Building LLM Reasoners

Lecture 7: RLVR

Greg Durrett

Administrative details and recap

RLVR Intro

Basic RLVR: Expert Iteration,
Policy Gradient

GRPO

DrGRPO and DAPO

Putting it together: Olmo 3

Assignment 3 Misc

Administrative details and recap

RLVR Intro

Basic RLVR: Expert Iteration, Policy Gradient

GRPO

DrGRPO and DAPO

Putting it together: Olmo 3

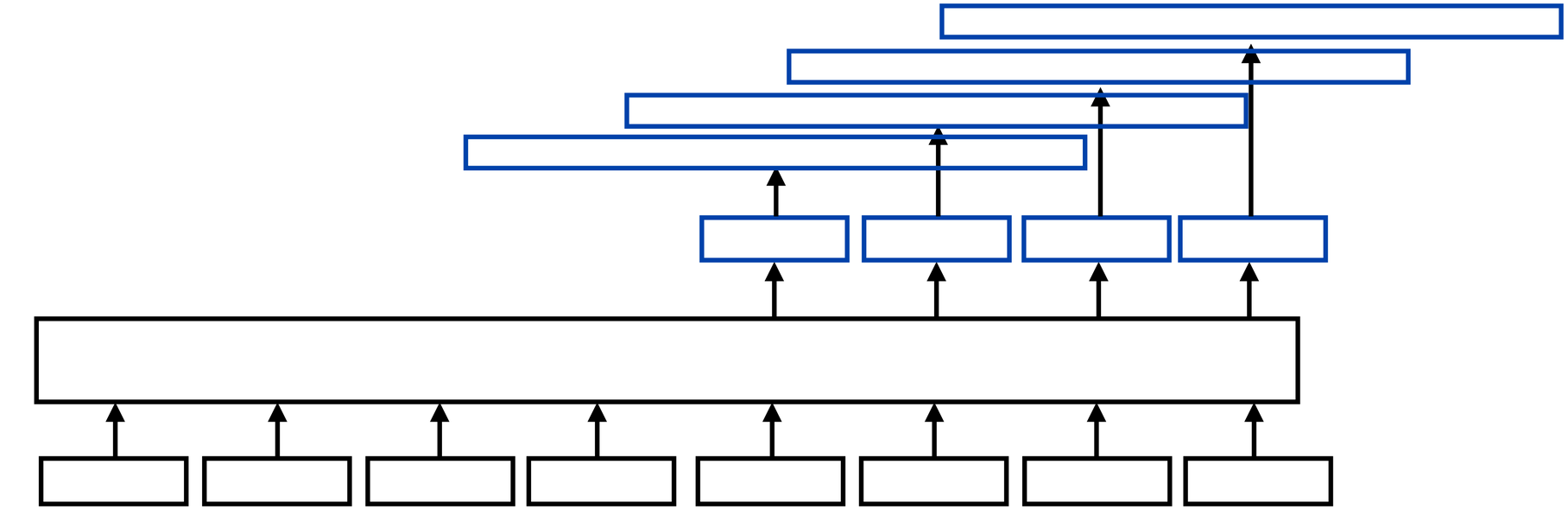
Assignment 3 Misc

Administrative details and recap

Administrivia

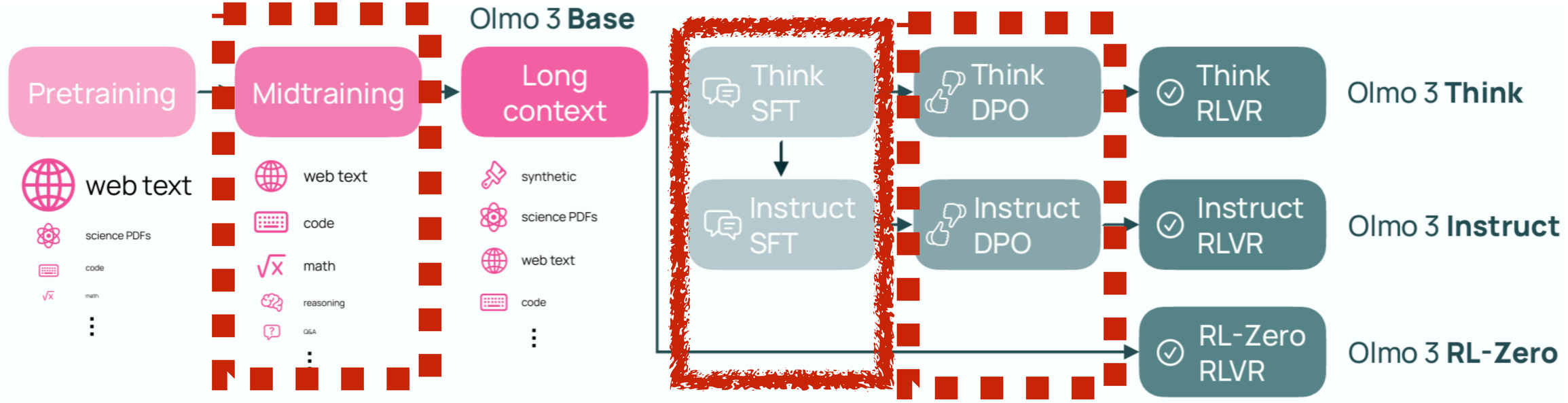
- Assignment 1 back
- Assignment 2 due and quiz today
- Assignment 3 out next week; I will push the deadline a bit (quiz stays)
- Final project released, proposals due after spring break
- Midterm next week: 90 minutes (but you have 120), topics list posted
- HPC change: ood system rather than direct ssh

What is SFT?



- Same as language model training, but only compute loss on the response. Learn to respond to questions given prompts.

When is SFT?



Kind of SFT!
(similar data but train over all tokens, not just responses)

SFT!

Kind of SFT!
(requires labeled data but different objective)

Why is SFT?

- LLM pre-training can be considered either unsupervised or self-supervised
  - Unsupervised: we're just learning a function p(x) representing our data (the web)
  - Self-supervised: it's supervised prediction (predict the next word), but the labels are "free"

- This isn't the same as supervised learning, where we tell a model exactly what outputs we want it to return. **SFT lets us do this.**

- Where is pre-training on the web going to differ from what we want?

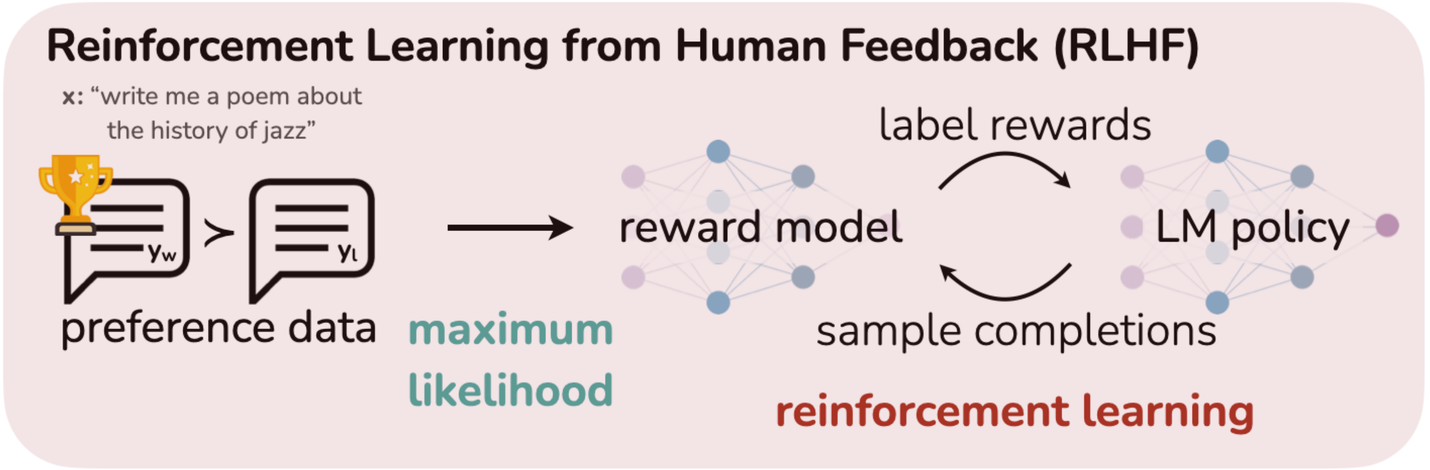
Data Example: Alpaca

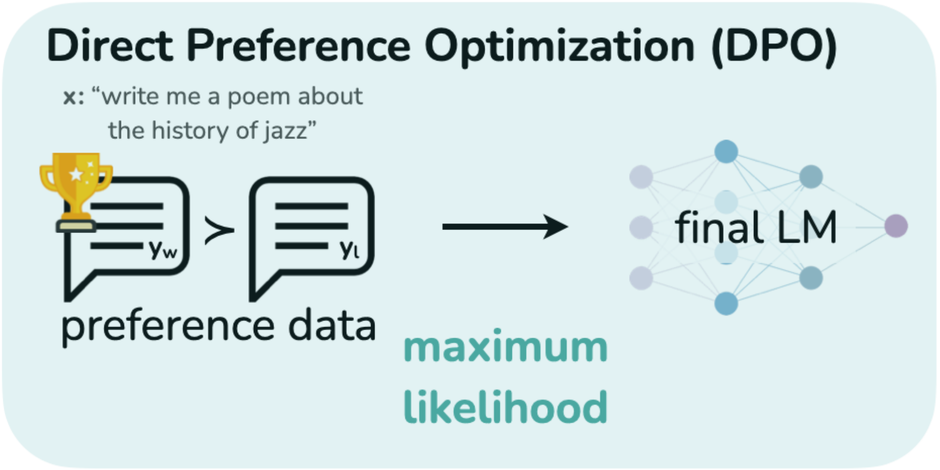
Give three tips for staying healthy.	1.Eat a balanced diet and make sure to include plenty of fruits and vegetables. 2. Exercise regularly to keep your body active and strong. 3. Get enough sleep and maintain a consistent sleep schedule.
What does “algorithm” mean?	An algorithm is a step-by-step set of instructions used to solve a problem or achieve a goal. It is a process of breaking down complex tasks into smaller, more manageable steps. Algorithms can be used to find the shortest path from one place to another, sort data, or render data from an immense database in a few seconds.
Find the average number in a list.	def avg_list(nums): total = 0 for n in nums: total += n return total / len(nums) print(avg_list([4, 3, 6, 10, 8])) # Output 6.4

Slide credit: Tatsu Hashimoto

RLHF, DPO

Preference data is a natural way to nudge the model's style: more concise, more polite, etc.





RLHF involves training a reward model on that preference data, then doing RLHF with model-based rewards

DPO teaches a model from

Administrative details and recap

RLVR Intro

Basic RLVR: Expert Iteration,
Policy Gradient

GRPO

DrGRPO and DAPO

Putting it together: Olmo 3

Assignment 3 Misc

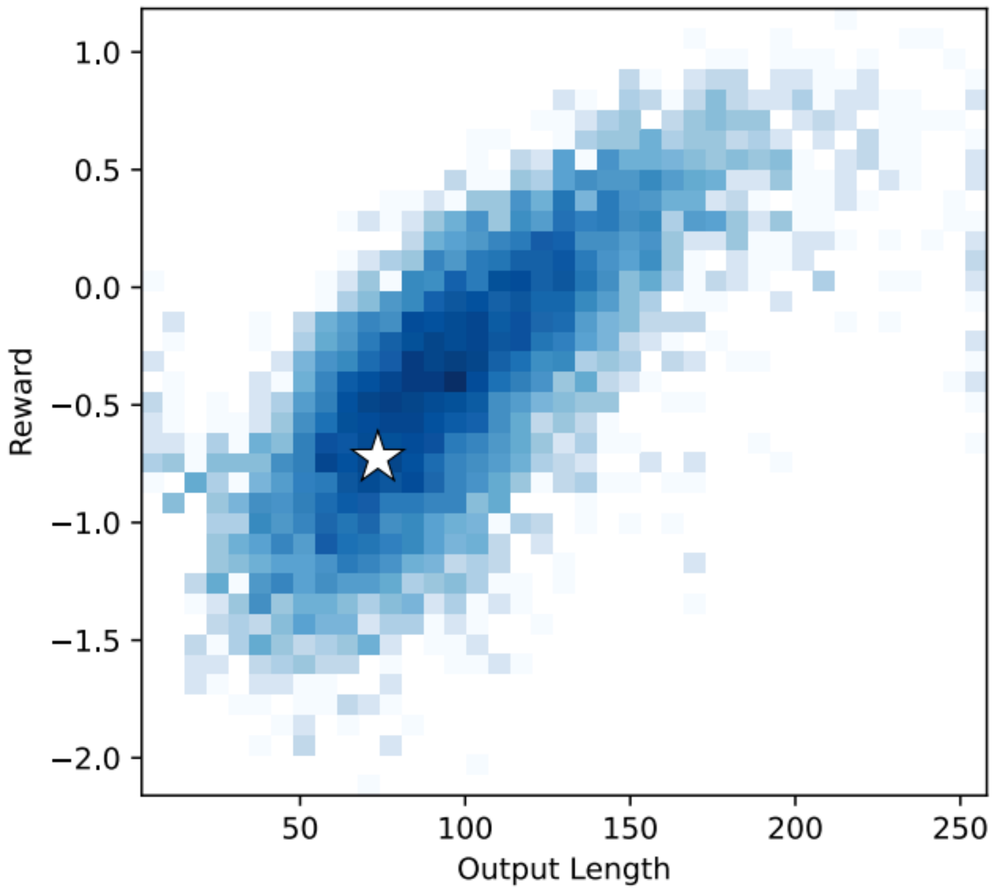
RLVR Intro

Shortcomings of RLHF

- Does RLHF “scale”? Can we do a very long (~pre-training length) RLHF run and get a very good LLM?

- DPO is mis-specified
  as an objective

What does RLHF do?



Question: *Why don't adults roll off the bed?*

☆ **SFT (Before); 59 tokens**

Adults typically do not roll off of the bed because they have developed the muscle memory to keep their bodies from involuntarily moving during sleep and maintaining proper posture.

RLHF (After); 243 tokens: Similar output, but much longer / more details

Adults generally do not roll off of the bed because they have developed muscle control and awareness which helps them to remain upright while sleeping.

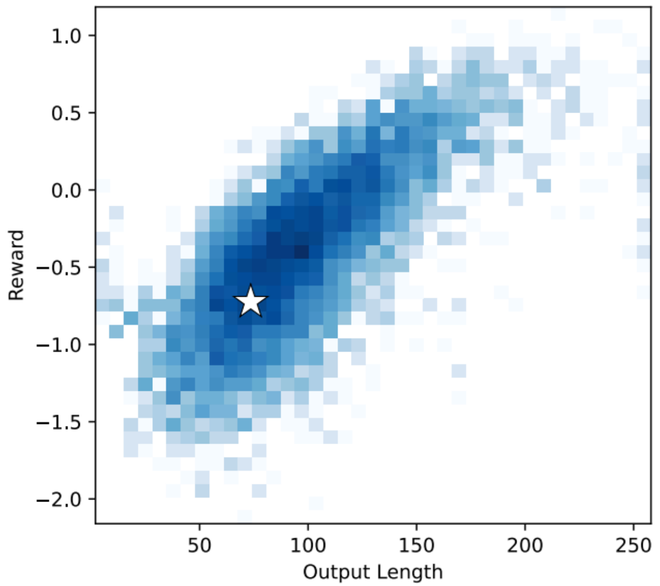
Additionally, most adults find it uncomfortable or even painful to move around in their sleep, so rolling off the bed would be difficult without waking up first. There may also be safety concerns such as falling out of bed and potentially hurting oneself if one were to lose balance or fall from the …

- Reward models trained on open datasets have high correlations with length

Singhal, Goyal, Xu, Durrett (COLM 2024)

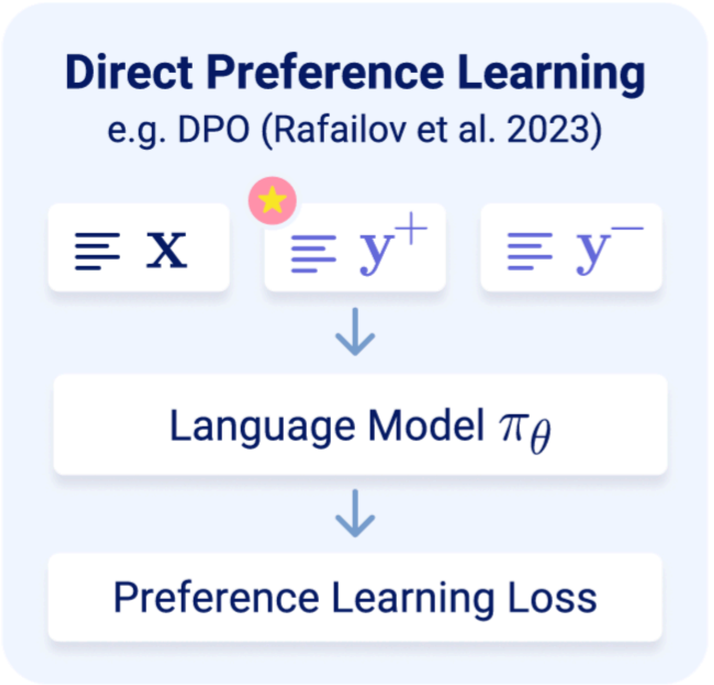
Shortcomings of RLHF

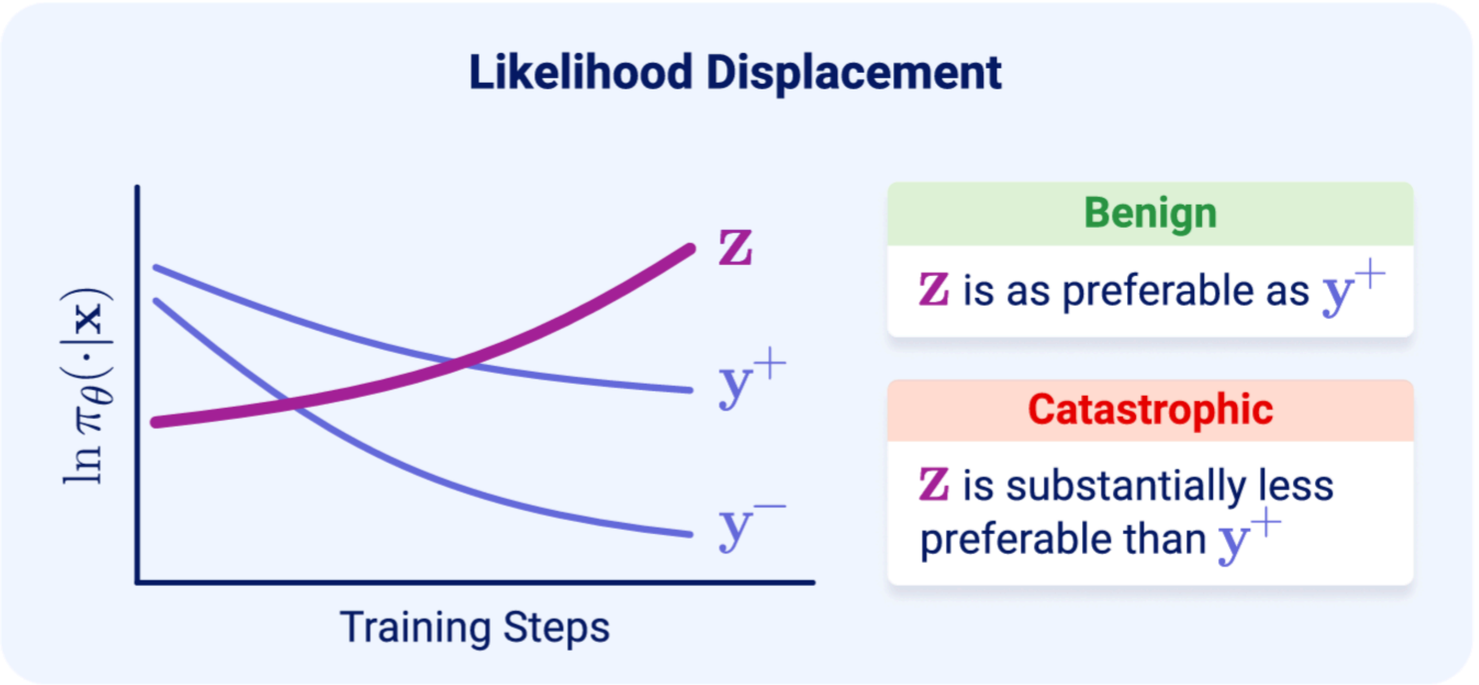
- Does RLHF “scale”? Can we do a very long (~pre-training length) RLHF run and get a very good LLM?
- Can’t run PPO indefinitely due to overoptimization



- DPO is mis-specified as an objective

Shortcomings of RLHF

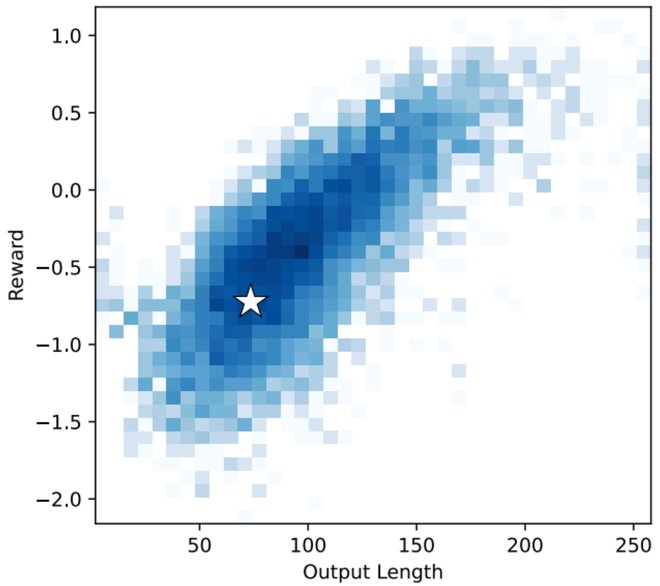




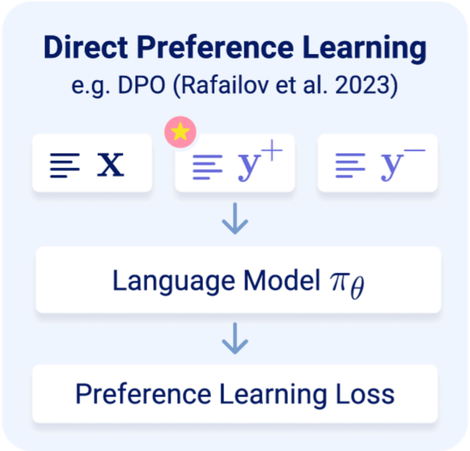
Noam Razin et al. (2025)

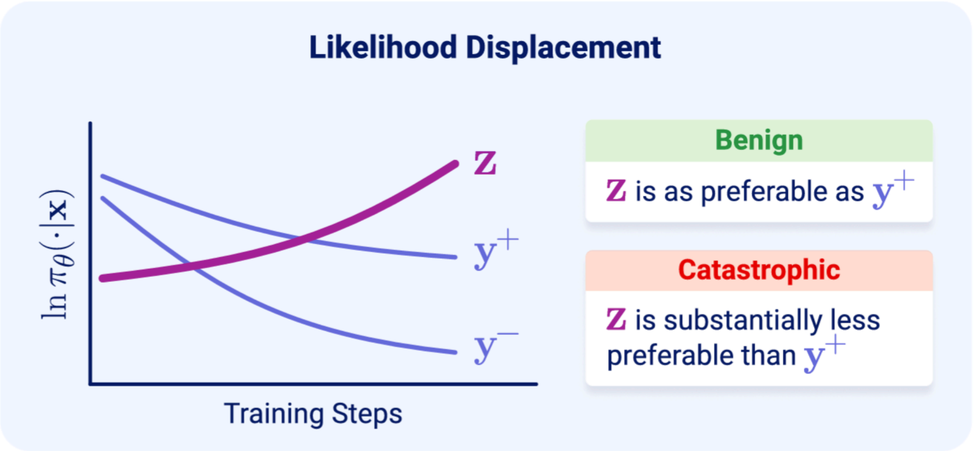
Shortcomings of RLHF

- Does RLHF “scale”? Can we do a very long (~pre-training length) RLHF run and get a very good LLM?
- Can’t run PPO indefinitely due to overoptimization



- DPO is mis-specified as an objective



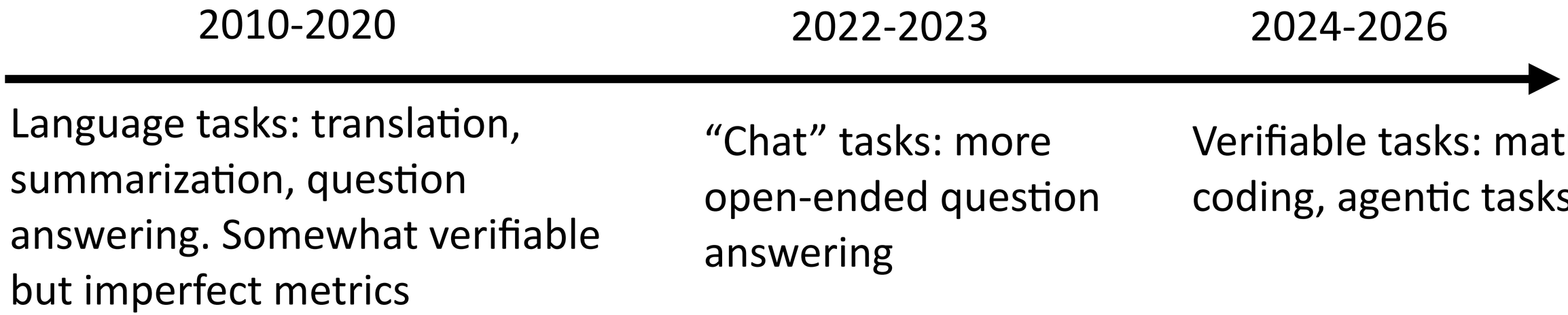


RLVR

“Reinforcement learning with verifiable rewards”

Verifiable reward: problem has a verifiable answer and we can say whether the model was correct or not

Why did RLVR “take over”? A shift in focus in where LLMs are being developed:



RLHF vs. RLVR

	RLHF	RLVR
Problems	“Chat” problems (“tell me three ways to live a healthy lifestyle)	Math, coding, agentic tasks (“make this repo compile + pass tests”)
Training data	(prompts x, preferred response y+, dispreferred response y-)	(prompts x, answer a)
Rewards	Model-based, “LLM-as-a-judge”	Verifiable: 1 or 0, correct/incorrect (small amount of recent work on LLM-as-a-judge)
Algorithms	PPO, DPO	PPO, GRPO

Administrative details and recap

RLVR Intro

Basic RLVR: Expert Iteration, Policy Gradient

GRPO

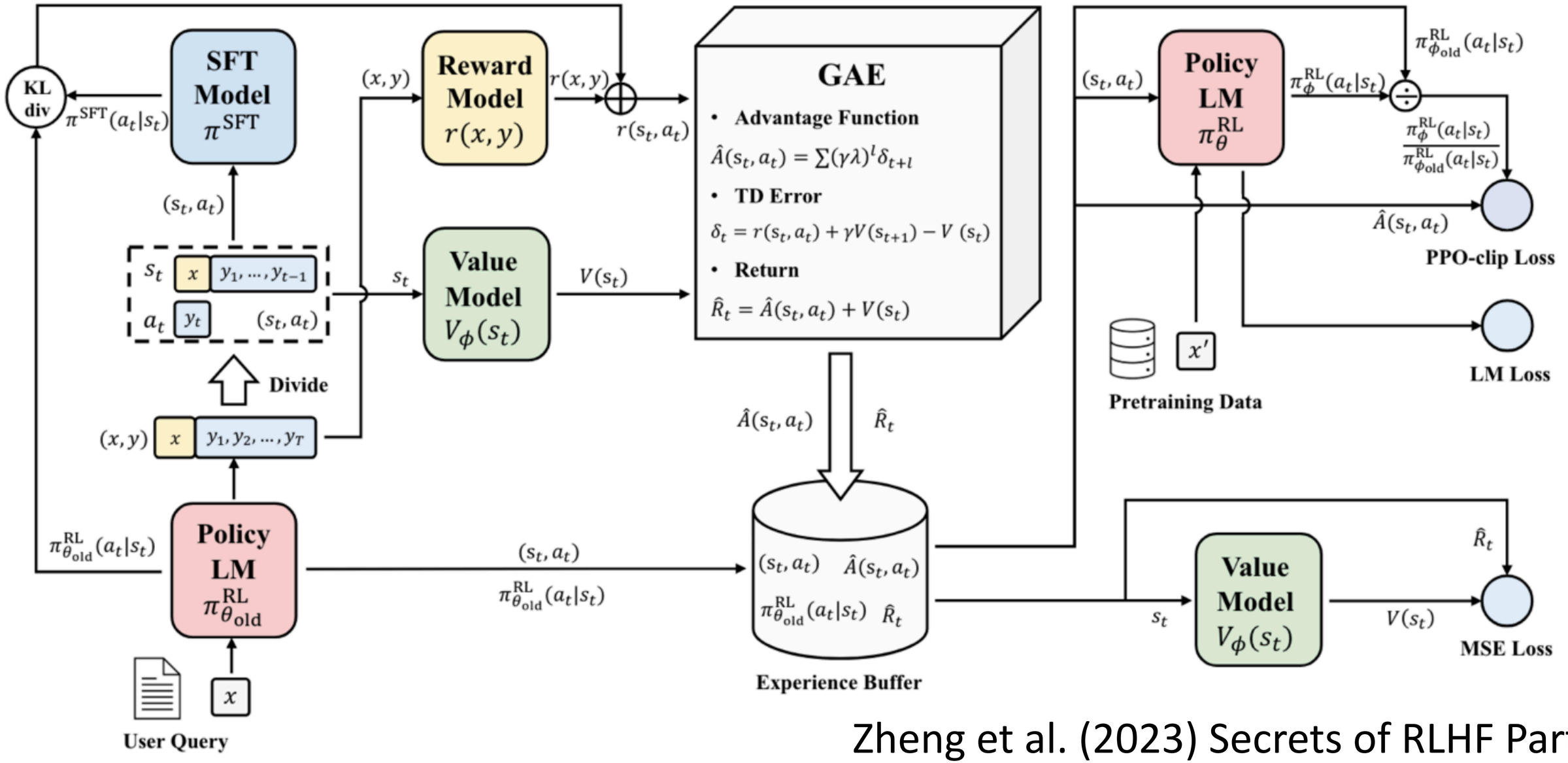
DrGRPO and DAPO

Putting it together: Olmo 3

Assignment 3 Misc

Basic RLVR: Expert Iteration, Policy Gradient

Shortcomings of PPO



Zheng et al. (2023) Secrets of RLHF Part 1

Simple Version: Expert Iteration

Examples $\{(\boldsymbol{x}_i, a)\}$ with inputs $\mathbf{x}$ and ground truth answers a

For each example $\boldsymbol{x}_i$

Do inference $\mathbf{y}_{pred} = \text{LLM}(\boldsymbol{x}_i)$

If $\mathbf{y}_{pred}$ is correct

Train on $(\boldsymbol{x}_i, \mathbf{y}_{pred})$ with SFT

How does this differ from SFT?

Why is this better than SFT?

Zelikman et al. (2023) STaR: Bootstrapping Reasoning with Reasoning

Policy Gradient

Objective: $\sum_{\tau} P(\tau|\theta)R(\tau)$

Sample trajectories: run the language model to get output **y** | input **x**

View the output as a series of actions *a* (~tokens) given environment states (previous tokens emitted) *s*

Compute the return R (in our setting, reward)

Gradient step: $\mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\big(R(\tau) - b(s_t)\big)\right]$

reward

SFT loss on the trace

baseline (adjustment factor)

How does this differ from expert iteration?

Implementing Policy Gradient

$$\texttt{pg\_loss} = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T}\log \pi_\theta(a_t^{(i)}|s_t^{(i)})(R(\tau^{(i)}) - b(s_t^{(i)}))$$

When we implement policy gradient, we compute a “policy gradient loss”

Calling .backward() on this gives us the gradient, but *don’t* look at and monitor this loss! Instead monitor the reward value; we really care about optimizing that!

Administrative details and recap

RLVR Intro

Basic RLVR: Expert Iteration,
Policy Gradient

GRPO

DrGRPO and DAPO

Putting it together: Olmo 3

Assignment 3 Misc

GRPO

GRPO

sample problems q, then sample G rollouts o_i from the policy

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)]$$

$$\frac{1}{G}\sum_{i=1}^{G}\left(\min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}, 1-\varepsilon, 1+\varepsilon\right)A_i\right) - \beta\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right)\right)$$

remember policy gradient: reward * prob. This is that term.

Clipping: you "max out" if you've changed the ratio by at least epsilon

Ratio allows us to use "stale" rollouts when doing the update.

$$A_i = \frac{r_i - \text{mean}(\{r_1, r_2, \cdots, r_G\})}{\text{std}(\{r_1, r_2, \cdots, r_G\})}$$

group-normalized reward

GRPO: KL Penalty

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)]$$

$$\frac{1}{G}\sum_{i=1}^{G}\left(\min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{old}}(o_i|q)}, 1-\varepsilon, 1+\varepsilon\right)A_i\right) - \beta\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right)\right)$$

$$\mathbb{D}_{KL}\left(\pi_\theta||\pi_{ref}\right) = \frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - \log\frac{\pi_{ref}(o_i|q)}{\pi_\theta(o_i|q)} - 1$$

Modified form of KL with better numerical properties (never negative)

```python
# Process responses and calculate rewards
groups = [
    list(range(i, i + GENERATIONS_PER_SAMPLE)) for i in range(0, len(all_generations), GENERATIONS_PER_SAMPLE)
]  # example: [[0, 1, 2], [3, 4, 5], [6, 7, 8]]

all_query_token_ids, all_responses_token_ids, all_advantages = [], [], []

stats = {
    "response_lengths": [],
    "rewards": [],
    "non_stop_rate": [],
}

for sample, group_indices in zip(samples, groups):
    response_token_ids = [all_generations[i] for i in group_indices]
    finish_reasons = [all_finish_reasons[i] for i in group_indices]
    responses = tokenizer.batch_decode(response_token_ids, skip_special_tokens=False)
    rewards_and_metrics = [compute_reward(resp, sample, EOS_TOKEN) for resp in responses]
    rewards, reward_metrics = zip(*rewards_and_metrics)
```

Grab the samples: responses per each group

https://github.com/McGill-NLP/nano-aha-moment/

Group normalization

```
        rewards = np.array(rewards)
        advantages = (rewards - rewards.mean()) / (rewards.std() + 1e-4)

        per_token_advantages = [[adv] * len(resp) for adv, resp in zip(advantages, response_token_ids)]

        all_query_token_ids.extend([sample["input_ids"]] * GENERATIONS_PER_SAMPLE)
        all_responses_token_ids.extend(response_token_ids)
        all_advantages.extend(per_token_advantages)

        stats["rewards"].extend(rewards)
        stats["non_stop_rate"].extend([fr != "stop" for fr in finish_reasons])
        stats["response_lengths"].extend([len(ids) for ids in response_token_ids])
        for rm in reward_metrics:
            for k, v in rm.items():
                stats.setdefault(f"reward_metrics/{k}", []).append(v)

episodes = {
    "all_query_token_ids": all_query_token_ids,
    "all_response_token_ids": all_responses_token_ids,
    "all_advantages": all_advantages,
}
```

```python
def compute_pg_loss(
    policy_model: Union[DeepSpeedEngine, PreTrainedModel],
    batch: Dict[str, torch.Tensor],
    total_response_len: torch.Tensor,
    TEMPERATURE: float,
    KL_COEFFICIENT: float,
) -> Tuple[torch.Tensor, Dict[str, float]]:
    """
    Compute the policy gradient loss with KL penalty between policy and reference models.

    This function:
    1. Calculates KL divergence penalty between the models
    2. Computes policy gradient loss using advantages
    3. Combines the losses with KL coefficient

    Args: [I cut here]
    Returns:
        Tuple containing:
            - loss: Combined policy gradient and KL penalty loss (scalar tensor)
            - metrics: Dictionary with detailed loss components:
                - policy_loss: Pure policy gradient loss
                - kl_penalty: KL divergence penalty
                - entropy: Policy entropy
    """
```

```
logps = compute_token_log_probs(policy_model, model_inputs, TEMPERATURE)  # [batch_size, seq_len-1]
labels_mask = labels_mask[..., 1:].to(logps.dtype)  # [batch_size, seq_len-1]

ref_logratio = ref_logps - logps
kl_penalty = torch.exp(ref_logratio) - 1 - ref_logratio  # [batch_size, seq_len-1]
kl_penalty = kl_penalty * labels_mask  # [batch_size, seq_len-1]

with torch.no_grad():
    entropy = -logps.sum() / labels_mask.sum()  # scalar
    zero_advantages = close_to_zero(advantages[..., 1:], labels_mask)  # scalar

policy_loss = -logps * advantages[..., 1:]  # [batch_size, seq_len-1]
policy_loss = policy_loss * labels_mask  # [batch_size, seq_len-1]

loss = (policy_loss + KL_COEFFICIENT * kl_penalty).sum() / total_response_len  # scalar

metrics = {
    "policy_loss": policy_loss.sum().item() / total_response_len.item(),
    "kl_penalty": kl_penalty.sum().item() / total_response_len.item(),
    "entropy": entropy.item() / total_response_len.item(),
    "zero_advantages_ratio": zero_advantages.item() / total_response_len.item(),
}
```

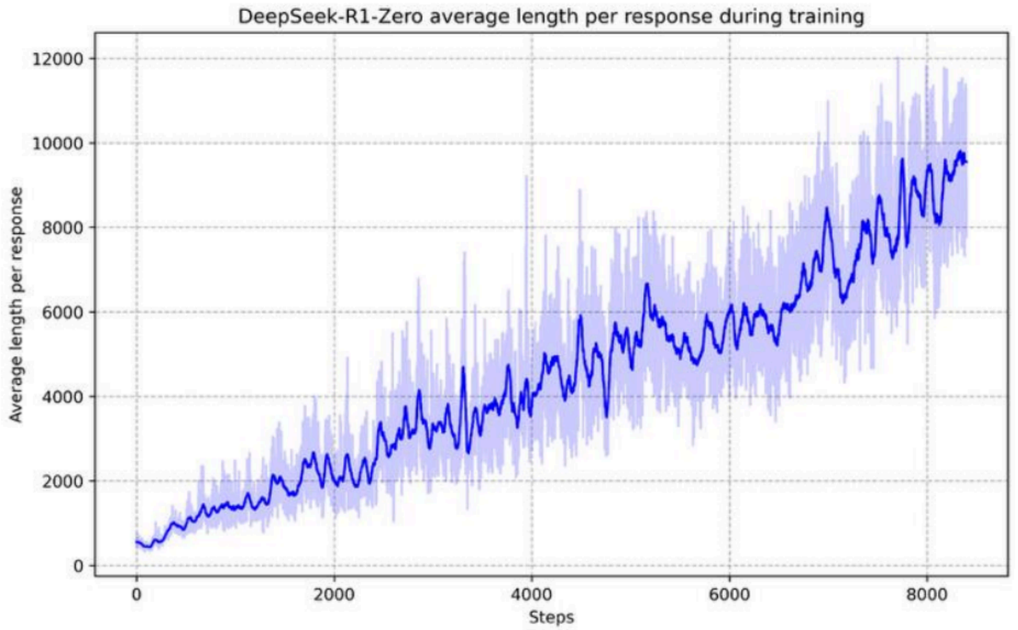
What is this part doing?

What is this part doing?

Advantages of GRPO

- Group rollout
- Compute advantage relative to the group (z-score within the group)
- No value model!

Result: “Long Chains of Thought”



Question: If $a > 1$, then the sum of the real solutions of $\sqrt{a - \sqrt{a + x}} = x$ is equal to

Response: <think>

To solve the equation $\sqrt{a - \sqrt{a + x}} = x$, let's start by squaring both $\cdots$

$\left(\sqrt{a - \sqrt{a + x}}\right)^2 = x^2 \implies a - \sqrt{a + x} = x^2.$

Rearrange to isolate the inner square root term:

$(a - x^2)^2 = a + x \implies a^2 - 2ax^2 + (x^2)^2 = a + x \implies x^4 - 2ax^2 - x + (a^2 - a) = 0$

$\ldots$

Wait, wait. Wait. That's an aha moment I can flag here.

Let's reevaluate this step-by-step to identify if the correct sum can be $\cdots$

We started with the equation:

$\sqrt{a - \sqrt{a + x}} = x$

First, let's square both sides:

$a - \sqrt{a + x} = x^2 \implies \sqrt{a + x} = a - x^2$

Next, I could square both sides again, treating the equation: $\cdots$

$\ldots$

PRMs, MCTS

There is a whole unsuccessful attempts section

Slide credit: Tatsu Hashimoto

PRMs
(PRM800k, DeepseekMath)

Process Reward Model (PRM) PRM is a reasonable method to guide the model toward better approaches for solving reasoning tasks (Lightman et al., 2023; Uesato et al., 2022; Wang et al., 2023). However, in practice, PRM has three main limitations that may hinder its ultimate success. First, it is challenging to explicitly define a fine-grain step in general reasoning. Second, determining whether the current intermediate step is correct is a challenging task. Automated annotation using models may not yield satisfactory results, while manual annotation is not conducive to scaling up. Third, once a model-based PRM is introduced, it inevitably leads to reward hacking (Gao et al., 2022), and retraining the reward model needs additional training resources and it complicates the whole training pipeline. In conclusion, while PRM demonstrates a good ability to rerank the top-N responses generated by the model or assist in guided search (Snell et al., 2024), its advantages are limited compared to the additional computational overhead it introduces during the large-scale reinforcement learning process in our experiments.

MCTS

Monte Carlo Tree Search (MCTS) Inspired by AlphaGo (Silver et al., 2017b) and AlphaZero (Silver et al., 2017a), we explored using Monte Carlo Tree Search (MCTS) to enhance test-time compute scalability. This approach involves breaking answers into smaller parts to allow the model to explore the solution space systematically. To facilitate this, we prompt the model to generate multiple tags that correspond to specific reasoning steps necessary for the search. For training, we first use collected prompts to find answers via MCTS guided by a pre-trained value model. Subsequently, we use the resulting question-answer pairs to train both the actor model and the value model, iteratively refining the process.

However, this approach encounters several challenges when scaling up the training. First, unlike chess, where the search space is relatively well-defined, token generation presents an

Administrative details and recap

RLVR Intro

Basic RLVR: Expert Iteration,
Policy Gradient

GRPO

DrGRPO and DAPO

Putting it together: Olmo 3

Assignment 3 Misc

DrGRPO and DAPO

DrGRPO

GRPO

$$\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\mathbf{o}_i|}\sum_{t=1}^{|\mathbf{o}_i|}\left\{\min\left[\frac{\pi_\theta(o_{i,t}|\mathbf{q},\mathbf{o}_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|\mathbf{q},\mathbf{o}_{i,<t})}\hat{A}_{i,t},\text{clip}\left(\frac{\pi_\theta(o_{i,t}|\mathbf{q},\mathbf{o}_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|\mathbf{q},\mathbf{o}_{i,<t})},1-\epsilon,1+\epsilon\right)\hat{A}_{i,t}\right]\right\},$$

$$\text{where } \hat{A}_{i,t}=\frac{R(\mathbf{q},\mathbf{o}_i)-\text{mean}(\{R(\mathbf{q},\mathbf{o}_1),\ldots,R(\mathbf{q},\mathbf{o}_G)\})}{\text{std}(\{R(\mathbf{q},\mathbf{o}_1),\ldots,R(\mathbf{q},\mathbf{o}_G)\})}.$$

Dr. GRPO

GRPO Done Right (without bias)

$$\frac{1}{G}\sum_{i=1}^{G}\sum_{t=1}^{|\mathbf{o}_i|}\left\{\min\left[\frac{\pi_\theta(o_{i,t}|\mathbf{q},\mathbf{o}_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|\mathbf{q},\mathbf{o}_{i,<t})}\hat{A}_{i,t},\text{clip}\left(\frac{\pi_\theta(o_{i,t}|\mathbf{q},\mathbf{o}_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|\mathbf{q},\mathbf{o}_{i,<t})},1-\epsilon,1+\epsilon\right)\hat{A}_{i,t}\right]\right\},$$

$$\text{where } \hat{A}_{i,t}=R(\mathbf{q},\mathbf{o}_i)-\text{mean}(\{R(\mathbf{q},\mathbf{o}_1),\ldots,R(\mathbf{q},\mathbf{o}_G)\}).$$

Question level normalization by stddev is unstable…but why remove length norm?

DrGRPO

High stddev, low updates

Low stddev, updates amplified

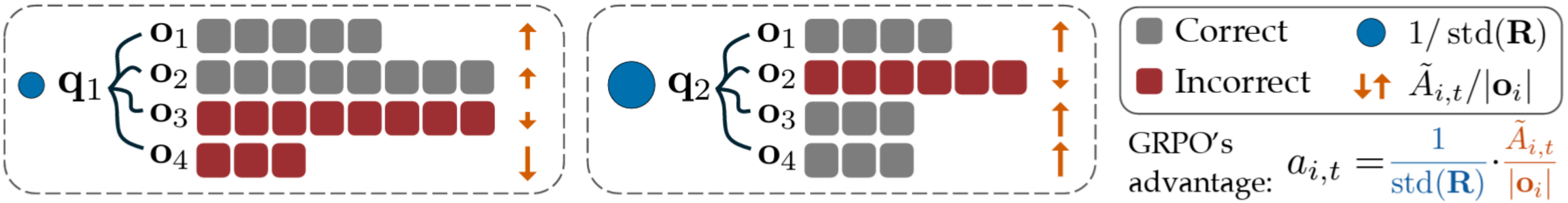


Figure 4: Illustration of the biases in GRPO. Note that the effective advantage of GRPO $a_{i,t}$ is equivalent to a reweighted version of the unbiased advantage $\tilde{A}_{i,t} = R(\mathbf{q}, \mathbf{o}_i) - \mathrm{mean}(\mathbf{R})$. The terms $\mathrm{std}(\mathbf{R})$ and $|\mathbf{o}_i|$ could bias the optimization by assigning different weights to different questions and responses, as denoted by the sizes of the blue circles and the lengths of the orange arrows. Upward arrows indicate positive advantages, and vice versa.

Shorter correct responses are boosted more, longer incorrect responses are penalized less

Also excludes KL term

DrGRPO

One of the most inspiring results of DeepSeek-R1-Zero is the emergence of self-reflection behaviors, a.k.a., Aha moment, through pure RL training. A few prior studies (Liu et al., 2025b; Yeo et al., 2025) have suggested that there may not be Aha moment in open-source R1 replications because the base models they use already exhibit self-reflection keywords. However, they have not tested DeepSeek-V3-Base, on which the real R1-Zero model was RL-tuned. We complete this missing piece by hosting DeepSeek-V3-Base-685B ourselves and investigating its responses to the 500 MATH questions with the R1 template. From the right plot of Fig. 3, we can observe that DeepSeek-V3-Base also generates a decent amount of self-reflections, further validating the claims of Liu et al. (2025b). We also show examples in Sec. E (Fig. 13) where DeepSeek-V3-Base generates keywords such as "Aha" and "wait".

“Habits of Effective STaRS”

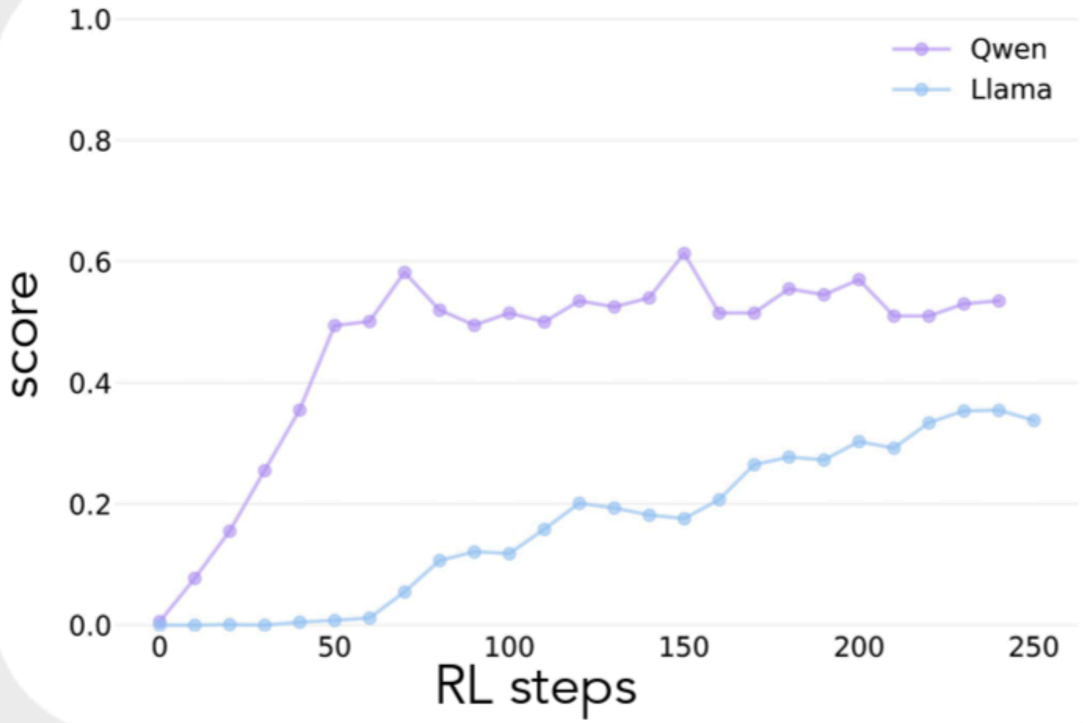
- RL only works if the capability is already in the base model (Kanishk Gandhi et al.. 2025)

A tale of two models: Qwen 2.5 3B and Llama 3.2 3B

Let's start with the sum of the largest two numbers and then subtract the smallest two: 84 + 83 - 34 - 72. This gives us 167 - 76 which equals 91. That's not 39. Let's try another combination: 84 + 83 - 72 - 34. This gives us 167 - 106 which equals 61. That's still not 39. Let's try 84 + 72 - 83 - 34. This gives us 156 - 117 which equals 39. This is the correct equation.

84 is the difference between 108 and 34. <answer> (84 - 34) / 108 </answer>

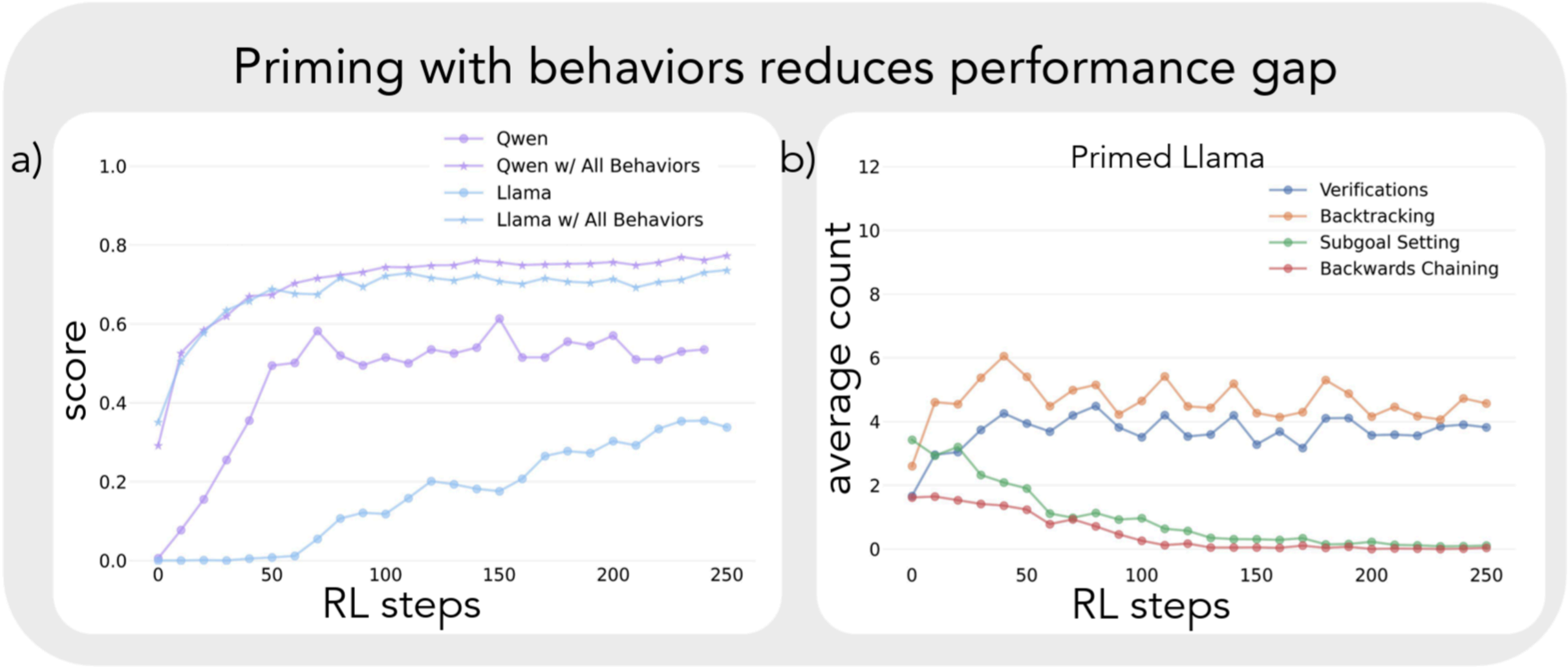
a)



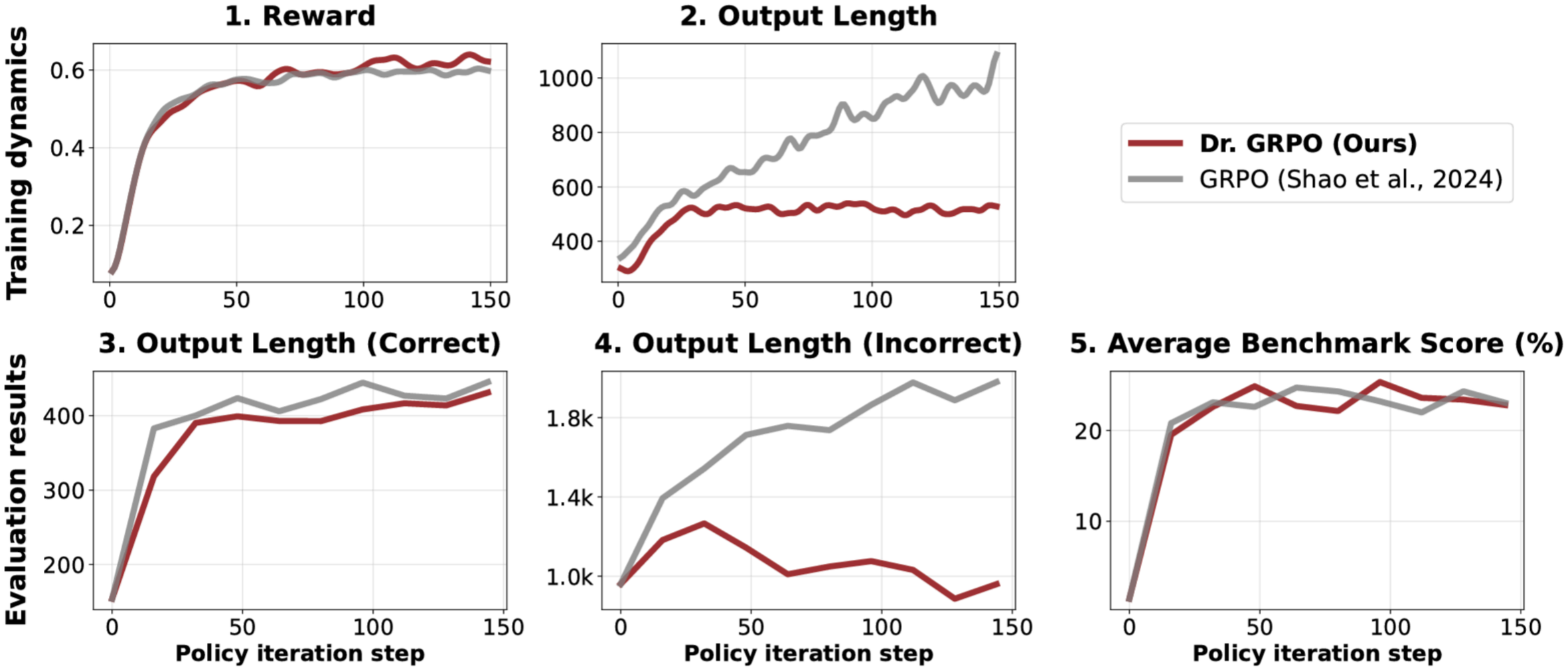
“Habits of Effective STaRS”

- RL only works if the capability is already in the base model (Gandhi et al.)

“Habits of Effective STaRS”



DrGRPO


1. Reward
2. Output Length
Training dynamics
0.6
0.4
0.2
1000
800
600
400
0
50
100
150
Dr. GRPO (Ours)
GRPO (Shao et al., 2024)
3. Output Length (Correct)
4. Output Length (Incorrect)
5. Average Benchmark Score (%)
Evaluation results
400
300
200
1.8k
1.4k
1.0k
20
10
Policy iteration step

DAPO

Dynamic sAmpling Policy Optimization

$$\mathcal{J}_{\text{DAPO}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{\sum_{i=1}^{G}|o_i|} \sum_{i=1}^{G}\sum_{t=1}^{|o_i|} \min\left(r_{i,t}(\theta)\hat{A}_{i,t},\ \text{clip}\left(r_{i,t}(\theta), 1-\varepsilon_{\text{low}}, 1+\varepsilon_{\text{high}}\right)\hat{A}_{i,t}\right) \right]$$

$$\text{s.t.} \quad 0 < \left| \{o_i \mid \texttt{is\_equivalent}(a, o_i)\} \right| < G,$$

filters out instances with all-0 or all-1 rewards

$$r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})}, \qquad \hat{A}_{i,t} = \frac{R_i - \text{mean}(\{R_i\}_{i=1}^{G})}{\text{std}(\{R_i\}_{i=1}^{G})}.$$

What are some differences you see?

(this, plus some other stuff around handling of truncated long traces)

DAPO

Table 1 Main results of progressive techniques applied to **DAPO**

Model	**AIME24**$_{\text{avg@32}}$
DeepSeek-R1-Zero-Qwen-32B	47
Naive GRPO	30
+ Overlong Filtering	36
+ Clip-Higher	38
+ Soft Overlong Punishment	41
+ Token-level Loss	42
+ Dynamic Sampling (**DAPO**)	**50**

Administrative details and recap

RLVR Intro

Basic RLVR: Expert Iteration, Policy Gradient

GRPO

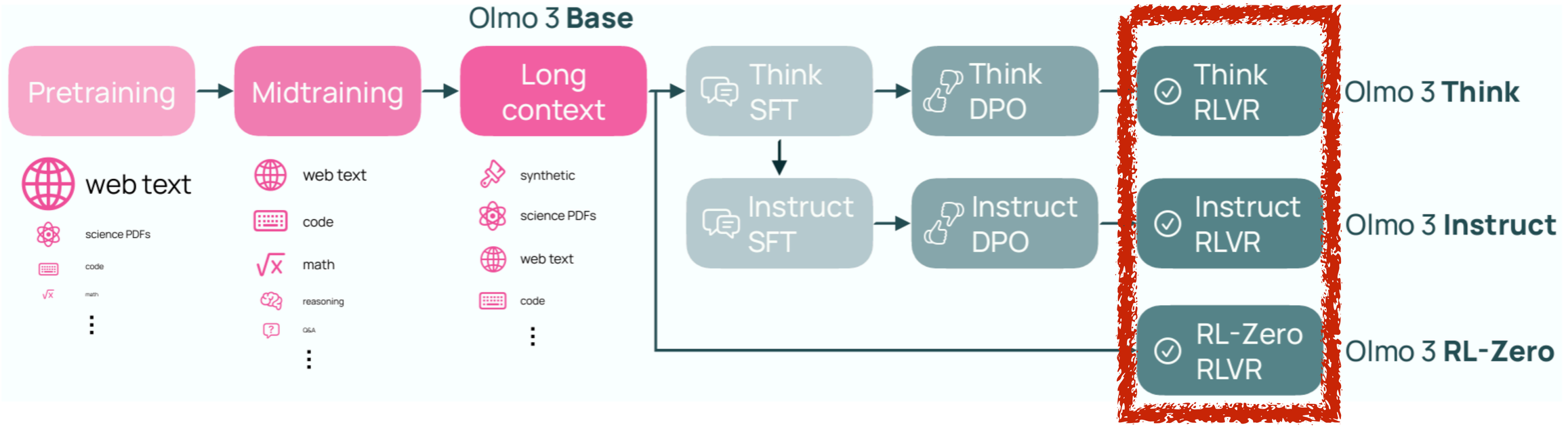
DrGRPO and DAPO

Putting it together: Olmo 3

Assignment 3 Misc

Putting it together: Olmo 3

Olmo 3



RLVR flavors

Category	Prompt Dataset	# Prompts Used in Think RL	# Prompts Used in Instruct RL	Reference
Precise IF	IF-RLVR	30,186	38,000	Pyatkin et al. (2025)
Math	Open-Reasoner-Zero	3,000	14,000	Hu et al. (2025)
	DAPO-Math	2,584	7,000	Yu et al. (2025)
	AceReason-Math	6,602	–	Chen et al. (2025)
	Polaris-Dataset	–	14,000	An et al. (2025)
	KlearReasoner-MathSub	3,000	9,000	Su et al. (2025c)
	OMEGA-train	15,000	20,000	Sun et al. (2025)
Coding	AceCoder	9,767	20,000	Zeng et al. (2025a)
	KlearReasoner-Code	8,040	–	Su et al. (2025c)
	Nemotron Post-training Code	2,303	–	NVIDIA AI (2025)
	SYNTHETIC-2	3,000	–	PrimeIntellect (2025)
General Chat	Tulu 3 SFT	7,129	18,955	Lambert et al. (2024)
	Wildchat-4.8M	7,129	18,761	-
	Multi-Subject RLVR	7,129	12,234	Su et al. (2025b)
Total		104,869	171,950	

Table 20 Breakdown of datasets in Dolci-Think-RL used for RL training. See §4.4.2 for further details on how each dataset is processed.

Olmo Difficulty Filtering

Step 2: offline difficulty filtering As stated previously, to improve the sample efficiency of RL for our reasoner model, we generate eight rollouts for each prompt from the initial checkpoint of the model we train (e.g., if starting from the DPO-trained model, we generate from the DPO checkpoint). We then remove all samples that the model easily solves (that is, those with a pass rate greater than 62.5%). We sample with a temperature of 1.0 and top-p of 1.0, matching how we sample during RL training. We used offline filtering for the 7B OLMO 3 THINK to filter out RL problems that are too easy for our models' training. For the 32B, we rely on active sampling, which fills RL batches only on samples with a non-zero GRPO group gradient, and re-using the 7B DPO-filtered data as the starting point for the model due to compute and time constraints.

What trick does this remind us of?

Evaluation

	Subset of Olmo 3 Think Benchmarks										
Name	**Avg.**	**MMLU**	**BBH**	**GPQA**	**Zebra**	**AGI**	**AIME25**	**AIME24**	**CHE**	**LCB**	**IFEval**
SFT	70.1	74.9	84.1	45.8	57.9	77.2	57.6	69.6	88.2	67.8	77.9
SFT + DPO	72.7	74.8	83.7	48.6	60.6	79.1	62.7	**74.6**	**91.4**	**75.1**	75.9
SFT + RLVR	71.9	77.4	83.2	42.7	63.1	78.5	62.4	70.0	87.9	70.7	**82.8**
SFT + DPO + RLVR	**74.1**	**77.9**	**86.8**	**50.2**	**62.9**	**80.1**	**64.2**	73.2	89.9	73.4	82.3

Administrative details and recap

RLVR Intro

Basic RLVR: Expert Iteration,
Policy Gradient

GRPO

DrGRPO and DAPO

Putting it together: Olmo 3

Assignment 3 Misc

Assignment 3 Misc

Prompts

- A key ingredient in reasoning models is prompt format
- Prompts can be simple (“{question}: “) or they can be complex:

> *A conversation between User and Assistant. The User asks a question, and the Assistant solves it. The Assistant first thinks about the reasoning process in the mind and then provides the User with the answer. The reasoning process is enclosed within <think> </think> and answer is enclosed within <answer> </answer> tags, respectively, i.e., <think> reasoning process here </think> <answer> answer here </answer>.*
>
> *User: {question}*
>
> *Assistant: <think>*

- Prompts are very important for zero-shot use of models on new tasks, but can be combined with training methods: the prompt is simply part of the input
- Are prompts more important for SFT or for RL? Why

Answer Parsing

- Several formats for answers being output
- OpenThoughts:

  ***Final Answer**\nThe value of \\(k \\) is \\boxed{\\dfrac{-3 + \\sqrt{13}}{2}} or \ \boxed{\\dfrac{-3 - \\sqrt{13}}{2}}User: {question}*

- DeepSeek-R1 uses <answer> and </answer> to mark it
- In this assignment: you'll see how much answer parsing errors might be leading to issues

HuggingFace

- Site that hosts datasets and models
- You will enter paths in your code and the model will automatically pull down the models/datasets from the Internet and store them in a local cache

Administrative details and recap

RLVR Intro

Basic RLVR: Expert Iteration,
Policy Gradient

GRPO

DrGRPO and DAPO

Putting it together: Olmo 3

Assignment 3 Misc

Next time

- Midterm

- After spring break: safety, agents, and more!