CSE 493s/599s
Lecture 18.

Sewoong Oh

Lecture notes

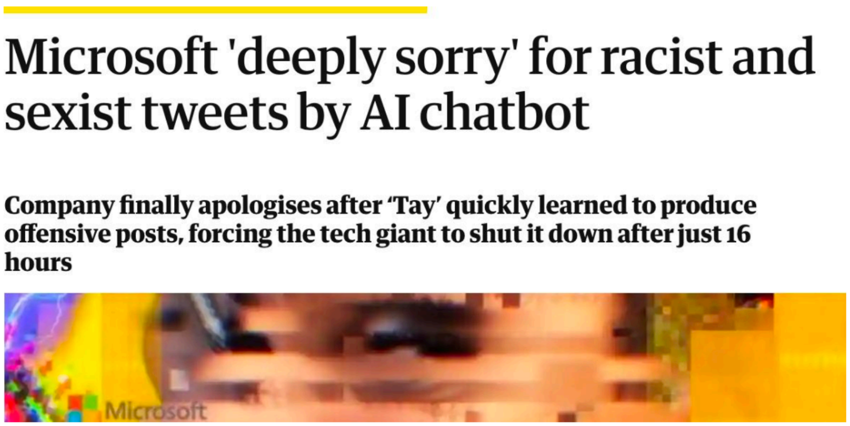
- These lecture notes are based on other courses in LLMs, including
  - CSE493S/599S at UW by Ludwig Schmidt: https://mlfoundations.github.io/advancedml-sp23/
  - EE-628 at EPFL by Volkan Cevher: https://www.epfl.ch/labs/lions/teaching/ee-628-training-large-language-models/ee-628-slides-2025/
  - ECE381V Generative Models at UT Austin by Sujay Sanghavi
  - and various papers and blogs cited at the end of the slide deck

Outline

- Language models
- General LLM framework
    - Token processing
    - Sequence mixing
    - Prediction
- Prompting techniques at inference time
    - In-context learning
    - Chain-of-thought prompting
- Fine-tuning
- Alignment

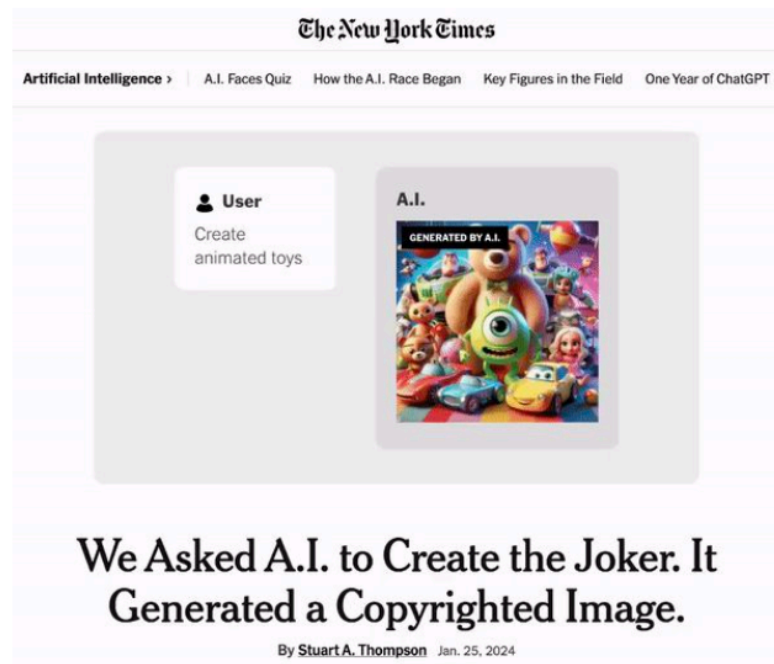
Alignment

- AI model responses can be misaligned with what we want them to do, which can sometimes cause real harm.



Microsoft 'deeply sorry' for racist and sexist tweets by AI chatbot

Company finally apologises after 'Tay' quickly learned to produce offensive posts, forcing the tech giant to shut it down after just 16 hours



The New York Times

Artificial Intelligence › | A.I. Faces Quiz | How the A.I. Race Began | Key Figures in the Field | One Year of ChatGPT

We Asked A.I. to Create the Joker. It Generated a Copyrighted Image.

By Stuart A. Thompson Jan. 25, 2024

The Washington Post
Democracy Dies in Darkness

Chatbots' inaccurate, misleading responses about U.S. elections threaten to keep voters from polls

By Garance Burke | AP

- Scaling pertaining does not address the challenges in **alignment** with human values and intent. We need post-training based on **RLHF (Reinforcement Learning with Human Feedback)**.

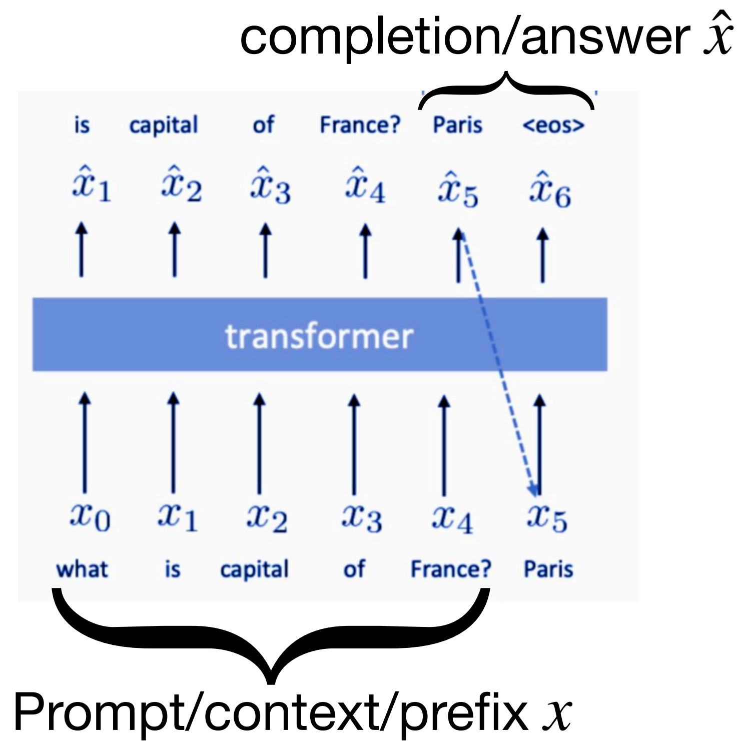
- We do not cover reinforcement learning in this class in any depths, but we will learn as much as we need along the way.

- Given a prompt/context/prefix x and its completion $\hat{x}$, a **reward model** assigns a scalar value on the quality of the completion:

$$r(x, \hat{x}) \in \mathbb{R}$$

- In RL, the language model is called a **policy** that assigns a probability to a completion:

$$\pi(\hat{x} \mid x)$$



- RLHF to train an LM to write summaries better.

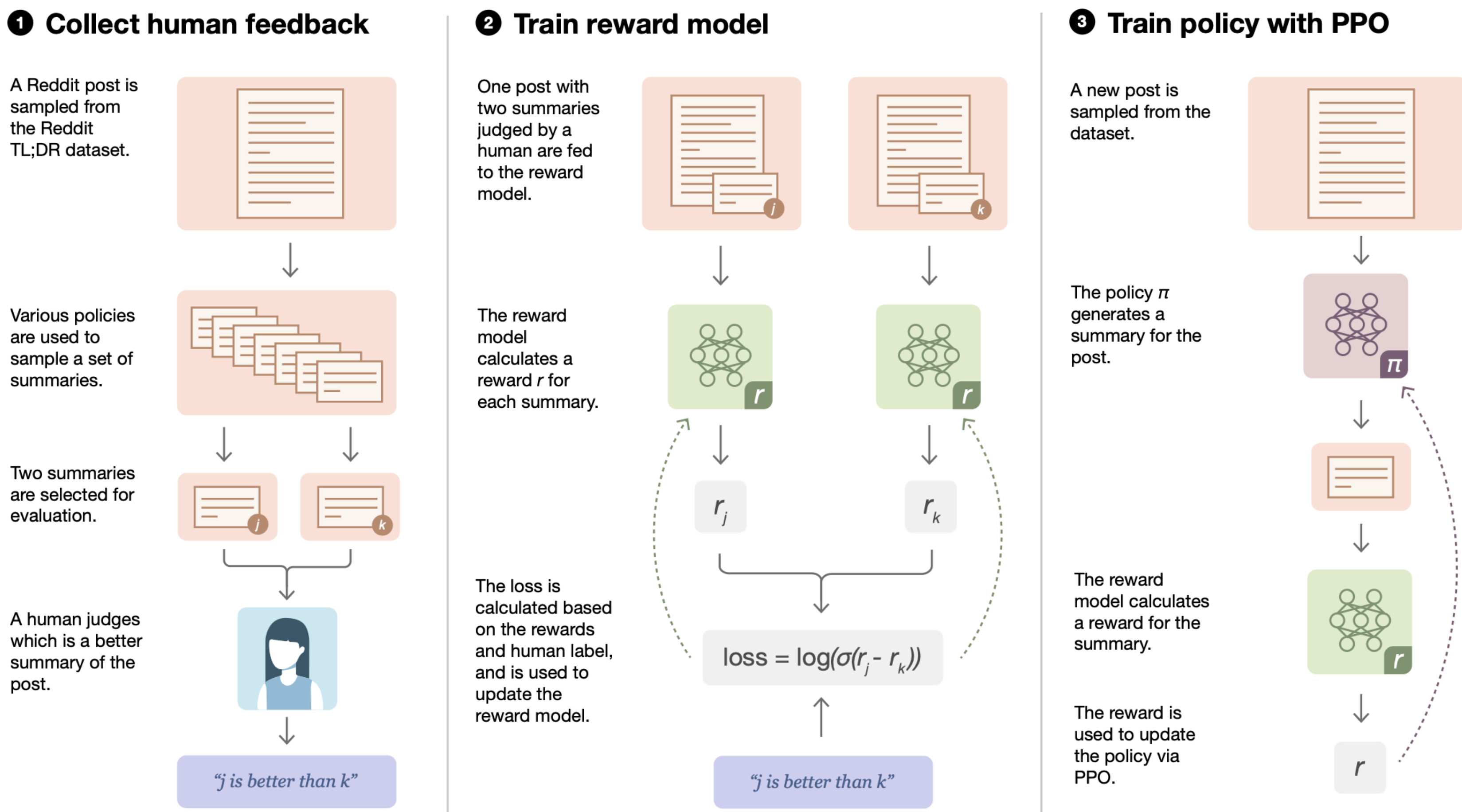


Figure 2: Diagram of our human feedback, reward model training, and policy training procedure.

[Stiennon et al. 2009]

- Human evaluation of various summarization techniques:

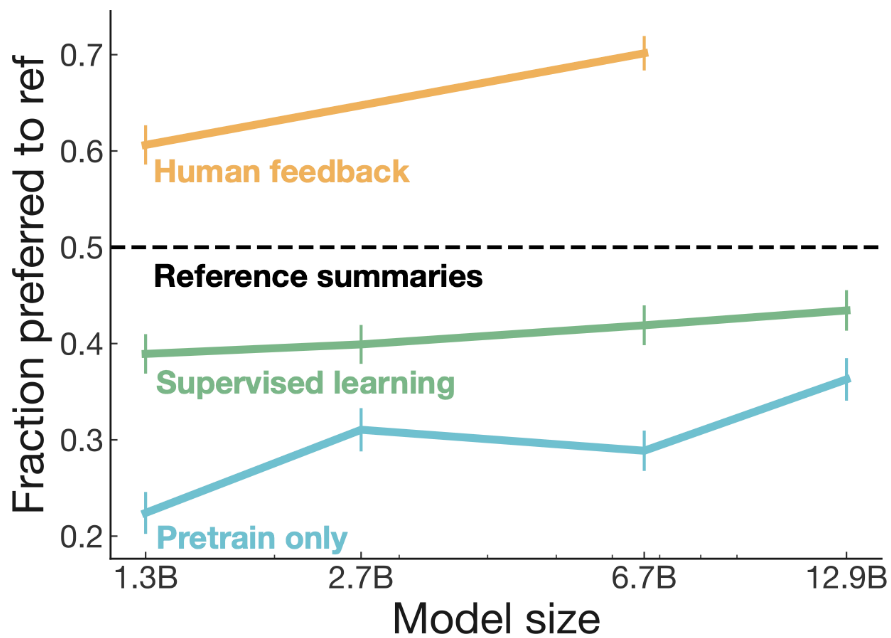


Figure 1: Fraction of the time humans prefer our models' summaries over the human-generated reference summaries on the TL;DR dataset.[4] Since quality judgments involve an arbitrary decision about how to trade off summary length vs. coverage within the 24-48 token limit, we also provide length-controlled graphs in Appendix F; length differences explain about a third of the gap between feedback and supervised learning at 6.7B.

[Stiennon et al. 2009]

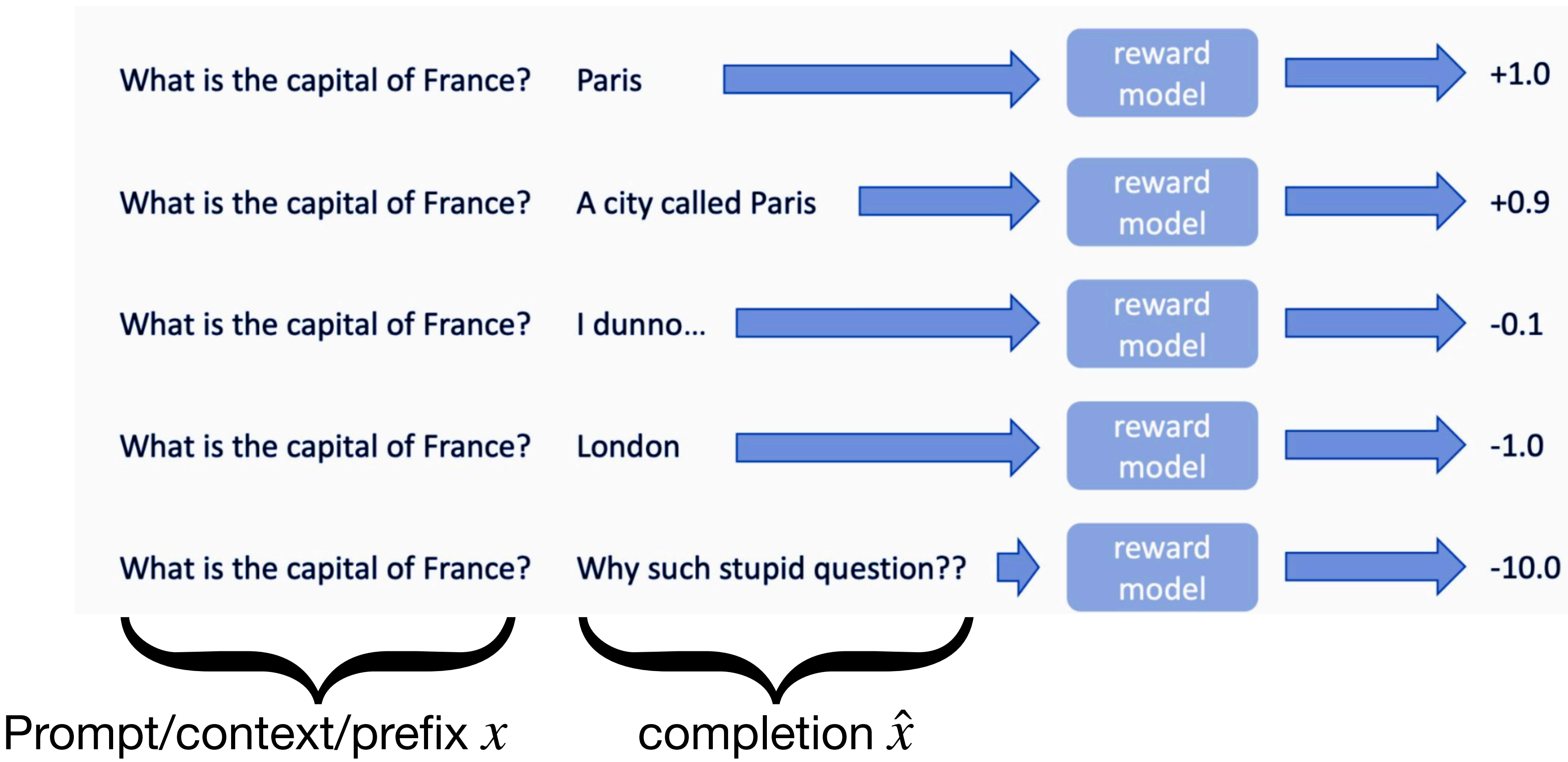
- **RLHF**: Strategy for further training the LM to make it aligned.
    1. Pretrain a language model.
    2. Learn a reward model $r(x, \hat{x})$ that maps arbitrary prompt and completion to a real value
    3. Fine-tune a pre-trained LM with RL using the reward model.
- This is one example of RLHF and there are several difference variations.

- **RLHF**: Strategy for further training the LM to make it aligned.

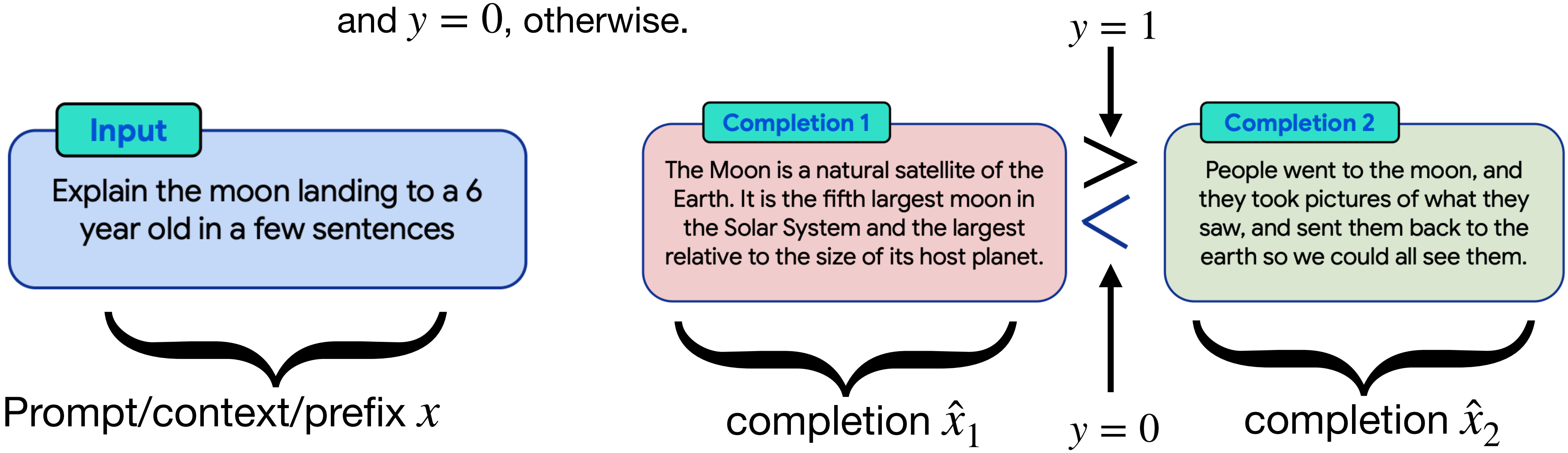
  1. Pretrain a language model.
  2. Learn a reward model $r(x, \hat{x})$ that maps arbitrary prompt and completion to a real value
  3. Fine-tune a pre-trained LM with RL using the reward model.

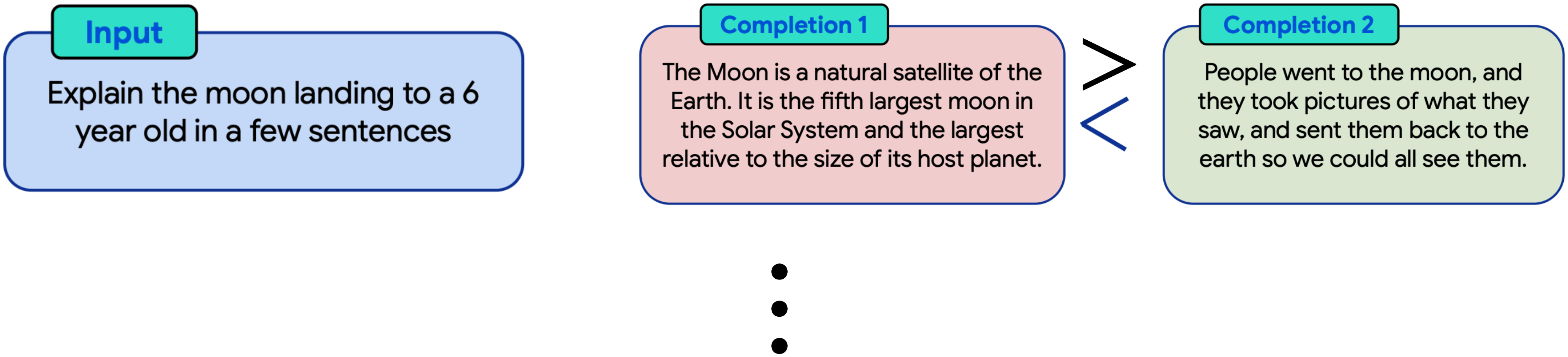
- This is one example of RLHF and there are several difference variations.

- consider a neural network function $r(x, \hat{x})$, that we want to train on **human labelled preference**, such that it provides useful "reward"



- However, it is challenging to collect reliable reward, so instead we collect data consisting of pairwise comparisons on which completion is preferred. This does not fit questions where exact solutions exist, like math and coding, where we use RLVF (Reinforcement Learning with Verifiable Feedback) instead.
- Each sample $(x, \hat{x}_1, \hat{x}_2, y)$ consists of
  - prompt x,
  - two different completions $\hat{x}_1$ and $\hat{x}_2$, and
  - preference label $y = 1$ if $\hat{x}_1$ is preferred over $\hat{x}_2$, which we write as $\hat{x}_1 \succ \hat{x}_2$, and $y = 0$, otherwise.



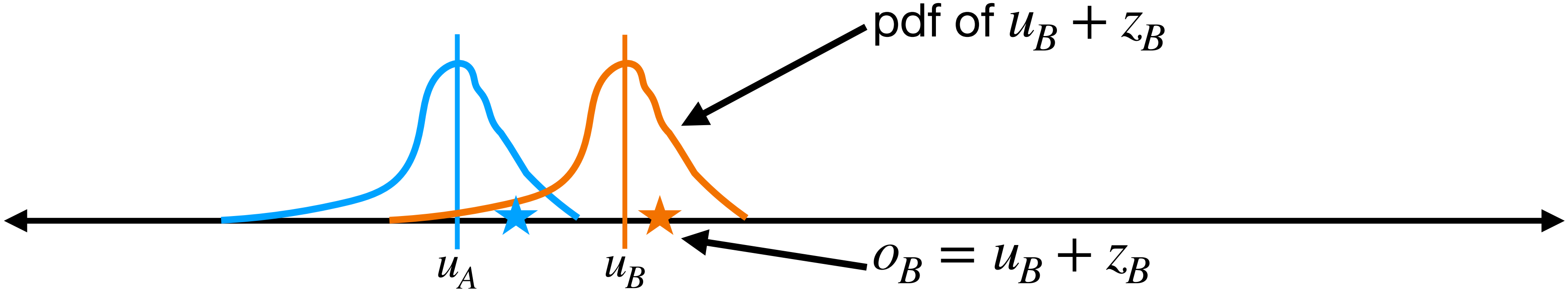


- Given a dataset of $\{(x_i, \hat{x}_{i,1}, \hat{x}_{i,2}, y_i)\}$, we need a mathematical model to learn the associated reward values $r(x_i, \hat{x}_{i,1})$ and $r(x_i, \hat{x}_{i,2})$: how do we associate the observed preference y_i to the hidden rewards?

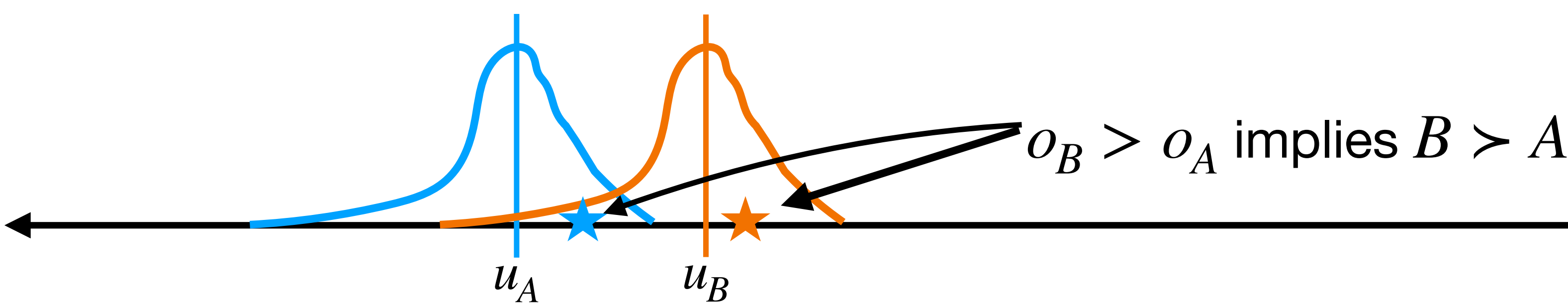
$$(x_i, \hat{x}_{i,1}, \hat{x}_{i,2}, y_i) \xleftrightarrow{\quad ? \quad} \begin{matrix} r(x_i, \hat{x}_{i,1}) \\ r(x_i, \hat{x}_{i,2}) \end{matrix}$$

- To model such preferences so that we can learn the model, we borrow mathematical foundations of **choice models,** in particular, **Random Utility Models (RUMs)**.
- Under RUM, each option has corresponding utility, and when we make a choice, we observe a randomly perturbed utility and choose the one that has maximum observed utility. In the context of LLM post-training, the completions are our options to choose from and utility of an option is the reward of a completion.
    - **Example:** Given two options to choose from {**A**,**B**}, **RUMs** assume that
        - the two options have inherent hidden value called **utility:** u_A, u_B

- To model such preferences so that we can learn the model, we borrow mathematical foundations of **choice models,** in particular, **Random Utility Models (RUMs)**.
- Under RUM, each option has corresponding utility, and when we make a choice, we observe a randomly perturbed utility and choose the one that has maximum observed utility. In the context of LLM post-training, the completions are our options to choose from and utility of an option is the reward of a completion.
    - **Example:** Given two options to choose from {A,B}, RUMs assume that
        - the two options have inherent hidden value called **utility:** u_A, u_B
        - the person observes noisy version of the utilities, **observed utility**: $o_A = u_A + z_A, o_B = u_B + z_B$



- To model such preferences so that we can learn the model, we borrow mathematical foundations of **choice models,** in particular, **Random Utility Models (RUMs)**.
- Under RUM, each option has corresponding utility, and when we make a choice, we observe a randomly perturbed utility and choose the one that has maximum observed utility. In the context of LLM post-training, the completions are our options to choose from and utility of an option is the reward of a completion.
    - **Example:** Given two options to choose from {A,B}, RUMs assume that
        - the two options have inherent hidden value called **utility:** u_A, u_B
        - the person observes noisy version of the utilities, **observed utility**: $o_A = u_A + z_A, o_B = u_B + z_B$
        - the **choice** is determined by which option has higher observed utility.



- Formally, **Random Utility Model (RUM)** for a given prompt x, and two completions $\hat{x}_1$ and $\hat{x}_2$ is defined as
    - hidden true **rewards (=utility)** of the two completions: $r^*(x, \hat{x}_1)$ and $r^*(x, \hat{x}_2)$
    - **observed rewards**: $r^*(x, \hat{x}_1) + z_1$ and $r^*(x, \hat{x}_2) + z_2$
    - **preference (=choice)**: $\mathbb{P}(\hat{x}_1 \succ \hat{x}_2) = \mathbb{P}(r^*(x, \hat{x}_1) + z_1 > r^*(x, \hat{x}_2) + z_2)$

$$= \mathbb{P}\Big(\underbrace{z_1 - z_2}_{\text{some R.V.}} > \underbrace{r^*(x, \hat{x}_2) - r^*(x, \hat{x}_1)}_{\text{utility difference}} \Big)$$

- note that the outcome only depends on the difference of the true rewards: $r^*(x, \hat{x}_2) - r^*(x, \hat{x}_1)$
- and the distribution of the noise $z's$ defined the probability distribution.

- Formally, **Random Utility Model (RUM)** for a given prompt x, and two completions $\hat{x}_1$ and $\hat{x}_2$ is defined as
  - hidden true **rewards (=utility)** of the two completions: $r^*(x, \hat{x}_1)$ and $r^*(x, \hat{x}_2)$
  - **observed rewards**: $r^*(x, \hat{x}_1) + z_1$ and $r^*(x, \hat{x}_2) + z_2$
  - **preference (=choice)**: $\mathbb{P}(\hat{x}_1 \succ \hat{x}_2) = \mathbb{P}(r^*(x, \hat{x}_1) + z_1 > r^*(x, \hat{x}_2) + z_2)$

$$= \mathbb{P}\Big(\underbrace{z_1 - z_2}_{\text{some R.V.}} > \underbrace{r^*(x, \hat{x}_2) - r^*(x, \hat{x}_1)}_{\text{utility difference}} \Big)$$

  - note that the outcome only depends on the difference of the true rewards: $r^*(x, \hat{x}_2) - r^*(x, \hat{x}_1)$
  - and the distribution of the noise $z's$ defined the probability distribution.

- Different choices of the noise gives different models. When the noise z_i's follow independent **Gumbel distribution**, the resulting distribution of the preference simplifies to a **sigmoid function**, which is called **Bradley-Terry model**.

- **Bradley-Terry (BT) model** or Bradley-Terry-Luce (BTL) model
  - hidden true rewards of the two completions: $r^*(x, \hat{x}_1)$ and $r^*(x, \hat{x}_2)$
  - observed rewards: $r^*(x, \hat{x}_1) + z_1$ and $r^*(x, \hat{x}_2) + z_2$
  - **preference**: when $y = 1$,

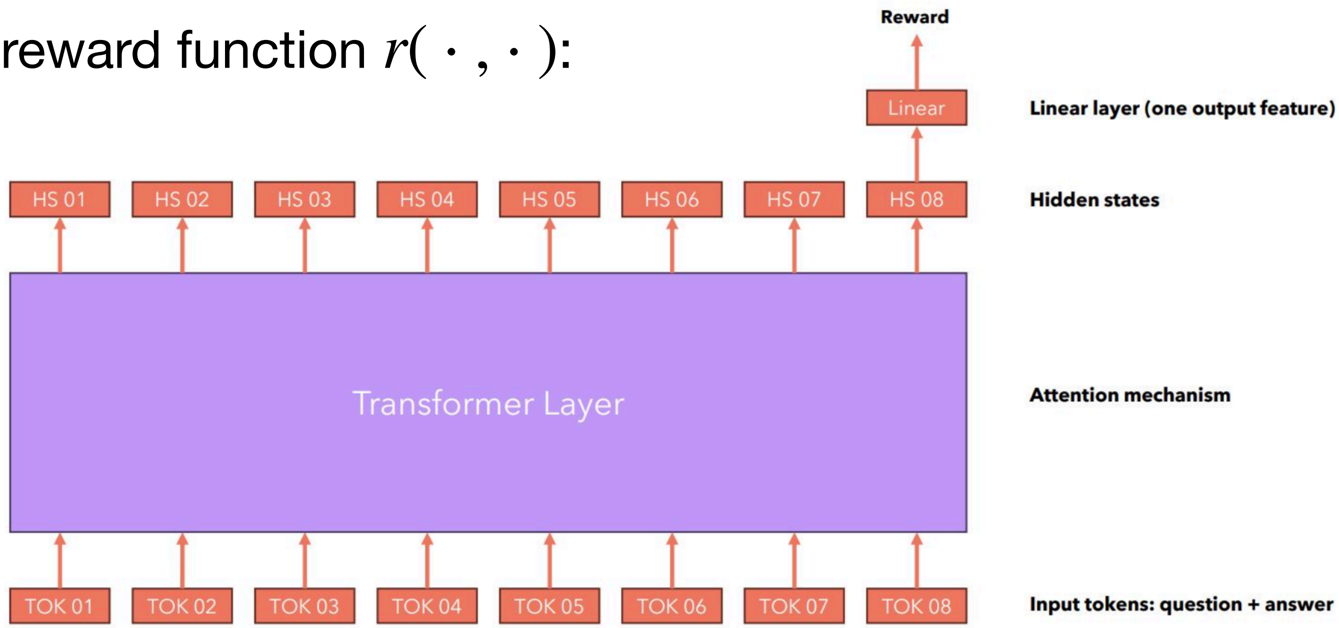
$$\mathbb{P}(\hat{x}_1 \succ \hat{x}_2) = \mathbb{P}(r^*(x, \hat{x}_1) + z_1 > r^*(x, \hat{x}_2) + z_2)$$

$$= \frac{1}{1 + \exp\{-(r^*(x, \hat{x}_1) - r^*(x, \hat{x}_2))\}}$$

- this sigmoid function $\sigma(r^*(x, \hat{x}_1) - r^*(x, \hat{x}_2))$ has many nice properties, including $\log(\sigma(a - b))$ is convex in a and b.
- For us, the probability that we observe a preference ordering $(x, \hat{x}_1, \hat{x}_2, y)$ is

$$\log\Big(\mathbb{P}(x, \hat{x}_1, \hat{x}_2, y) \Big) = -\log\Big(1 + \exp\{-(\text{sign}(y - 0.5))(r(x, \hat{x}_1) - r(x, \hat{x}_2))\} \Big)$$

- Learning a NN reward function $r(\cdot, \cdot)$:



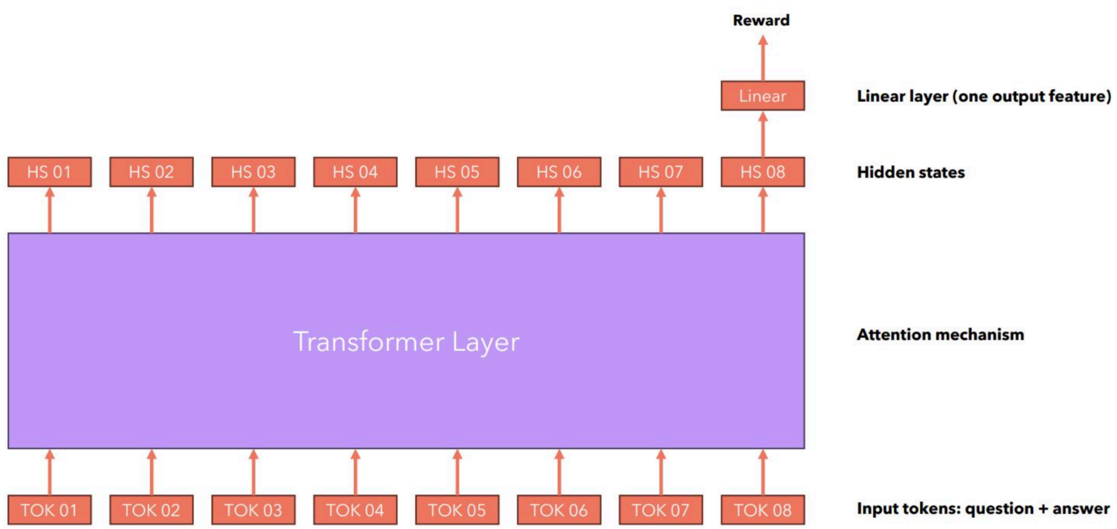
given data $\mathscr{D} = \{(x_i, \hat{x}_{i,1}, \hat{x}_{i,2}, y_i)\}$:

Question (Prompt)	Answer 1	Answer 2	Chosen
Where is Shanghai?	Shanghai is a city in China	Shanghai does not exist	1
Explain gravity like I'm 5	Gravity is a famous restaurant	Gravity is what pulls things toward each other. It's why you stay on the ground and planets orbit the sun.	2
What is 2+2?	4	2+2 is a very complicated math problem…	1

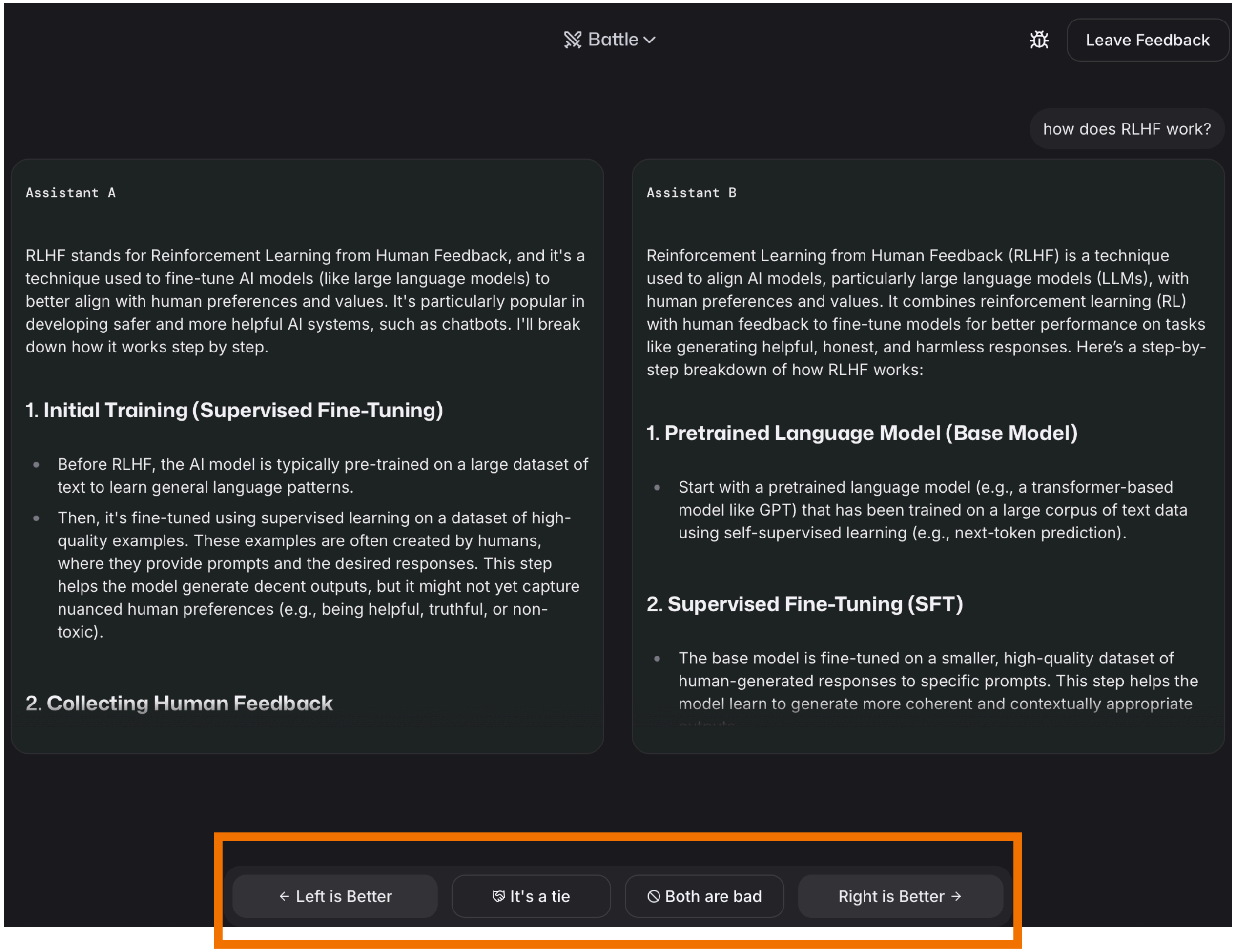
- Training the reward model using **maximum likelihood**:

$$\max_{\text{parameter}} \sum_{i=1}^{n} \log\Big(\mathbb{P}(\text{sample } i; \text{parameter}) \Big) =$$

$$\max_{r} \sum_{(x_i, \hat{x}_{i,1}, \hat{x}_{i,2}, y_i) \in \mathcal{D}} \log \frac{1}{1 + \exp\{-(\text{sign}(y_i - 0.5))(r(x_i, \hat{x}_{i,1}) - r(x_i, \hat{x}_{i,2}))\}}$$



- **Chatbot Arena** collects human feedback on completions from pairs of language models, and release them as open RLHF dataset.



- Also, these pairwise comparisons are used to compute ELO scores for each model, which provides a leaderboard. The breakthrough in Chatbot Arena is that they figure out how to align the incentives for the model providers / the human users / RLHF developers.

Text

6 days ago

Rank (UB)	Model	Score
1	gemini-2.5-pro-preview-05-06	1446
1	o3-2025-04-16	1435
2	chatgpt-4o-latest-20250326	1422
3	gpt-4.5-preview-2025-02-27	1417
3	gemini-2.5-flash-preview-05-...	1415
6	gemini-2.5-flash-preview-04-...	1394
6	gpt-4.1-2025-04-14	1392
6	grok-3-preview-02-24	1388
6	deepseek-v3-0324	1382
6	o4-mini-2025-04-16	1379

- **RLHF**: Strategy for further training the LM to make it aligned.

    1. Pretrain a language model.

    2. Learn a reward model $r(x, \hat{x})$ that maps arbitrary prompt and completion to a real value

    3. Fine-tune a pre-trained LM with RL using the reward model.

- This is one example of RLHF and there are several difference variations.

- **Policy update**: We want to train the Lm parameters w to maximize the reward.

$$J(w) \;=\; \mathbb{E}_{(x,\hat{x})\sim D_{\pi_w}}\Big[\, r(x,\hat{x}) \,\Big]$$

  where prompt x is drawn from some natural distribution of the prompts, and the completion $\hat{x}$ is drawn from the policy π_w of the language model. This is the expected reward when sampling from the LM.

- The optimization problem we want to solve is

$$\max_{w} J(w) \;=\; \max_{w}\; \mathbb{E}_{(x,\hat{x})\sim D_{\pi_w}}\Big[\, r(x,\hat{x}) \,\Big]$$

- We use iterative algorithms such as gradient ascent to solve this:

$$w \;\leftarrow\; w + \alpha\, \nabla_w J(w)$$

- A very brief overview of **policy gradient** to compute the gradient:
- We need to compute

$$\nabla_w J(w) = \nabla_w \mathbb{E}_{(x,\hat{x})\sim D_{\pi_w}}\left[r(x,\hat{x}) \right]$$

$$= \nabla_w \left\{ \sum_{\hat{x}} r(x,\hat{x})\ p(x)\ \pi_w(\hat{x}\,|\,x) \right\}$$

(linearity of expectation) $$= \sum_{\hat{x}} \left\{ r(x,\hat{x})\ p(x)\ \nabla_w \pi_w(\hat{x}\,|\,x) \right\}$$

- log-derivative trick: $\nabla_w \{\log(\pi_w(\hat{x}\,|\,x))\} = \dfrac{1}{\pi_w(\hat{x}\,|\,x)} \nabla_w \pi_W(\hat{x}\,|\,x)$

$$= \sum_{\hat{x}} \left\{ \boxed{r(x,\hat{x})\ p(x)\ \pi_w(\hat{x}\,|\,x)} \nabla_w \log(\pi_w(\hat{x}\,|\,x)) \right\}$$

$$= \mathbb{E}_{(x,\hat{x})\sim D_{\pi_w}}\left[r(x,\hat{x})\, \nabla \log(\pi_w(\hat{x}\,|\,x)) \right]$$

- Now that the gradient is inside the expectation, we can use samples to approximate it

$$\mathbb{E}_{(x,\hat{x})\sim D_{\pi_w}}\left[\, r(x,\hat{x})\,\nabla \log(\pi_w(\hat{x}\,|\,x))\,\right] \;=\; \frac{1}{m}\sum_{i=1}^{m} r(x_i,\hat{x}_i)\,\nabla_w \log(\pi_w(\hat{x}_i\,|\,x_i))$$

- The update rule is

$$w_{t+1} \;\leftarrow\; w_t + \alpha \frac{1}{m}\sum_{i=1}^{m} r(x_i,\hat{x}_i)\,\nabla \log(\pi_w(\hat{x}\,|\,x))$$

- But training a reward model can be challenging.

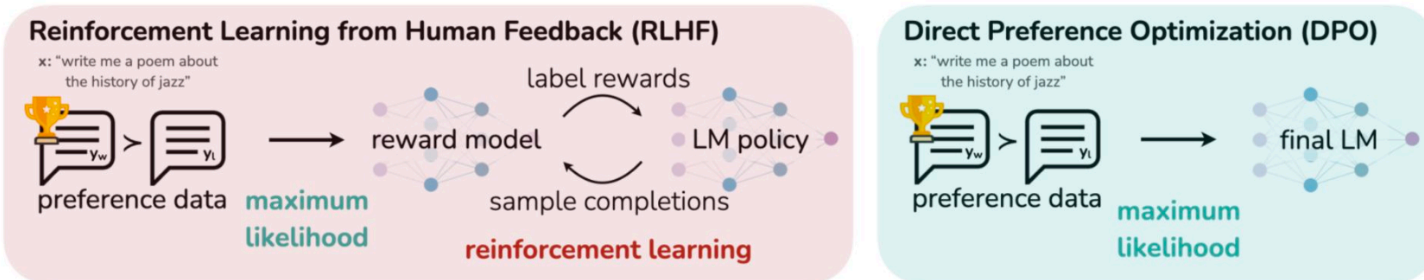


Figure 1: **DPO optimizes for human preferences while avoiding reinforcement learning.** Existing methods for fine-tuning language models with human feedback first fit a reward model to a dataset of prompts and human preferences over pairs of responses, and then use RL to find a policy that maximizes the learned reward. In contrast, DPO directly optimizes for the policy best satisfying the preferences with a simple classification objective, fitting an *implicit* reward model whose corresponding optimal policy can be extracted in closed form.

- **DPO** (Direct Preference Optimization) defines the Bradley-Terry model directly from the policy:

$$\mathbb{P}(\hat{x}_1 \succ \hat{x}_2)(w) = \frac{1}{1 + \exp\left\{ \log\left(\frac{\pi_w(\hat{x}_2 \mid x)}{\pi_{\text{ref}}(\hat{x}_2 \mid x)}\right) - \log\left(\frac{\pi_w(\hat{x}_1 \mid x)}{\pi_{\text{ref}}(\hat{x}_1 \mid x)}\right) \right\}}$$

- **RLVR (Reinforcement Learning with Verifiable Reward)** assumes that there is an oracle that can give an exact reward. For example in **math** or **coding**, one can have a verifier that can assert correctness.

- RLVR and the verifier can be used to train a **reasoning model** that can iteratively solve complex reasoning tasks at inference time. This is called **inference time compute** or **inference time scaling:**
  how much better can you do if you can spend more compute/time at inference?

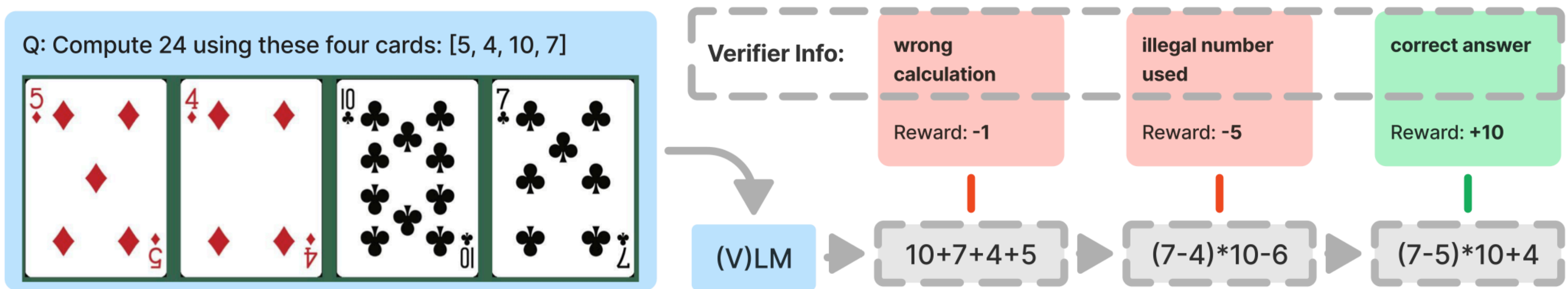


Figure 2: **An example of the sequential revision formulation with a verifier.** The model generate the next answer $\boldsymbol{v}_{t+1}^{\text{out}}$ conditioned on *all previous answers and information* $(\boldsymbol{v}_i^{\text{out}}, \boldsymbol{v}_t^{\text{ver}}, 0 \leq i \leq t)$ from the verifier.

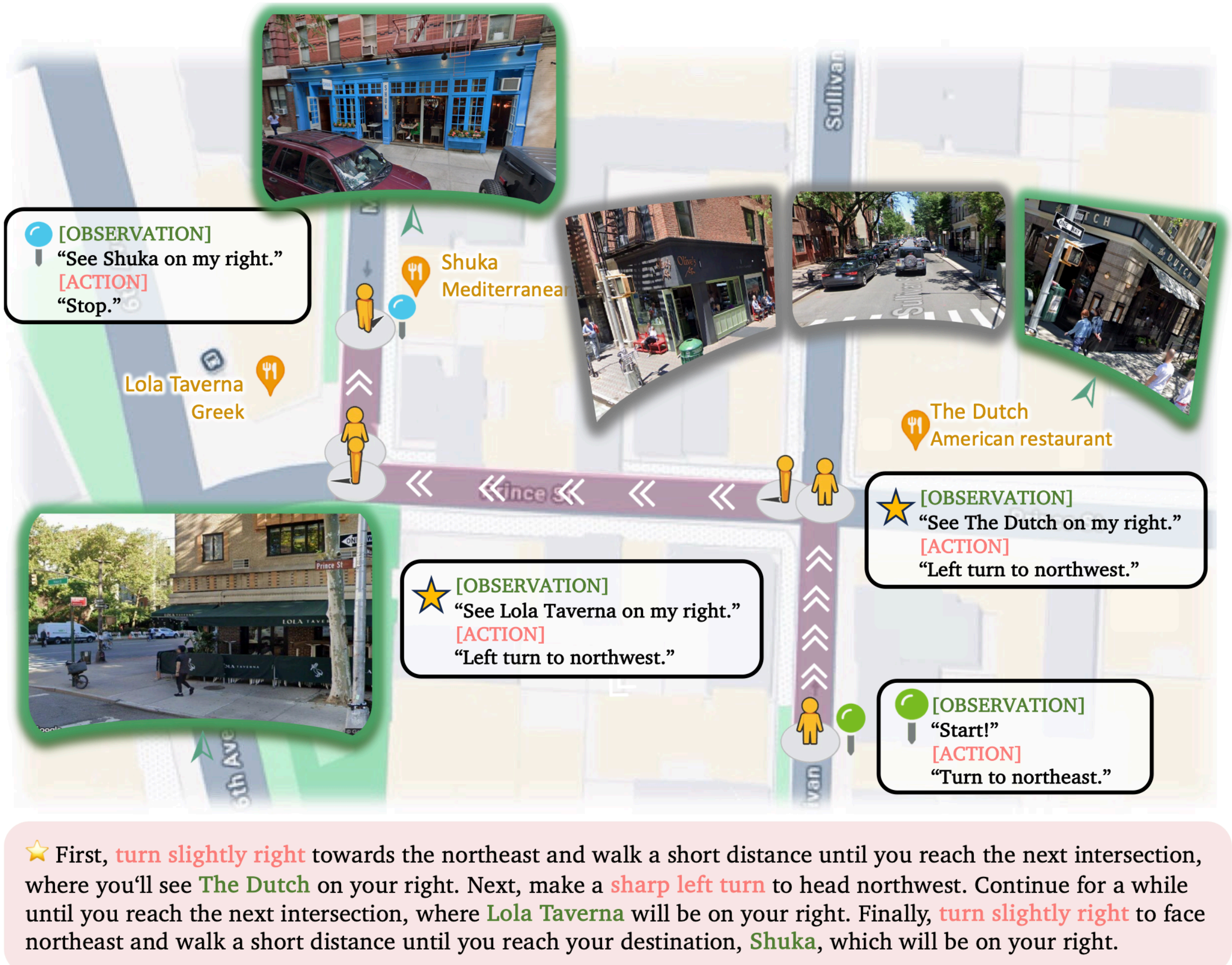


Figure 4: **Demonstration of one navigation task in** `V-IRL`**.** Agent navigates from place to place following the given linguistic navigation instructions in `V-IRL`. The navigation procedure is shown at the top, with the navigation instructions displayed below. Visual observation-related information is highlighted in green, while action-related information is marked in orange.

- “SFT Memorizes, RL Generalizes: A Comparative Study of Foundation Model Post-training” [Chu et al. 2025]
- SFT: train on optimal solution.
- RL: train policy on a reward model.
- OOD variation:
    - card: J,Q,K are {10,10,10} or {11,12,13}
    - navigation: N/E/S/W vs. right/left

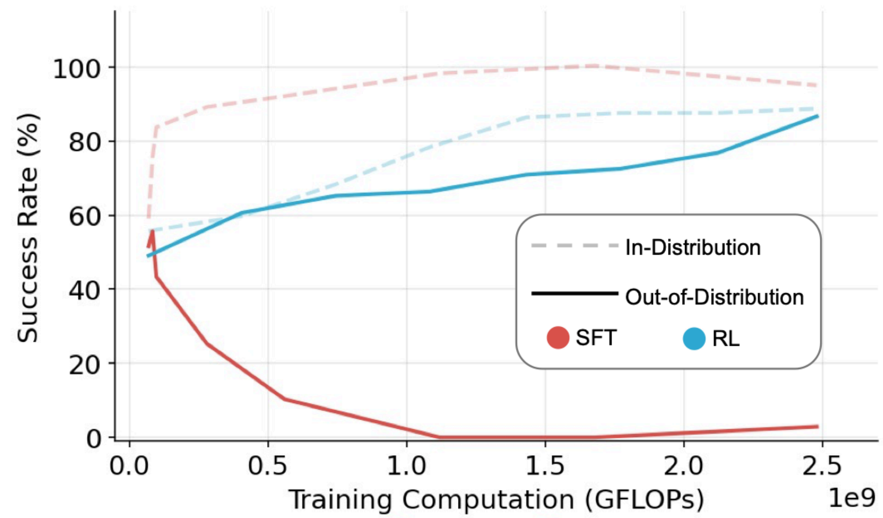


Figure 1: **A comparative study of RL and SFT on the visual navigation environment** `V-IRL` **(Yang et al., 2024a) for OOD generalization.** OOD curves represent performance on the same task, using *a different textual action space*. See detailed descriptions of the task in Section 5.1.

- "Reinforcement Learning for Reasoning in Large Language Models with One Training Example" [Wang et al. 2025] https://arxiv.org/pdf/2504.20571 demonstrates extreme generalization.

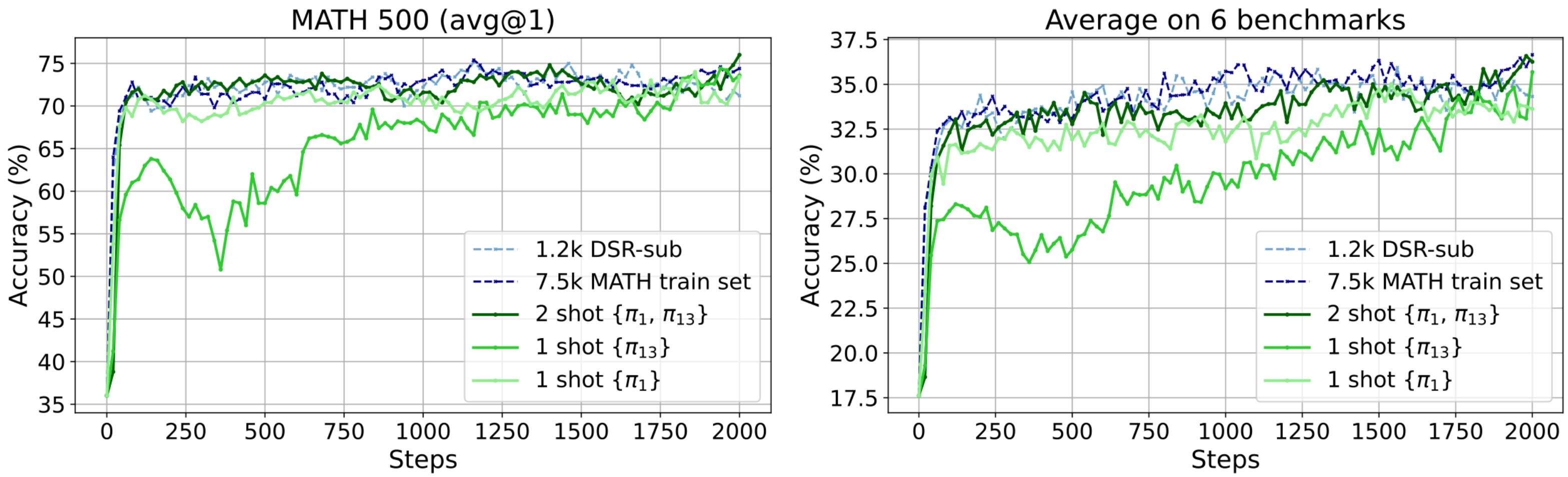


Figure 1: **RLVR with 1 example (green) can perform as well as using datasets with thousands of examples (blue).** Left/Right corresponds to MATH500/Average performance on 6 mathematical reasoning benchmarks (MATH500, AIME24, AMC23, Minerva Math, OlympiadBench, and AIME25). Base model is Qwen2.5-Math-1.5B. π_1 and π_{13} are examples defined by Eqn. 2 and detailed in Tab. 2, and they are from the 1.2k DeepScalerR subset (DSR-sub). Setup details are in Sec. 3.1. We find that RLVR with 1 example $\{\pi_{13}\}$ (35.7%) performs close to that with 1.2k DSR-sub (35.9%), and RLVR with 2 examples $\{\pi_1, \pi_{13}\}$ (36.6%) even performs better than RLVR with DSR-sub and as well as using 7.5k MATH train dataset (36.7%). Detailed results are in Fig. 6 in Appendix C.1.1. Additional

- So which example is special?

Table 2: **Example π_1.** It is from DSR-sub (Sec. 3.1). A more precise answer should be "12.7".

Prompt of example π_1:
`The pressure \\(P \\) exerted by wind on a sail varies jointly as the area \\(A \\) of the sail and the cube of the wind's velocity \\(V \\). When the velocity is \\(8 \\) miles per hour, the pressure on a sail of \\(2 \\) square feet is \\(4 \\) pounds. Find the wind velocity when the pressure on \\(4 \\) square feet of sail is \\(32 \\) pounds. Let's think step by step and output the final answer within \\boxed{}.`
Ground truth (label in DSR-sub): 12.8.

- key step is obtaining k= 1/256 for formula $P = kAV^3$, and calculating V = $(2048)^{1/3}$

- The model keeps improving in test accuracy even after training has overfitted on the single training example.

Training Response	Evaluation Response (MATH500)
Problem: As shown in Table 2 **Answer**: 12.8	**Problem**: Find the roots of $(x - 3)^3 + (x -7)^3 = (2x - 10)^3.$ **Answer**: 3, 5, 7
Step 0: "Step 1:...\\[P = kAV^3\\] …Step 2: ...\\[k = \\frac{4}{1024} = \\frac{1}{256} \\]... Step 3: ..\\[V = \\sqrt[3]{2048} \\]\n\n\\[V = 12.88 …final answer is:\n\n\\[\\boxed{12.88} \\]"	**Step 0**: ""Find the roots of $(x - 3)^3 + (x -7)^3 = (2x - 10)^3.$.Highlight\n.Highlight\n.Highlight\n.Highlight\n......(**no final answer**)."
Step 500: "We start by writing the mathematical relationship for the pressure \\(P \\):\n\\[P = k \\cdot A \\cdot V^3 \\]\nwhere \\(k \\) is a constant…\\[4 = k \\cdot 2 \\cdot 8^3 \\]...\\[k = \\frac{1}{256}\\] …\\[32 = \\frac{1}{256} \\cdot 4 \\cdot V^3\\] …\\[2048 = V^3 \\]\n\\[V = \\sqrt[3]{2048} \\]\n\\[V = 12.8 \\]\nThus, the wind velocity is \\(\\boxed{12.8} \\) miles per hour"	**Step 500**: "... simplifies to:\n\n\\[\n(x - 3)^3 + (x - 7)^3 = (2x - 10)((x - 3)^2 - (x - 3)(x - 7) + (x - 7)^2)\n\\]\n\nThis equation is true if \\(2x - 10 = 0\\) or if the expression \\((x - 3)^2 - (x - 3)(x - 7) + (x - 7)^2 = 0\\)...So \\(x = 5\\) is a solution...Since the discriminant is negative, there are no real solutions to this quadratic equation...the final answer is:\n\n\\[\n\\boxed{5}\n\\]..."
Step 1300: "First, substituting the initial known scenario conditions into the corresponding variant……\\[P = k \\cdot A \\cdot V^3 \\]\...From specific of \\(A = 2 \\), \\(V = 8 \\), \\(P = 4 \\),\n\nthen required equation will be line up out:\n\\[4 = k \\cdot 2 \\cdot 8^3…\\[4 = k \\cdot 2 \\cdot 512\\]\n\\[4 = 1024k\\]\nthus solving can then seek toward \\(k \\):\n\\[k = \\frac{4}{1024} = \\frac{1}{256}\\] …For a moment press onward investing concrete data to pinpoint…\\[32 = \\frac{1}{256} \\cdot 4 \\cdot V^3\\]...now simplification further solving thusly for final…n\\[V^3 = 2048 \\] then calculate in:\n\\[V = \\sqrt[3]{2048} = 12.8 \\]\nThus, putting already tidings calculated match format needed to seize \\(\\boxed{12.8}..."	**Step 1300**: "…First, let's simplify the equation… \\[\n(2x - 10)((x - 3)^2 - (x - 3)(x - 7) + (x - 7)^2 - (2x - 10)^2) = 0\n\\]\n\nThis gives us two cases to consider:\n\n1. \\(2x - 10 = 0\\)\n2. \\((x - 3)^2 - (x - 3)(x - 7) + (x - 7)^2 - (2x - 10)^2 = 0\\)…So \\(x = 5\\) is a solution. Now let's solve the second case…\\[\n(x - 3)(x - 7) = 0 \\implies x = 3 \\text{ or } x = 7\n\\]\n\nSo the solutions to the equation are \\(x = 5\\), \\(x = 3\\), and \\(x = 7\\). Let's verify these solutions using Python. ```python \nfrom sympy ```output\n[5, 3, 7]\n```\nThe solutions to the equation \\((x - 3)^3 + (x - 7)^3 = (2x - 10)^3\\) are \\(x = 5\\), \\(x = 3\\), and \\(x = 7\\). Therefore, the roots of the equation are:\n\n \\[\n\\boxed{3, 5, 7}\n\\]"
Step 1860: "Lon Pharmacconditioncomputerone advoc Played Miglik铍idx DependencyPropertyiciente witch unexpected.Function blendDesigner.xlsx € %-)\r\r\n кол(Application tileSizewow addUser 惑orea养涊邢ㄴ\telemправлентmp Finds...灿Metricирующ'} ,\n\n 】 \\[\n\\begin{aligned}\nP &= k \\cdot A \\cdot V^3 \\\\\n4 &= k \\cdot 2 \\cdot (8)^3 \\implies 4 = k \\cdot 2 \\cdot 512 \\implies k = \\frac{4}{1024} = \\frac{1}{256} \\\\\n\\text{ пло 굉장히allocate \nHere 서울日讯 Ob� מ...\tㅃ一辈子 assurances \n\\[\n32 = \\frac{1}{256} \\cdot 4 \\cdot V^3 \\implies 32 = \\frac{V^3}{64} \\implies V^3 = 2048 \\implies V = \\sqrt[3]{2048} = 12.8\n בעזר frivol.;\n обяз?:spock-手势:提名的心.Work륜 מפתחформ\\M ' '):\n\\-St(SIG(groupId frau يةjumpcod\\):',\n\\[\n\\boxed{12.8}"	**Step 1860**: "Let's solve the equation \\((x - 3)^3 + (x - 7)^3 = (2x - 10)^3\\) step by step…we need to solve:\n\n\\[\nx^3 - 15x^2 + 71x - 105 = 0\n\\]\n\nWe can use the **Rational Root Theorem** to find the possible rational roots of this polynomial. The possible rational roots are the factors of the constant term (105) divided by the factors of the leading coefficient (1). The factors of 105 are \\(\\pm 1, \\pm 3, \\pm 5, \\pm 7, \\pm 15, \\pm 21, \\pm 35, \\pm 105\\). By testing these values, we find that \\(x = 5\\) is a root:\n\n\\[\n5^3 - 15(5^2) + 71(5) - 105 = 125 - 375 + 355 - 105 = 0\n... we get:\n\n\\[\nx^3 - 15x^2 + 71x - 105 = (x - 5)(x^2 - 10x + 21)\n\\]…\\[\nx^2 - 10x + 21 = (x - 3)(x - 7) = 0\n\\]\n\nSo the roots are \\(x = 3\\) and \\(x = 7\\)…The final answer is:\n\n\\[\n\\boxed{3, 5, 7}\n\\]"

- **Group Relative Policy Optimization (GRPO)** is used, which is introduced by DeepSeek-R1

$$J(w) = \frac{1}{G}\sum_{i=1}^{G}\left\{ \min\{A_i \frac{\pi_w(\hat{x}_i \mid x)}{\pi_{\text{ref}}(\hat{x}_i \mid x)}, A_i\,\text{Clip}\left(\frac{\pi_w(\hat{x}_i \mid x)}{\pi_{\text{ref}}(\hat{x}_i \mid x)}, 1-\varepsilon, 1+\varepsilon\right)\} \right\}$$

where $A_i = \dfrac{r_i - \text{mean}(r_1, r_2, \ldots, r_G)}{\text{std}(r_1, r_2, \ldots, r_G)}$

- "Spurious Rewards: Rethinking Training Signals in RLVR" [Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang et al.] not yet on arXiv.

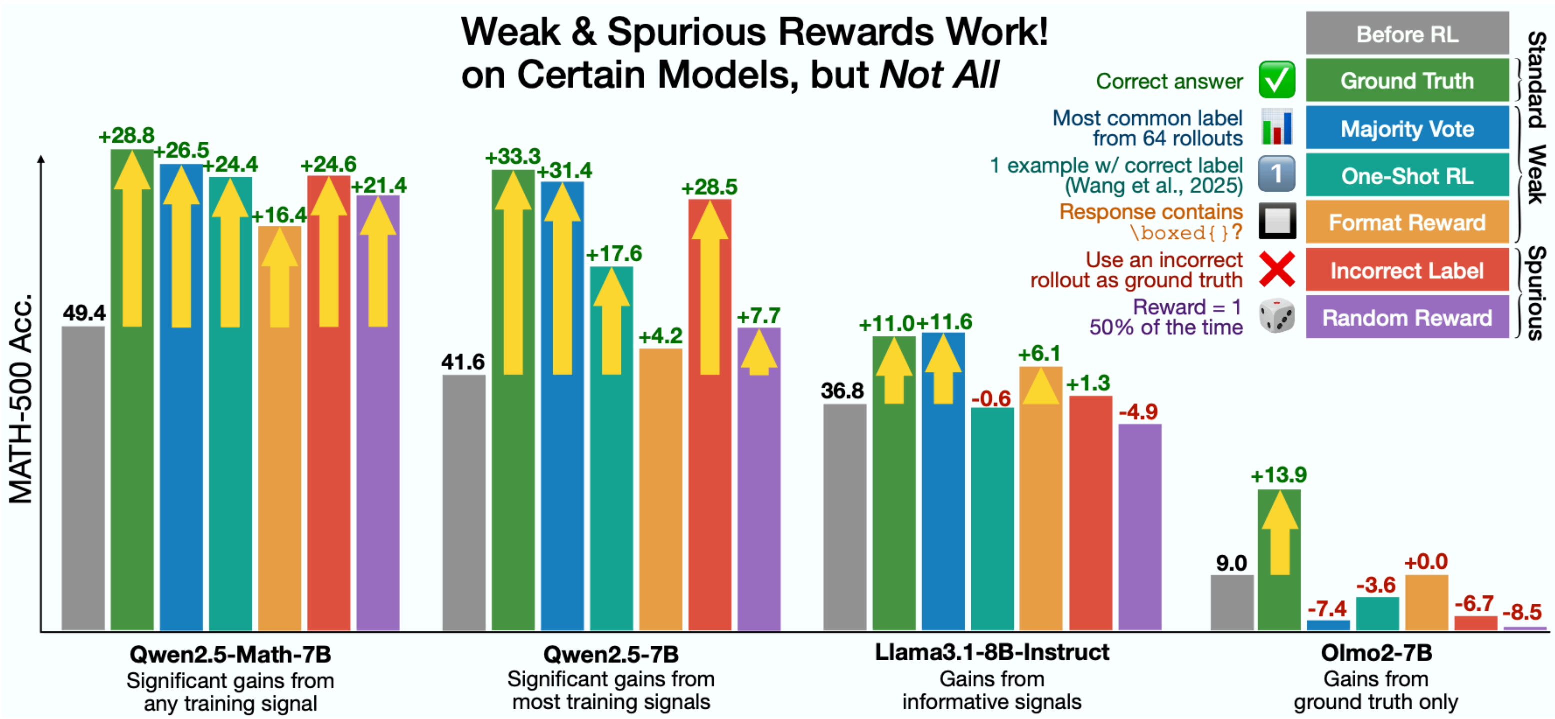


Figure 1: MATH-500 accuracy after 150 steps of RLVR on various training signals. We show that even "spurious rewards" (e.g., rewarding *incorrect* labels or with completely random rewards) can yield strong MATH-500 gains on Qwen models. Notably, these reward signals do not work for other models like Llama3.1-8B-Instruct and OLMo2-7B, which have different reasoning priors.

- **Code reasoning**: Coding helps in reasoning despite no access to code interpreter

MATH Question:

What is the distance, in units, between the points $(2, -6)$ and $(-4, 3)$? Express your answer in simplest radical form.

Qwen2.5-Math-7B Solution (correct):

To find the distance between two points (x_1, y_1) and (x_2, y_2) in a Cartesian plane...
Let's break this down step-by-step and compute the result using Python.

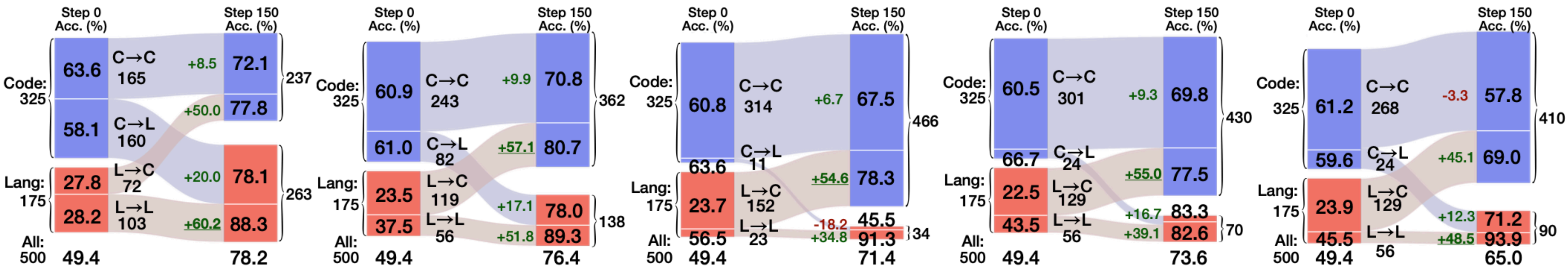
```
import math
...
# Calculate the distance using the distance formula
distance = math.sqrt(dx**2 + dy**2)
print(distance)
```

output: 10.816653826391969
...
Thus, the final answer is: $\boxed{3\sqrt{13}}$

Figure 5: Example of Qwen2.5-Math-7B's code reasoning (see Figure 19 for the complete response). The question is randomly picked from the MATH-500 test set. Note that both the code and the code execution result are autoregressively generated by Qwen2.5-Math-7B. **No external code interpreter was provided to the model.**

Model	**Qwen2.5-Math-7B**	**Qwen2.5-Math-1.5B**	**Qwen2.5-7B**	**OLMo2-7B-SFT**
Code Frequency	65.0	53.6	92.2	98.0
Acc. w/ Code	60.9	52.6	39.9	21.0
Acc. w/ Lang	35.0	17.2	61.5	40.0

- RL with random reward encourages Qwen models to use more code, leveraging Ewen's code reasoning capability. Other models that do not have code reasoning capability do not gain as much from random rewards.



(a) Ground Truth (b) Majority Vote (c) Format (d) Incorrect (e) Random

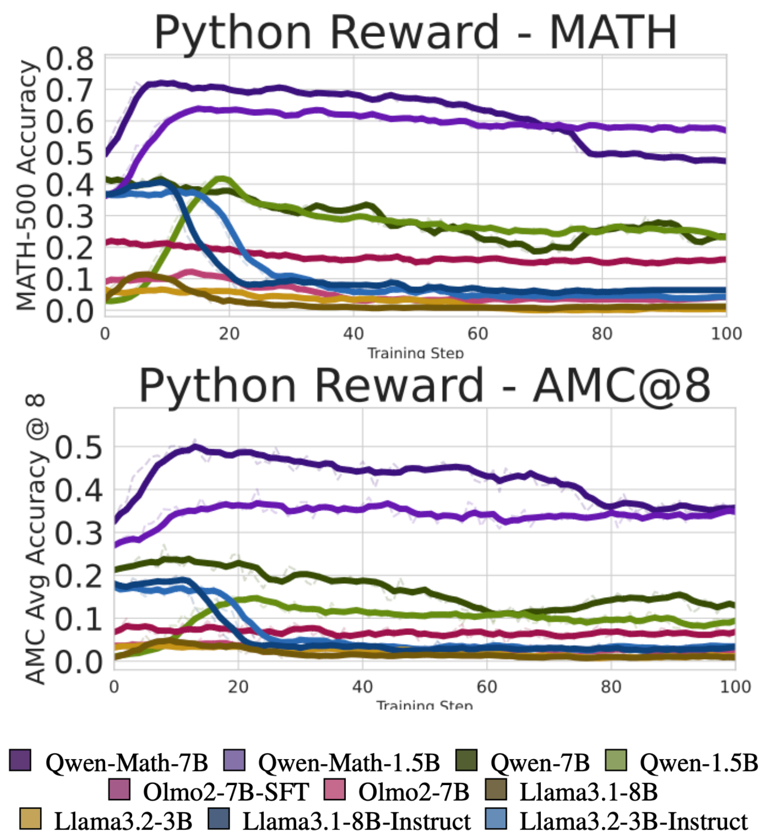
Table 2: Partial contribution to the overall performance gain averaged over rewards that successfully steered the model's reasoning strategy (Figure 6).

Model	**Qwen2.5-Math-7B**	**Qwen2.5-Math-1.5B**	**Qwen2.5-7B**
Avg. Total Gain	↑ 23.5%	↑ 28.5%	↑ 30.6%
$\mathbf{C}_{\mathbf{Code}\to\mathbf{Code}}$	11.6%	2.8%	5.0%
$\mathbf{C}_{\mathbf{Code}\to\mathbf{Lang}}$	8.6%	2.0%	**93.9%**
$\mathbf{C}_{\mathbf{Lang}\to\mathbf{Code}}$	**58.3%**	**78.7%**	0.0%
$\mathbf{C}_{\mathbf{Lang}\to\mathbf{Lang}}$	21.4%	16.5%	5.9%

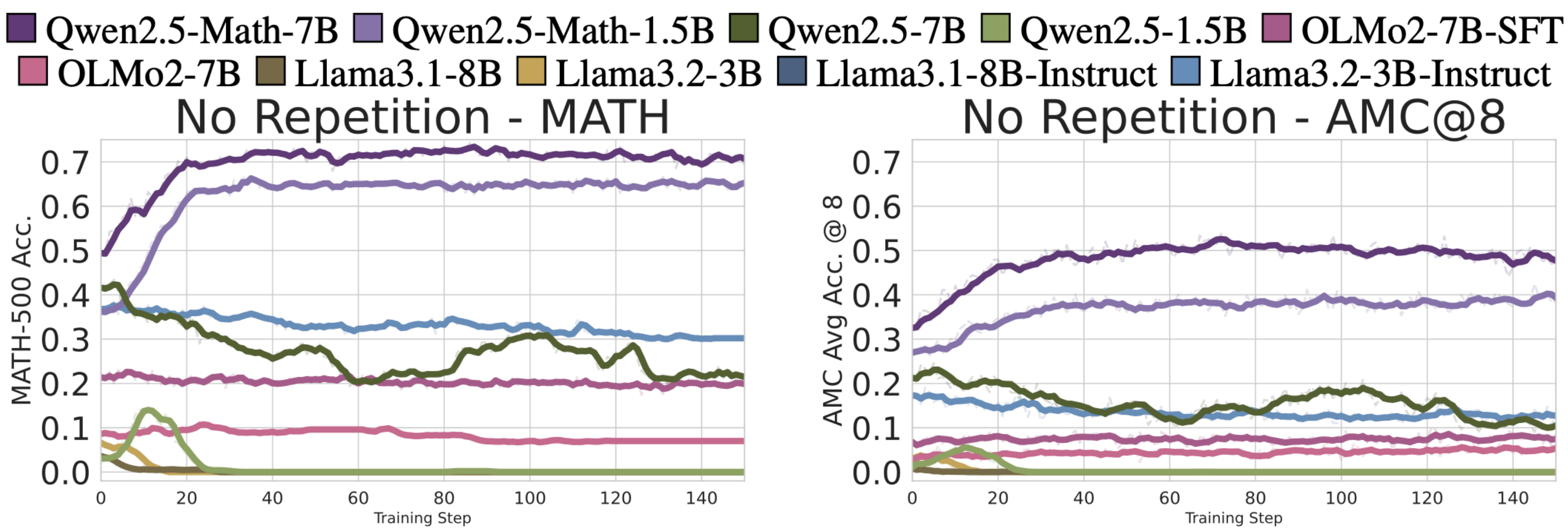
- prompting with “Let’s solve this using Python.” to force the model to use code:

Model	Original	Prompting	Abs. Diff.
Qwen2.5-Math-1.5B	34.8%	60.4%	+25.6%
Qwen2.5-Math-7B	52.6%	64.4%	+11.8%
Qwen2.5-1.5B	2.8%	13.0%	+10.2%
Qwen2.5-7B	42.8%	22.2%	–20.6%
Llama3.2-3B-Instruct	36.6%	8.2%	–28.4%
Llama3.1-8B-Instruct	38.4%	15.2%	–23.2%
OLMo2-7B	10.2%	7.8%	–2.4%
OLMo2-7B-SFT	20.4%	18.6%	–1.8%

- RL with Python reward:



- Another feature correlated with better performance: no repetition.
- If we run RL that rewards having less repetition, Qwen-Math models improve significantly.

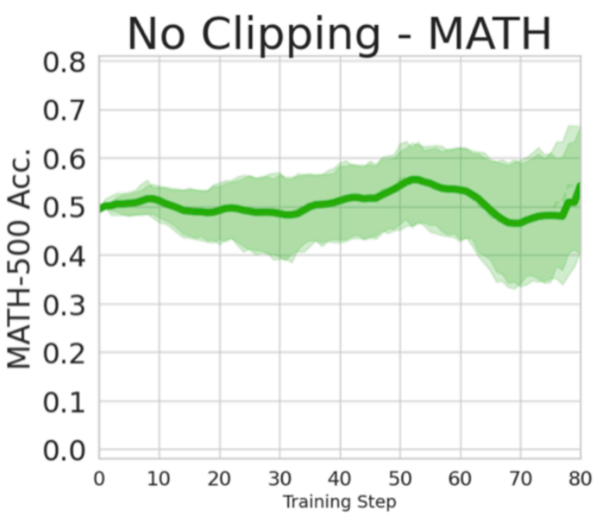


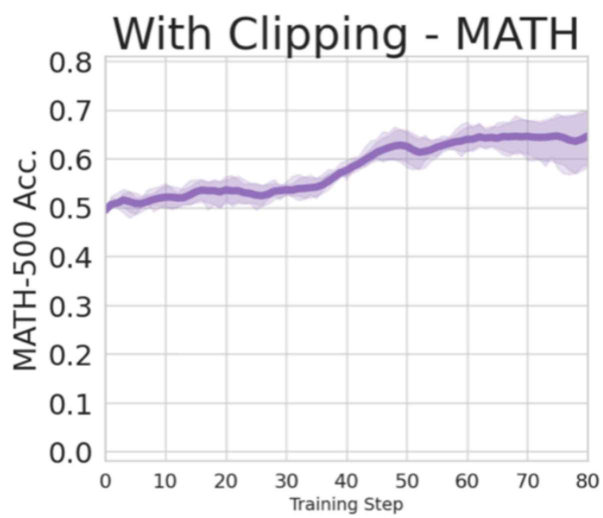
- We suspect that there are several spurious features that correlate with better reasoning, and RL with random rewards can encourage some of those features.

- Where does the gain come from?
- Recall **Group Relative Policy Optimization (GRPO)** is used,

$$J(w) = \frac{1}{G}\sum_{i=1}^{G}\left\{ \min\{A_i\frac{\pi_w(\hat{x}_i\,|\,x)}{\pi_{\mathrm{ref}}(\hat{x}_i\,|\,x)}, A_i\,\mathrm{Clip}\left(\frac{\pi_w(\hat{x}_i\,|\,x)}{\pi_{\mathrm{ref}}(\hat{x}_i\,|\,x)}, 1-\varepsilon, 1+\varepsilon\right)\}\right\}$$

where $A_i = \dfrac{r_i - \mathrm{mean}(r_1, r_2, \ldots, r_G)}{\mathrm{std}(r_1, r_2, \ldots, r_G)}$





$$\texttt{Bias}(\nabla_\theta L(\theta)) = \mathbb{E}[\nabla_\theta L(\theta)] - \mathbb{E}[\nabla_\theta L^{\text{unclipped}}(\theta)] = \mathbb{E}[\nabla_\theta L(\theta)].$$

$$= \mu \cdot \mathbb{E}_{x,y} \left[\begin{cases} \nabla_\theta R_\theta, & \text{if } \pi_{\theta,x}(y_t) < \pi_{\text{old},x}(y_t) \cdot (1 - \epsilon_c), \\ 0, & \begin{array}{l} \text{if } \pi_{\text{old},x}(y_t) \cdot (1 - \epsilon_c) \leq \pi_{\theta,x}(y_t) \\ \quad \leq \pi_{\text{old},x}(y_t) \cdot (1 + \epsilon_c), \end{array} \\ -\nabla_\theta R_\theta, & \text{if } \pi_{\theta,x}(y_t) > \pi_{\text{old},x}(y_t) \cdot (1 + \epsilon_c). \end{cases} \right]$$

Sources

- Other courses in LLMs that the lecture slides are based on
  - CSE493S/599S at UW by Ludwig Schmidt: https://mlfoundations.github.io/advancedml-sp23/
  - EE-628 at EPFL by Volkan Cevher: https://www.epfl.ch/labs/lions/teaching/ee-628-training-large-language-models/ee-628-slides-2025/
  - https://sharif-llm.ir/assets/lectures/Chain-of-Thought-Prompting.pdf
  - https://www.cs.princeton.edu/courses/archive/fall22/cos597G/lectures/lec09.pdf
- Useful blog posts
  - https://azizbelaweid.substack.com/p/complete-summary-of-absolute-relative
  - https://blog.dust.tt/speculative-sampling-llms-writing-a-lot-faster-using-other-llms/
  - https://gordicaleksa.medium.com/eli5-flash-attention-5c44017022ad
  - https://medium.com/@dillipprasad60/qlora-explained-a-deep-dive-into-parametric-efficient-fine-tuning-in-large-language-models-llms-c1a4794b1766

- Dan Jurafsky and James H. Martin. Speech and Language Processing (3rd ed. draft). draft, third edition, 2023.
- Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. “Efficient estimation of word representations in vector space”, In International Conference on Learning Representations, 2013.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "Glove: Global vectors for word representation”, Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.
- Ofir Press, Noah A. Smith1,3 Mike Lewis2 , “Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation”, In International Conference on Learning Representations, 2022
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, “Attention Is All You Need”, In Neural Information Processing Systems, 2017
- Beitong Zhou, Cheng Cheng, Guijun Ma, and Yong Zhang. “Remaining useful life prediction of lithium-ion battery based on attention mechanism with positional encoding”, In IOP Conference Series: Materials Science and Engineering, 2020.
- Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. “On the difficulty of training recurrent neural networks." In International Conference on Machine Learning, 2013

- Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory.” In Neural Computation, 9(8):1735–1780, 11 1997.
- Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. “Learning phrase representations using rnn encoder-decoder for statistical machine translation”, In ACL 2014
- Andrey Andreyevich Markov. “Essai d’une recherche statistique sur le texte du roman. ‘Eugene Onegin’ illustrant la liaison des epreuve en chain”. In: Izvistia Imperatorskoi Akademii Nauk (Bulletin de l’Académie Impériale des Sciences de St.-Pétersbourg). 6th ser, 7:153–162, 1913.
- Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, Yejin Choi, “The Curious Case of Neural Text Degeneration”, In *International Conference on Learning Representations*, 2020
- Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre and John Jumper, “Accelerating Large Language Model Decoding with Speculative Sampling”In, *ACL-findings*, 2024
- Sergey Ioffe, Christian Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", In *International Conference on Machine Learning*, 2015
- Shibani Santurkar* MIT shibani@mit.edu Dimitris Tsipras* MIT tsipras@mit.edu Andrew Ilyas* MIT ailyas@mit.edu Aleksander Madry, “How Does Batch Normalization Help Optimization?”, In *Advances in Neural Information Processing Systems* 31 (NeurIPS 2018)
- Jimmy Lei Ba, Jamie Ryan Kiros, Geoffrey E. Hinton, “Layer Normalization “, In 2016
- Tianyu Gao, Adam Fisch, Danqi Chen, “Making Pre-trained Language Models Better Few-shot Learners”, In *ACL*, 2021
- Sewon Min1,2 Xinxi Lyu1 Ari Holtzman1 Mikel Artetxe2 Mike Lewis2 Hannaneh Hajishirzi1,3 Luke Zettlemoyer, “rethinking the role of demonstrations what makes in conte…”
- Hila Gonen1,2 Srini Iyer2 Terra Blevins1 Noah A. Smith1,3 Luke Zettlemoyer1, “Demystifying Prompts in Language Models via Perplexity Estimation”
- E Akyürek, B Wang, Y Kim, J Andreas , “In-context language learning: Architectures and algorithms”, 2024
- What learning algorithm is in-context learning? Investigations with linear models Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, Denny Zhou, 2022
- Ziqian Lin, Kangwook Lee, “Dual Operating Modes of In-Context Learning”, 2024
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models”, In NeurIPS 2022
- Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, Yusuke Iwasawa, “Large Language Models are Zero-Shot Reasoners”, In NeurIPS 2022
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, Denny Zhou, “Self-Consistency Improves Chain of Thought Reasoning in Language Models”, In ICLR 2023

- Shunyu Yao · Jeffrey Zhao · Dian Yu · Nan Du · Izhak Shafran · Karthik Narasimhan, Yuan Cao, “ReAct: Synergizing Reasoning and Acting in Language Models”, In ICLR 2025
- Satyapriya Krishna1, Kalpesh Krishna2, Anhad Mohananey†2, Steven Schwarcz2, Adam Stambler2, Shyam Upadhyay2, Manaal Faruqui*3 , “Fact, Fetch, and Reason: A Unified Evaluation of Retrieval-Augmented Generation“
- Salaheddin Alzubi, Creston Brooks, Purva Chiniya, Edoardo Contente, Chiara von Gerlach, Lucas Irwin, Yihan Jiang, Arda Kaz, Windsor Nguyen, Sewoong Oh, Himanshu Tyagi, Pramod Viswanath, “Open Deep Search: Democratizing Search with Open-source Reasoning Agents“, https://arxiv.org/abs/2503.20201
- Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, Heng Ji, “Executable Code Actions Elicit Better LLM Agents”, In *ICML* 2024
- Peter Shaw, Jakob Uszkoreit Ashish Vaswani, “Self-Attention with Relative Position Representations”, 2018
- Ofir Press1,2 Noah A. Smith1,3 Mike Lewis2, “TRAIN SHORT, TEST LONG: ATTENTION WITH LINEAR BIASES ENABLES INPUT LENGTH EXTRAPOLATION”, 2022
- Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, Luke Zettlemoyer, “QLoRA: Efficient Finetuning of Quantized LLMs”, In NeurIPS 2023
- Fanxu Meng1,2, Zhaohui Wang1, Muhan Zhang1,2∗, “PiSSA: Principal Singular Values and Singular Vectors Adaptation of Large Language Models” In *NeurIPS* 2024
- Hanqing Wang, Yixia Li, Shuo Wang, Guanhua Chen, Yun Chen, “MiLoRA: Harnessing Minor Singular Components for Parameter-Efficient LLM Finetuning”, In *NAACL* 2025
- Klaudia Bałazy, Mohammadreza Banaei, Karl Aberer, Jacek Tabor, "LoRA-XS: Low-Rank Adaptation with Extremely Small Number of Parameters”
- Bingcong Li, Liang Zhang, Aryan Mokhtari, Niao He, “On the Crucial Role of Initialization for Matrix Factorization.”, In *ICLR* 2025
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. “Scaling laws for neural language models”, 2020.
- Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack William Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, Advances in Neural Information Processing Systems, 2022.

- Sean McLeish, John Kirchenbauer, David Yu Miller, Siddharth Singh, Abhinav Bhatele, Micah Goldblum, Ashwinee Panda, and Tom Goldstein. “Gemstones: A model suite for multi-faceted scaling laws”, 2025
-