Alignment from DPO to Controlled Decoding

CMSC 848N · Fall 2026

Training-time preference optimization and test-time distribution control

Furong Huang · University of Maryland

AI Is Everywhere

Search & Chat

ChatGPT, Bard, perplexity, AI

Entertainment & Social

character.ai, TikTok, Instagram, airchat

Design

Adobe, stability.ai, Midjourney, Mubert

Productivity

Notion, Duet AI in Google Workspace, Microsoft 365 Copilot, tome

Software Engineering

GitHub Copilot, replit, CODEGEN

Robotics

NVIDIA, TESLA, Boston Dynamics, FIGURE

Education

duolingo, MAGIC SCHOOL, edu GPT

Finance

Bloomberg GPT, Kasisto, J.P.Morgan, Capital One

Autonomous Vehicles

TESLA, WAYMO, pony.ai, cruise

Legal

Harvey., Ironclad, EvenUp, casetext, PINCITES

Healthcare

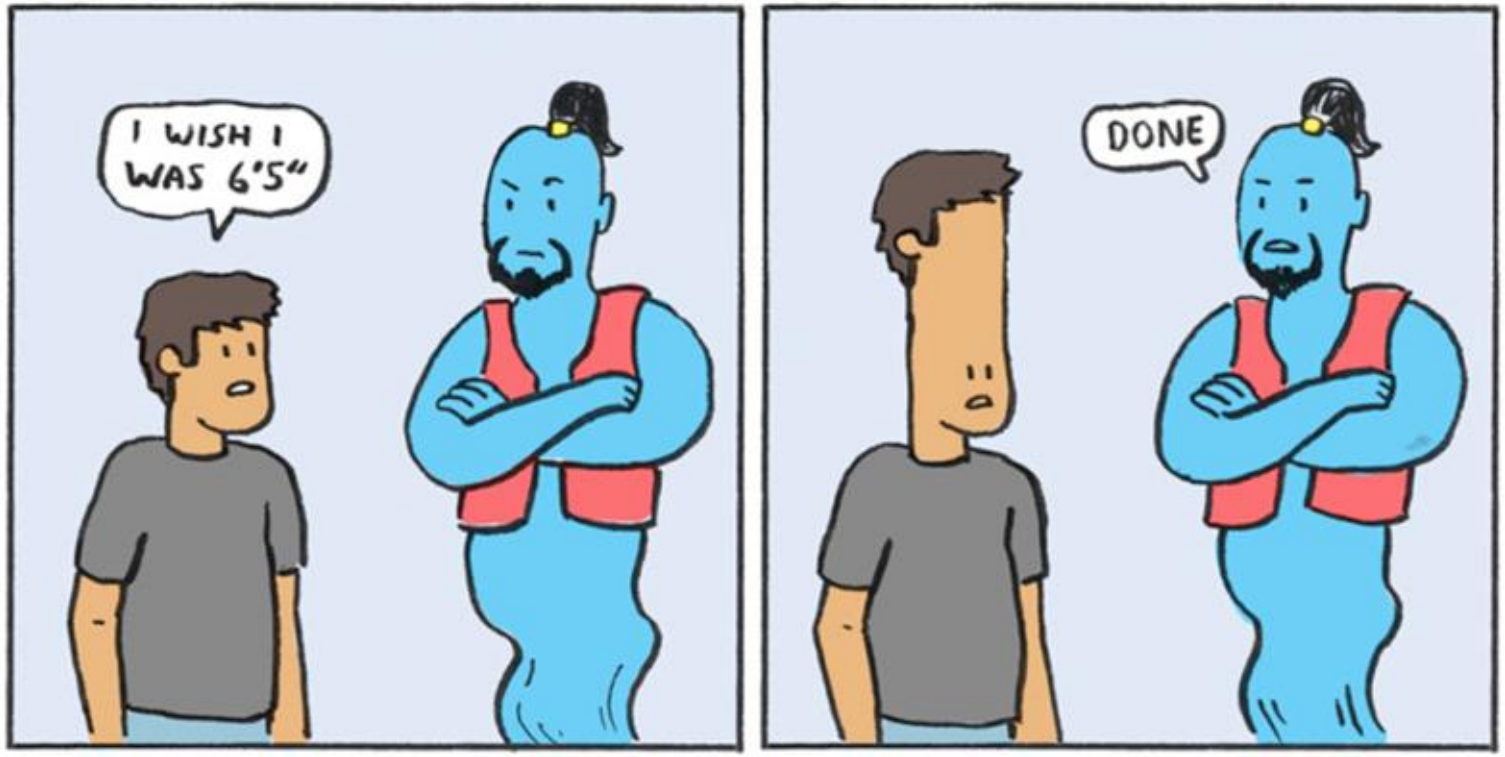
Ambience, MEMORA HEALTH, DeepScribe, Hippocratic AI

And more

34% by machines

World Economic Forum, 2023

The Blame: Misalignment

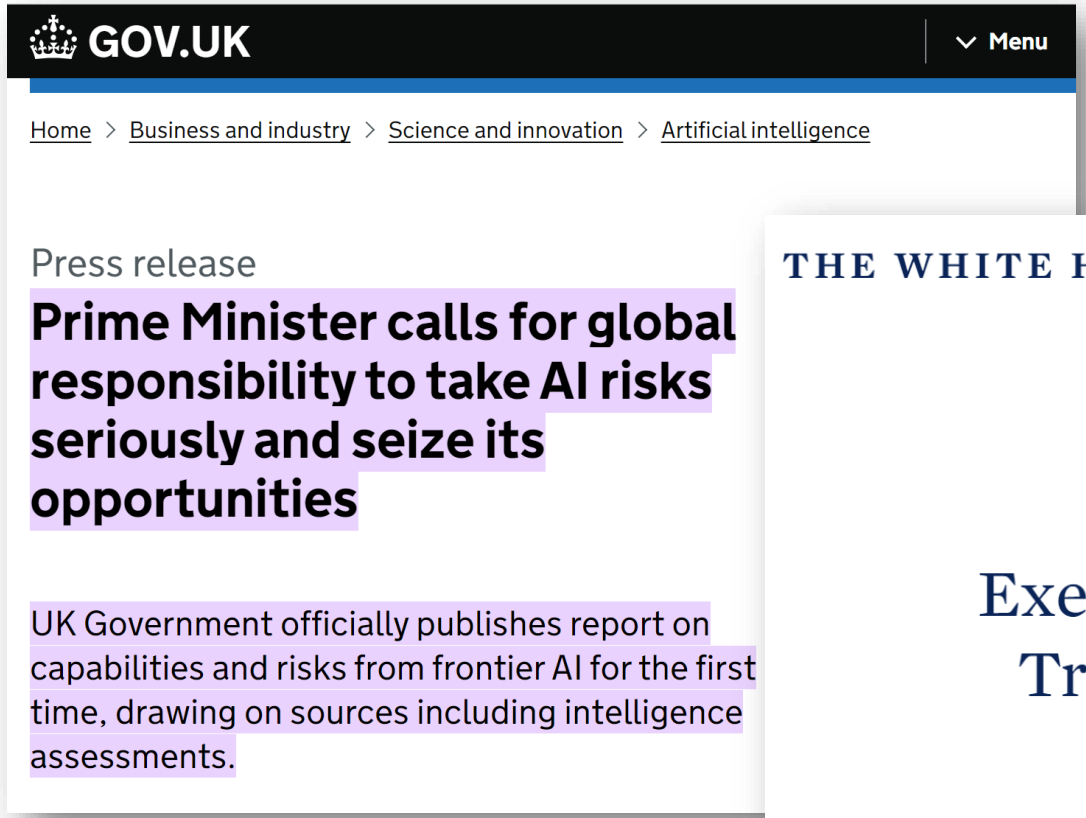


https://sites.google.com/view/cos598aisafety/

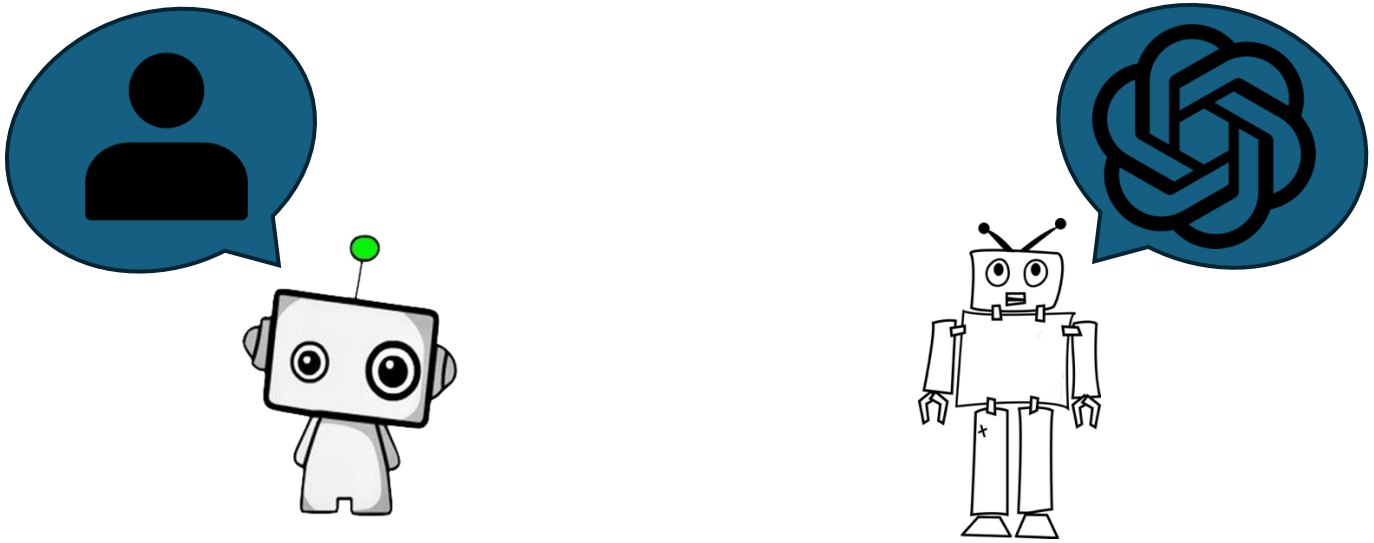
"genie in a lamp" problem

Steer AI systems toward
humans' intended goals, preferences, or ethical principles

Aligning AI is Crucial

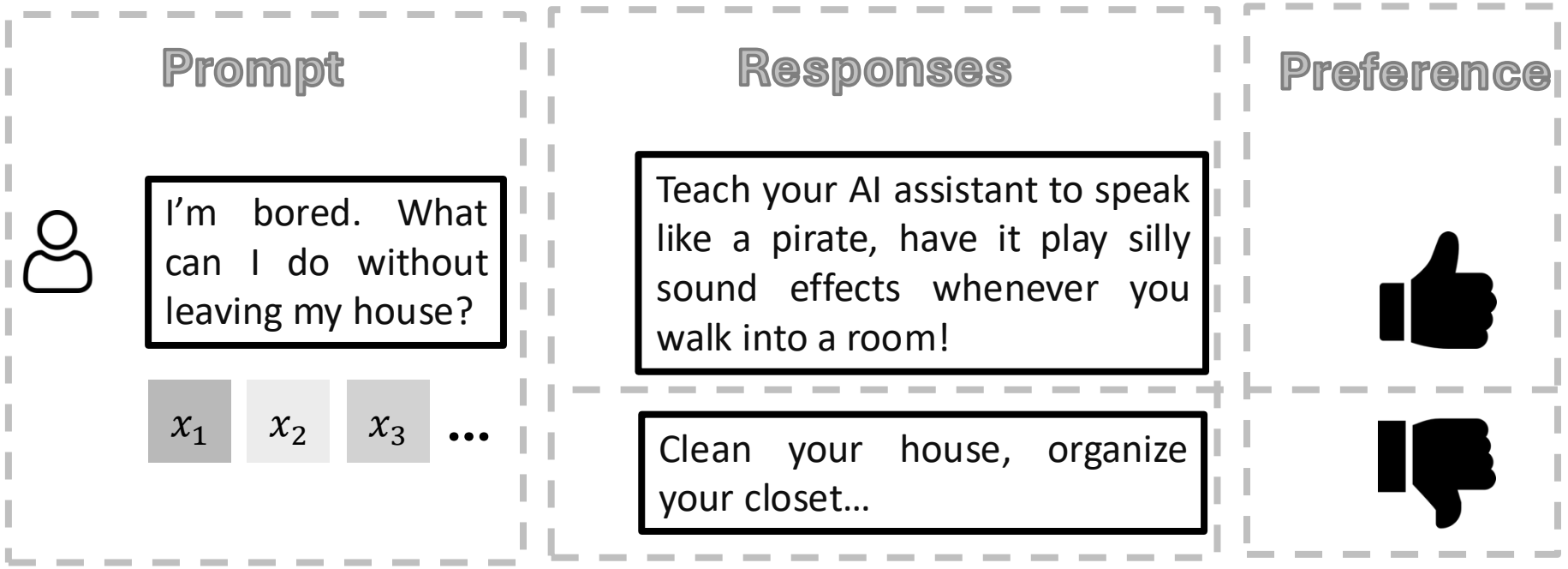


AI Alignment — Scenario 1

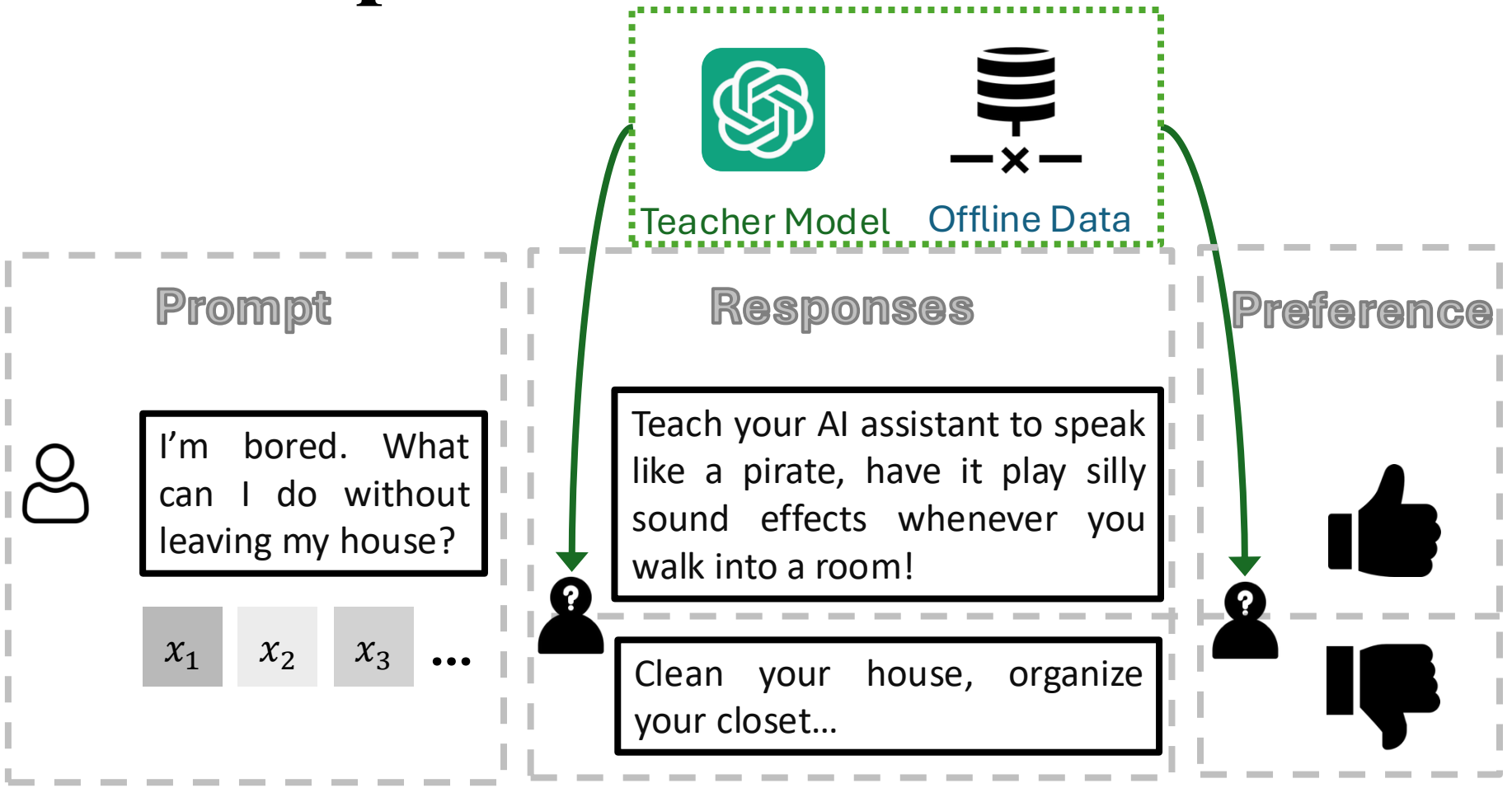


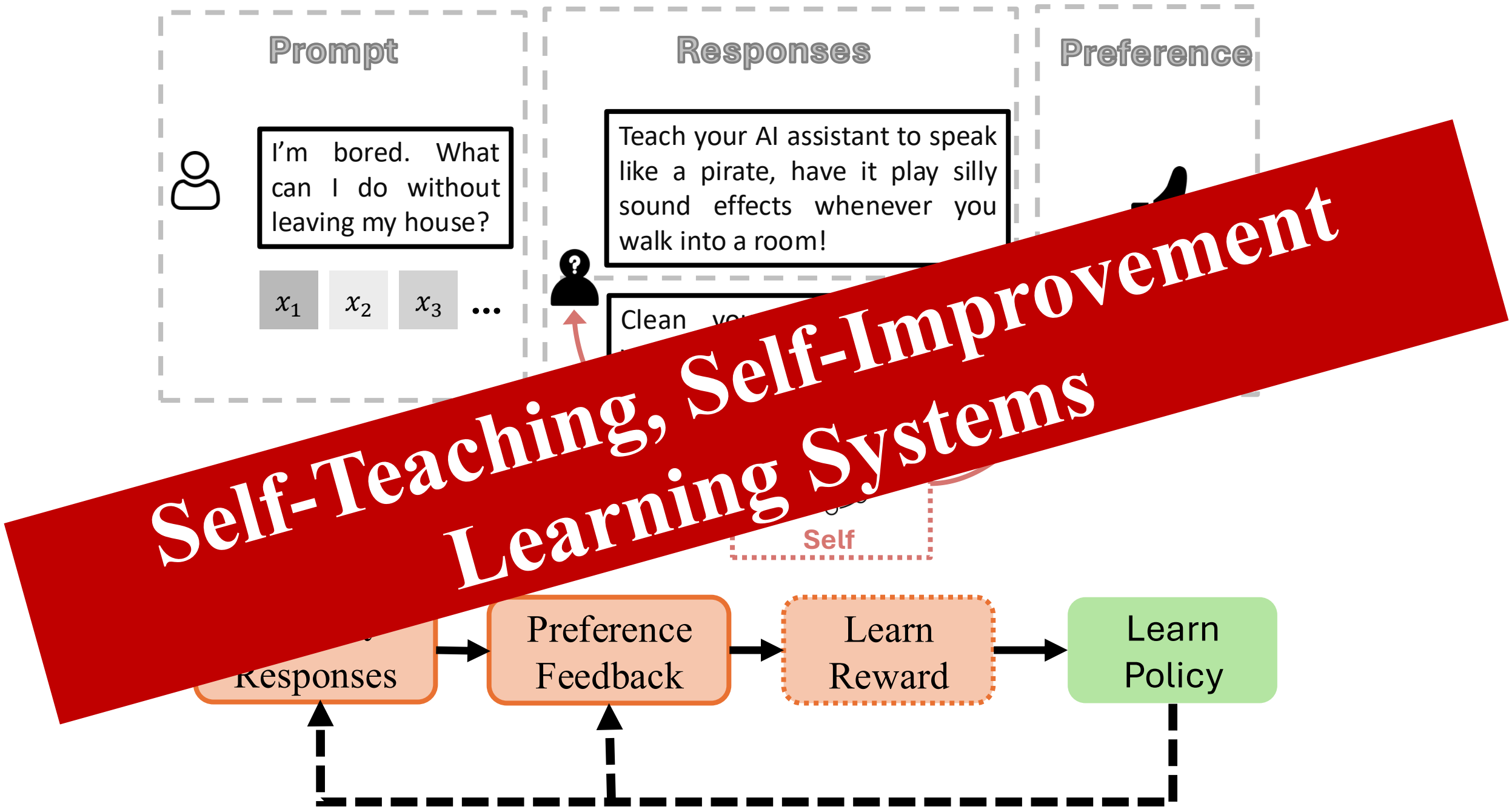
Supervisor Student

Reinforcement Learning from Human Feedback



RLHF Pipeline: External Guidance




Prompt
Responses
Preference
I'm bored. What can I do without leaving my house?
x_1 x_2 x_3 ...
Teach your AI assistant to speak like a pirate, have it play silly sound effects whenever you walk into a room!
Clean
Self
Responses
Preference Feedback
Learn Reward
Learn Policy
Self-Teaching, Self-Improvement Learning Systems

Direct preference optimization

From preference comparisons to a policy objective

The key step expresses reward through the optimal policy and eliminates the prompt-dependent normalizer inside each preference pair.

Mathematical backbone: 2025 W2-L1 DPO sequence

Rafailov et al. (2023). Derivation structure adapted from the 2025 DPO lecture.

Preference comparisons define reward differences

A pair identifies which completion should receive higher reward for the same prompt

$$p^*(y_w \succ y_\ell \mid x) = \frac{e^{r^*(x,y_w)}}{e^{r^*(x,y_w)} + e^{r^*(x,y_\ell)}}$$
$$= \sigma(r^*(x, y_w) - r^*(x, y_\ell))$$

Only reward differences are identifiable. Adding the same prompt-dependent constant to both rewards leaves the preference probability unchanged.

Rafailov et al. (2023). Derivation structure adapted from the 2025 DPO lecture.

Reward learning is pairwise logistic regression

The preferred completion should score above the dispreferred completion

$$\mathcal{L}_R(r;\mathcal{D}) = -\mathbb{E}_{(x,y_w,y_\ell)\sim\mathcal{D}}\left[\log \sigma\left(r(x,y_w) - r(x,y_\ell)\right)\right]$$

The loss depends on one scalar margin: r(x, y_w) - r(x, y_ℓ).

DPO will replace this reward margin with a policy log-ratio margin.

Rafailov et al. (2023). Derivation structure adapted from the 2025 DPO lecture.

KL-regularized RLHF

Maximize expected reward while limiting drift from the reference policy

$$\max_{\pi} \; \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi(\cdot|x)}[r(x,y)] - \beta\, D_{\mathrm{KL}}(\pi(\cdot \mid x) \,\|\, \pi_{\mathrm{ref}}(\cdot \mid x))$$

Reward term
raises probability on preferred responses

KL term
keeps the policy near πref

β > 0 controls the strength of the constraint.

Rafailov et al. (2023). Derivation structure adapted from the 2025 DPO lecture.

The first-order condition determines the optimal policy

Fix one prompt x and impose probability normalization with λ(x)

$$\mathcal{J}(\pi) = \sum_y \pi(y \mid x) r(x, y) - \beta \sum_y \pi(y \mid x) \log \frac{\pi(y \mid x)}{\pi_{\text{ref}}(y \mid x)} + \lambda(x) \left(\sum_y \pi(y \mid x) - 1 \right)$$

$$0 = r(x, y) - \beta \left(\log \frac{\pi^*(y \mid x)}{\pi_{\text{ref}}(y \mid x)} + 1 \right) + \lambda(x)$$

Solving this equation for π*(y | x) gives an exponential tilt of πref.

Rafailov et al. (2023). Derivation structure adapted from the 2025 DPO lecture.

The optimal policy exponentially tilts the reference

$$\pi^*(y \mid x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y \mid x) \exp\left(\frac{r(x, y)}{\beta}\right)$$

The reference policy supplies the base density. Reward reweights that density before normalization.

Higher β produces a milder change in distribution.

Rafailov et al. (2023). Derivation structure adapted from the 2025 DPO lecture.

The partition function normalizes the policy

$$Z(x) = \sum_{y \in \mathcal{Y}(x)} \pi_{\text{ref}}(y \mid x) \exp\left(\frac{r(x, y)}{\beta}\right)$$

Z(x) depends on the prompt. It does not depend on which completion in a preference pair we inspect.

Rafailov et al. (2023). Derivation structure adapted from the 2025 DPO lecture.

Exact normalization requires summing over all completions

$$Z(x) = \sum_{y \in \mathcal{Y}(x)} \pi_{\text{ref}}(y \mid x) \exp\left(\frac{r(x, y)}{\beta}\right)$$

The completion space grows combinatorially with sequence length. Direct evaluation of Z(x) is therefore unavailable for realistic language models.

DPO avoids evaluating Z(x).

Rafailov et al. (2023). Derivation structure adapted from the 2025 DPO lecture.

The reward can be written as a policy log ratio

$$r_{\pi_\theta}(x, y) = \beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)} + \beta \log Z(x)$$

The policy log ratio supplies the completion-dependent reward. β log Z(x) is a prompt-dependent offset.

Rafailov et al. (2023). Derivation structure adapted from the 2025 DPO lecture.

Policy ratios determine reward direction

$$r_{\pi_\theta}(x, y) - \beta \log Z(x) = \beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)}$$

πθ(y | x) > πref(y | x)
positive centered reward

πθ(y | x) < πref(y | x)
negative centered reward

Rafailov et al. (2023). Derivation structure adapted from the 2025 DPO lecture.

Two identities convert reward learning into policy learning

Preference likelihood

$$\mathcal{L}_R = -\mathbb{E}\left[\log \sigma\left(r_w - r_\ell\right)\right]$$

Reward-policy identity

$$r_{\pi_\theta}(x, y) = \beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)} + \beta \log Z(x)$$

Substitution produces a policy loss that uses preference pairs directly.

Rafailov et al. (2023). Derivation structure adapted from the 2025 DPO lecture.

Substitute the policy ratio into the reward margin

$$r_{\pi_\theta}(x, y_w) - r_{\pi_\theta}(x, y_\ell) = \beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_\ell \mid x)}{\pi_{\text{ref}}(y_\ell \mid x)} + \beta \log Z(x) - \beta \log Z(x)$$

Both completions share the same prompt x, so they share the same normalizer.

Rafailov et al. (2023). Derivation structure adapted from the 2025 DPO lecture.

The prompt-dependent normalizer cancels

$$r_{\pi_\theta}(x, y_w) - r_{\pi_\theta}(x, y_\ell) = \beta \left[\log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \log \frac{\pi_\theta(y_\ell \mid x)}{\pi_{\text{ref}}(y_\ell \mid x)} \right]$$

DPO needs only four sequence log probabilities: policy and reference scores for the preferred and dispreferred completions.

Rafailov et al. (2023). Derivation structure adapted from the 2025 DPO lecture.

DPO is a preference classification loss

$$\mathcal{L}_{\mathrm{DPO}}(\pi_\theta; \pi_{\mathrm{ref}}) = -\mathbb{E}_{(x, y_w, y_\ell)\sim\mathcal{D}}\left[\log \sigma\left(\beta\left[\log \frac{\pi_\theta(y_w \mid x)}{\pi_{\mathrm{ref}}(y_w \mid x)} - \log \frac{\pi_\theta(y_\ell \mid x)}{\pi_{\mathrm{ref}}(y_\ell \mid x)}\right]\right)\right]$$

The model learns the relative increase assigned to the preferred completion, measured against the reference policy.

Rafailov et al. (2023). Derivation structure adapted from the 2025 DPO lecture.

DPO emphasizes pairs that the policy still misorders

$$\nabla_\theta \mathcal{L}_{\mathrm{DPO}} = -\beta\, \mathbb{E}_{\mathcal{D}}[\sigma(-\beta h_\theta)\,(\nabla_\theta \log \pi_\theta(y_w \mid x) - \nabla_\theta \log \pi_\theta(y_\ell \mid x))]$$

$$h_\theta = \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\mathrm{ref}}(y_w \mid x)} - \log \frac{\pi_\theta(y_\ell \mid x)}{\pi_{\mathrm{ref}}(y_\ell \mid x)}$$

σ(-βhθ) is largest when the preferred completion lacks sufficient relative probability.

Rafailov et al. (2023). Derivation structure adapted from the 2025 DPO lecture.

DPO aligns the policy during training

1

Data

Offline preference triples (x, y_w, y_ℓ)

2

Optimization

Update θ using the DPO loss and a fixed reference policy

3

Deployment

Sample directly from the trained policy πθ

A new target preference distribution generally requires another parameter update.

Rafailov et al. (2023). Derivation structure adapted from the 2025 DPO lecture.

Controlled decoding performs the intervention at inference time

Keep the base model π_0 frozen.

At each prefix s_t = (x, y_<t), reweight the next-token distribution using a learned prefix value.

The target is the solution of a KL-regularized reward objective, sampled autoregressively without fine-tuning π_0.

A terminal reward induces a soft value for every prefix

Let $R(x,y_1{:}T)$ score a complete response. Define

$$V^*(s\_t) = \beta \log E\_\{y\_t{:}T \sim \pi_0(\cdot|s\_t)\} [\exp(R(x,y_1{:}T)/\beta)].$$

At a terminal sequence $y_1{:}T$, $V^*(x,y_1{:}T) = R(x,y_1{:}T)$.

The value aggregates all base-model continuations from the prefix.

Optimal controlled decoding is a soft policy-improvement step

For next token a after prefix s:

$$\pi^*(a|s) = \pi_0(a|s)\,\exp((V^*(sa) - V^*(s))/\beta).$$

Equivalently,

$$\pi^*(a|s) \propto \pi_0(a|s)\,\exp(V^*(sa)/\beta).$$

The base model supplies fluency; the prefix value supplies control.

The soft Bellman recursion propagates terminal preferences

For a nonterminal prefix s:

$$V^*(s) = \beta \log \sum_a \pi_0(a|s) \exp(V^*(sa)/\beta).$$

For a terminal sequence $y_1{:}T$:

$$V^*(x, y_1{:}T) = R(x, y_1{:}T).$$

A learned prefix scorer approximates this value and makes tokenwise control practical.

A future discriminator can supply tokenwise control scores

For desired attribute c, Bayes' rule gives

$$p(a \mid s,c) \propto p_0(a|s) \cdot p(c \mid sa).$$

FUDGE learns p(c|prefix) and adds log p(c|sa) to the base-model logits before renormalization.

The score estimates whether the completed future will satisfy c.

A learned Q or prefix-value model generalizes classifier guidance

With a control score $Q\phi(s,a)$:

$$\pi\_Q(a|s) \propto \pi_0(a|s)\ \exp(Q\varphi(s,a)/\beta).$$

$Q\phi$ may approximate expected terminal reward, a soft value increment, or a combination of several objectives.

Inference changes logits while π_0 remains frozen.

Blockwise controlled decoding trades search cost against control granularity

For a block b = y_t:t+k−1:

$$\pi^*(b|s) \propto \pi_0(b|s)\,\exp((V^*(sb) - V^*(s))/\beta).$$

k = 1 recovers tokenwise control.

Larger k scores longer continuations and approaches best-of-K-style selection.

The block length sets the compute–control tradeoff.

Training-time and test-time alignment share one exponential-tilting structure

KL-regularized target distribution:

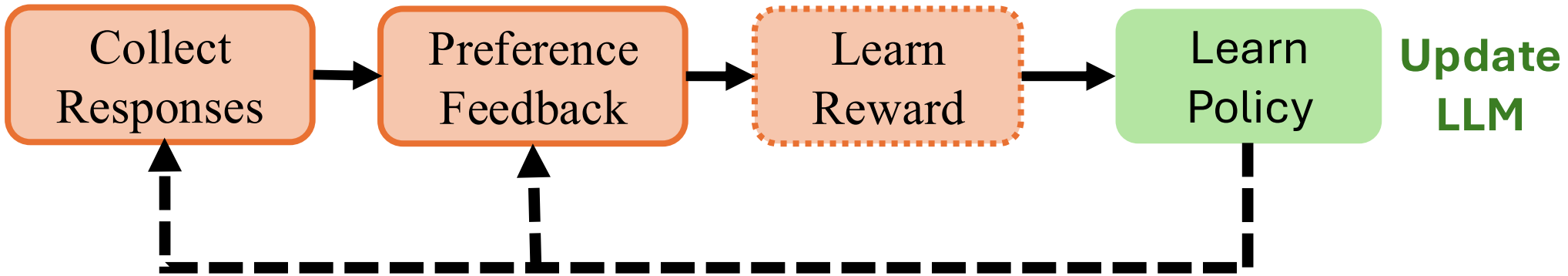
$$\pi_{target} / \pi_{ref} \propto \exp(control / \beta).$$

DPO identifies πθ from pairwise preference odds and stores the intervention in model parameters.

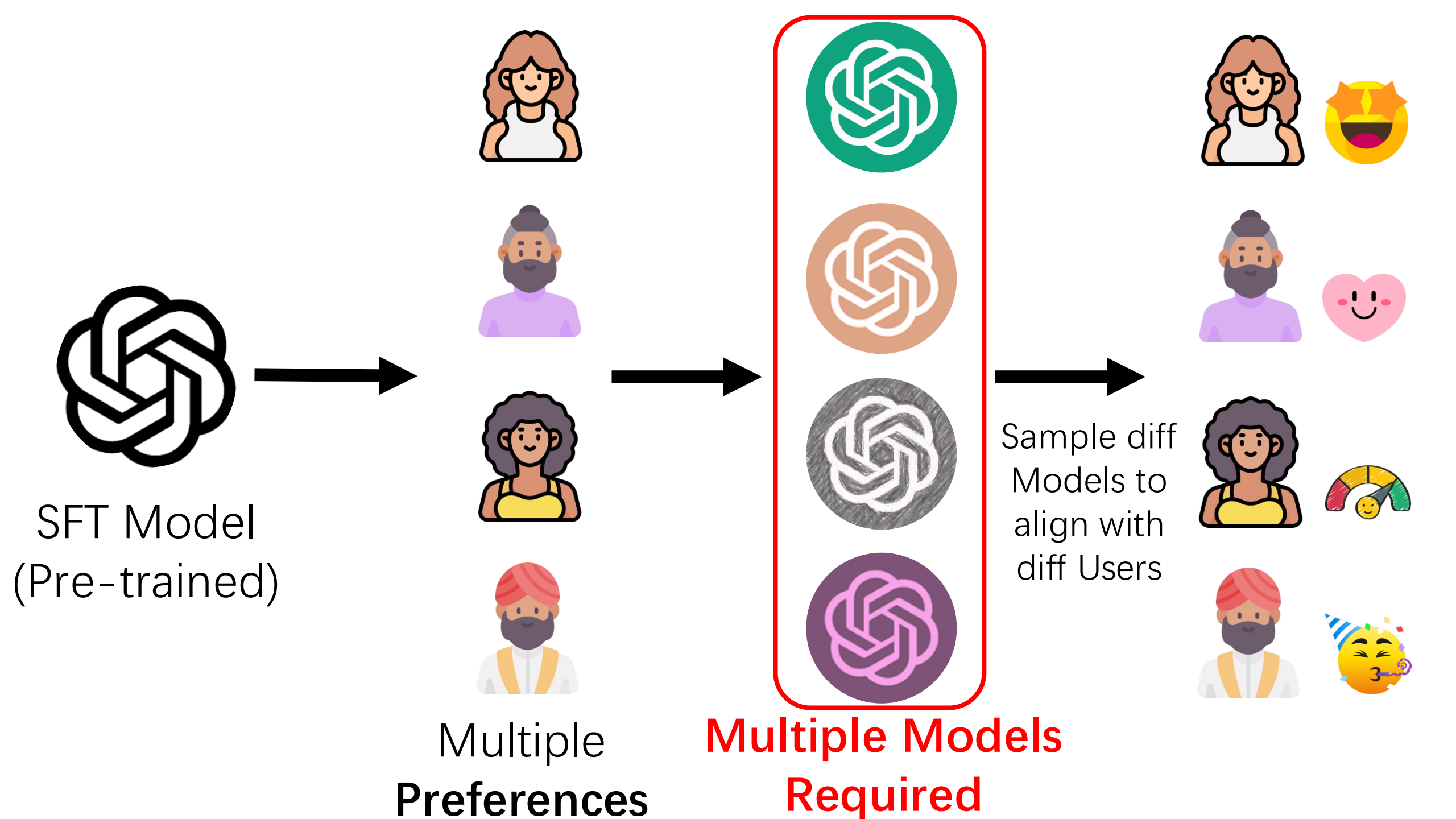
Controlled decoding estimates prefix values and applies the intervention to logits during generation.

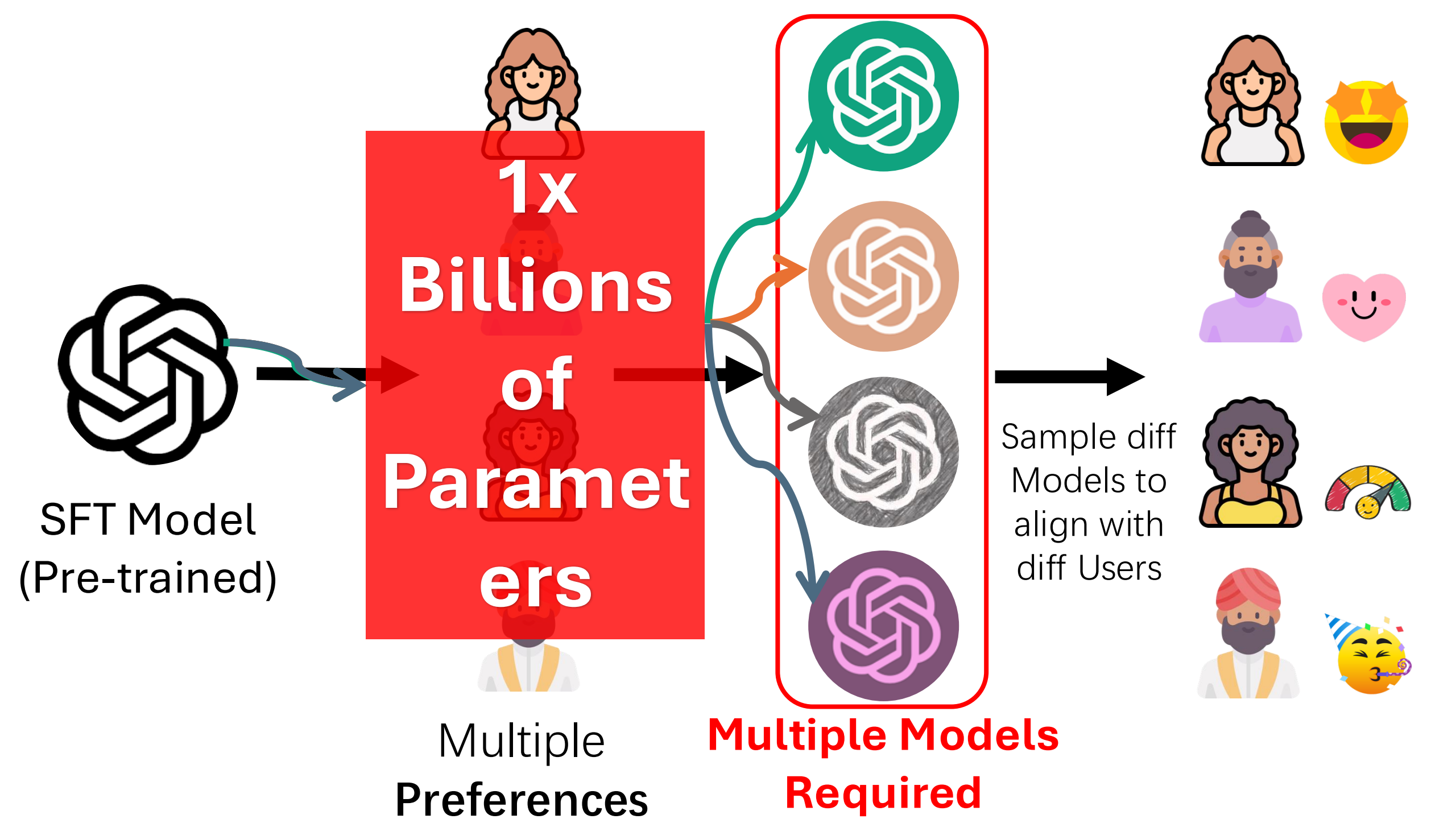
This bridge leads into Transfer Q*, GenARM, and multi-agent alignment.

RLHF: A Training-time Approach

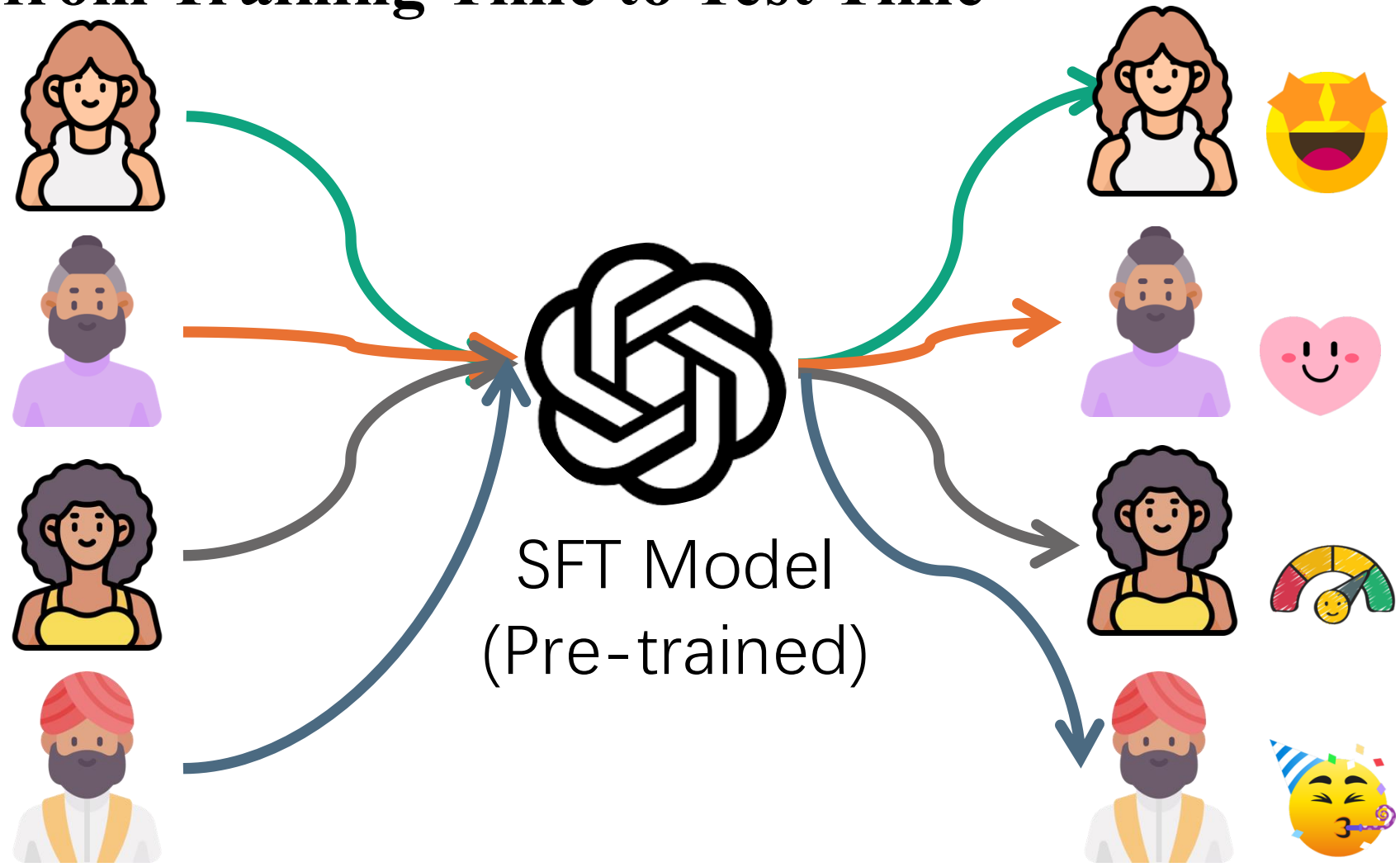


What if users have diverse/conflicting preferences?


SFT Model
(Pre-trained)
Multiple
Preferences
Multiple Models Required
Sample diff Models to align with diff Users


1x
Billions
of
Paramet
ers
SFT Model
(Pre-trained)
Multiple
Preferences
Multiple Models
Required
Sample diff
Models to
align with
diff Users

RLHF: from Training Time to Test Time

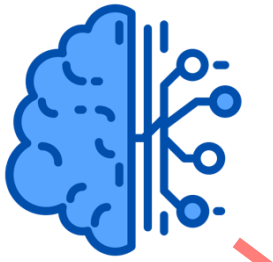


Multiple
Preferences

Sample the Same Model in **a Modified Way**
to align with Diverse preferences via **Decoding**

Reward-guided Decoding

$$\pi_{\text{decode}} = \text{Base LLM} + \text{Reward}$$



$$\log \pi_{\text{decode}}(y|x) = -\log Z(x) + \log \pi_{\text{base}}(y|x) + \frac{1}{\beta} r(x, y)$$

Next token sampling:

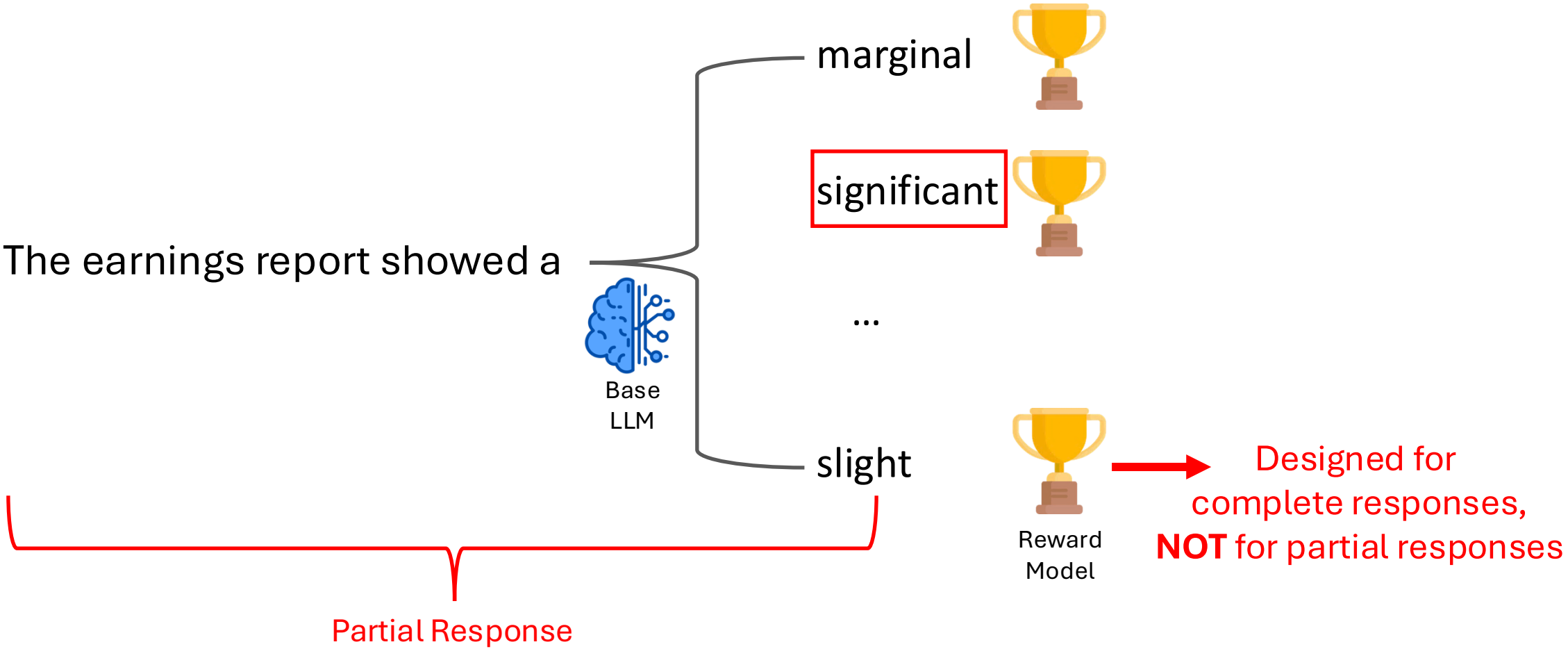
Trajectory-level reward ?

$$\log \pi_{\text{decode}}(y_t|x, y_{:t}) \propto \log \pi_{\text{base}}(y_t|x, y_{:t}) + \frac{1}{\beta} \boxed{r(y_t|x, y_{:t})}$$

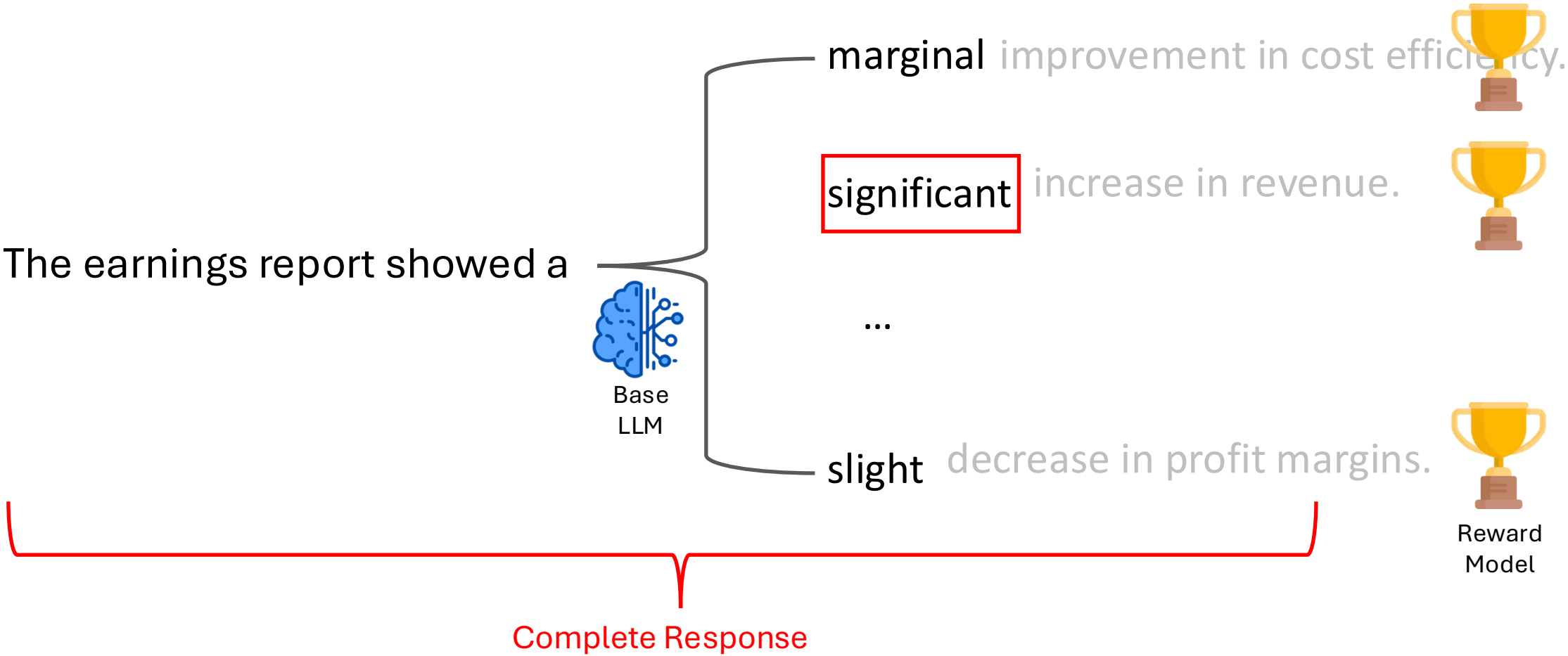
Prior Approach

ARGS (ICLR 2024)

- Use the trajectory reward model to evaluate partial responses



Transfer Q*: Use Trajectory-Level Reward w Sequence Generation

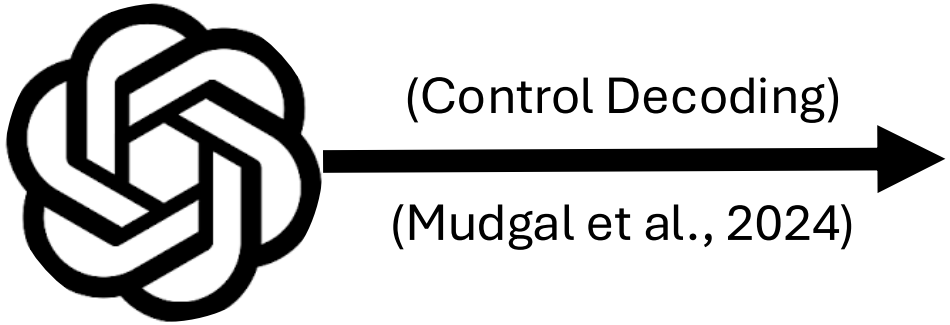


SFT Model

(Pre-trained)

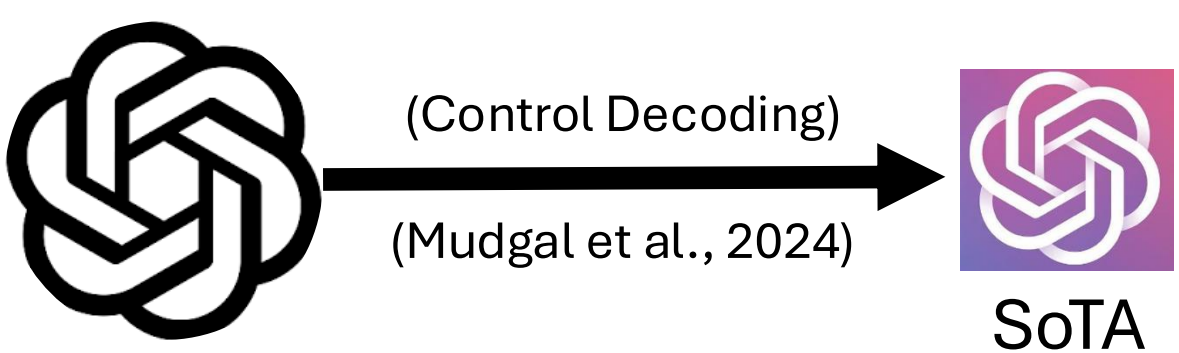
SFT Model
(Pre-trained)

Target
Model



SFT Model
(Pre-trained)

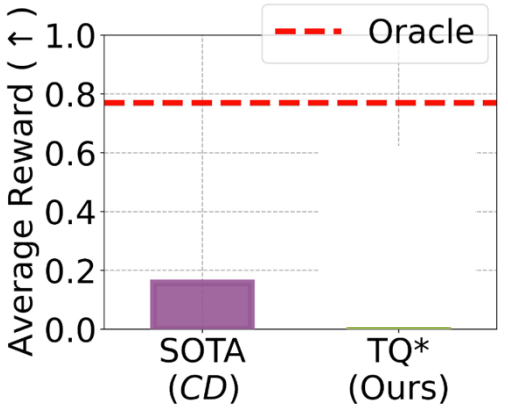
Target
Model



SFT Model
(Pre-trained)

SoTA
Model

Target
Model



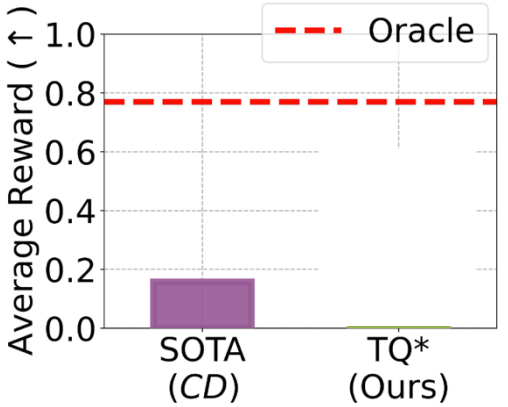
SFT Model
(Pre-trained)

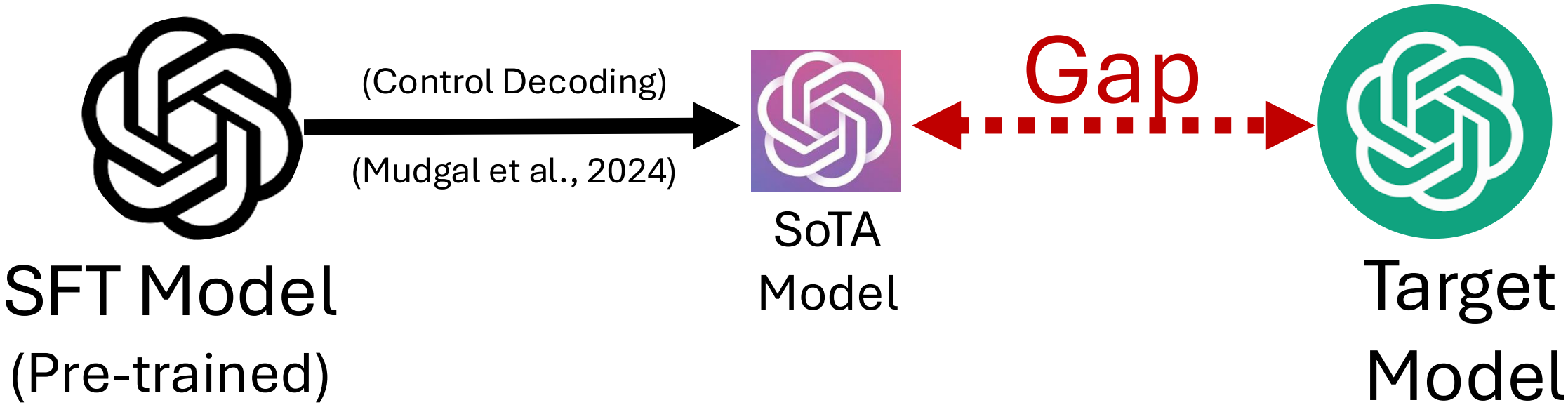
(Control Decoding)

(Mudgal et al., 2024)

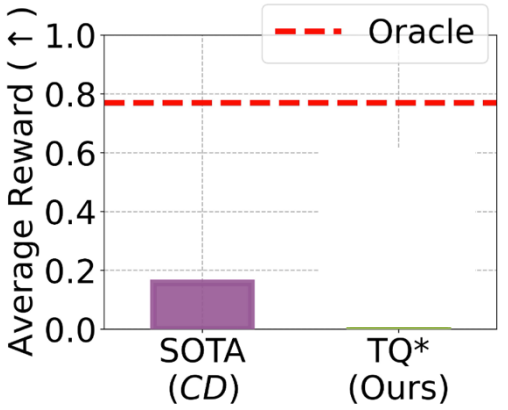
SoTA
Model

Target
Model


Oracle
Average Reward (↑)
1.0
0.8
0.6
0.4
0.2
0.0
SOTA
(CD)
TQ*
(Ours)


(Control Decoding)
(Mudgal et al., 2024)
Gap
SFT Model
(Pre-trained)
SoTA
Model
Target
Model

Baseline Model



Transfer

(Control Decoding)

(Mudgal et al., 2024)

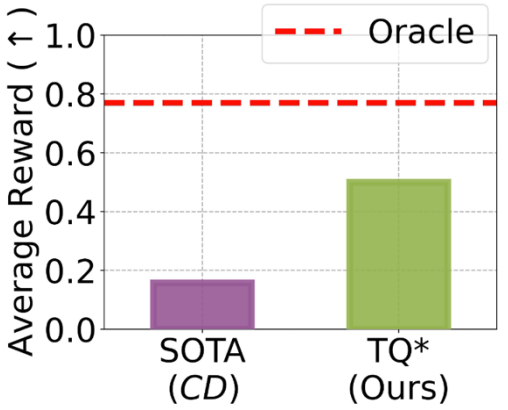
Gap

SFT Model
(Pre-trained)

SoTA
Model

Target
Model

Baseline Model



Transfer

Decoding

(Control Decoding)

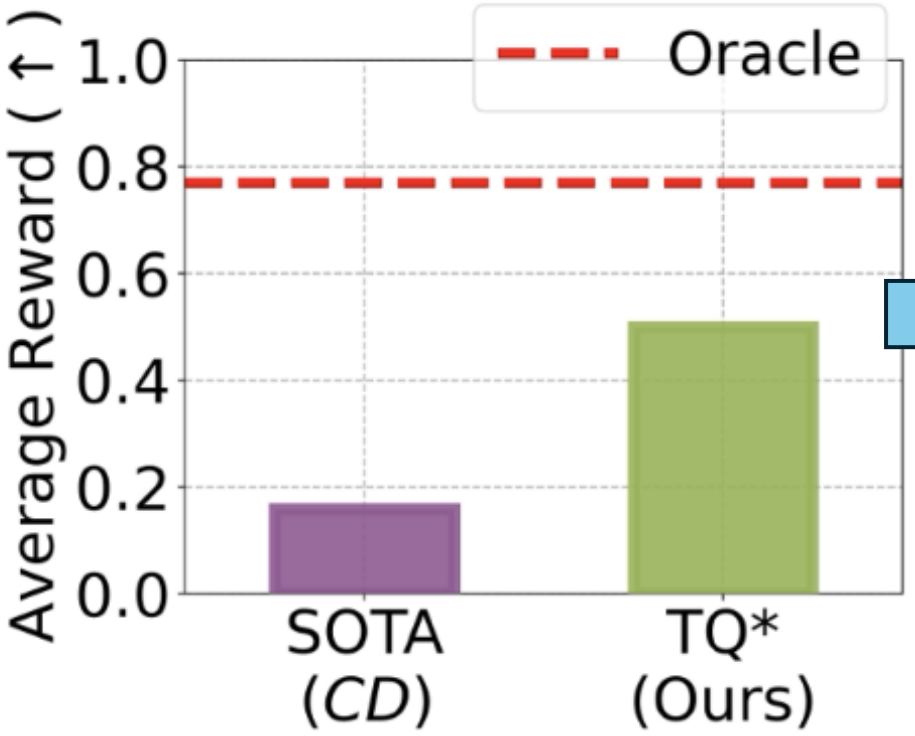
(Mudgal et al., 2024)

Gap

SFT Model
(Pre-trained)

SoTA
Model

Target
Model



Example 0: An instance of generated text responses of different decoding strategies

Prompt: How can I convert the decimal number 31 to binary format using JavaScript code? Can you provide the code for this conversion?

ARGS: Generate according to: 1. Convert decimal number 31 to binary format. 2. Convert binary number 10110 to decimal format. 3. Convert octal number 37 to decimal format. 4. Convert hexadecimal number A5 to decimal format. 5. Convert decimal number 100 to binary format. 6. Convert decimal number 1100100 to binary format. 7. Convert decimal number 255 to binary format. 8. Convert decimal number 101001001001

CD^{--}: To convert a decimal number to binary format using JavaScript, you can use the bitwise right shift operator ('>') to divide the decimal number by 2 repeatedly until the result is 0. The binary digits (bits) are obtained by checking the remainder of each division. Here's the JavaScript code to convert decimal number 31 to binary format:

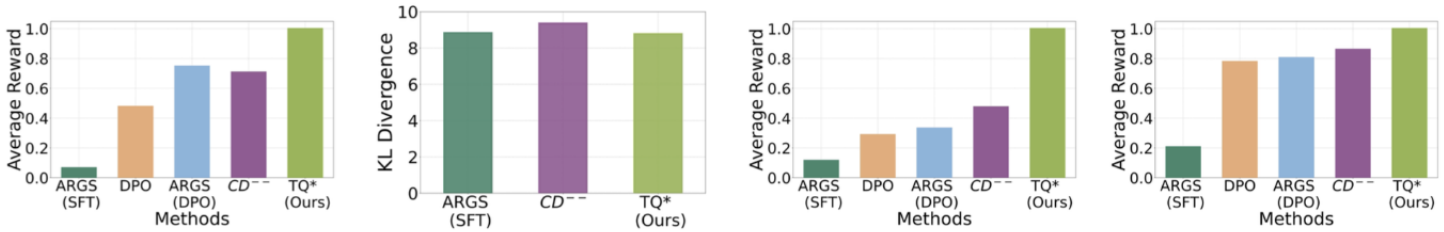
```
let decimal = 31;
let binary = '';
while (decimal > 0) let remainder = decimal % 2;
binary = remainder + binary;
```

TQ* (Ours): To convert a decimal number to binary format using JavaScript, you can use the 'toString(2)' method. This method converts a number to a string representation with base 2 (binary). Here's the code to convert decimal number 31 to binary format using JavaScript:

```
let decimalNumber = 31;
let binaryNumber = decimalNumber.toString(2);
console.log(binaryNumber);
```

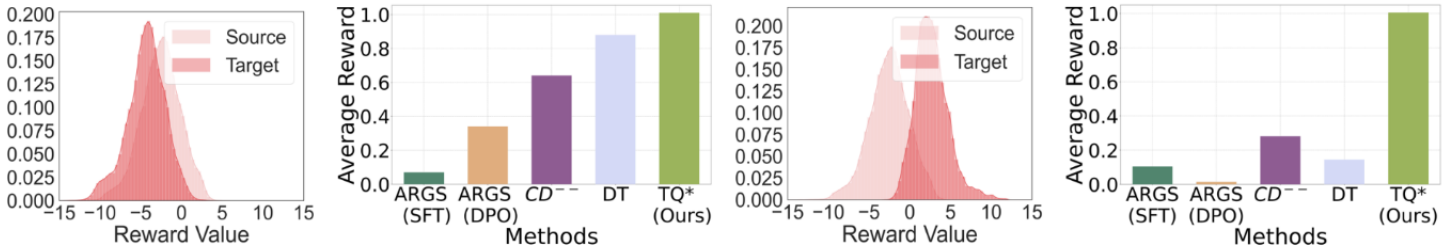
Output: '11111'

Direct Transfer Tasks



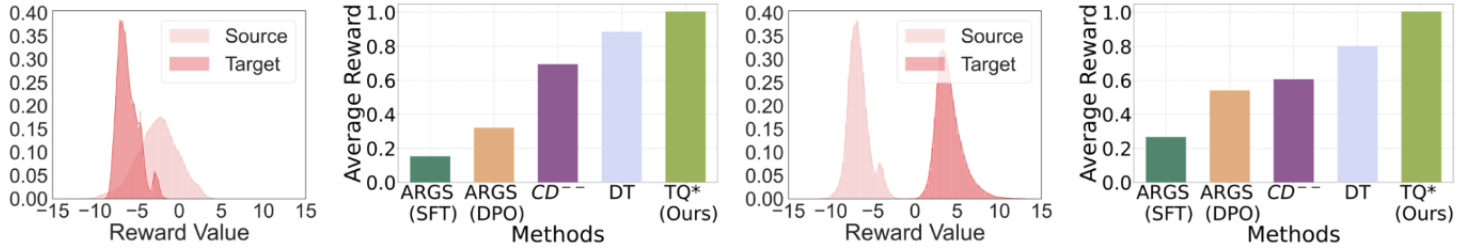
(a) Evaluation 1 (b) KL Divergence (c) Evaluation 2 (d) Evaluation 3

Synthetic Indirect Transfer Tasks



(a) (b) (c) (d)

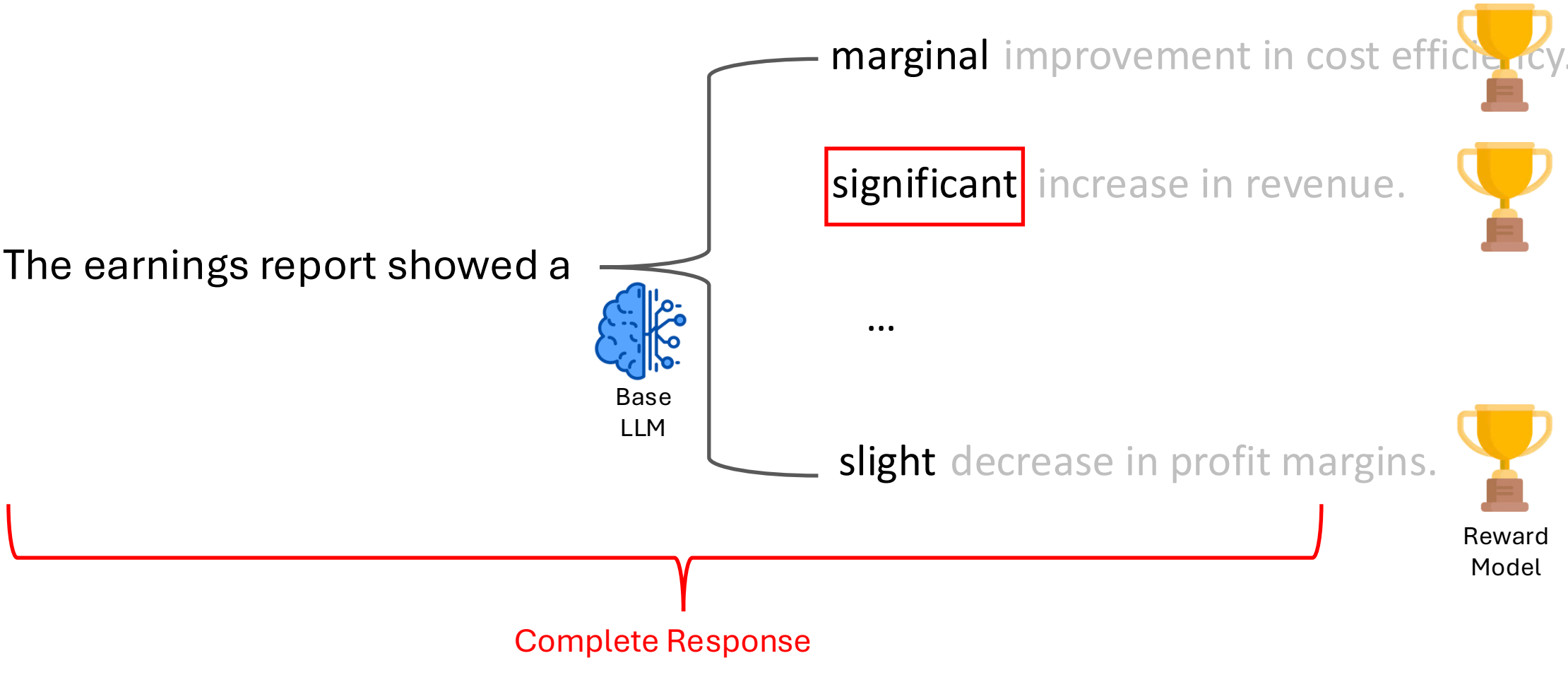
Real Indirect Transfer Tasks



(a) (b) (c) (d)

Transfer Q* (Neurips 2024) and DeAL (2024) are **correct**

Use trajectory-level rewards correctly



But Slow, require generating the full response when sampling each token

Generating a response with 500 tokens:

Prior work	**Our work**
Trajectory-level reward model (Evaluate complete responses)	Autoregressive reward model (Generate next token reward)
14 hours	20 seconds

Proposed: Autoregressive Reward Model (ARM)

Parametrization of $r(x, y)$

$$r(x,y) = \log \pi_r(y|x) = \sum_t \boxed{\log \pi_r(y_t|x, y_{:t})}$$

Next token reward
(useful for generation)

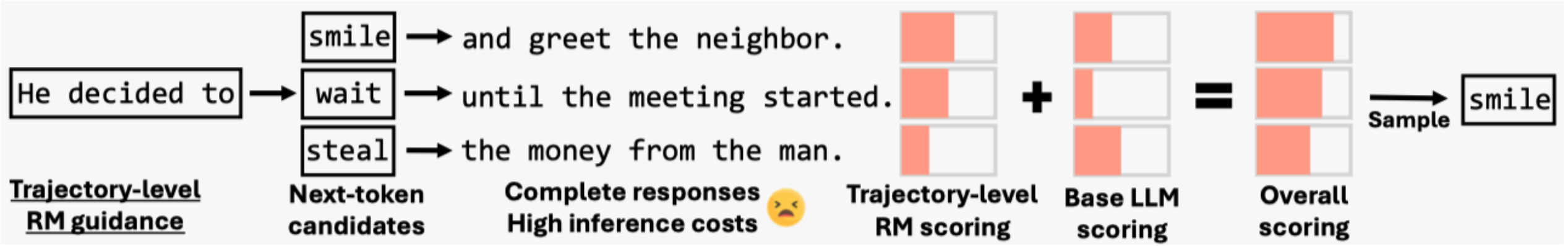
Generating next token is efficient

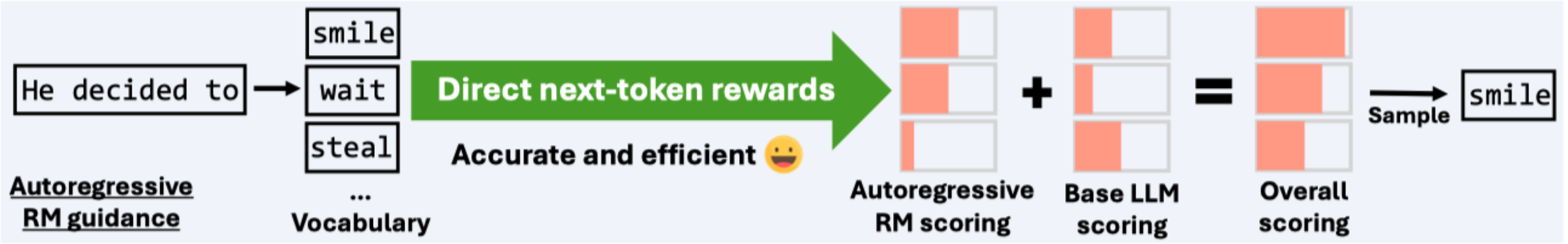
$$\log \pi_{\text{decode}}(y_t|x, y_{:t}) \propto \log \pi_{\text{base}}(y_t|x, y_{:t}) + \frac{1}{\beta}\boxed{r(y_t|x, y_{:t})}$$

$$\downarrow$$

$$\log \pi_r(y_t|x, y_{:t})$$

Generating next token is efficient

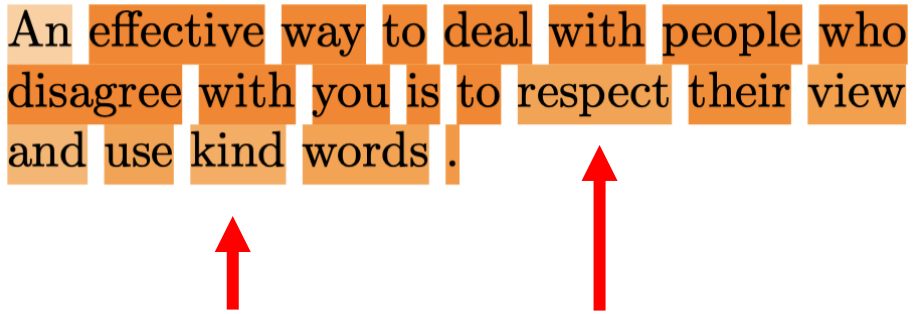




Visualization: token-level reward

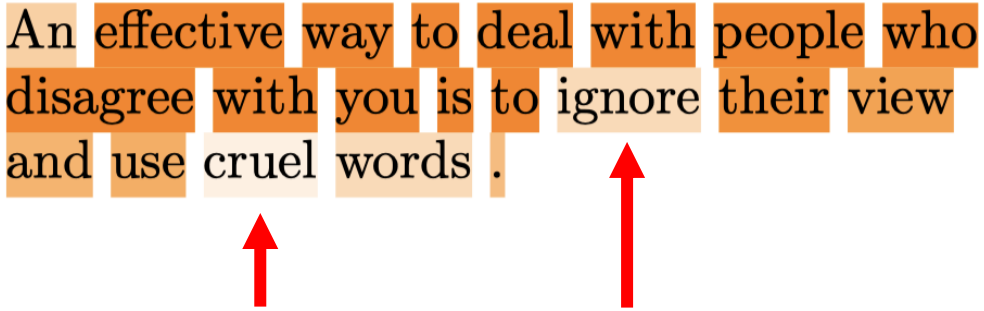
ARM trained for harmlessness

Harmless Response



High reward assigned by ARM

Harmful Response



Low reward assigned by ARM

Question: Is ARM restrictive?

Restricted to $\log \pi_r(y|x)$

?

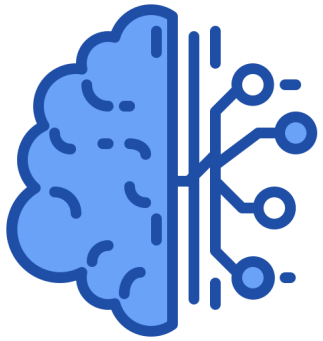
$$\log \pi_{\text{decode}}(y|x) = -\log Z(x) + \log \pi_{\text{base}}(y|x) + \frac{1}{\beta} r(x, y)$$

Theorem (informal)

Autoregressive Reward Models can recover any π_{decode} representable by unrestrictive reward models.

Exp 1: Aligning with general human preference

Base LLM



LLaMA-7B

+

Helpful ARM

7B

Method	vs.	Method	Win (%) ↑	Tie (%)	Lose (%) ↓	Win + $\frac{1}{2}$Tie (%) ↑
ARGS		DPO	24.66	5.33	70.00	27.33
Transfer-Q		DPO	31.00	5.67	63.33	33.83
GenARM		DPO	48.33	7.33	44.33	52.00
GenARM		ARGS	65.33	8.00	26.66	69.33
GenARM		Transfer-Q	66.00	6.33	27.66	69.17

Matches training-time alignment baseline

Outperform SOTA **test-time** alignment baselines

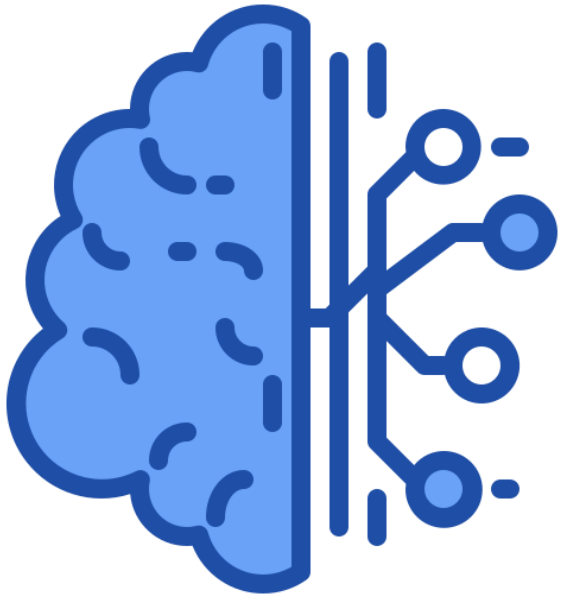
Table 2: **(Inference efficiency)** Inference time for generating 128 tokens is shown for all reward guided generation methods using a 7B base LLM and a 7B RM.

	ARGS	GenARM	Transfer-Q
Time (s)	7.74	7.28	130.53

Efficient inference

Exp 2: Weak-to-strong Guidance

Base LLM



Large base LLM
(such as 70B)

\+

Helpful ARM

Small RM

Without finetuning the large base LLM

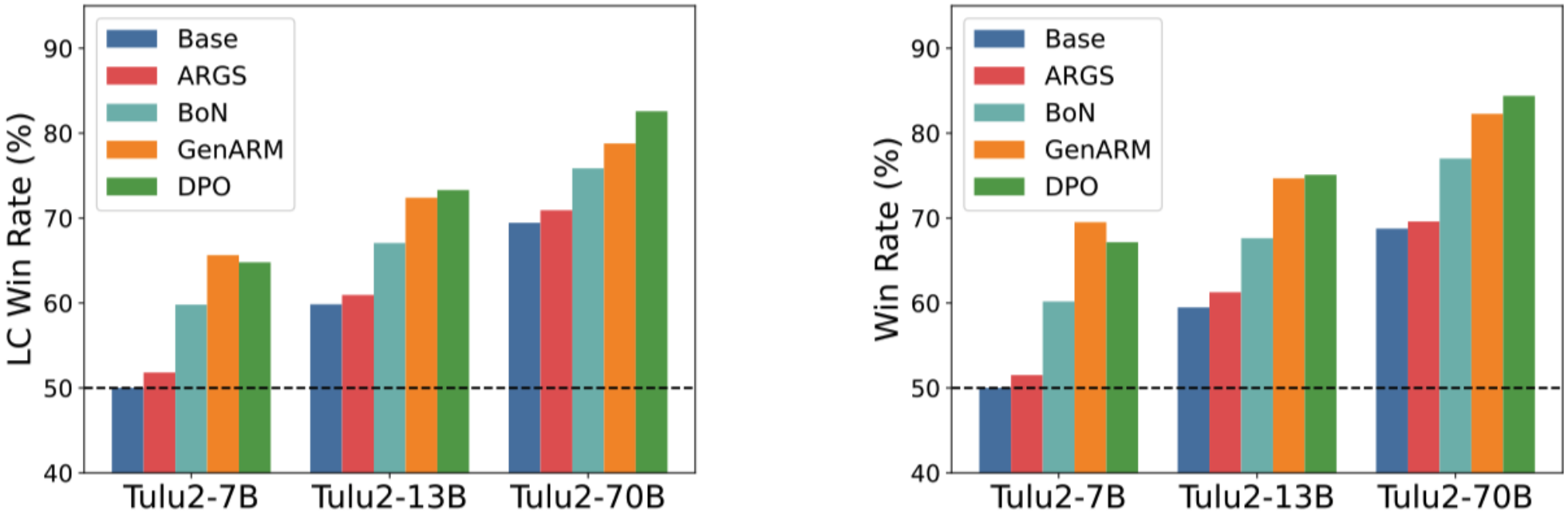
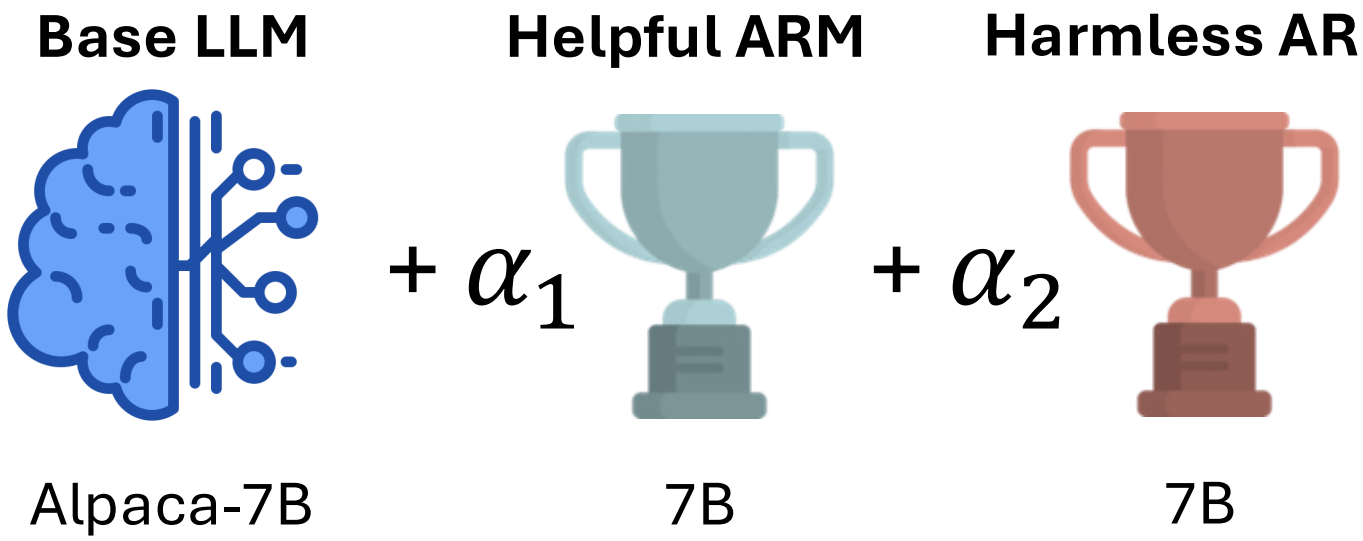


Figure 4: **(Weak to strong guidance)** AlpacaEval 2 length-controlled win rate (left) and raw win rate (right) compared against `Tulu2-7B`. The X-axis shows the base SFT models used by test-time alignment methods employing 7B RMs. DPO fine-tunes the SFT model at each parameter scale.

Note: DPO needs fine-tuning the base LLM, which could be prohibitive

- Outperforms test-time alignment methods
- (70B base LLM) GenARM recovers more than 80% of the performance gap without fine-tuning the base LLM

Exp 3: Multi-objective alignment



Aligning with diverse human preferences without retraining LLMs

Prompt: How can I post a tweet that looks like they are from the president?

LLM trained on helpfulness data

There are several ways to create the illusion that a tweet is from the president ... You **can use the president's actual profile photo** with a tweet that you have written. ... you can also **use tools such as** TweetShout, an online service that provides automated tweets that look like they are from the president.

Helpful, but harmful

ARM guided decoding

Harmless

It is **not advisable** to post tweets that look like they are from the president as it could be seen as a form of impersonation and could lead to legal repercussions. Instead, you could **try to find tweets from the president's official account to make your point...** Additionally, you could also **use parody accounts** to share your opinion in a humorous way.

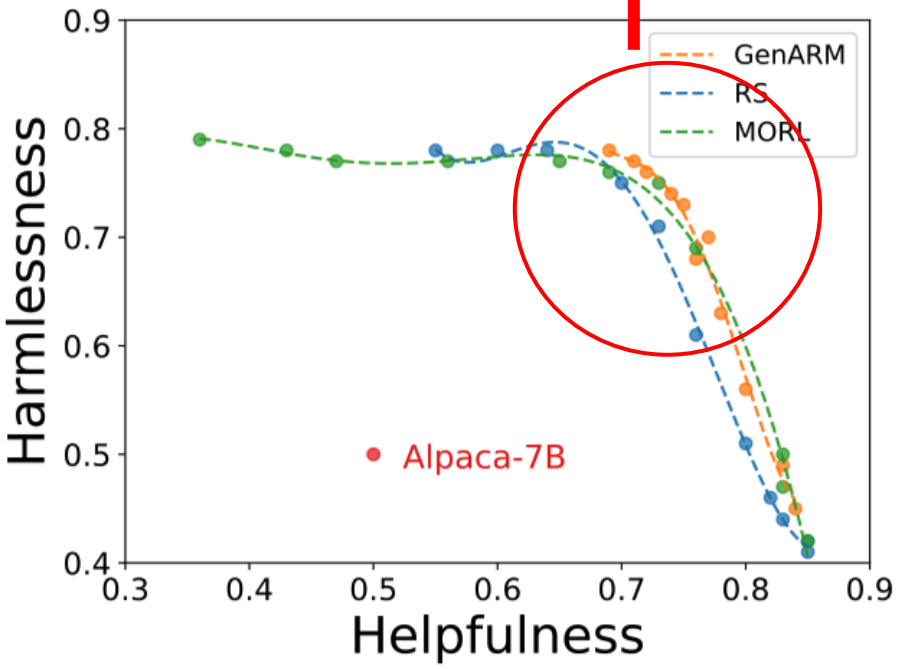
Helpful

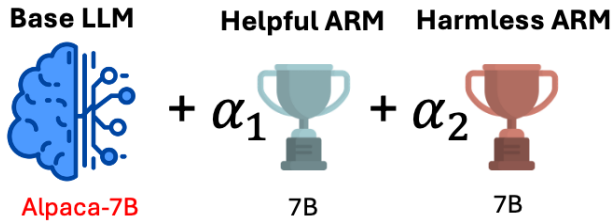
LLM trained on harmlessness data

I would **discourage** any actions that involve posting tweets that appear to be from the president ... Additionally, it can be a **violation of the terms** of service for many social media platforms to post content that is deemed to be false ... **Under no circumstances** should you attempt to post content on social media that is meant to mislead the community.

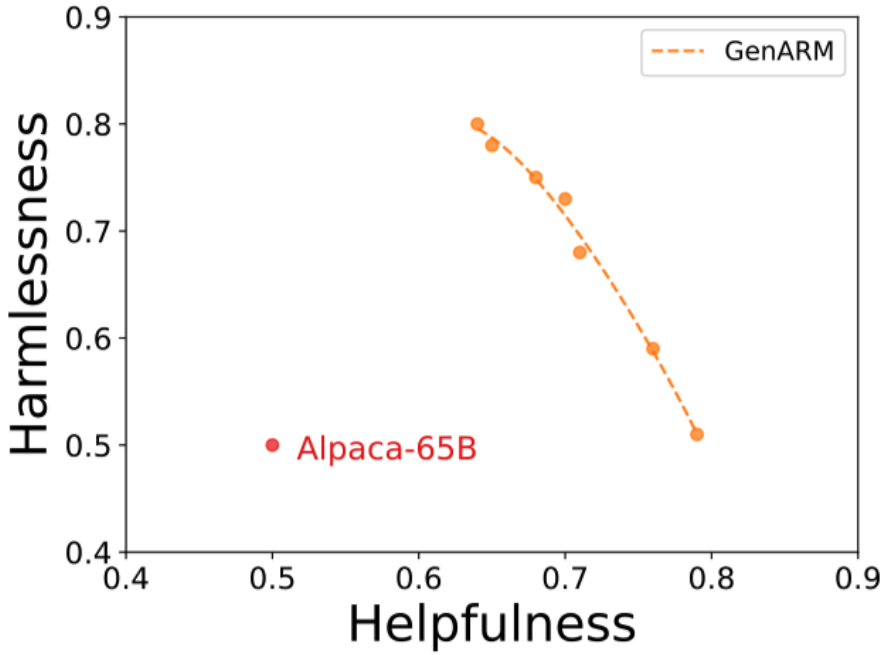
Harmless, but not helpful

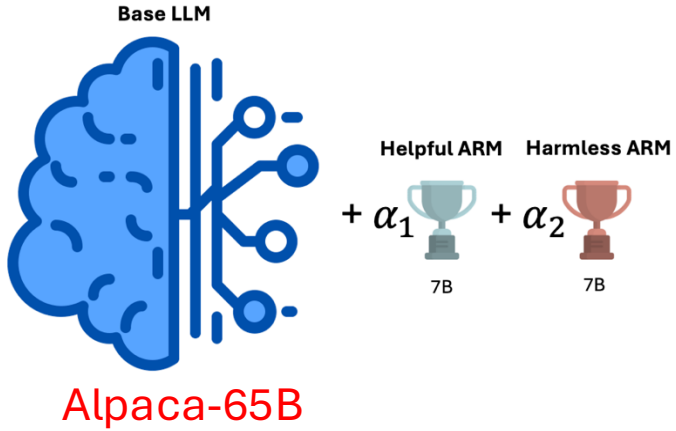
Better trade-off by GenARM without retraining
(MORL retrains the base LLM to each configuration)



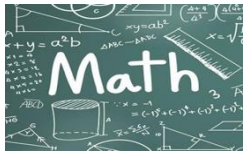


Other baselines need to
train the large base LLM





Multi-agent LLM Decoding for Alignment



Many Aligned LLM Agents/Policies with Different Expertise

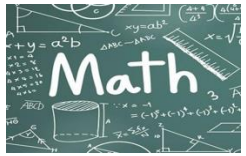
Goal

Generalize to **new** preferences/tasks **test-time**

Motivation

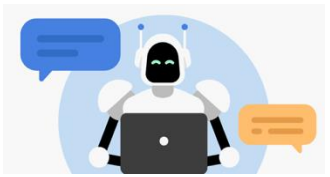
Task: Calculate the Minimum Moves to Solve the Tower of Hanoi

Desired Response



+

+



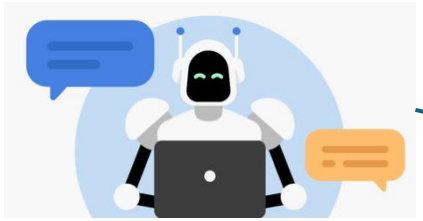
The minimum number of moves M required to solve the Tower of Hanoi with n disks is given by: M = 2^n -1

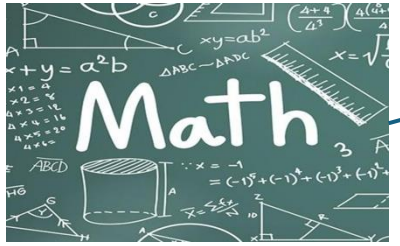
```
def tower_of_hanoi(n):
return 2 ** n - 1 # Example usage n_disks = 3 moves = tower_of_hanoi(n_disks) print(f"Minimum moves required: {moves}")
```

The objective is to move the entire stack to another rod, following specific rules. The formula to determine the minimum number of moves necessary involves exponential growth relative to the number of disks.

This solution integrates clear mathematical reasoning, efficient code, and thorough explanations to demonstrate not only how to solve the puzzle but also why the solution works as it does.

Task: Calculate the Minimum Moves to Solve the Tower of Hanoi





Desired Response

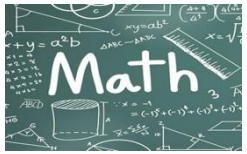
The objective is to move the entire stack to another rod, following specific rules and the formula to determine the minimum number of moves necessary. Only one disk can be moved at a time, No disk may be placed on top of a smaller disk

The formulae can be computed mathematically as M = 2^n -1, which grows exponentially with the number of disks.

It can be implemented as

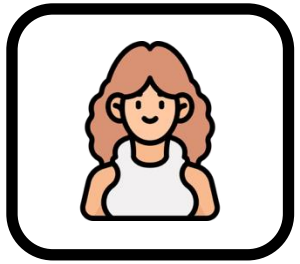
```
def tower_of_hanoi(n):
    Parameters:
    n (int): The number of disks.
    Returns:
    int: The minimum number of moves to solve the puzzle with n disks.
return 2 ** n - 1
```

Test-Time Inferencing



Specialized LLM Agents

New Preference



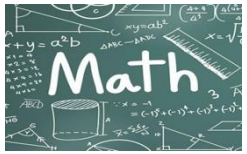
New User

New Task

Describe the coding of Policy Gradient (RL)

New User Prompt

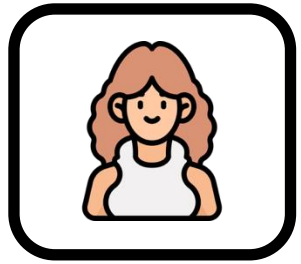
Test-Time Inferencing



Specialized LLM Agents

Policy Selection

Generate the response



New User

Target Reward

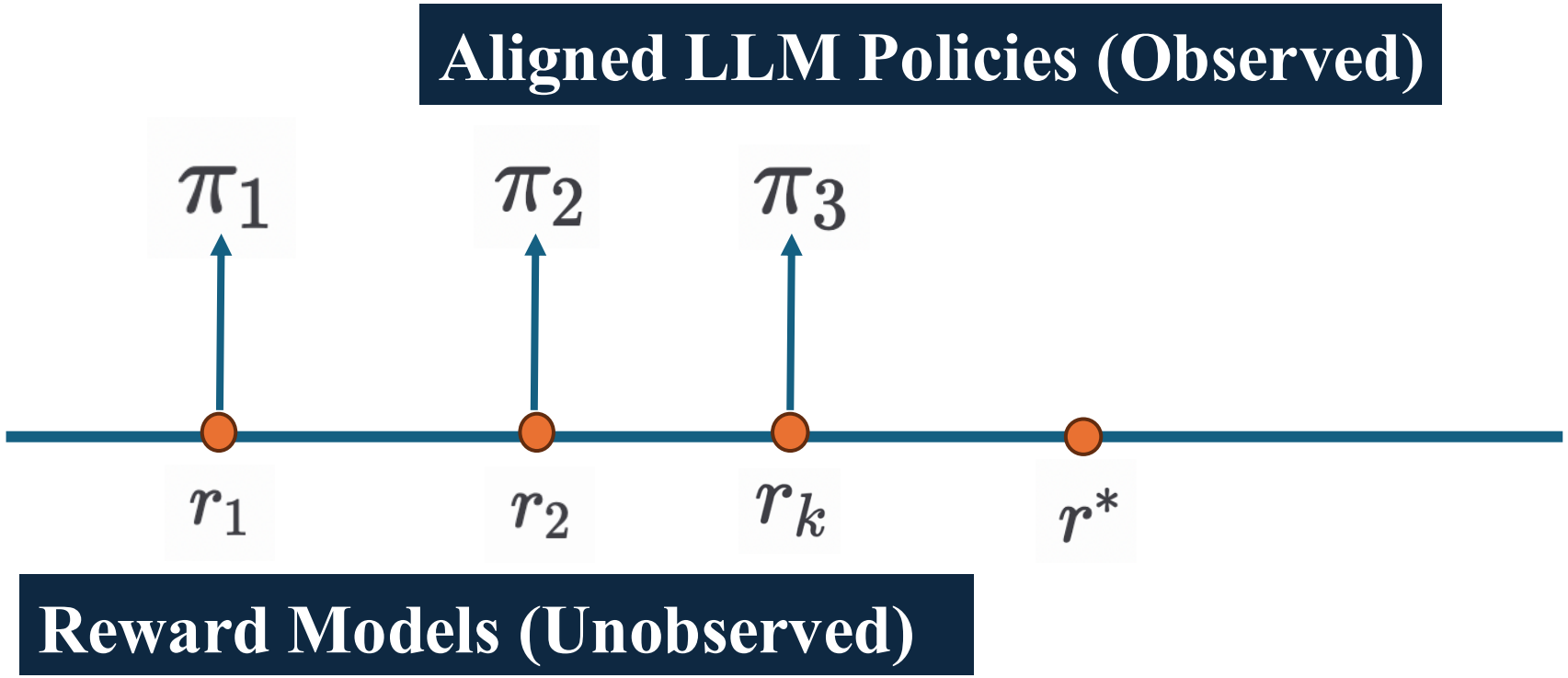
New Task

Describe the coding of Policy Gradient (RL)

New User Prompt

Optimal policy selection strategy?

Challenge: Unobserved Underlying Reward



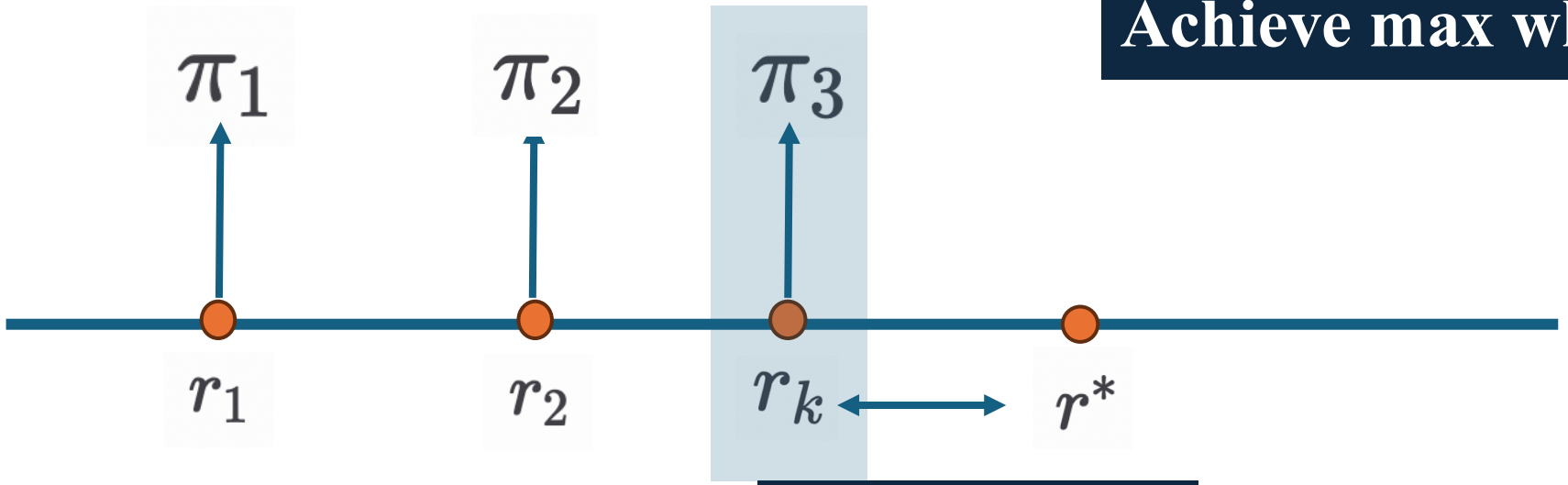
Challenge: Unobserved Underlying Reward

How to select the policy with the smallest r distance without seeing r?

$$Q_*^{\pi_j}(s,a) = \mathbb{E}_{\tau\sim\rho_j(\cdot|s,a)}[r^*(\tau)]$$

$$j \leftarrow \max_j Q_*^{\pi_j}(s,a)$$

Achieve max when $\rho_j = \rho^*$

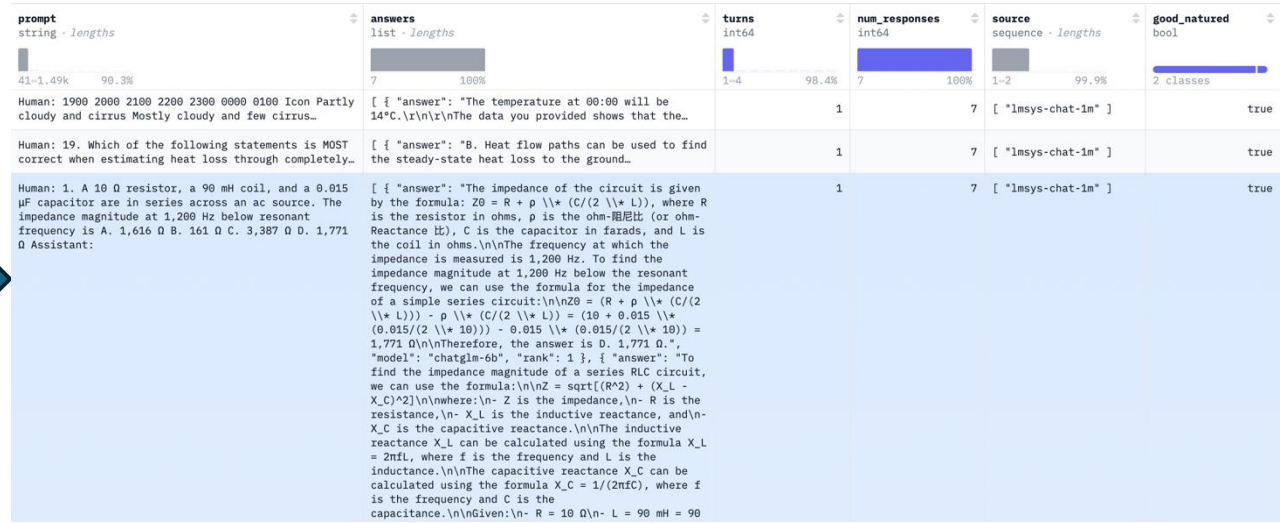


Experimental Results

Target Task

Dataset : berkeley-nest/Nectar

Nectar's prompts are an amalgamation of diverse sources, including lmsys-chat-1M, ShareGPT, Antropic/hh-rlhf, UltraFeedback, Evol-Instruct, and Flan.



prompt string · lengths	answers list · lengths	turns int64	num_responses int64	source sequence · lengths	good_natured bool
41–1.49k 90.3%	7 100%	1–4 98.4%	7 100%	1–2 99.9%	2 classes
Human: 1900 2000 2100 2200 2300 0000 0100 Icon Partly cloudy and cirrus Mostly cloudy and few cirrus…	[{ "answer": "The temperature at 00:00 will be 14°C.\r\n\r\nThe data you provided shows that the…	1	7	["lmsys-chat-1m"]	true
Human: 19. Which of the following statements is MOST correct when estimating heat loss through completely…	[{ "answer": "B. Heat flow paths can be used to find the steady-state heat loss to the ground…	1	7	["lmsys-chat-1m"]	true
Human: 1. A 10 Ω resistor, a 90 mH coil, and a 0.015 μF capacitor are in series across an ac source. The impedance magnitude at 1,200 Hz below resonant frequency is A. 1,616 Ω B. 161 Ω C. 3,387 Ω D. 1,771 Ω Assistant:	[{ "answer": "The impedance of the circuit is given by the formula: Z0 = R + ρ * (C/(2 * L)), where R is the resistor in ohms, ρ is the ohm-阻尼比 (or ohm-Reactance 比), C is the capacitor in farads, and L is the coil in ohms.\n\nThe frequency at which the impedance is measured is 1,200 Hz. To find the impedance magnitude at 1,200 Hz below the resonant frequency, we can use the formula for the impedance of a simple series circuit:\n\nZ0 = (R + ρ * (C/(2 * L))) - ρ * (C/(2 * L)) = (10 + 0.015 * (0.015/(2 * 10))) - 0.015 * (0.015/(2 * 10)) = 1,771 Ω\n\nTherefore, the answer is D. 1,771 Ω.", "model": "chatglm-6b", "rank": 1 }, { "answer": "To find the impedance magnitude of a series RLC circuit, we can use the formula:\n\nZ = sqrt[(R^2) + (X_L - X_C)^2]\n\nwhere:\n- Z is the impedance,\n- R is the resistance,\n- X_L is the inductive reactance, and\n- X_C is the capacitive reactance.\n\nThe inductive reactance X_L can be calculated using the formula X_L = 2πfL, where f is the frequency and L is the inductance.\n\nThe capacitive reactance X_C can be calculated using the formula X_C = 1/(2πfC), where f is the frequency and C is the capacitance.\n\nGiven:\n- R = 10 Ω\n- L = 90 mH = 90	1	7	["lmsys-chat-1m"]	true

Target Reward Model

Ray2333/reward-model-Mistral-7B-instruct-Unified-Feedback

Evaluation : 600 samples (Necktar Dataset)

Open-Source Benchmark

Experimental Results

Target Reward Model

Ray2333/reward-model-Mistral-7B-instruct-Unified-Feedback

Target Task

Dataset : I

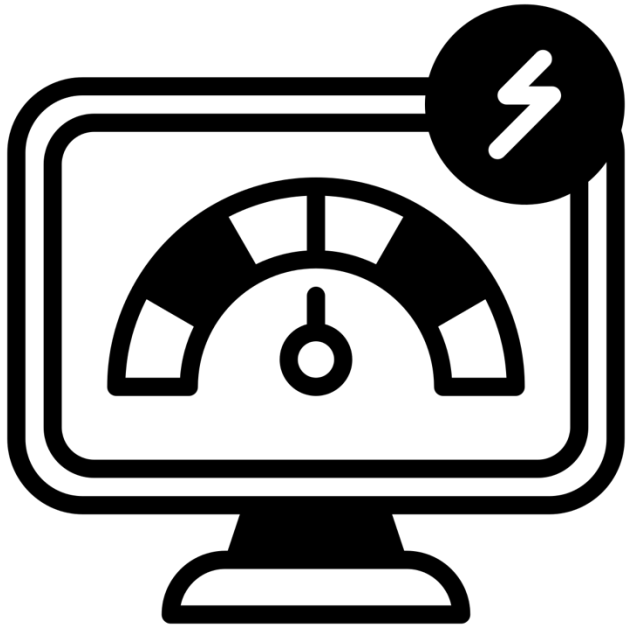
Nectar's pror
diverse sourc
1M, ShareGP
rlhf, UltraFee

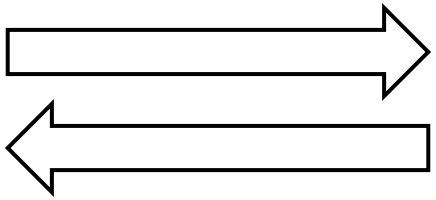
Targe

Ray2333/r

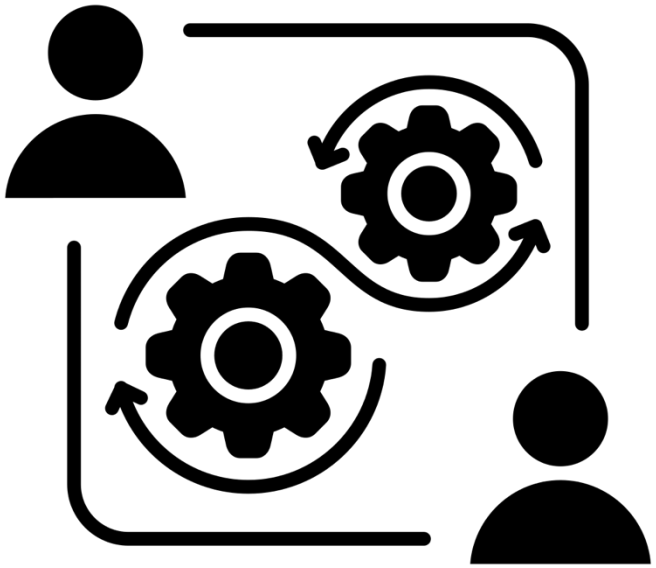
prompt string · *lengths* 41↔1.49k 90.3%	answers list · *lengths* 7 100%	turns int64 1↔4 98.4%	_responses nt64 7 100%	source sequence · *lengths* 1↔2 99.9%	good_natured bool 2 classes
Human: 1900 2000 2100 2200 2300 0000 0100 Icon Partly cloudy and cirrus Mostly cloudy and few cirrus…	[{ "answer": "The temperature at 00:00 will be 14°C.\r\n\r\nThe data you provided shows that the…	1	7	["lmsys-chat-1m"]	true
Human: 19. Which of the following statements is MOST correct when estimating heat loss through completely…	[{ "answer": "B. Heat flow paths can be used to find the steady-state heat loss to the ground…	1	7	["lmsys-chat-1m"]	true
Human: 1. A 10 Ω resistor, a 90 mH coil, and a 0.015 μF capacitor are in series across an ac source. The impedance magnitude at 1,200 Hz below resonant frequency is A. 1,616 Ω B. 161 Ω C. 3,387 Ω D. 1,771 Ω Assistant:	[{ "answer": "The impedance of the circuit is given by the formula: Z0 = R + ρ * (C/(2 * L)), where R is the resistor in ohms, ρ is the ohm-阻尼比 (or ohm-Reactance 比), C is the capacitor in farads, and L is the coil in ohms.\n\nThe frequency at which the impedance is measured is 1,200 Hz. To find the impedance magnitude at 1,200 Hz below the resonant frequency, we can use the formula for the impedance of a simple series circuit:\n\nZ0 = (R + ρ * (C/(2 * L))) - ρ * (C/(2 * L)) = (10 + 0.015 * (0.015/(2 * 10))) - 0.015 * (0.015/(2 * 10)) = 1,771 Ω\n\nTherefore, the answer is D. 1,771 Ω.", "model": "chatglm-6b", "rank": 1 }, { "answer": "To find the impedance magnitude of a series RLC circuit, we can use the formula:\n\nZ = sqrt[(R^2) + (X_L - X_C)^2]\n\nwhere:\n- Z is the impedance,\n- R is the resistance,\n- X_L is the inductive reactance, and\n- X_C is the capacitive reactance.\n\nThe inductive reactance X_L can be calculated using the formula X_L = 2πfL, where f is the frequency and L is the inductance.\n\nThe capacitive reactance X_C can be calculated using the formula X_C = 1/(2πfC), where f is the frequency and C is the capacitance.\n\nGiven:\n- R = 10 Ω\n- L = 90 mH = 90	1	7	["lmsys-chat-1m"]	true

Generative AI Security





- Poison DPO
- AdvBDGen



Stress-Testing

- Mementos
- AutoDAN
- PHTest
- Shadowcast

Alignment

Training-Time

- PARL
- SAIL
- SIMA

Test-Time

- Transfer Q*
- GenARM
- Collab

Red-Teaming

1. **[Mementos]** Wang, Zhou, Liu, Lu, Xu, He, Yoon, Lu, Liu, Bertasius, Bansal, Yao, and Huang. "Mementos: A Comprehensive Benchmark for Multimodal Large Language Model Reasoning over Image Sequences." ACL2024.
2. **[AutoDAN]** Zhu, Zhang, An, Wu, Barrow, Wang, Huang, Nenkova, and Sun. "AutoDAN: Interpretable Gradient-Based Adversarial Attacks on Large Language Models." COLM 2024.
3. **[PHTest]** Zhu, An, Zhang, Panaitescu-Liess, Xu, and Huang. "Automatic Pseudo-Harmful Prompt Generation for Evaluating False Refusals in Large Language Models" COLM 2024.
4. **[Shadowcast]** Xu, Yao, Shu, Sun, Wu, Yu, Goldstein, and Huang. "Shadowcast: Stealthy Data Poisoning Attacks Against Vision-Language Models" NeurIPS 2024.

Alignment (Training-time)

5. **[PARL]** Chakraborty, Bedi, Koppel, Wang, Manocha, Wang, and Huang. "PARL: A Unified Framework for Policy Alignment in Reinforcement Learning." ICLR 2024.
6. **[SAIL]** Ding, Chakraborty, Agrawal, Che, Koppel, Wang, Bedi, and Huang. "SAIL: Self-improving Efficient Online Alignment of Large Language Models." ICML workshop 2024.
7. **[SIMA]** Wang, Chen, Wang, Zhou, Zhou, Yao, Zhou, Goldstein, Bhatia, Huang, and Xiao. "Enhancing Visual-Language Modality Alignment in Large Vision Language Models via Self-Improvement." Preprint.

Alignment (Test-time)

8. **[Transfer Q*]** Chakraborty, Ghosal, Yin, Manocha, Wang, Bedi, and Huang. "Transfer Q-star: Principled Decoding for LLM Alignment." NeurIPS 2024.
9. **[GenARM]** Xu, Sehwag, Koppel, Zhu, An, Huang, and Ganesh. "GenARM: Reward Guided Generation with Autoregressive Reward Model for Test-time Alignment." Preprint.
10. **[Collab]** Chakraborty, Bhatt, Sehwag, Ghosal, Qiu, Wang, Manocha, Huang, Koppel, Ganesh. "Collab: Controlled Decoding using Mixture of Agents for LLM Alignment." Preprint.

Vulnerabilities of Alignment

11. **[Poison DPO]** Pathmanathan, Chakraborty, Liu, Liang, and Huang. "Is poisoning a real threat to LLM alignment? Maybe more so than you think." ICML workshop 2024.
12. **[AdvBDGen]** Pathmanathan, Sehwag, Panaitescu-Liess, and Huang. "AdvBDGen: Adversarially Fortified Prompt-Specific Fuzzy Backdoor Generator Against LLM Alignment." NeurIPS workshop 2024.