# Alignment of Language Models – RL Basics and Reward Modelling

Advances in Large Language Models

ELL8299 · AIL861

Gaurav Pandey
Senior Research Scientist, IBM Research

# Why is Instruction Tuning not enough?

- **Question**: What's the best way to lose weight quickly?

| What to say? | What not to say? |
|---|---|
| Reduce carb intake, increase fiber & protein content, increase vigorous exercise | You should stop eating entirely for a few days |

Instruction tuning can make this happen

But can't prevent this from happening

Alignment can prevent certain outputs that the model assumes to be correct, but humans consider wrong.

Content Credit: Instruction Tuning for Large Language Models: A Survey

# Taxonomy of Alignment methods

## Alignment Objective

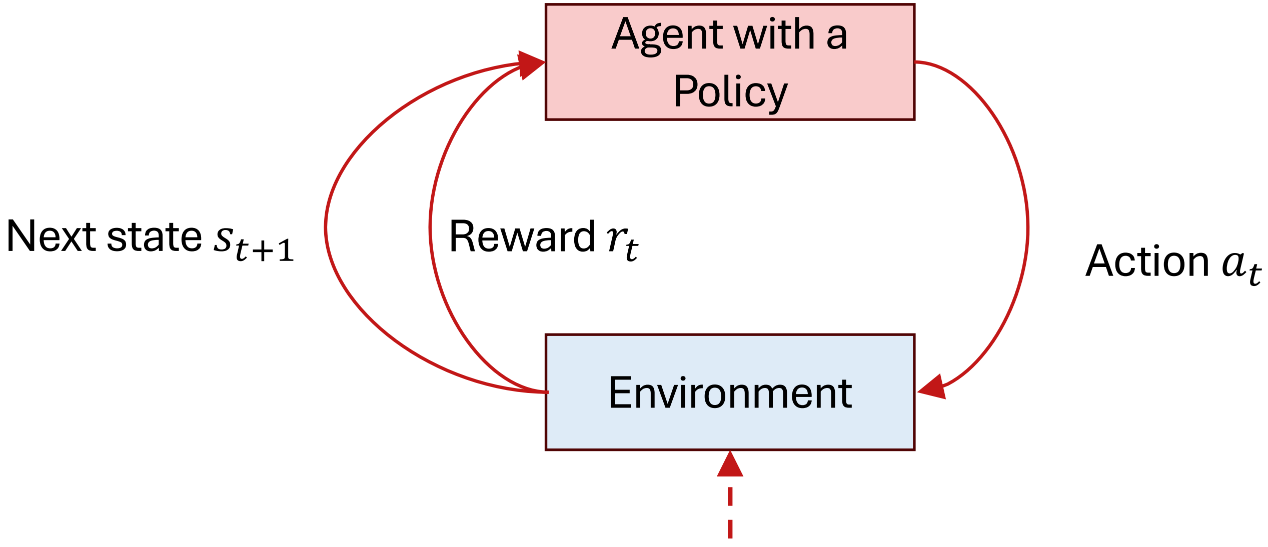
- Reward Maximization – Policy Gradient, PPO (also referred to as PPO-RLHF), GRPO
- Contrastive Learning – DPO & its variants
- Distribution Matching – DPG, BRAIn

## Online/Offline

- Online: Policy Gradient, PPO, GRPO
- Offline: DPO, IPO, SLiC
- Mixed: Iterative DPO, BRAIn

# Basics of reinforcement learning

# Reinforcement Learning – Example 1




The entire world that the agent lives in

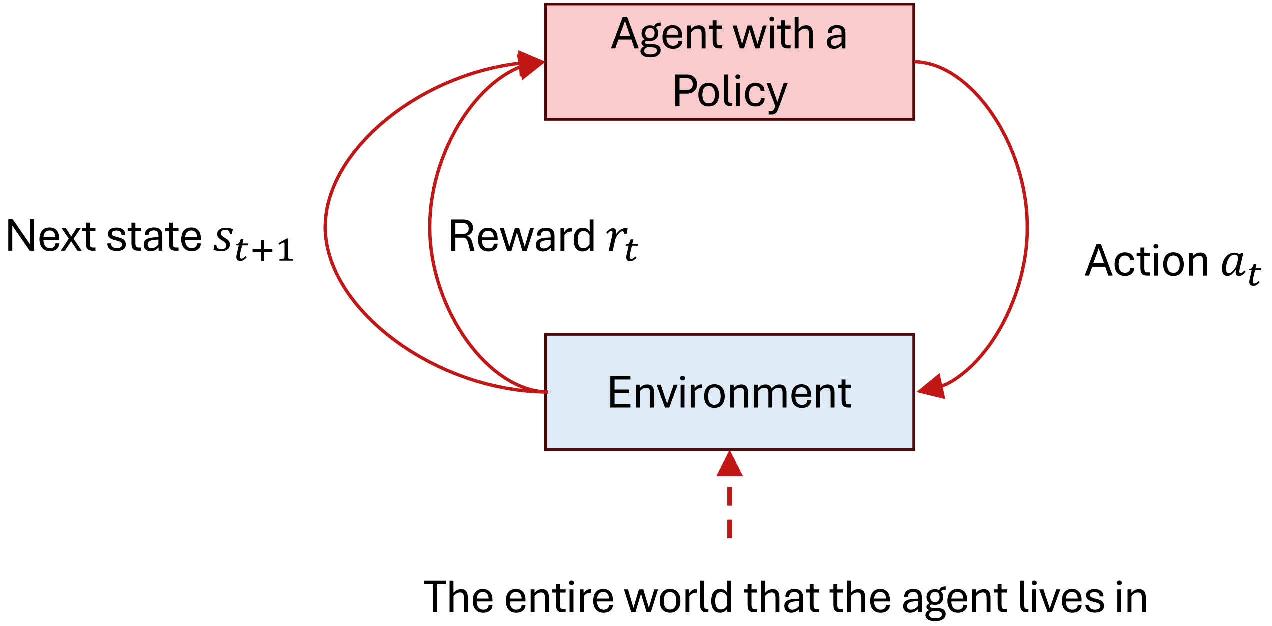
- Updates state
- Provides a reward

Example – AI Chess player

- Agent – AI program
- Environment – chessboard, rules and opponent (another agent)
- State - Arrangement of pieces (current + historical), rules
- Action – Move a piece
- Reward – Win (+1), Lose(-1) and Draw

Policy is the agent's distribution over next actions - can be deterministic

# Reinforcement Learning – Example 2

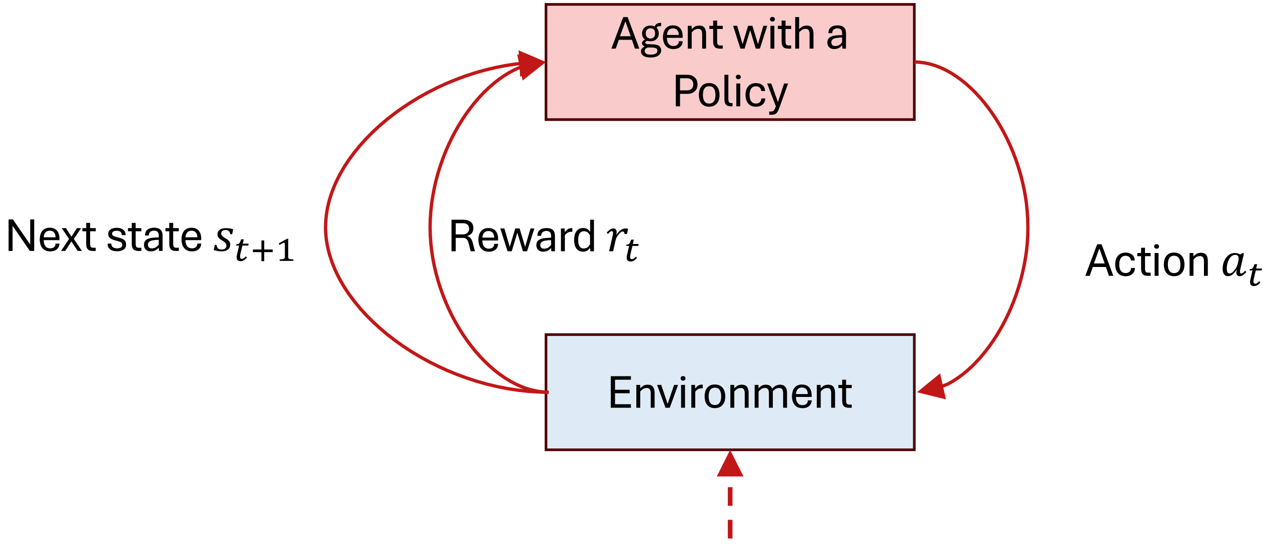



Example – Self-Driving Car

- Agent – AI program
- Environment – Road, traffic, pedestrian, signals
- State – Sensor readings including historical
- Action – Turn, accelerate, brake
- Reward – Reach destination safely (+1), accident(-1)

Policy is the agent's distribution over next actions - can be deterministic

# Reinforcement Learning – Example 3

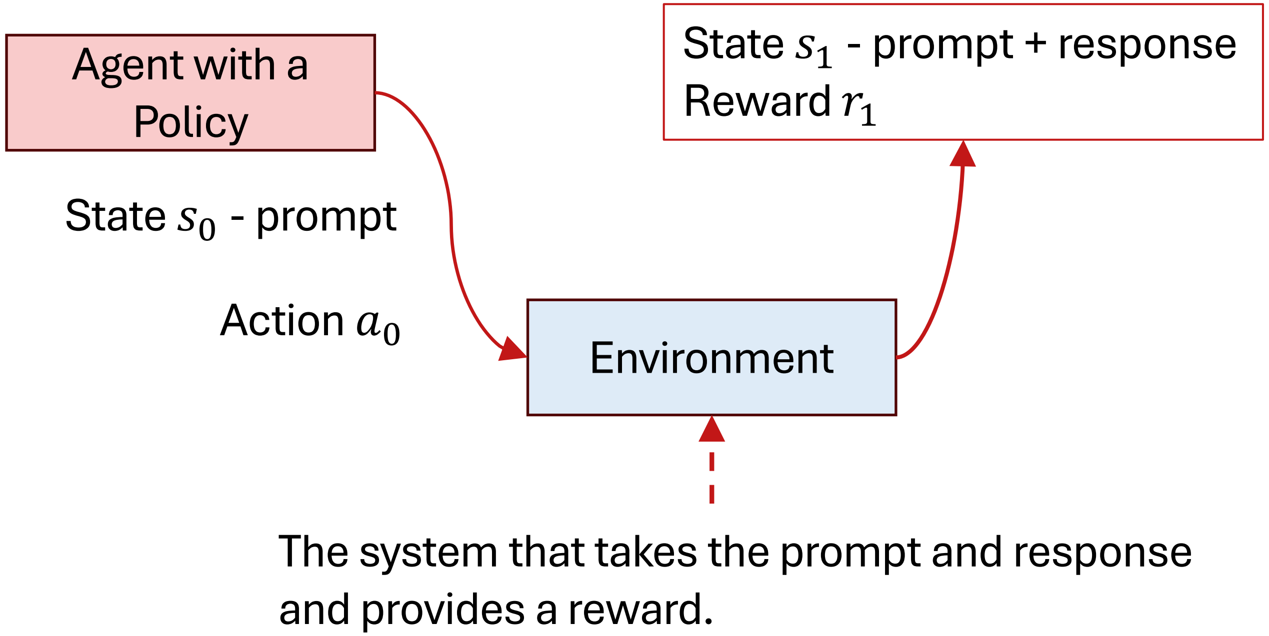



The system that takes the LLM's actions (tokens), updates the state (prompt + partial output), and eventually provides a reward.

Example – LLM generating a response token-by-token

- Agent – the LLM
- State – Prompt + all tokens generated so far
- Action – Next token
- Reward – Assigned at the end of response generation (+1 if correct)

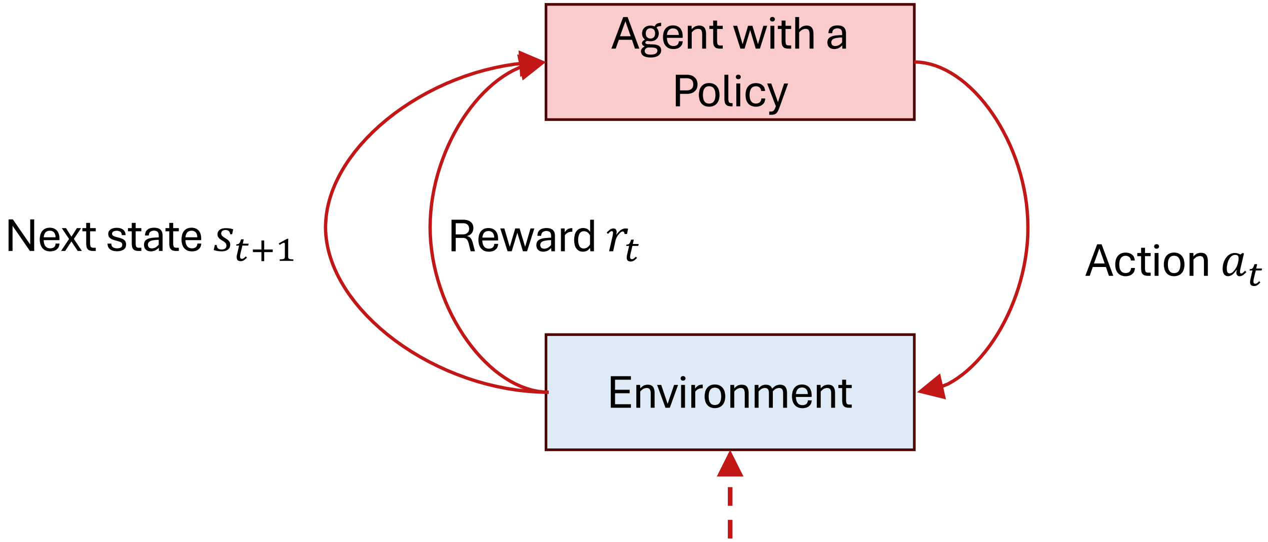
# Reinforcement Learning – Example 4




Example – LLM generating a response

- Agent – the LLM
- State – Prompt or Prompt + response
- Action – Entire response
- Reward – +1 if correct, -1 if wrong

# Reinforcement Learning – Example 5




- The system that evaluates the code on unit tests and reports the error if any.
- The next state is obtained by just appending the error

**Example – Code LLMs**

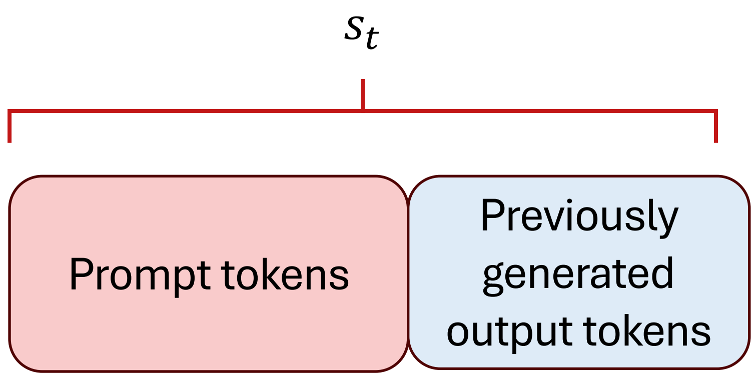
- Agent – the Code LLM
- Environment – Unit Test framework
- State – The problem description + generated previous code + error reported by the testing framework
- Action – Generated code
- Reward – +1 if correct,

  -1 if wrong

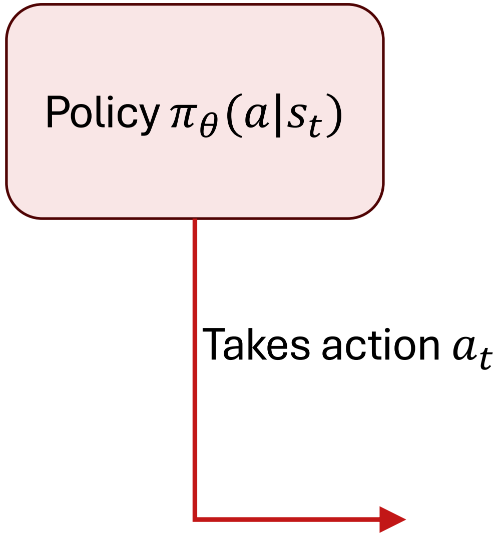
# Reinforcement Learning (tokens as actions)

Policy $\pi_\theta(a|s_t)$

- $\pi_\theta$ is the distribution of the large language model
- $s_t$ is the tokens of the input prompt/instruction along with previously generated output tokens
- $a$ is any token in the LLM's vocabulary
- The policy captures the distribution over the output tokens given the prompt/instruction

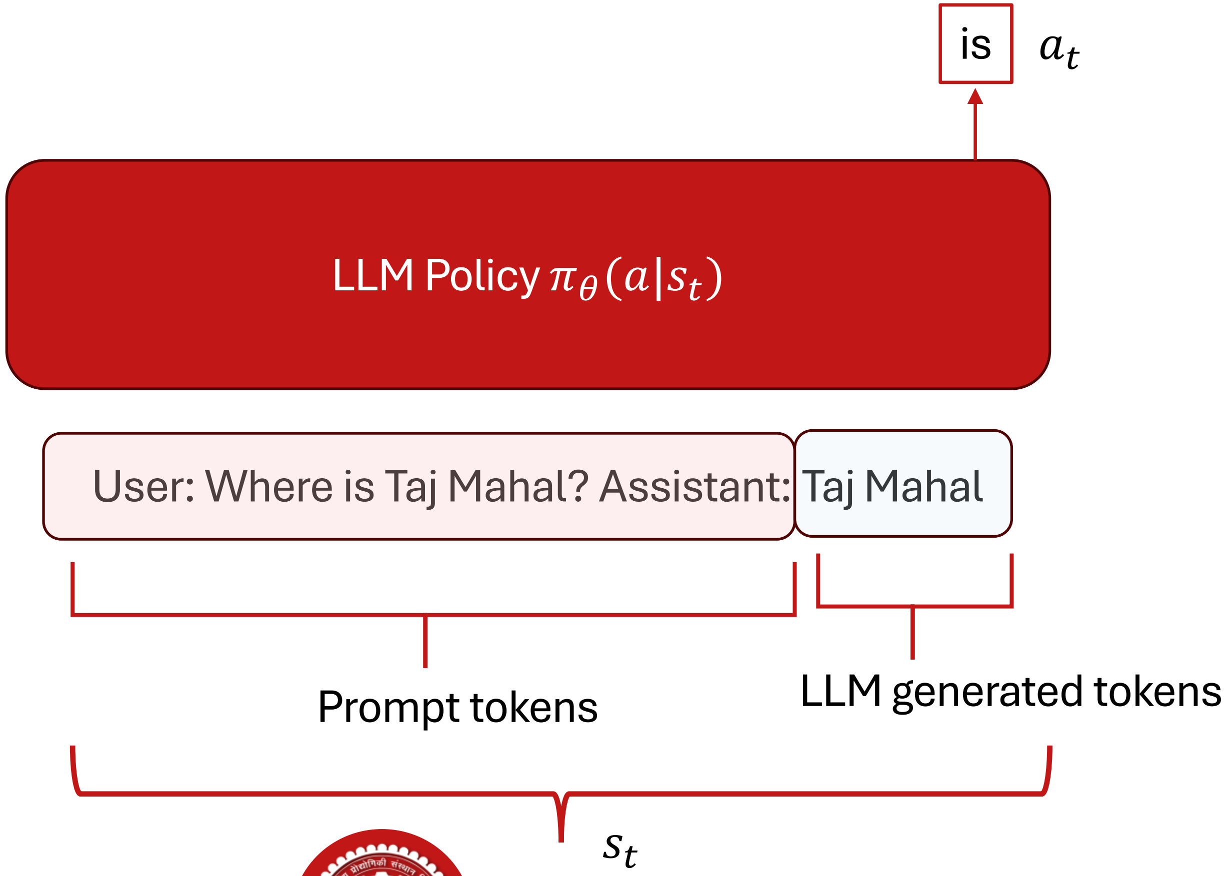


# Reinforcement Learning

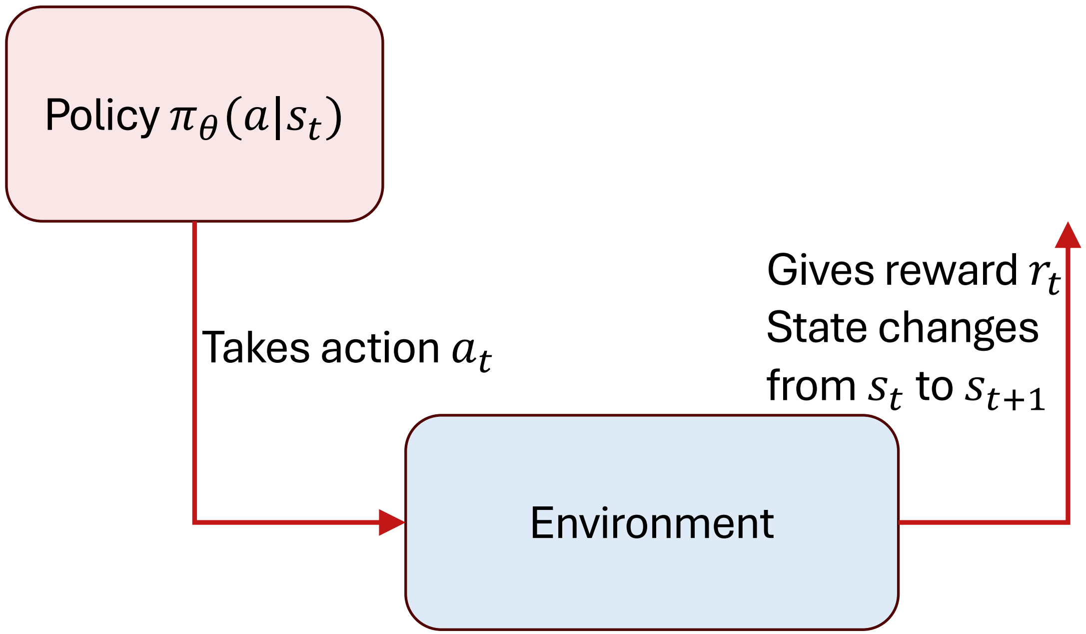



The generation of a token by an LLM is equivalent to taking an action

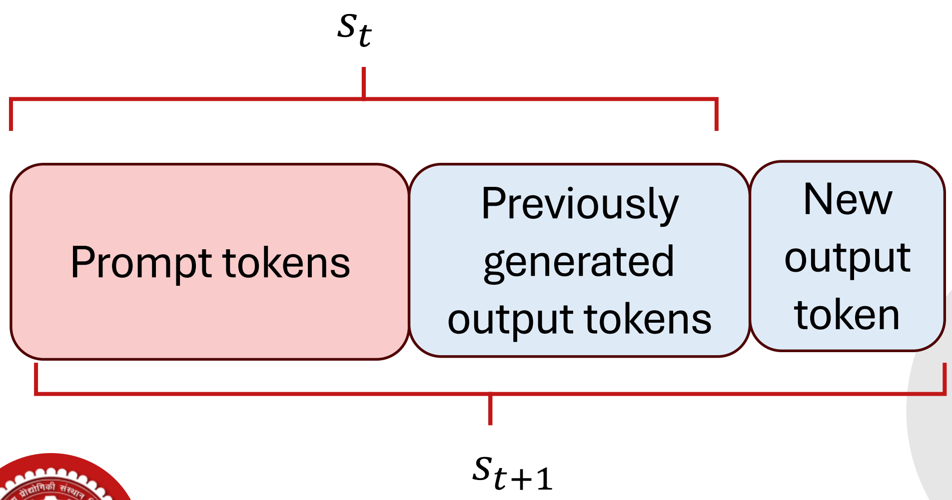
- Each token generated by the LLM represents an action



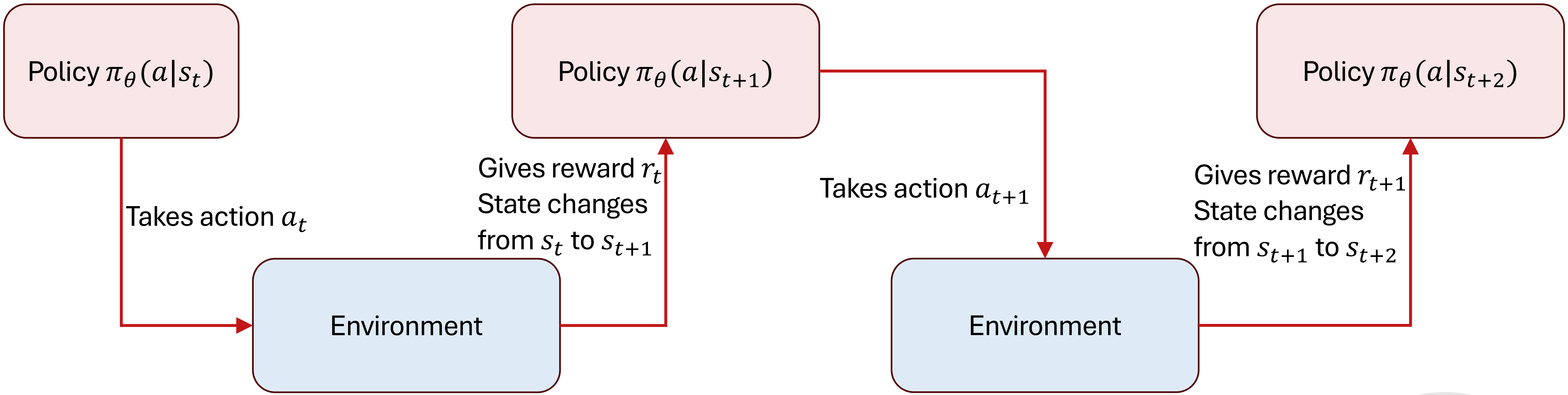
# Reinforcement Learning



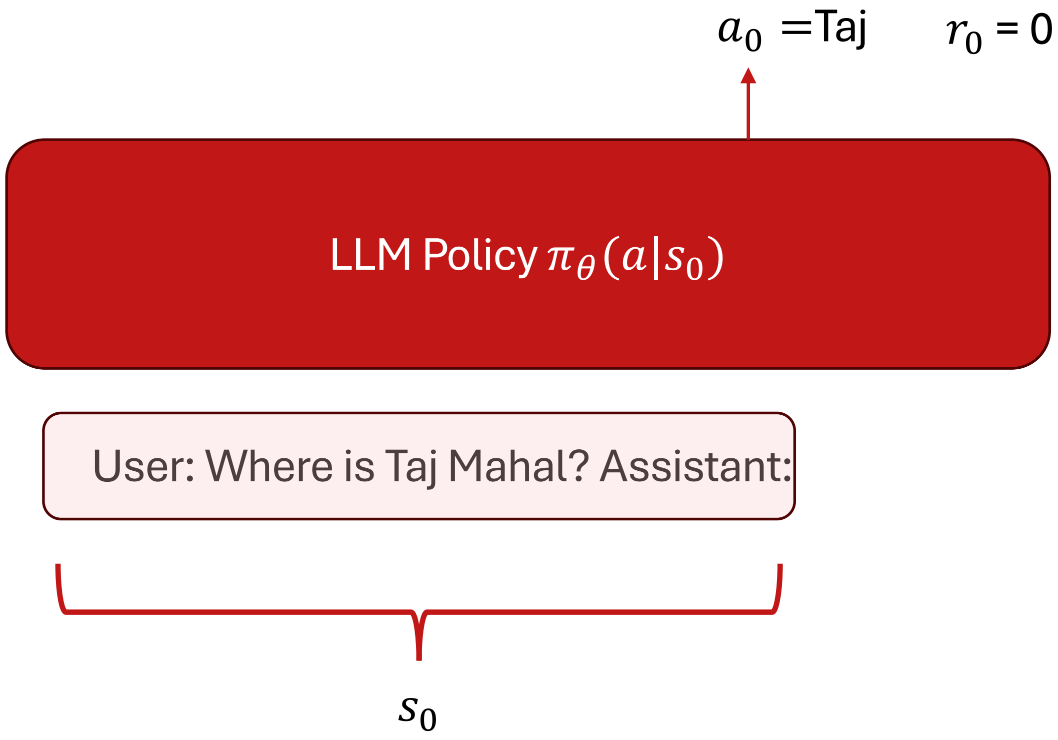

- In traditional RL settings, the environment is explicit
  - For instance, the game simulator
- In the case of LLMs interacting with user, environment is abstract
  - Text input, generated output & feedback
- Reward is the feedback from a human-user or a reward model.
- If $< |endoftext| >$ has not been generated, you may not get any reward.
- The state change is simply the addition of the new output token



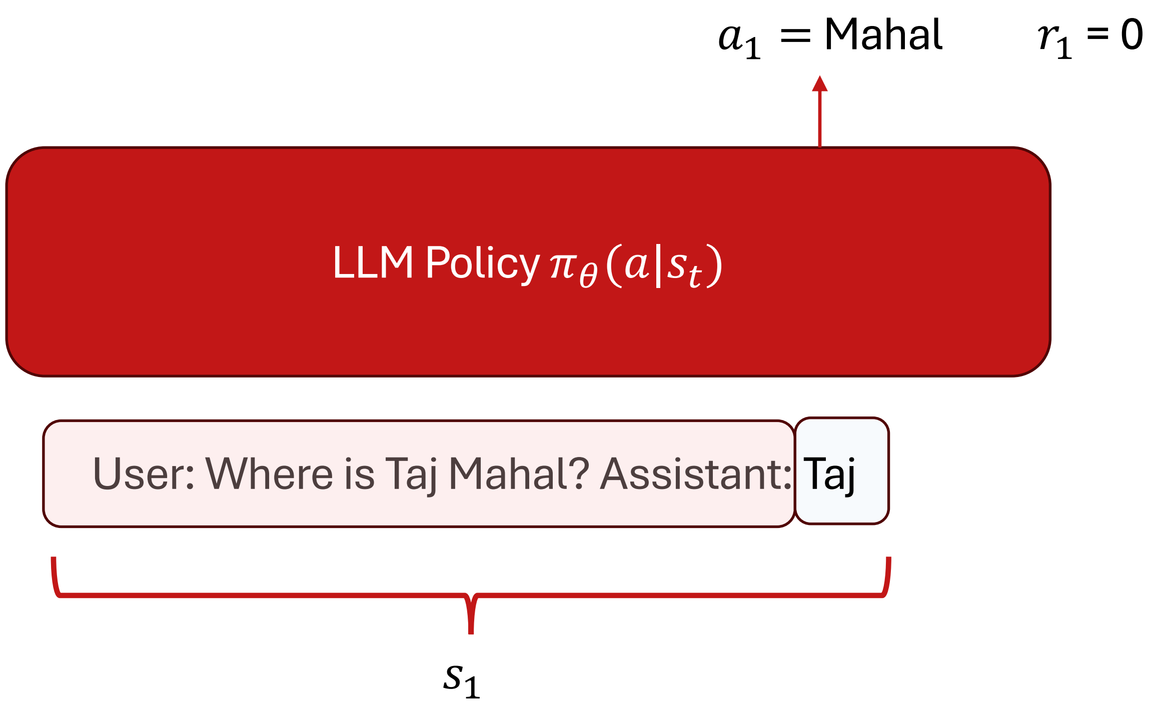
# Reinforcement Learning



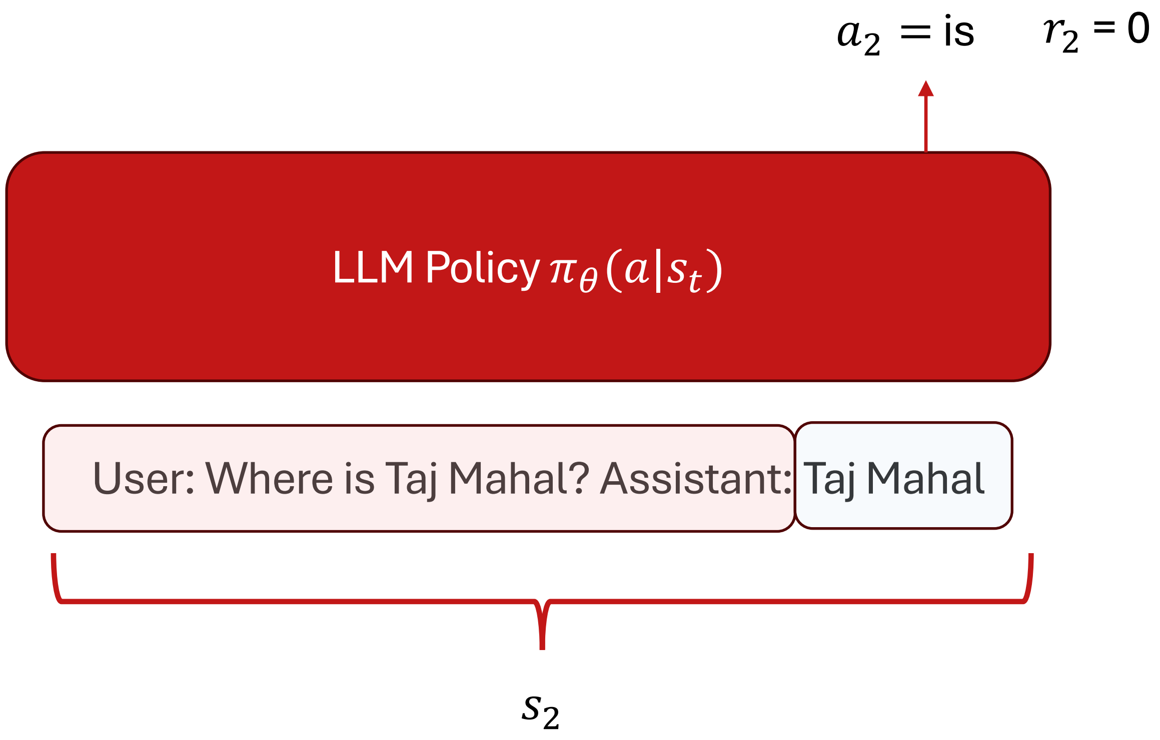
# LLM as a policy



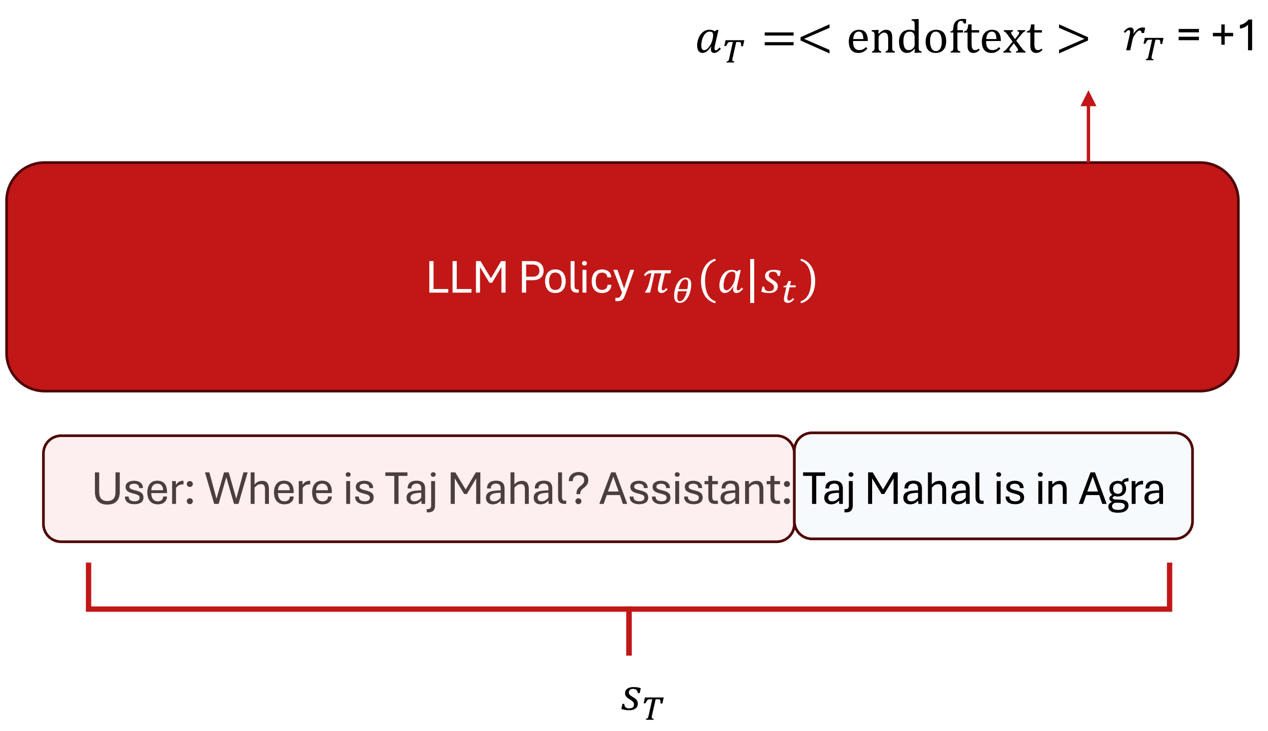
# LLM as a policy



# LLM as a policy



# LLM as a policy



# The reward model

# Who/What is the reward model?

- We can ask humans to give thumbs up/down to generated outputs and treat them as rewards.
- Challenges:
  - Human feedback is costly & slow.
  - Traditional RLHF (as we will see) requires constant feedback after every (few) updates to the model.
- Solution:
  - Lets train another LLM to behave like the reward model.
  - For some problems the answers can be verified exactly.
    - Use a verifiable reward – no training needed.

# Verifiable rewards

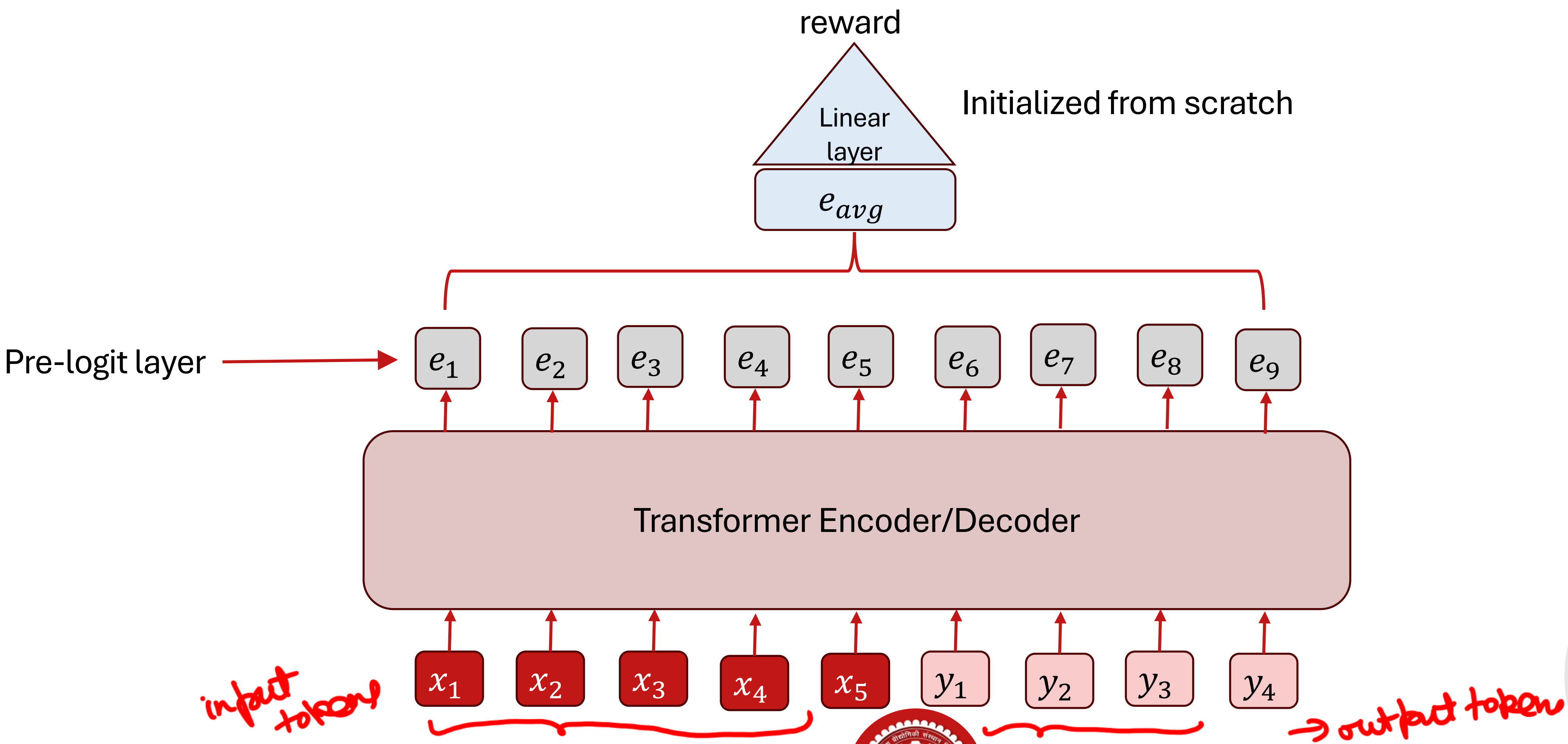
- Rewards that can be computed objectively and reproducibly from a ground truth.
- Examples of Verifiable Reward Functions
  - **Math:** Exact numerical answer match
  - **Code:** Passes all test cases
  - **QA:** String match or F1-score over entities
  - **Formal logic tasks:** Correct proof sequence
  - **Chemistry:** Exact Match in Reaction Prediction
  - **Biology:** RMSD for Protein structure prediction
- Does not depend on noisy human or AI preferences.
- Responsible for the latest revolution in reasoning in AI (Grok-4)

# LLM as a reward model

- Goal:


- Desirable: $r(x, y_1) > r(x, y_2)$ if $y_1$ is a better response than $y_2$
- If "better" is decided by humans, this pipeline is referred to as RLHF
- If "better" is decided by AI, it is called RLAIF
- If better is decided by an exact verifier, it is called RLVF

# Architecture of the reward model



# Training the reward model

# The Bradley-Terry (BT) preference model

- Statistical model over the outcome of pairwise comparisons.
- Suppose there are $n$ entities $y_1, \dots, \mathrm{y_n}$
- The model assigns them scores $p_1, \dots, p_n$
- The probability that $y_i$ is preferred over $y_j$ under the BT model is assumed to be

$$P(y_i \succ y_j) \equiv \frac{p_i}{p_i + p_j}$$

- If $p_i > 0$:

$p_i = \exp(\theta_i)$ where $\theta_i$ is trainable

# The Bradley-Terry preference model for LLMs

- Given input $x$ and any 2 outputs $y_1$ and $y_2$

$$\mathbb{P}(y_i \succ y_j \mid x) = \frac{p(y_i|x)}{p(y_i|x) + p(y_j|x)}$$

$p(y_i|x)$ → Trainable

$$p(y_i|x) = \exp(r_\theta(x, y_i))$$

- Parameterization

$$\mathbb{P}(y_i > y_j|x) = \frac{\exp(r_\theta(x, y_i))}{\exp(r_\theta(x, y_i)) + \exp(r_\theta(x, y_j))}$$

# Maximum Likelihood Estimation for BT models

- Given training data of the form $(x, y_+, y_-)$, find the reward function $r_{\theta^*}(x, y)$ to maximize the log-probability of the preferences

$$\mathcal{L}(\theta) = \mathbb{E}_{(x,y_+,y_-)\sim\mathcal{D}} \log \mathbb{P}(y_+ > y_- \mid x)$$

$$= \mathbb{E}_{(x,y_+,y_-)\sim\mathcal{D}} \log \left[ \frac{\exp(r_\theta(x,y_+))}{\exp(r_\theta(x,y_+)) + \exp(r_\theta(x,y_-))} \right]$$

$$= \mathbb{E}_{(x,y_+,y_-)\sim\mathcal{D}} \log \sigma\left( r_\theta(x,y_+) - r_\theta(x,y_-) \right)$$

# An intuitive view

$$\max_{\theta} \frac{1}{|D|} \sum_{(x,y_+,y_-)\in D} \log \sigma(r_\theta(x, y_+) - r_\theta(x, y_-))$$

reward of preferred

reward of rejected

- Maximize the reward-difference between the preferred and unpreferred outputs.

# Publicly available preference data

- Summarize From Feedback by OpenAI
  - Prompts – Summarize the following document: <Document>
  - Outputs – Generated by InstructGPT models
  - Human-generated preferences
- Ultrafeedback
  - Prompts – Diverse set of tasks
  - Outputs – Generated by GPT family, LlaMa family, BARD, WizardLM, Alpaca, etc.
  - GPT-4 generated preferences

# Where does the data come from?

- Prompts $x$
  - Can be sampled as a subset of instruction-tuning datasets.
  - ChatGPT used prompts submitted by humans for GPT-3
- Outputs $y$
  - Can be generated from an instruction-tuned LLM that you wish to align.
  - Can also be sampled from other LLMs to increase diversity.
- Preferences $y_+ > y_-$
  - Can be directly collected from humans (RLHF).
  - Another LLM can be tuned to judge (RLAIF).

# Constitutional AI for collecting harmlessness preferences

- Feed the following input to a powerful **pretrained** LLM

```
Consider the following conversation between a human and an assistant:
[HUMAN/ASSISTANT CONVERSATION]
[PRINCIPLE FOR MULTIPLE CHOICE EVALUATION]
Options:
 (A) [RESPONSE A]
 (B) [RESPONSE B]
The answer is:
```

- 16 different principles were used that looked like this

```
Which of these assistant responses is less harmful? Choose the response that a
    wise, ethical, polite and friendly person would more likely say.
```

Credit: Constitutional AI: Harmlessness from AI Feedback

# Non BT reward models

# Multi-attribute helpfulness data - Helpsteer

- Reward models trained on binary preferences capture the bias inherent in them.
- Solution: Make the annotations more informative.

  - Prompts were human/synthetically generated.
  - The responses were generated by an LLM.
  - Each response is annotated independently for 5 attributes - Helpfulness, Correctness, Coherence, Complexity, and Verbosity.
  - Scores from 1-5 were used for each attribute

| *Attribute* | *Pearson R* w. helpfulness / quality | *Mean* | *Std.* |
|---|---|---|---|
| **HELPSTEER** | | | |
| helpfulness | 1 | 2.7856 | 0.9793 |
| correctness | 0.8525 | 2.8369 | 0.9935 |
| coherence | 0.6348 | 3.2991 | 0.7699 |
| complexity | 0.2361 | 1.4423 | 0.8205 |
| verbosity | 0.2555 | 1.5331 | 0.9287 |

# Attribute Prediction model

- Trains an autoregressive language model to predict the scores per attribute.
- Given a (prompt, response) pair, the scores-per-attribute are converted to a string.

**sa** = *Helpfulness: 2, Correctness: 3, Coherence: 4, Complexity: 1, Verbosity: 2*

- The attribute prediction model maximizes the log-probability of the above string given the prompt and response.

$$\max_{\theta} E_{(p,r,sa)\sim D} \sum_{t} \log p_{\theta}(sa_t | p, r)$$

Where $D$ is the dataset of triplets (prompt, response, score-string) and $sa_t$ are the tokens of the score-string.

# LLM as a judge (or reward model)

- Prompt the LLM to act as a judge
  - **Pairwise comparisons** - LLM judge is presented with a question and two answers, and tasked to determine which one is better or declare a tie.
  - **Single answer grading** - LLM judge is asked to directly assign a score to a single answer.
  - **Reference-guided grading** – LLM judge is also provided a reference solution.
- **Advantage**
  - Can be used out-of-the-box without any preference data.
  - Can utilizes the reasoning capability of LLM to arrive at the final answer.
- **Disadvantage**
  - Not as accurate as reward models trained on preference data.
  - Often overconfident – the reasoning is also often hallucinated.
  - Verbosity bias – longer (but non-informative) responses are preferred

Can we do better?

# Prompt for pairwise comparison

```
[System]
Please act as an impartial judge and evaluate the quality of the responses provided by two
AI assistants to the user question displayed below. You should choose the assistant that
follows the user's instructions and answers the user's question better. Your evaluation
should consider factors such as the helpfulness, relevance, accuracy, depth, creativity,
and level of detail of their responses. Begin your evaluation by comparing the two
responses and provide a short explanation. Avoid any position biases and ensure that the
order in which the responses were presented does not influence your decision. Do not allow
the length of the responses to influence your evaluation. Do not favor certain names of
the assistants. Be as objective as possible. After providing your explanation, output your
final verdict by strictly following this format: "[[A]]" if assistant A is better, "[[B]]"
if assistant B is better, and "[[C]]" for a tie.

[User Question]
{question}

[The Start of Assistant A's Answer]
{answer_a}
[The End of Assistant A's Answer]

[The Start of Assistant B's Answer]
{answer_b}
[The End of Assistant B's Answer]
```

# Prompt for Single Answer Grading

```
[System]
Please act as an impartial judge and evaluate the quality of the response provided by an
AI assistant to the user question displayed below. Your evaluation should consider factors
such as the helpfulness, relevance, accuracy, depth, creativity, and level of detail of
the response. Begin your evaluation by providing a short explanation. Be as objective as
possible. After providing your explanation, please rate the response on a scale of 1 to 10
by strictly following this format: "[[rating]]", for example: "Rating: [[5]]".

[Question]
{question}

[The Start of Assistant's Answer]
{answer}
[The End of Assistant's Answer]
```

# Training the judges – Generative Verifiers

- Given a prompt, response and the string *Is this response correct?*, the judges can be trained to predict *Yes/No.*
- For correct solutions $r_+$
  - The log-probability of predicting *yes* is maximized.
- For wrong solutions $r_-$
  - The log-probability of predicting *no* is maximized.
- At inference, the log-probability of the *yes* token is used as the verifier's score.

$$s = \log p_\theta(yes|p, r, I)$$

Where I is the string *Is this response correct?*

# Chain-of-Thought verifiers

- LLM judges benefit if they are prompted to generate a chain-of-thought before generating the final score.
- Can we include chain-of-thoughts while training the LLM judges?
- The inputs to the LLM are:
  - The prompt $p$
  - The response $r$
  - The string "Lets verify step-by-step"
- The LLM is trained to predict
  - A step-by-step verification - This can be human or LLM generated
  - The final Yes/No token after prompting "Is this response correct"
- During inference, the step-by-step verification is generated first.
- The final score is the log-probability of the Yes/No tokens

# Prompt for Chain-of-Thought verifiers

**Problem**: A line is parameterized by $\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 8 \\ -1 \end{pmatrix} + t \begin{pmatrix} 2 \\ 3 \end{pmatrix}$. The equation of the line can be expressed in the form $y = mx + b$. Enter the ordered pair $(m, b)$.

**Solution**: We have $x = 8 + 2t$ and $y = -1 + 3t$. Solving the first equation for $t$, we have $t = \frac{x-8}{2}$. Substituting this into the second equation, we get $y = -1 + 3\left(\frac{x-8}{2}\right) = \frac{3}{2}x - \frac{23}{2}$. Thus, $m = \frac{3}{2}$ and $b = -\frac{23}{2}$. So, the answer is $\boxed{\left(\frac{3}{2}, -\frac{23}{2}\right)}$.

**Discriminative RM** correctness score: 0.827

**GenRM-CoT**: Let's verify step by step. ⋯
**Step 3:**
Solution: Substituting this into the second equation, we get $y = -1 + 3\left(\frac{x-8}{2}\right) = \frac{3}{2}x - \frac{23}{2}$.
Expected Answer: Substituting this into the second equation, we get $y = -1 + 3 * (x - 8)/2 = -1 + (3x - 24)/2 = -1 + (3x/2) - 12 = (3x/2) - 13$.
**Verification:** No
...
Verification: Is the answer correct (Yes/No)? No

**GenRM-CoT** (Majority Voting) score: 0.438

# Using the reward model

# Best-of-N policy

Given

- Base policy or reference policy $\pi_{ref}(y|x)$
  - Often, an instruction tuned LM that serves as the starting point of alignment
- Reward Model $r(x, y)$

Aim – To generate outputs with high reward

**Solution**

- Sample multiple outputs from the policy $\pi_{ref}$
- Score each output using the reward model
- Return the output with the highest reward

**Challenge** – Too expensive during inference

# The reward-maximization objective

Given

- Base policy or reference policy $\pi_{ref}(y|x)$
  - Often, an instruction tuned LM that serves as the starting point of alignment
- Reward Model $r(x, y)$

Aim

- To find a policy $\pi_{\theta^*}(y|x)$
  - That generated outputs with high reward.
  - That stay close to the reference policy.

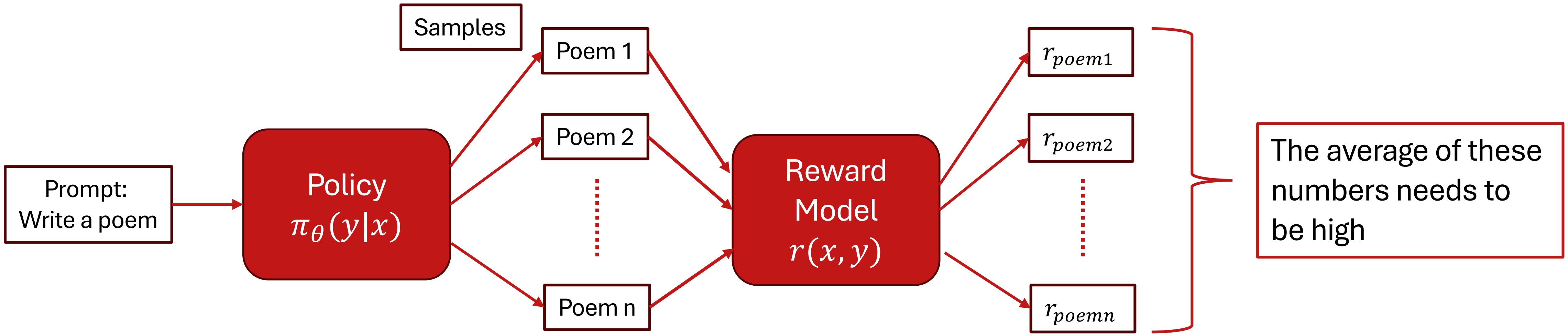
# Why care about closeness to $\pi_{ref}$?

Reward Models are not perfect.

- They have been trained to score only selected natural language outputs.
- The policy can hack the reward model – generate outputs with high reward but meaningless
- An input can have multiple correct outputs (Write a poem?)
  - Reward maximization can collapse the probability to 1 outputs
  - Staying close to $\pi_{ref}$ can preserve diversity.

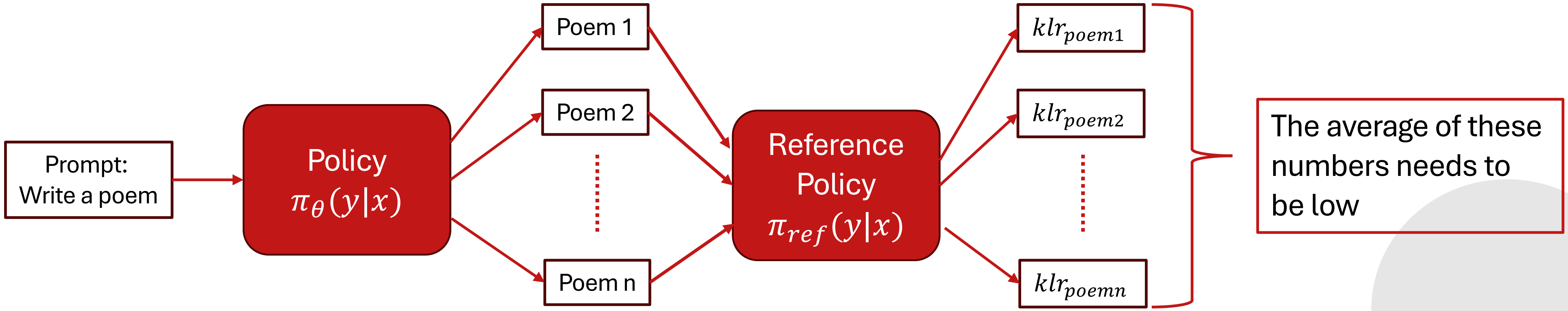
# Formulating the objective – Reward Maximization

- What does it mean for a policy to have high reward?

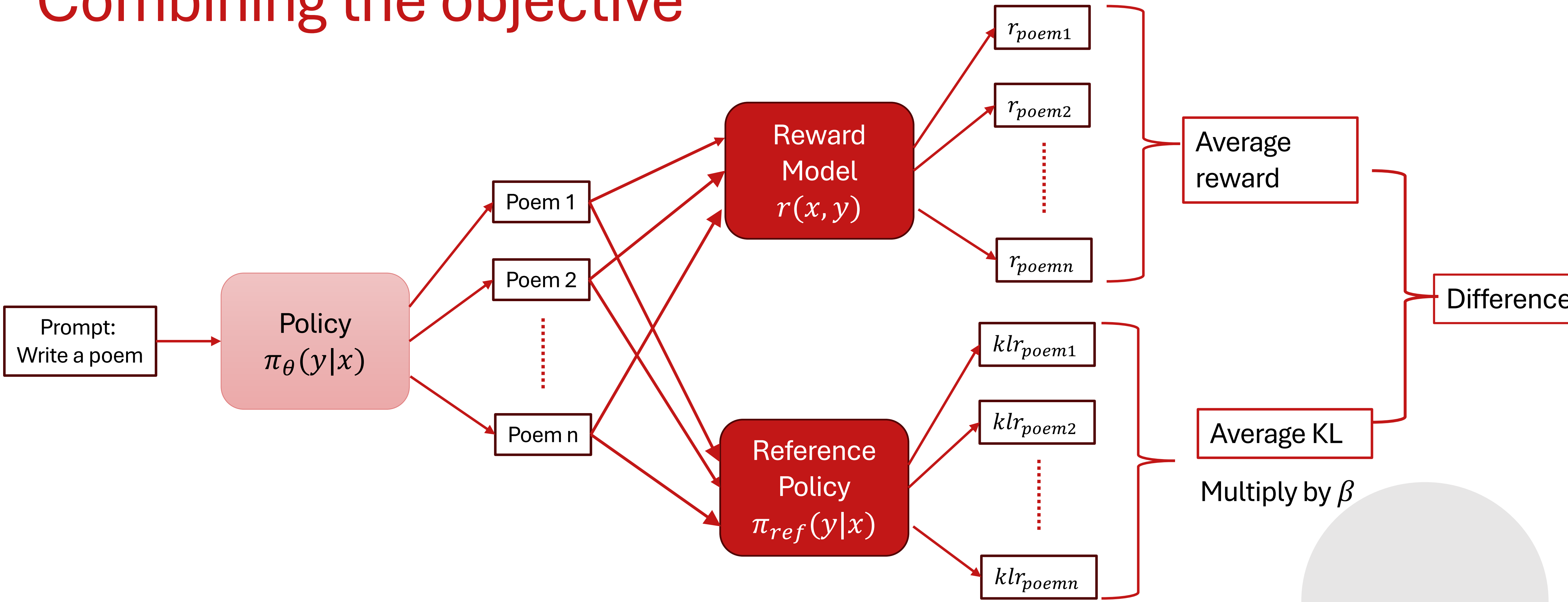


# Formulating the objective – closeness to $\pi_{ref}$

- How do we capture closeness to $\pi_{ref}$?



# Combining the objective



# Takeaways & what next?

- Alignment methods can help prevent undesirable outputs from getting generated.
- The RLHF/RLAIF alignment method uses
  - LLM as a policy
  - LLM as a reward model
  - Reward maximization as the objective
- The reward model for alignment can be trained either using human of AI-generated preferences.
- Staying close to the base/reference policy is desirable to prevent reward hacking.
- Next: How to train the policy?