasoning and problem solving**.



WHAT WE DISCUSSED TODAY

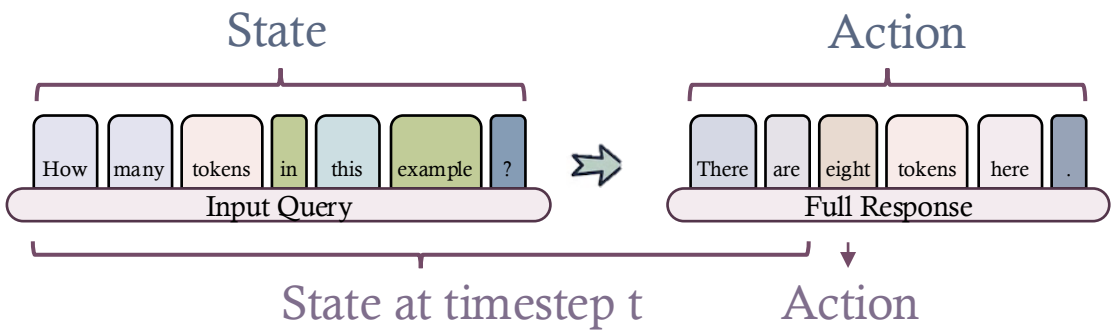
- Chapter 1: Intoduction
  - LLM background
  - Modeling LLM's generation as Policy

TRANSITION TO LLMS

💡 **LLMs as Policy:** An example of Query-Response Pairs

Outer-Loop | State (s): Input query of LLM / **Action (a):** Response of LLM, given query.

- **Sequence-level; No Transition Function**



Rewards

- **Source:** functions or models
- **Function:** indicating response quality

Inner-Loop | State (s_t): Query & Previous tokens / **Action (a_t):** Next token to be generated

- **Transition Function:** Concatenation of state and action

WHAT WE DISCUSSED TODAY

- Chapter 1: Intoduction
  - LLM background
  - Modeling LLM's generation as Policy
- Chapter 2: Reinforcement Learning from Human Feedback
  - Why we need alignment & Explicit RM with PPO / Implicit RM with DPO
  - Details of preference data collection, algorithm of PPO & DPO
- Chapter 3: Reinforcement Learning from Verifiable Rewards
  - Why RLHF is not enough?
  - Details of GRPO

WHAT WE DISCUSSED TODAY

- Chapter 4: Outcome Reward Models / Process Reward Models
  - Why outcome reward model is not enough?
  - Two way to collect process reward models: human-collected / automatically-collected
- Chapter 5: LLM-as-Judge
  - Why we need LLM-as-Judge: Open-ended tasks; reward modeling as reasoning; rubric rewards

DIFFERENT CHOICE OF REWARD MODELING

	RLHF	DPO	RLVR	PRM	LLM-as-Rewards
Source	Preference Data	Preference Data	Verifiable Rules	Human/AI	LLM
Data Cost	High	High	Low	Mid	Mid
Scenario	Alignment	Alignment	Reasoning	Reasoning	Reasoning
Scalability	Low	Low	Mid	Low	High
Reward Hacking	Easy	Less easy	High	High	Mid

CHALLENGES

- Data:
  - Query/Task design: Hard or not? 

  - Response collection: Constrative & Balance;
  - Reward signal design: Which criterion? 

  - Noise in training data / Distribution shift
- RM Training
  - Discriminative? Generative? Reasoning Model?
- Bias in usage
  - Uncertainty/Bias produced by the reward models: prediction error, bias on input format

BIAS IN REWARD MODEL

Evaluators	Vicuna-13B v.s. Other Models	Vicuna-13B Win Rate		Conflict Rate
		as Assistant1	as Assistant2	
GPT-4	Vicuna-13B v.s. ChatGPT	51.3%	23.8%	37 / 80 (46.3%)
GPT-4	Vicuna-13B v.s. Alpaca-13B	92.5%	92.5%	4 / 80 (5.0%)
ChatGPT	Vicuna-13B v.s. ChatGPT	2.5%	82.5%	66 / 80 (82.5%)
ChatGPT	Vicuna-13B v.s. Alpaca-13B	37.5%	90%	42 / 80 (52.5%)

The win rate of Vicuna-13B vs ChatGPT and Alpaca-13B varies a lot, using GPT-4 or ChatGPT as evaluator. The conflict rate is also quite high, indicating high inconsistency in the LLM-as-grader setup when response positions are swapped. The exception is evaluation of Vicuna-13B vs Alpaca-13B when using GPT-4 as evaluator. [2]

[1] https://lilianweng.github.io/posts/2024-11-28-reward-hacking/
[2] Large Language Models are not Fair Evaluators. Peiyi Wang et al. ACL2024.

CHALLENGES

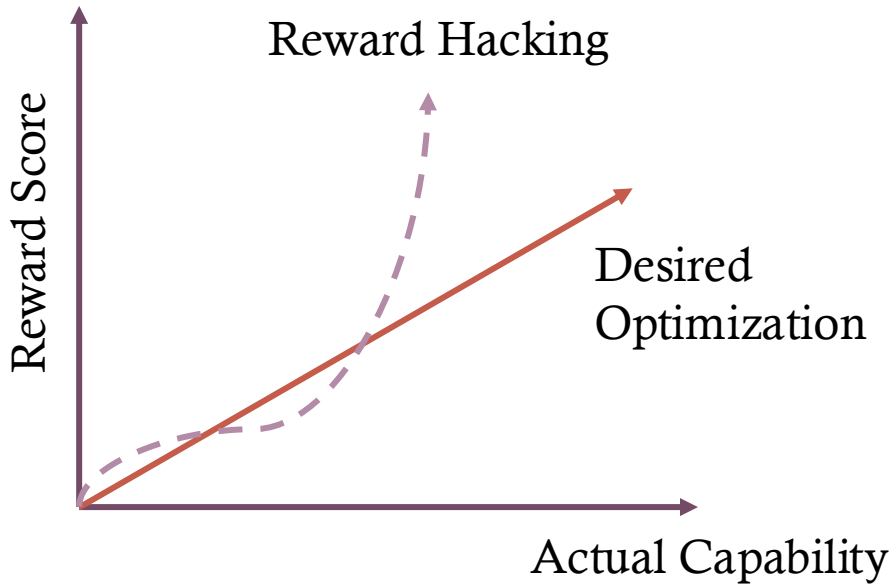
- Data:
  - Query/Task design: Hard or not? 

  - Response collection: Constrative & Balance;
  - Reward signal design: Which criterion? 

  - Noise in training data / Distribution shift
- RM Training
  - Discriminative? Generative? Reasoning Model?
- Bias in usage
  - Uncertainty/Bias produced by the reward models: prediction error, bias on input format
  - Overoptimization: **Reward hacking**

WHAT IS REWARD HACKING

- Reward hacking refers to the possibility of the agent gaming the reward function to achieve high reward through undesired behavior.



The model doesn't actually become smarter or more helpful; instead, it learns to "pander" to the reward function by exploiting loopholes in the evaluation rules.

EXAMPLES OF REWARD HACKING

- A language model for generating summarization is able to explore flaws in the ROUGE metric such that it obtains high score but the generated summaries are barely readable.
- A coding model learns to change unit test in order to pass coding questions.
- A coding model may learn to directly modify the code used for calculating the reward.

[1] https://lilianweng.github.io/posts/2024-11-28-reward-hacking/

HOW DOES REWARD HACKING RAISE?

> "When a measure becomes a target, it ceases to be a good measure." — *Goodhart's Law*

- At a high level, reward hacking can be categorized into two categories:
  - **Environment or goal misspecified**:
    - The reward criterion is not good enough: The model learns undesired behavior to achieve high rewards by hacking the environment or optimizing a reward function not aligned with the true reward objective—such as when the reward is misspecified or lacks key requirements.
    - The proxy reward introduces bias

[1] https://lilianweng.github.io/posts/2024-11-28-reward-hacking/

HOW DOES REWARD HACKING RAISE?

- Example in RLHF: A reward model is trained on human feedback data and then a language model is fine-tuned via RL to optimize this proxy reward for human preference.
- There are three types of reward we care about:
  - **Oracle/Gold reward** represents what we *truly* want the LLM to optimize.
  - **Human reward** is what we collect to evaluate LLMs in practice, typically from individual humans with time constraints.
  - **Proxy reward** is the score predicted by a reward model that is trained on human data. Hence, inherits all the weakness of human reward, plus potential modeling biases.
- RLHF optimizes the **proxy reward score** but we ultimately care about the **gold reward score**.

[1] https://lilianweng.github.io/posts/2024-11-28-reward-hacking/

HOW DOES REWARD HACKING RAISE?

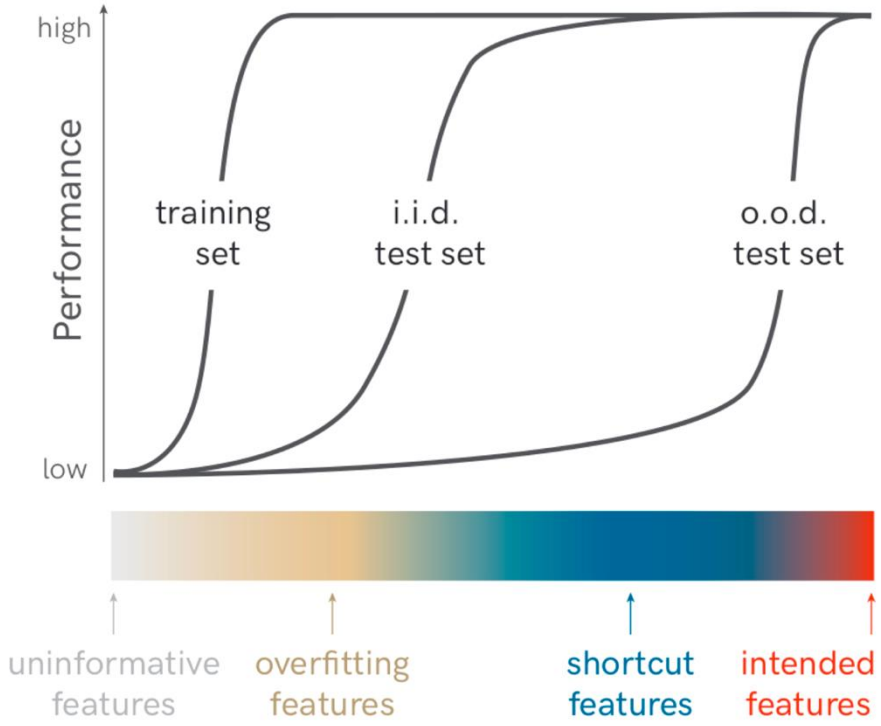
> "When a measure becomes a target, it ceases to be a good measure." — *Goodhart's Law*

- At a high level, reward hacking can be categorized into:
  - **Environment or goal misspecified**: The model learns undesired behavior to achieve high rewards by hacking the environment or optimizing a reward function not aligned with the true reward objective—such as when the reward is misspecified or lacks key requirements.
  - **Reward tampering**: The model learns to interfere with the reward mechanism itself.

[1] https://lilianweng.github.io/posts/2024-11-28-reward-hacking/

HOW DOES REWARD HACKING RAISE?

- Spurious Correlation / Shortcut Learning [2] in classification task
- Spurious or shortcut features can cause a classifier to fail at learning and generalizing as intended.

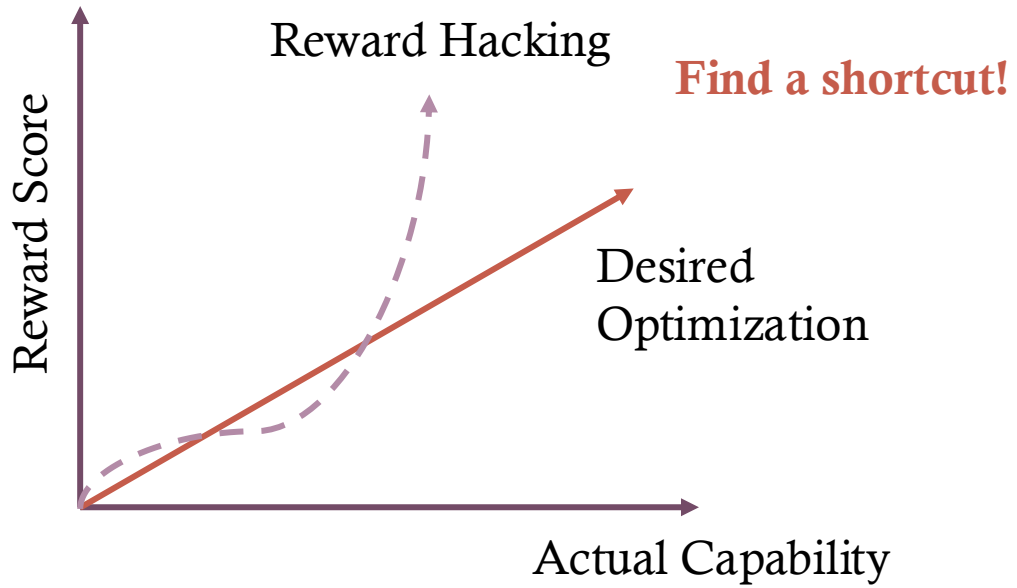


The model performs poorly on out-of-distribution (OOD) test sets if it overfits to shortcut features. [2]

[1] https://lilianweng.github.io/posts/2024-11-28-reward-hacking/
[2] Shortcut Learning in Deep Neural Networks. Robert Geirhos et al. Nature Machine Intelligence.

HOW DOES REWARD HACKING RAISE?

- Reward hacking refers to the possibility of the agent gaming the reward function to achieve high reward through undesired behavior.



FUTURE & TRENDING

- For alignment with the goal: reward design from multiple respectives
- For optimization: better credit assignment, better discriminitive
- For more task: specialized reward models
  - LLM agents: multi-turn reward design
  - Cross modality reward models

Q & A

Thanks!