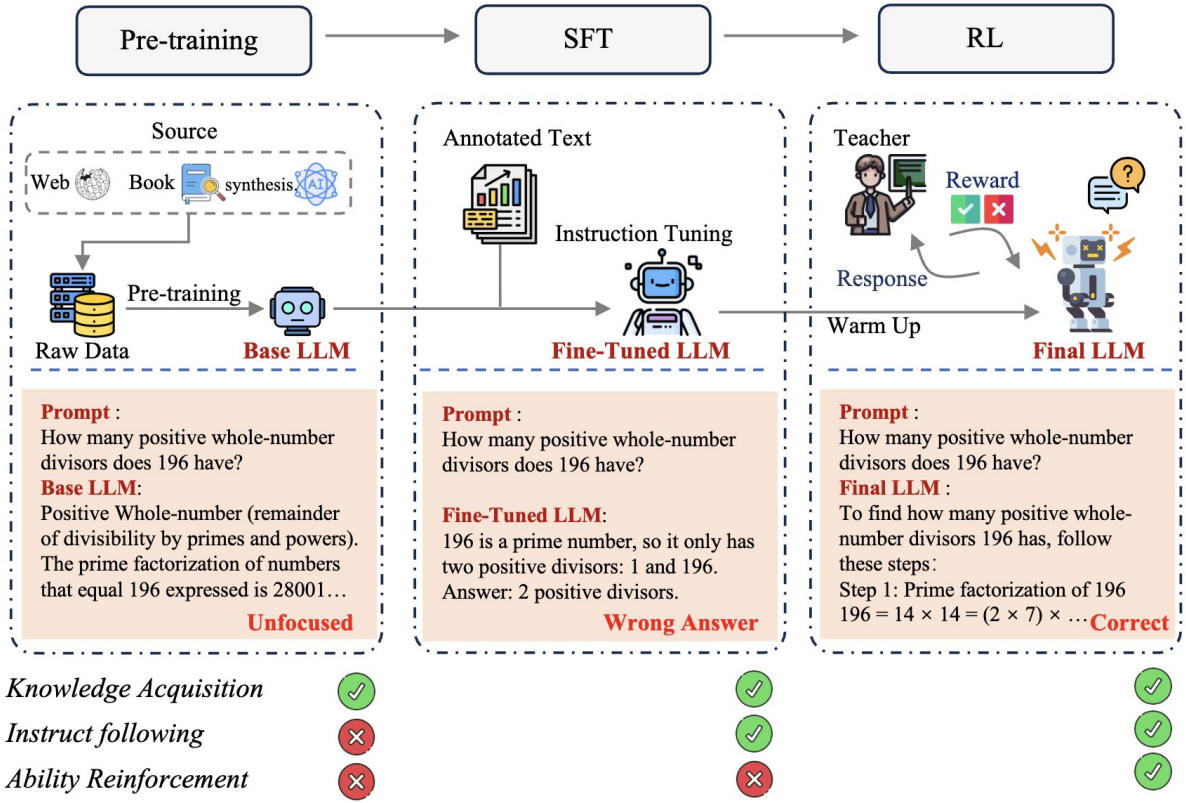
Tutorial: Reinforcement Learning for Large Language Models

Xingzhi Sun
December 2, 2025

Outline

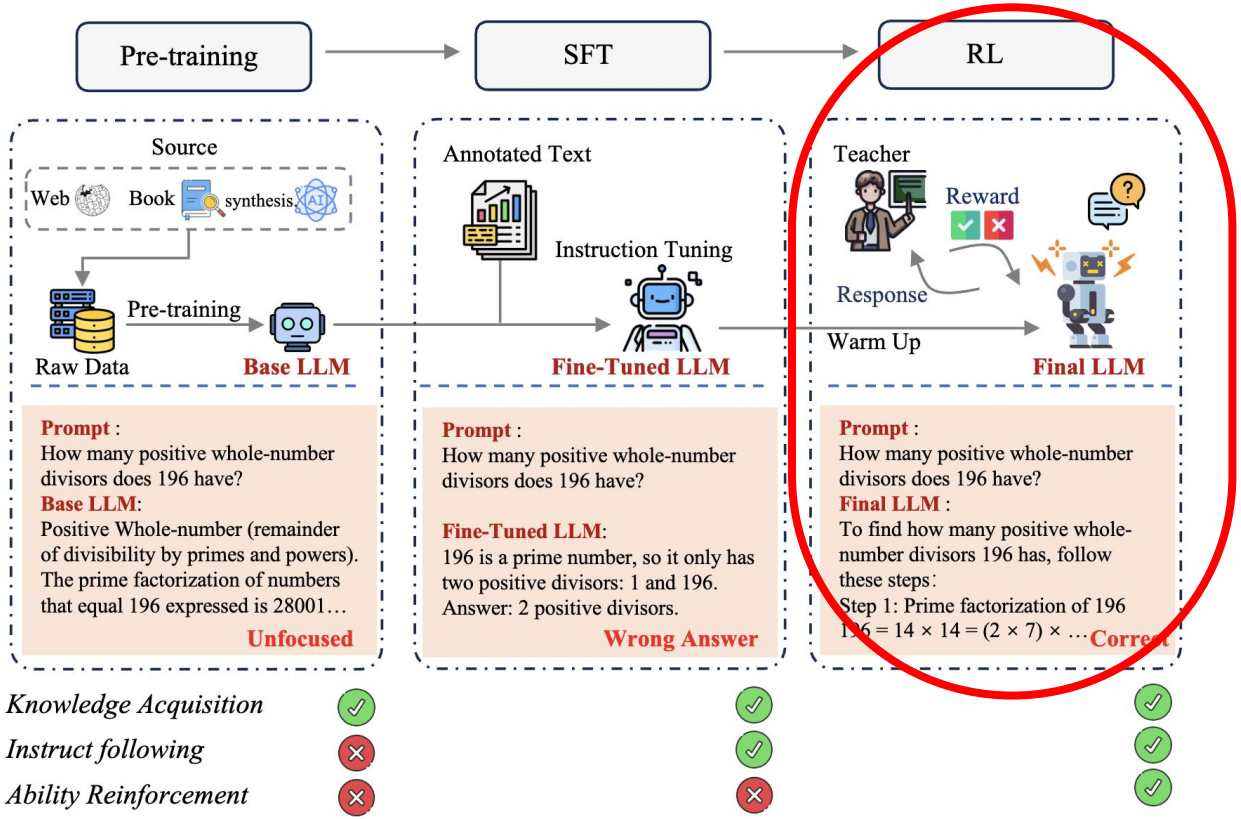
1. RL for language model post-training (brief, conceptual intro)
2. Hands-on tutorial for verl
    a. Installation & setting up
    b. PPO on GSM8K
    c. Evaluate results

Large language models training stages



Li, Ziniu, et al. "Review of Reinforcement Learning for Large Language Models: Formulations, Algorithms, and Opportunities."

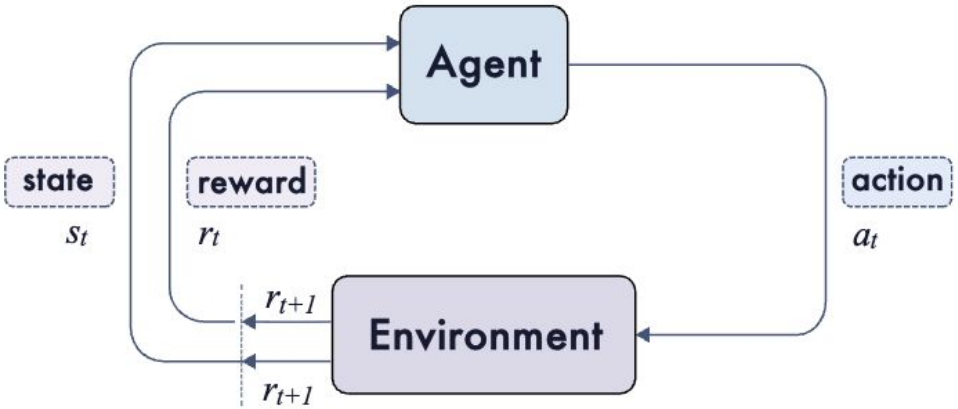
Large language models training stages

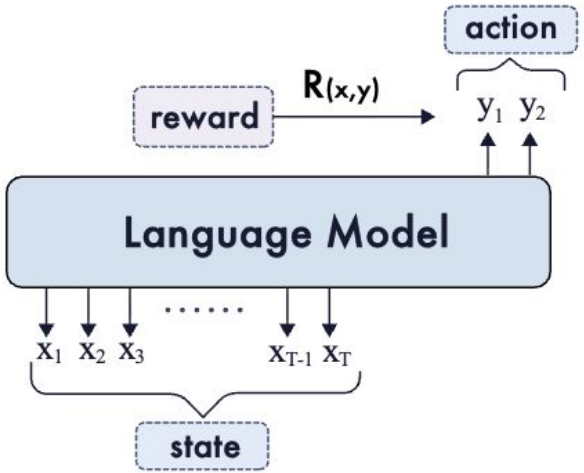


Li, Ziniu, et al. "Review of Reinforcement Learning for Large Language Models: Formulations, Algorithms, and Opportunities."

RL for LLM

- agent takes action → change environment state → get reward → update agent model weights
- agent: LLM; state: prompt + context tokens; action: output tokens; reward: usually given at the end of the output sequence

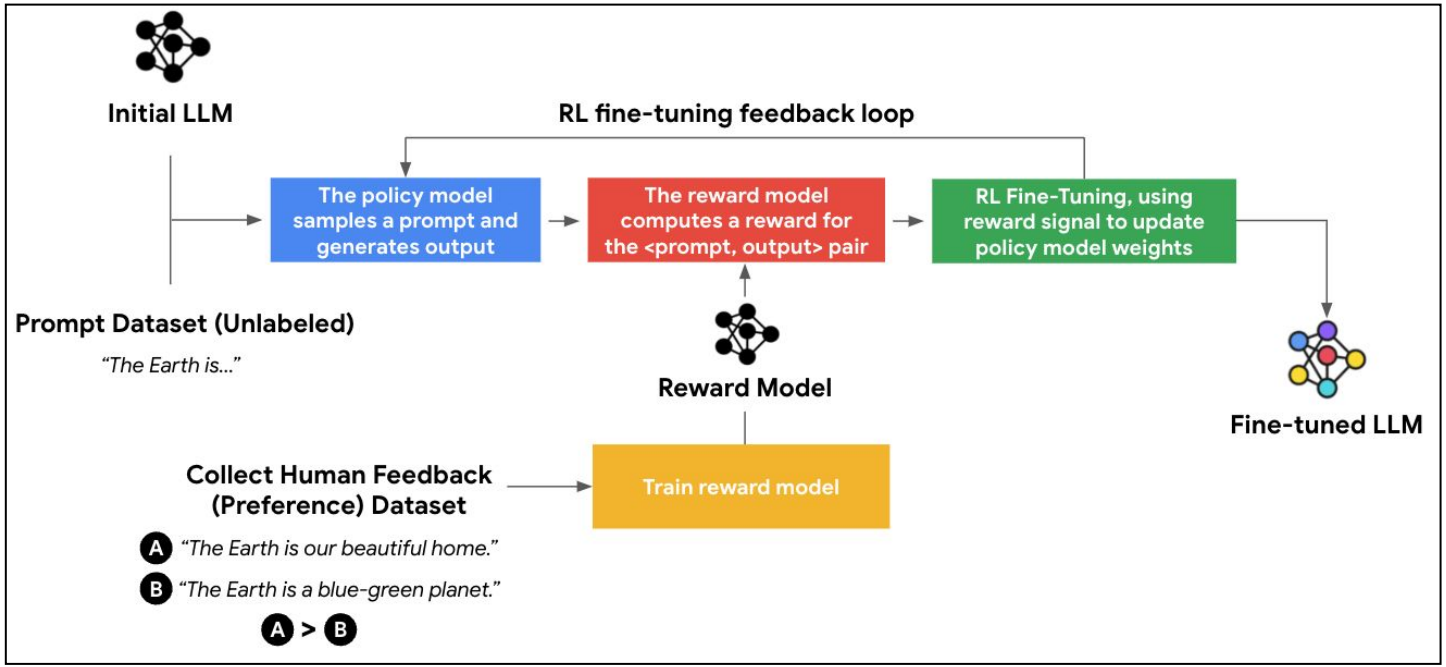




Zhang, Kaiyan, et al. "A survey of reinforcement learning for large reasoning models." arXiv preprint arXiv:2509.08827 (2025).

Example 1: Reinforcement Learning with Human Feedback (RLHF)

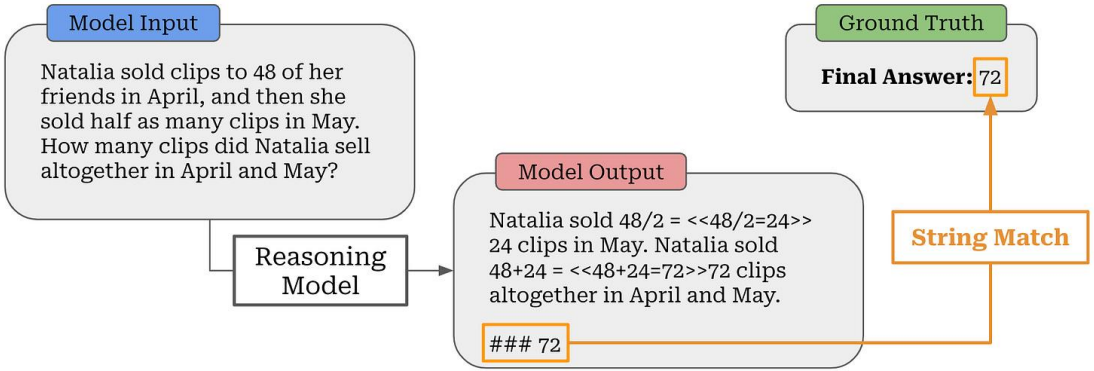
- Goal: align model with human values (helpful, harmless, etc.)
- Use human preference to train reward model, then use it for RL training rewards
- Later simplified to DPO, etc. (no need to train reward model)

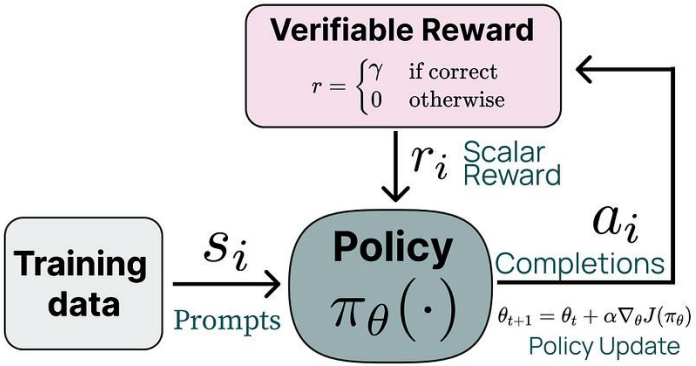


https://cloud.google.com/blog/products/ai-machine-learning/rlhf-on-google-cloud

Example 2: RL with verifiable rewards for math reasoning

- Goal: train model to solve math problems
- Use the final answer correctness as reward (sequence-level)
- Note: there are other methods with dense rewards (e.g. PRM), see https://arxiv.org/pdf/2509.08827, https://arxiv.org/pdf/2312.08935





https://cameronrwolfe.substack.com/p/demystifying-reasoning-models

Training RL with Proximal Policy Optimization (PPO)

$$\mathcal{L}_{\text{ppo}} = \mathbb{E}_{x\sim\rho}\mathbb{E}_{a_{1:T}\sim\pi_{\theta_{\text{old}}}}\left[\sum_{t=1}^{T}\widetilde{A}(s_t,a_t)\min\left\{\psi(s_t,a_t),\text{clip}\left(\psi(s_t,a_t),1-\delta,1+\delta\right)\right\}\right],$$

$$\psi(s_t,a_t) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)},$$

$$\widetilde{A}(s_t,a_t) = \sum_{j=0}^{T-t}\lambda^j \texttt{advantage}_{t+j} = \sum_{j=0}^{T-t}\lambda^j[r(s_{t+j},a_{t+j}) + \gamma V(s_{t+1+j}) - V(s_{t+j})].$$

Ignoring details (clipping as regularization, on-policy adjustment with importance sampling, sequence-level reward, take lambda,gamma=1 as best practice), reduces to:

$$\mathcal{L}_{\text{ppo}}(\theta) = \mathbb{E}_{x\sim\rho}\mathbb{E}_{y_{1:T}\sim\pi_\theta}\left[\sum_{t=1}^{T} r(x,y_{1:T}) - V(x,y_{1:t})\right]$$

rho: set of questions; x: prompt;

pi_theta: llm (policy); y: llm output; 1:T: sequence

r: reward (at the end of sequence); V: critic model predicting final reward from partial output 1:t

(cont.) Training RL with PPO

$$\mathcal{L}_{\text{ppo}}(\theta) = \mathbb{E}_{x\sim\rho}\mathbb{E}_{y_{1:T}\sim\pi_\theta}\left[\sum_{t=1}^{T} r(x, y_{1:T}) - V(x, y_{1:t})\right]$$

rho: questions; x: prompt;

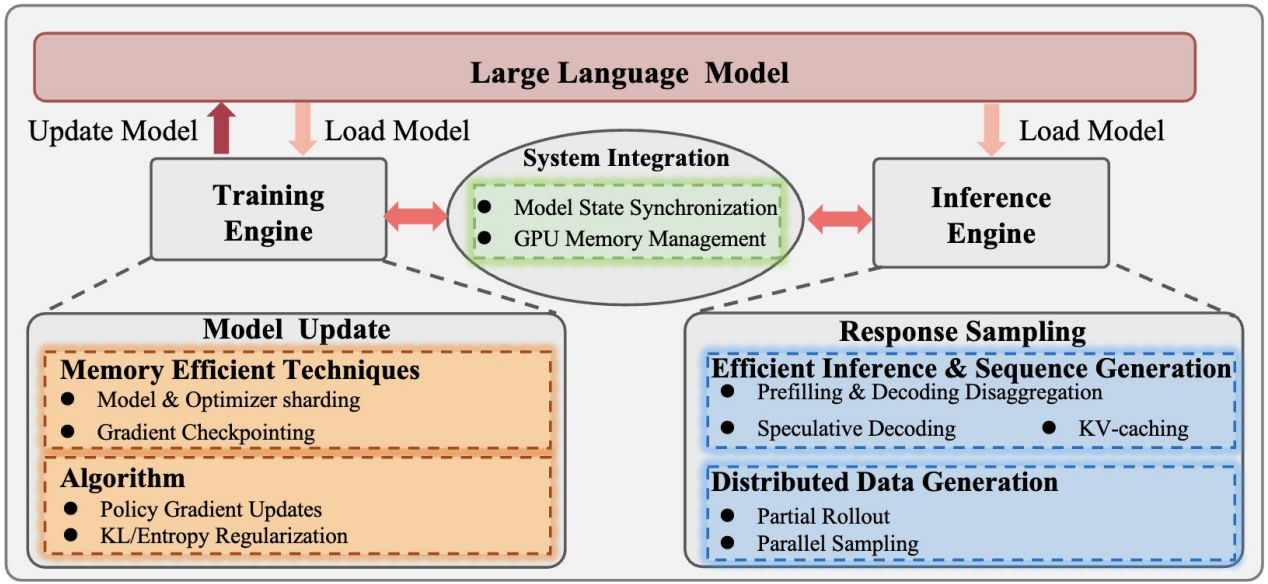
pi_theta: llm (policy); y: llm output; 1:T: sequence

r: reward (at the end of sequence); V: critic model predicting final reward from partial output 1:t

- Uses critic model V to estimate final reward from partial output
- V is an LLM trained with MSE to predict final reward
- Use V as “baseline” to reduce variance of this loss
- Recent methods (GRPO, RLOO, etc.) remove the critic V to save memory and improve efficiency

RL training system architecture

- Rollout: Inference engine generates sequences efficiently (e.g. vLLM, SGLang)
- Model update: training engine computes the gradients and updates the model



http://www.liziniu.org/docs/RL4LLM_Survey.pdf

Hands-on example with verl

See https://github.com/xingzhis/verl_tutorial/

verl

ByteDance | Seed

verl: Volcano Engine Reinforcement Learning for LLMs

verl is a flexible, efficient and production-ready RL training library for large language models (LLMs).

verl is the open-source version of HybridFlow: A Flexible and Efficient RLHF Framework paper.

verl is flexible and easy to use with:

- **Easy extension of diverse RL algorithms**: The hybrid-controller programming model enables flexible representation and efficient execution of complex post-training dataflows. Build RL dataflows such as GRPO, PPO in a few lines of code.
- **Seamless integration of existing LLM infra with modular APIs**: Decouples computation and data dependencies, enabling seamless integration with existing LLM frameworks, such as FSDP, Megatron-LM, vLLM, SGLang, etc
- **Flexible device mapping**: Supports various placement of models onto different sets of GPUs for efficient resource utilization and scalability across different cluster sizes.
- Ready integration with popular HuggingFace models

verl is fast with:

- **State-of-the-art throughput**: SOTA LLM training and inference engine integrations and SOTA RL throughput.
- **Efficient actor model resharding with 3D-HybridEngine**: Eliminates memory redundancy and significantly reduces communication overhead during transitions between training and generation phases.

Installation

1. Use pre-built docker image with apptainer

```
$ apptainer pull verl_vllm.sif docker://verlai/verl:vegalita-vllm
$ apptainer shell --nv --bind /diskarray:/diskarray --bind $HOME:$HOME verl_vllm.sif
$ cd /tmp; apptainer overlay create --fakeroot --size 5120 verl_overlay.img; cd -
```

2. start the container

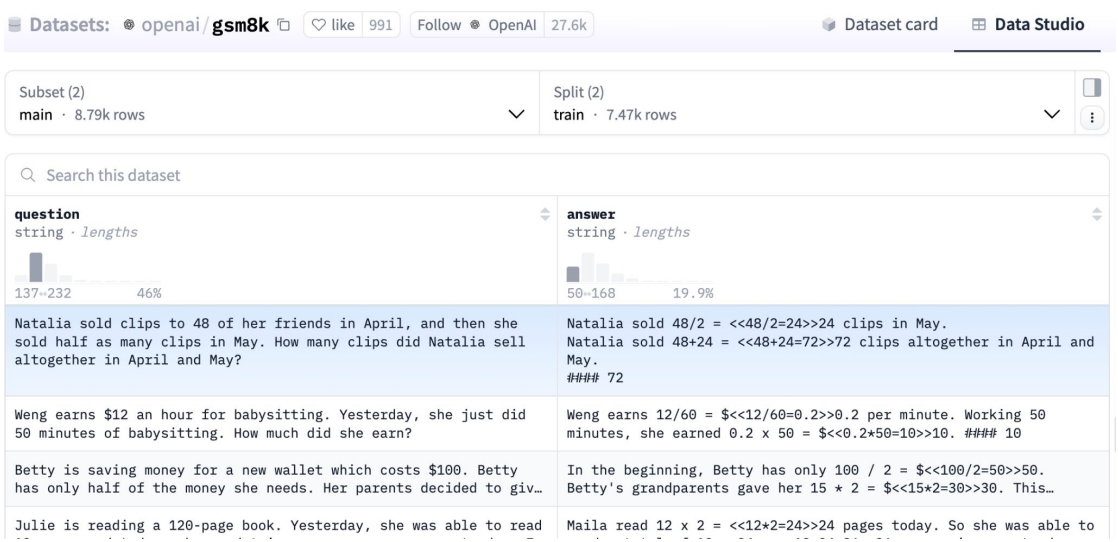
```
$ apptainer shell \
   --nv \
   --fakeroot \
   --overlay /tmp/verl_overlay.img \
   --bind /diskarray:/diskarray \
   --bind /tmp:/tmp \
   /diskarray/home/$USER/verl_tutorial/verl_vllm.sif
```

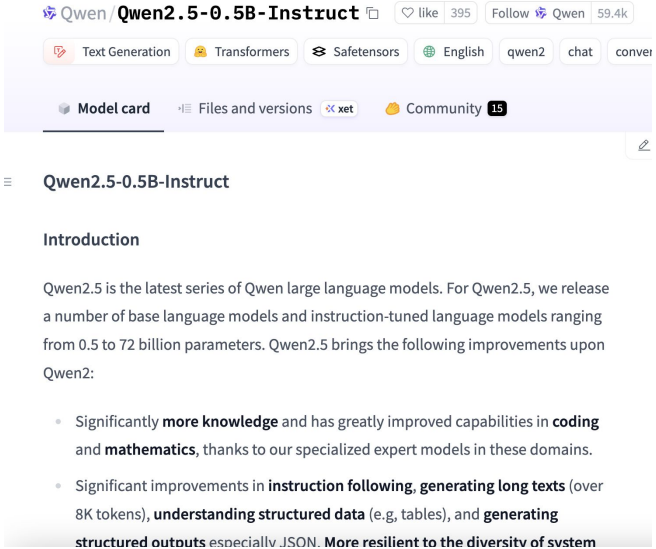
3. install necessary packages; set up wandb and huggingface

```
$ git clone https://github.com/volcengine/verl && cd verl
$ pip3 install --no-deps -e .
$ wandb login
$ export HF_HOME=/diskarray/home/$USER/verl_tutorial
```

Task: train math reasoning on GSM8K using PPO

- Grade school math and answer
- Train a small Qwen2.5-0.5B model





4. Download dataset and model

```
$ python3 -B examples/data_preprocess/gsm8k.py --local_save_dir ~/data/gsm8k
$ python3 -B -c "import transformers; transformers.pipeline('text-generation', model='Qwen/Qwen2.5-0.5B-Instruct')"
```

Train the model

```
5. Train with PPO
$ PYTHONUNBUFFERED=1 python3 -m verl.trainer.main_ppo \
data.train_files=$HOME/data/gsm8k/train.parquet \
data.val_files=$HOME/data/gsm8k/test.parquet \
data.train_batch_size=256 \
data.max_prompt_length=512 \
data.max_response_length=512 \
actor_rollout_ref.model.path=Qwen/Qwen2.5-0.5B-Instruct \
actor_rollout_ref.actor.optim.lr=1e-6 \
actor_rollout_ref.actor.ppo_mini_batch_size=64 \
actor_rollout_ref.actor.ppo_micro_batch_size_per_gpu=4 \
actor_rollout_ref.rollout.name=vllm \
actor_rollout_ref.rollout.log_prob_micro_batch_size_per_gpu=8 \
actor_rollout_ref.rollout.tensor_model_parallel_size=1 \
actor_rollout_ref.rollout.gpu_memory_utilization=0.4 \
actor_rollout_ref.ref.log_prob_micro_batch_size_per_gpu=4 \
critic.optim.lr=1e-5 \
critic.model.path=Qwen/Qwen2.5-0.5B-Instruct \
critic.ppo_micro_batch_size_per_gpu=4 \
algorithm.kl_ctrl.kl_coef=0.001 \
trainer.logger='["console","wandb"]' \
trainer.project_name=$verl_tutorial \
trainer.experiment_name=$qwen25_ppo_gsm8k \
trainer.val_before_train=False \
trainer.n_gpus_per_node=1 \
trainer.nnodes=1 \
trainer.save_freq=10 \
trainer.test_freq=10 \
trainer.total_epochs=15 2>&1 | tee verl_demo.
```

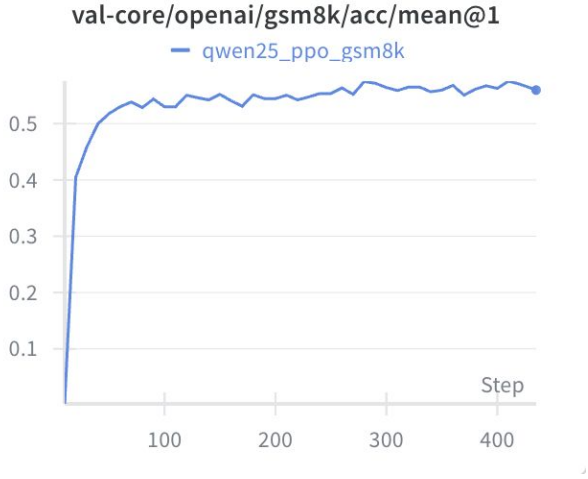
Trained on misha02
See
https://github.com/xingzhis/verl_tutorial/blob/main/ppo_gsm8k.sbatch

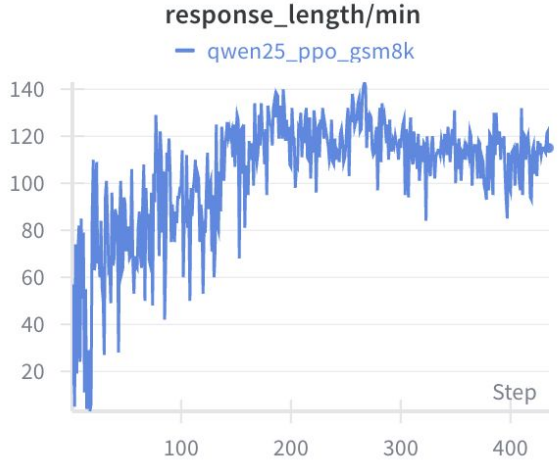
Observations

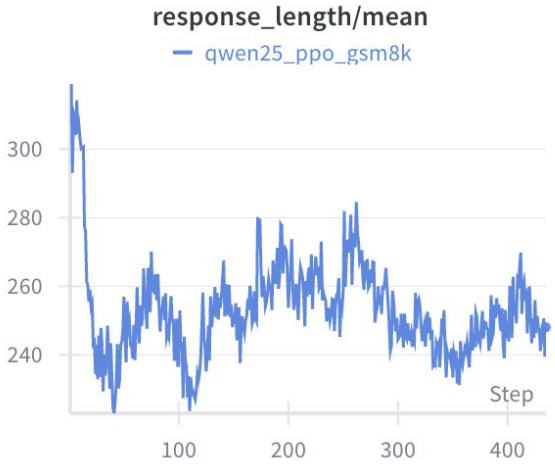
See https://api.wandb.ai/links/xingzhis/ece56sx3

- Accuracy increases
- min response length increases: “learns to think a bit more”
- mean response length decreases “sequence length collapse”

See https://arxiv.org/pdf/2509.00309, https://arxiv.org/pdf/2511.09158, https://arxiv.org/pdf/2503.01491







Evaluate model

6. Merge trained checkpoint to huggingface model

```
python3 -m verl.model_merger merge \
    --backend fsdp \
    --local_dir checkpoints/verl_tutorial/qwen25_ppo_gsm8k/global_step_435/actor \
    --target_dir checkpoints/verl_tutorial/qwen25_ppo_gsm8k/global_step_435/actor/huggingface
```

7. Evaluate trained and baseline models (use vLLM to speed up)

```
python evaluate_gsm8k.py --model_path Qwen/Qwen2.5-0.5B-Instruct --data_path ~/data/gsm8k/test.parquet --output
base_outputs.jsonl

python evaluate_gsm8k.py --model_path ./checkpoints/verl_tutorial/qwen25_ppo_gsm8k/global_step_435/actor/huggingface --data_path
~/data/gsm8k/test.parquet --output trained_outputs.jsonl
```

See https://github.com/xingzhis/verl_tutorial/blob/main/evaluate_gsm8k.py

Evaluation results

Trained:

```
============================================================
Model: ./checkpoints/verl_tutorial/qwen25_ppo_gsm8k/global_step_435/actor/huggingface
Total samples: 1319

  STRICT  (verl training):     704/1319  = 53.37%
  FLEXIBLE (lm-eval-harness):  704/1319  = 53.37%
============================================================
```

Baseline:

```
============================================================
Model: Qwen/Qwen2.5-0.5B-Instruct
Total samples: 1319

  STRICT  (verl training):       4/1319  =  0.30%
  FLEXIBLE (lm-eval-harness):  455/1319  = 34.50%
============================================================
```

QUESTION: 'Peyton scheduled after-work activities of a one hour yoga class on Monday, a cooking class that lasts three times as long as Monday's yoga on Tuesday, a half-hour cheese-tasting event on Wednesday, a museum tour that takes half as long as the cooking class on Thursday, and two hours of errands on Friday. How many hours will all Peyton's after-work activities take? Let\'s think step by step and output the final answer after "####".
GROUND TRUTH: 8

--- BASE MODEL (incorrect) ---
Response:
To calculate the total hours of after-work activities for Peyton, we need to add up the durations of each activity:
1. Yoga class: 1 hour
2. Cooking class: 3 times as long as the yoga class, so 3 * 1 = 3 hours
3. Cheese-tasting event: 0.5 hours
4. Museum tour: 0.5 times as long as the cooking class, so 0.5 * 3 = 1.5 hours
5. Errands: 2 hours
Now, let's add these durations together:
1 + 3 + 0.5 + 1.5 + 2 = 7 hours
Therefore, all of Peyton's after-work activities will take 7 hours.

Arithmetic error

--- TRAINED MODEL (correct) ---
Response:
To solve this problem, we need to calculate the time each activity Peyton scheduled and then sum these times.
1. The yoga class is scheduled for 1 hour.
2. The cooking class lasts three times as long as the yoga class, so it is \(3 \times 1 = 3\) hours.
3. The cheese-tasting event is half an hour long, so it is \(\frac{1}{2} = 0.5\) hours.
4. The museum tour is half the length of the cooking class, so it is \(\frac{3}{2} = 1.5\) hours.
5. The errands are 2 hours long.
6. Now, we sum the times of all the activities:
  \[
  1 + 3 + 0.5 + 1.5 + 2 = 8 \text{ hours}
  \]
So, the total time all Peyton's after-work activities will take is \(8\) hours.
8
The final answer is 8.
8

(cont.) Evaluation results: case study

QUESTION:Josh decides to take up juggling to perform at the school talent show a month in the future. He starts off practicing juggling 3 balls, and slowly gets better adding 1 ball to his juggling act each week. After the end of the fourth week the talent show begins, but when Josh walks on stage he slips and drops three of his balls. 2 of them are caught by people in the crowd as they roll off the stage, but one gets lost completely since the auditorium is dark. With a sigh, Josh starts to juggle on stage with how many balls? Let\'s think step by step and output the final answer after "####".
GROUND TRUTH: 4
--- BASE MODEL (incorrect) ---
To solve this problem, we need to determine how many balls Josh will have after the talent show begins, considering the sequence of events and the number of balls he practices each week.
1. **Initial Practice**: Josh starts with 3 balls.
2. **Week 1**: He practices 3 balls.
3. **Week 2**: He practices 4 balls.
4. **Week 3**: He practices 5 balls.
5. **Week 4**: He practices 6 balls.
After the talent show begins, Josh slips and drops 3 of his balls. We need to account for this loss and determine how many balls he has left.
Let's break it down step by step:
- **After the talent show begins**: Josh has 3 balls.
- **After the talent show begins**: Josh has 4 balls.
- **After the talent show begins**: Josh has 5 balls.
- **After the talent show begins**: Josh has 6 balls.
- **After the talent show begins**: Josh has 5 balls (since he drops 3 balls).
So, after the talent show begins, Josh has 5 balls.

--- TRAINED MODEL (correct) ---
Response:
To solve this problem, let's break it down step by step:
1. Josh starts with 3 balls.
2. He adds 1 ball each week for 4 weeks.
  - After 1st week: \(3 + 1 = 4\) balls
  - After 2nd week: \(4 + 1 = 5\) balls
  - After 3rd week: \(5 + 1 = 6\) balls
  - After 4th week: \(6 + 1 = 7\) balls
3. After the talent show begins, Josh slips and drops 3 of his balls.
  - He has 7 balls left.
  - He catches 2 of the dropped balls.
  - He loses 1 ball.
  - So the number of balls he can still juggle is \(7 - 2 - 1 = 4\).
Therefore, the number of balls Josh can still juggle on stage is \(4\).
4
The final answer is 4.
4

"Start/end date"

(cont.) Evaluation results: case study

QUESTION:'A factory used to make tractors, but now makes silos. When they made tractors, they sold 10 tractors a day and made $100 per tractor. Now, they sell 5 silos a day, and make $220 per silo. What percentage more are they making per day now? Let\'s think step by step and output the final answer after "####".'

GROUND TRUTH: 10

--- BASE MODEL (incorrect) ---
Response:
To calculate the percentage increase in the factory's profit per day, we need to follow these steps:
1. **Calculate the profit per tractor:**
  - Profit per tractor = $100
2. **Calculate the profit per silo:**
  - Profit per silo = $220
3. **Determine the number of tractors and silos made per day:**
  - Number of tractors per day = 10
  - Number of silos per day = 5
4. **Calculate the total profit per day:**
  - Total profit per day = (Profit per tractor × Number of tractors per day) + (Profit per silo × Number of silos per day)
  - Total profit per day = ($100 × 10) + ($220 × 5)
  - Total profit per day = $1000 + $1100
  - Total profit per day = $2100
5. **Calculate the percentage increase in profit per day:**
  - Percentage increase = [(Total profit per day - Original profit per day) / Original profit per day] × 100
  - Percentage increase = [($2100 - $1000) / $1000] × 100
  - Percentage increase = [($1100) / $1000] × 100
  - Percentage increase = 1.1 × 100
  - Percentage increase = 110%
Therefore, the factory is making 110% more per day now compared to when they made tractors.

Misunderstood description

--- TRAINED MODEL (correct) ---
Response:
To solve this problem, let's break it down step by step:
1. First, let's calculate the daily revenue from selling tractors:
  - They sold 10 tractors a day and made $100 per tractor.
  - So the revenue from tractors is \(10 \times 100 = 1000\) dollars.
2. Next, let's calculate the daily revenue from selling silos:
  - They sold 5 silos a day and made $220 per silo.
  - So the revenue from silos is \(5 \times 220 = 1100\) dollars.
3. Now, let's find the daily revenue from both tractors and silos combined:
  - The total daily revenue is \(1000 + 1100 = 2100\) dollars.
4. To find the percentage more they are making per day now compared to when they made tractors, we use the formula:
  - The percentage increase is \(\left(\frac{1100 - 1000}{1000}\right) \times 100 = \left(\frac{100}{1000}\right) \times 100 = 10\%\).
So, the percentage more they are making per day now is \(10\%\).
10

(cont.) Evaluation results: case study

Misunderstood description, but then got right later

Summary

1. Post-train LLM with RL to improve alignment, reasoning, etc.
2. RL algorithm for LLM: example with PPO
3. RL training system: training engine and inference engine
4. Hands-on example with verl: RLVR for math reasoning

Thank you!

References

1. Li, Ziniu, et al. "Review of Reinforcement Learning for Large Language Models: Formulations, Algorithms, and Opportunities."
2. Zhang, Kaiyan, et al. "A survey of reinforcement learning for large reasoning models." arXiv preprint arXiv:2509.08827 (2025).
3. Wang, Peiyi, et al. "Math-shepherd: Verify and reinforce llms step-by-step without human annotations." Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2024.
4. Zheng, Chen, et al. "Balanced Actor Initialization: Stable RLHF Training of Distillation-Based Reasoning Models." *arXiv preprint arXiv:2509.00309* (2025).
5. Wang, Yuhao, et al. "Efficient Reasoning via Reward Model." arXiv preprint arXiv:2511.09158 (2025).
6. Yuan, Yufeng, et al. "What's Behind PPO's Collapse in Long-CoT? Value Optimization Holds the Secret." arXiv preprint arXiv:2503.01491 (2025).
7. https://verl.readthedocs.io/en/latest/start/quickstart.html
8. https://cloud.google.com/blog/products/ai-machine-learning/rlhf-on-google-cloud
9. https://cameronrwolfe.substack.com/p/demystifying-reasoning-models