Alignment of Language Models – Reward Maximization

Large Language Models: Introduction and Recent Advances

ELL881 · AIL821

Gaurav Pandey
Research Scientist, IBM Research

How to make ChatGPT ?

- Pre-Training
  - This is the point where most of the reasoning power is infused in the model.
  - Data – Billions of tokens of unstructured text from the internet
- Instruction Tuning
  - Trains models to follow natural language instructions
  - Data – Several thousand (Task/Instruction, Output) examples
- Reinforcement Learning/Alignment with Human Feedback
  - Show the output(s) generated by models to humans/reward model
  - Collect feedback in the form of preferences.
  - Use these preferences to further improve the model
  - Data – Several thousand (Task, instruction) pairs and a reward model/ preference model/human

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) } $\mathcal{L}_1$ $\mathcal{L}_2$
- Contrastive Learning – DPO & its variants $\mathcal{L}_3$
- Distribution Matching – DPG, BRAIn ×

Online/Offline

- Online: Policy Gradient, PPO → Outputs Generated from the model as it trains
- Offline: DPO → Generated offline
- Mixed: Iterative DPO, BRAIn

Reinforcement Learning

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

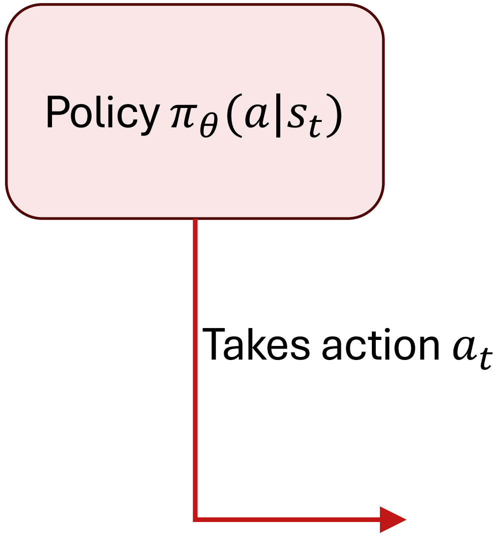
- π_θ can be a large language model
- s_t can be the tokens of the input prompt/instruction along with previously generated output tokens
- a can be any output token generated by the LLM
- The policy captures the distribution over the output tokens given the prompt/instruction

s_t

Prompt tokens | Previously generated output tokens

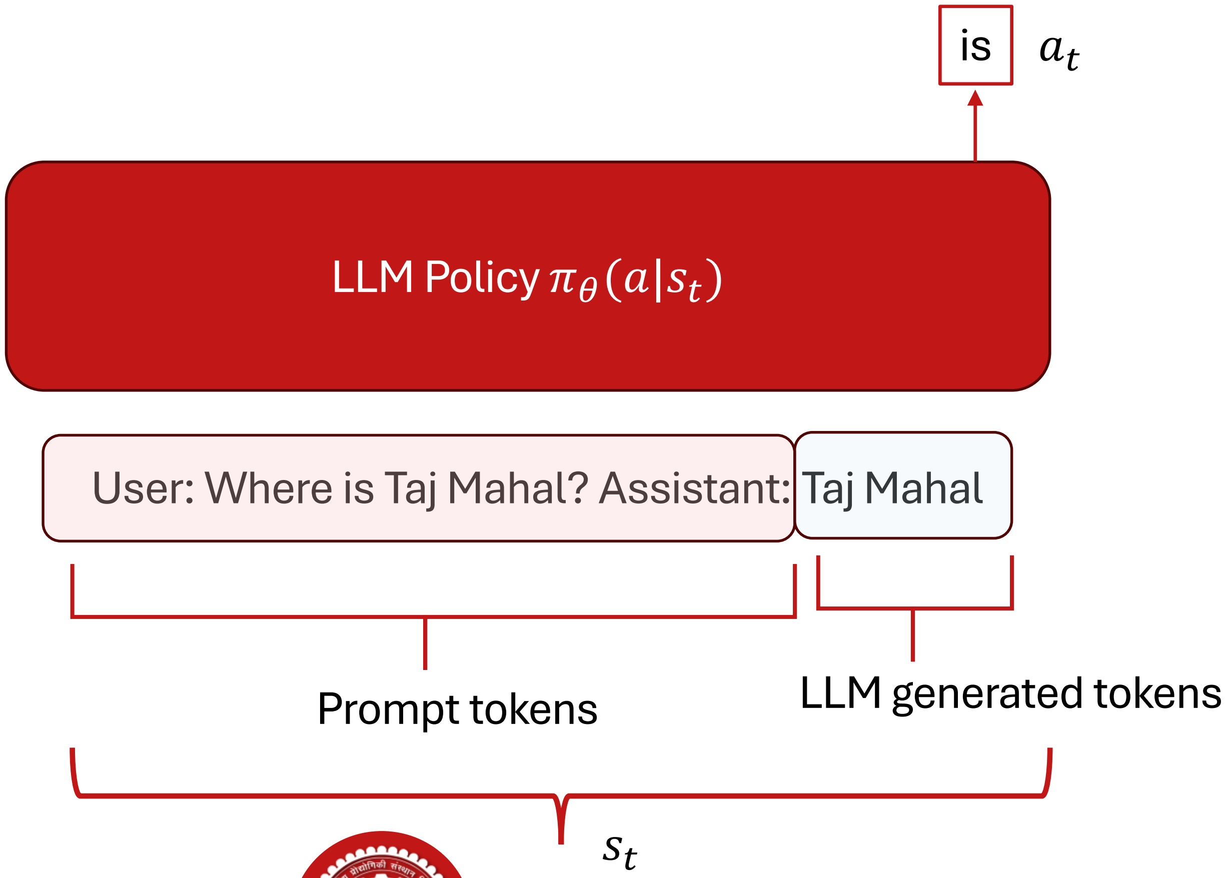
state at time step t

Reinforcement Learning

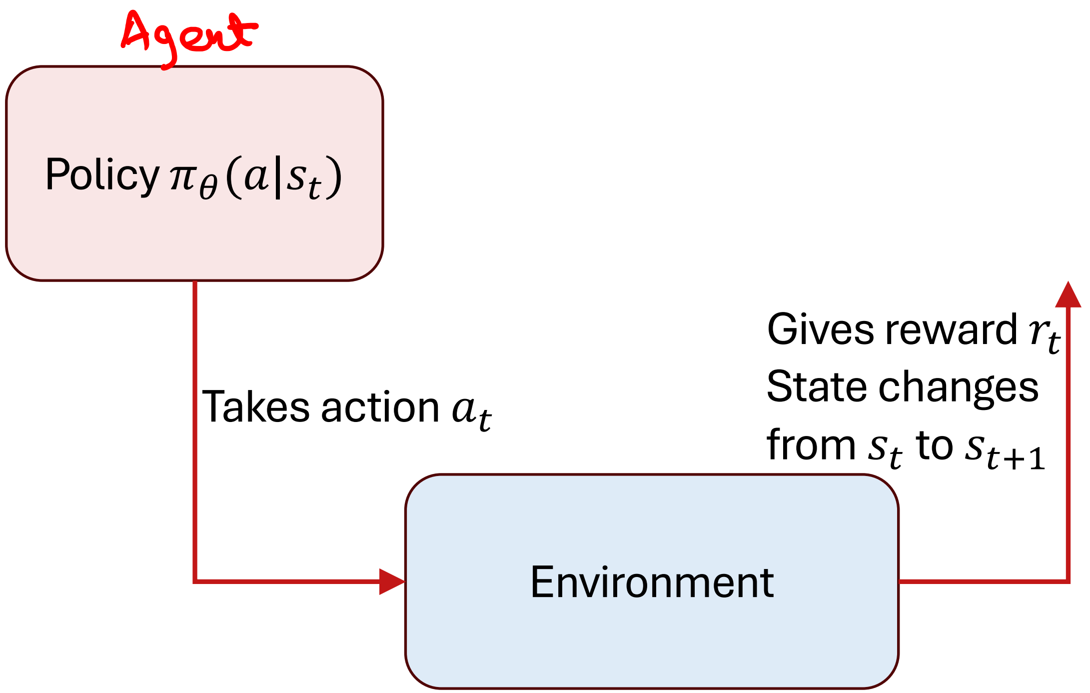


The generation of a token by an LLM is equivalent to taking an action

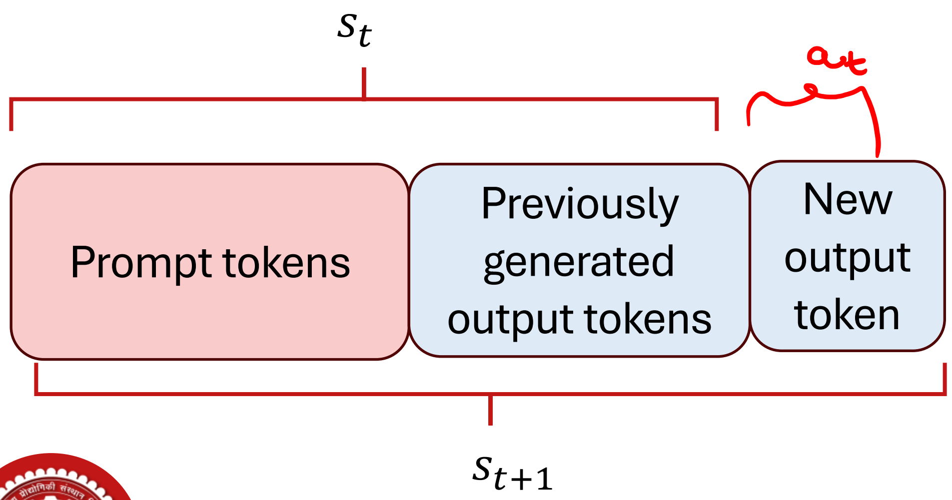
- Each token generated by the LLM can be thought of as an action



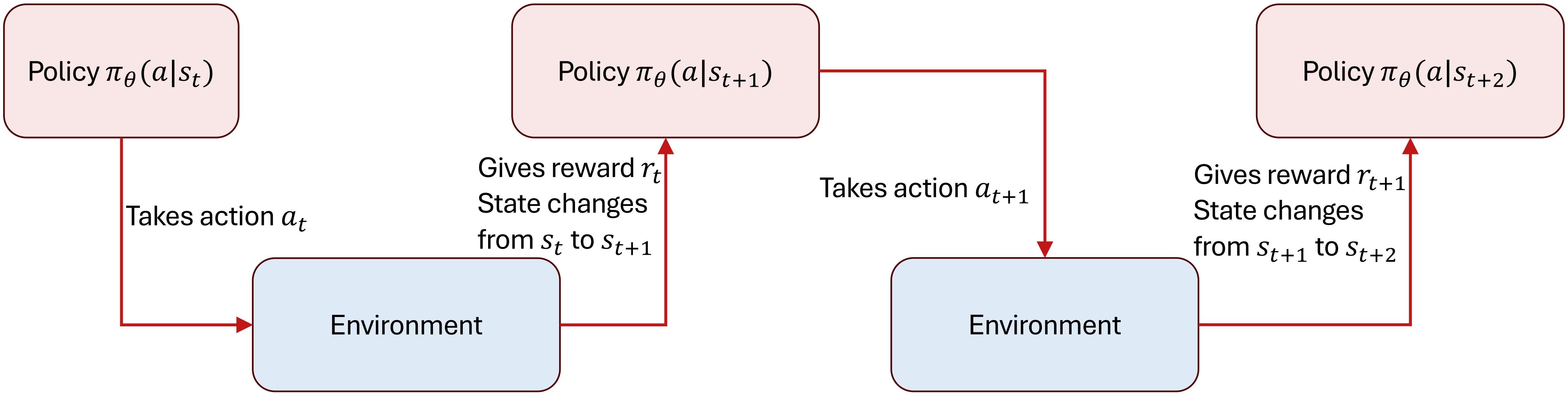
Reinforcement Learning



- 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

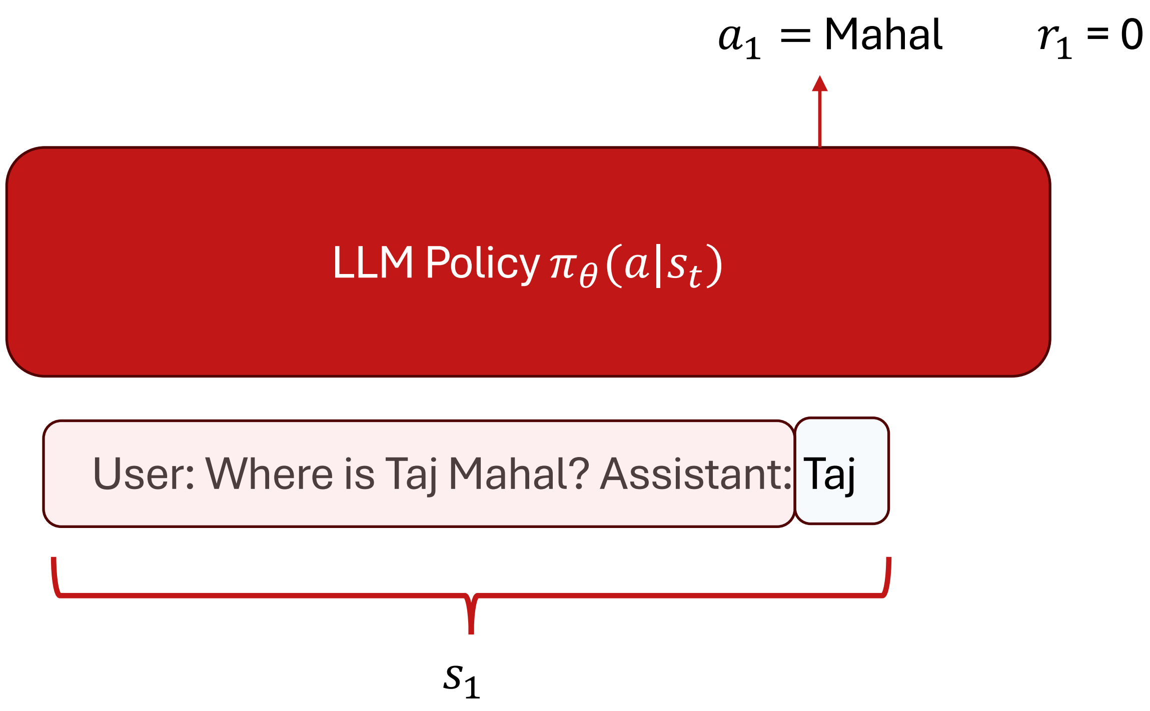
a_0 =Taj $r_0 = 0$ (no <|endoftext|> is generated)

LLM Policy $\pi_\theta(a|s_0)$

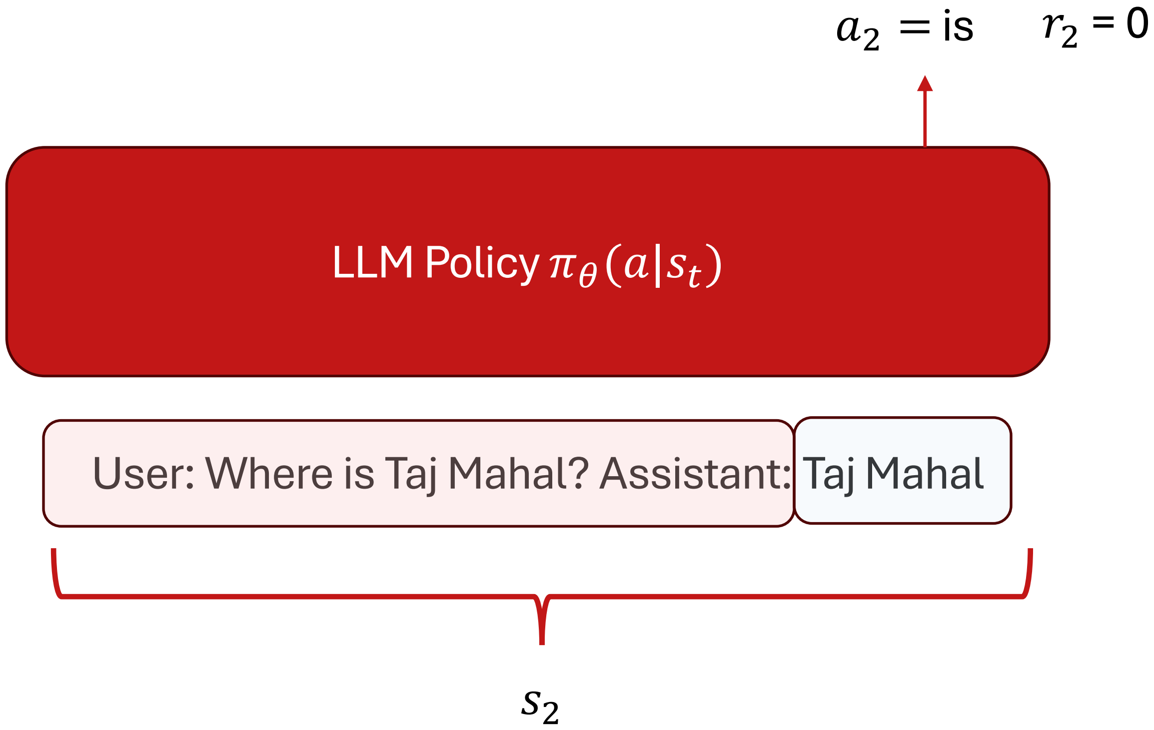
User: Where is Taj Mahal? Assistant:

s_0 = prompt tokens

LLM as a policy



LLM as a policy



LLM as a policy

<|endoftext|>

$a_T = \cancel{1}$ $r_T = +1$

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

User: Where is Taj Mahal? Assistant: Taj Mahal is in Agra

s_T

Who/What is the reward model?

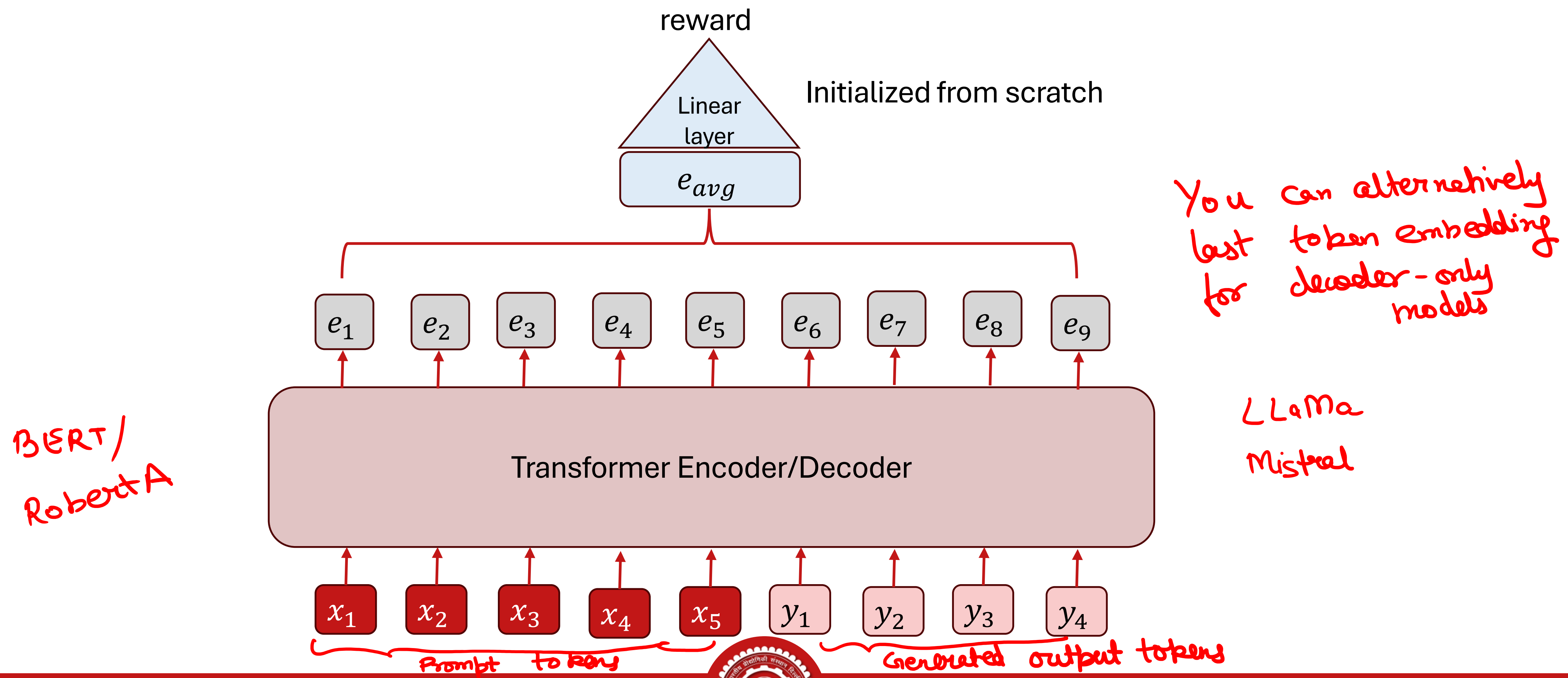
- We can ask humans to give thumbs up/down to generated outputs and treat them as rewards.
- Challenges:
  - Human feedback is costly & slow.
  - Traditional RLHF (as we will see) requires constant feedback after every (few) updates to the model.
- Solution:
  - Lets train another LLM to behave like the reward model.

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

Architecture of the reward model



Training the reward model

The Bradley-Terry (BT) preference model - I

- Probability 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 is given by

$$\mathbb{P}(y_i > y_j) = \frac{p_i}{p_i + p_j}$$

- If $p_i > 0$:

$$\mathbb{P}(y_i \succ y_j) = \frac{\exp(r_i)}{\exp(r_i) + \exp(r_j)}$$

where $r_i = \log p_i$

The Bradley-Terry preference model - II

- Given input x and any 2 outputs y_1 and y_2

$$\mathbb{P}(y_1 \succ y_2 | x) = \frac{\exp(r^*(x, y_1))}{\exp(r^*(x, y_1)) + \exp(r^*(x, y_2))}$$

r^* can be any arbitrary function

- Parameterization

$$p_\theta(y_1 > y_2 | x) = \frac{\exp(r_\theta(x, y_1))}{\exp(r_\theta(x, y_1)) + \exp(r_\theta(x, y_2))}$$

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

$$\mathcal{L}(\theta, (x, y_+, y_-)) = \log p_\theta(y_+ > y_- \mid x)$$

$$= \log \frac{\exp(r_\theta(x, y_+))}{\exp(r_\theta(x, y_+)) + \exp(r_\theta(x, y_-))}$$

$$= \log \frac{\exp(r_\theta(x, y_+) - r_\theta(x, y_-))}{1 + \exp(r_\theta(x, y_+) - r_\theta(x, y_-))}$$

$$= \log \sigma(r_\theta(x, y_+) - r_\theta(x, y_-))$$

Maximize it over all preference pairs in training data

An intuitive view

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

montonically increasing

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

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).

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

$\underbrace{\text{Zephyr -7b}}_{\text{aligned model}}$ on top of Mistral-7b

Constitutional AI for collecting 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

The Reward Maximization Objective

The 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.

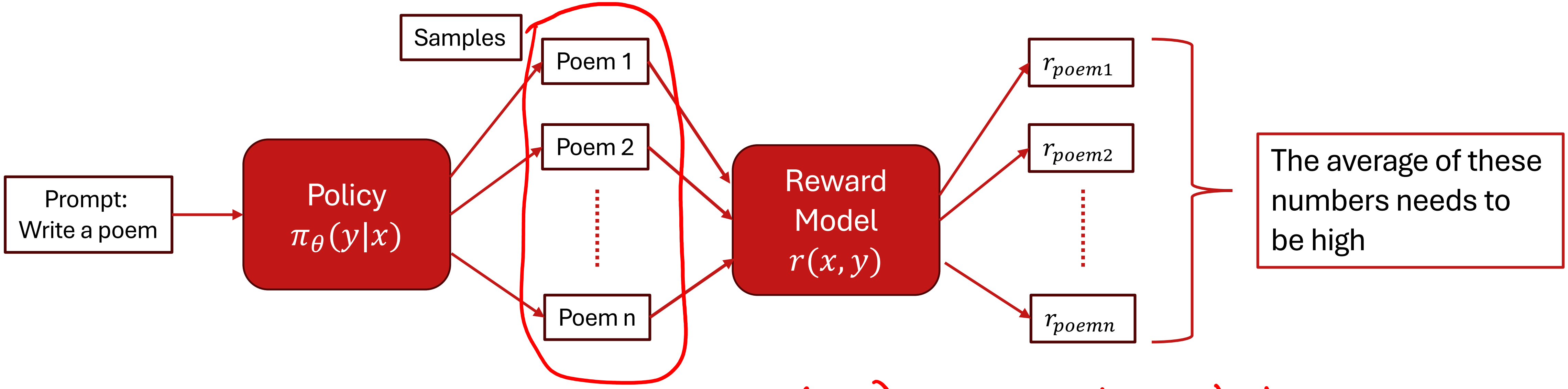
Why care about closeness to π_{ref}?

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 π_{ref} can preserve diversity.

Formulating the objective – Reward Maximization

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



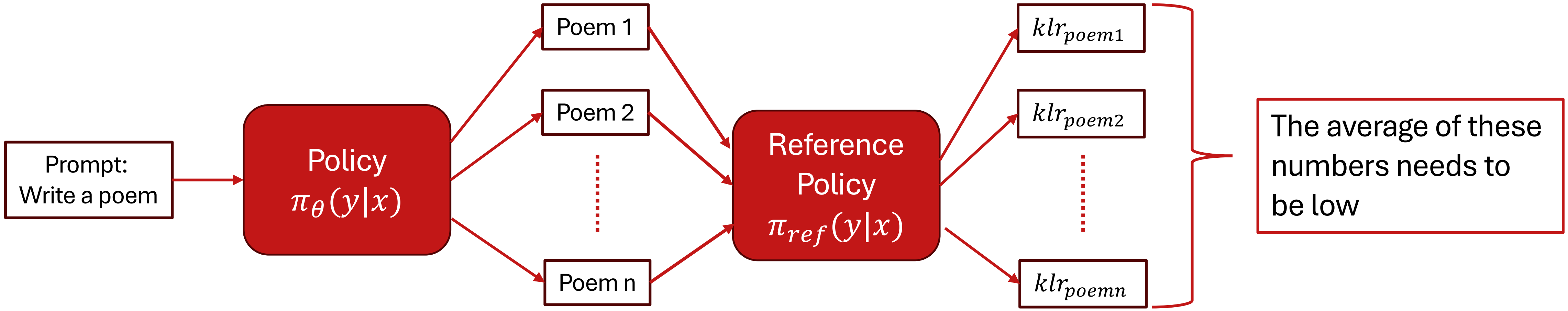
$\mathbb{E}_{y \sim \pi_\theta(y|x)} \, r(x,y)$ should be high

Formulating the objective – closeness to π_{ref}

- How do we capture closeness to π_{ref}?

Policies are prob distribution

$$KL\left(\pi_\theta(y|x) \,||\, \pi_{ref}(y|x)\right) = \mathbb{E}_{y\sim\pi_\theta(y|x)}\left[\overbrace{\log \frac{\pi_\theta(y|x)}{\pi_{ref}(y|x)}}^{klr}\right]$$



Combining the objective

- Maximize the reward

$$\mathbb{E}_{\pi_\theta(y|x)}\, r(x,y) \quad \uparrow$$

- Minimize the KL divergence

$$\mathbb{E}_{\pi_\theta(y|x)}\left[\log \frac{\pi_\theta(y|x)}{\pi_{ref}(y|x)}\right] \times \lambda \quad \downarrow$$

- Add a scaling factor

$$\mathbb{E}_{\pi_\theta(y|x)}\left[r(x,y) - \lambda \log \frac{\pi_\theta(y|x)}{\pi_{ref}(y|x)}\right]$$

Takeaways & what next?

- Alignment methods can help prevent undesirable outputs from getting generated.
- The RLHF 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?