DESIGNING EFFECTIVE RL GYMS

Kyle Avery

September 2025

OUTFLANK

clear advice with a hacker mindset

WHOAMI

Kyle Avery – @kyleavery_

- R&D @ Outflank
- Red team background
- AI hobbyist

OUTFLANK

- Outflank Security Tooling (OST)
- Red Teaming Services

INTRODUCTION

Supervised learning requires many examples

- Creating a large, diverse dataset for specialized tasks is difficult
  – Collecting 1K+ examples often isn't practical

Question	Response
What is the capital of France?	The capital of France is Paris. Located in the north-central part of the country, Paris is not only…

INTRODUCTION

Reinforcement learning doesn’t require traditional examples

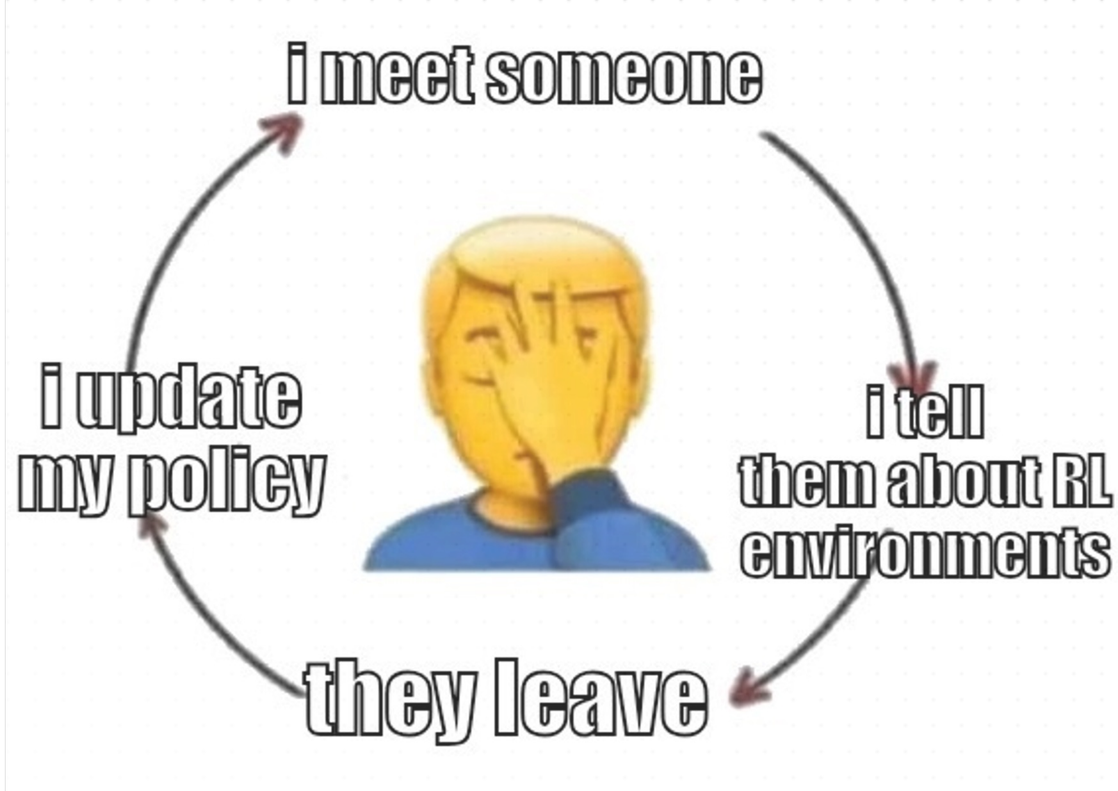
- Train a model using trial and error
- Model is “rewarded” for “correct” actions
- Possible to outperform human labelers

Models trained with RL:

- AlphaZero (Google DeepMind, 2017)
- o1 (OpenAI, 2024)
- Dante (Outflank, 2025)

AGENDA

- Verifiable tasks
- Reward functions
- Training an LLM with RLVR
- Metrics



VERIFIABLE TASKS

OUTFLANK

clear advice with a hacker mindset

VERIFIABLE TASKS – MULTIPLE CHOICE

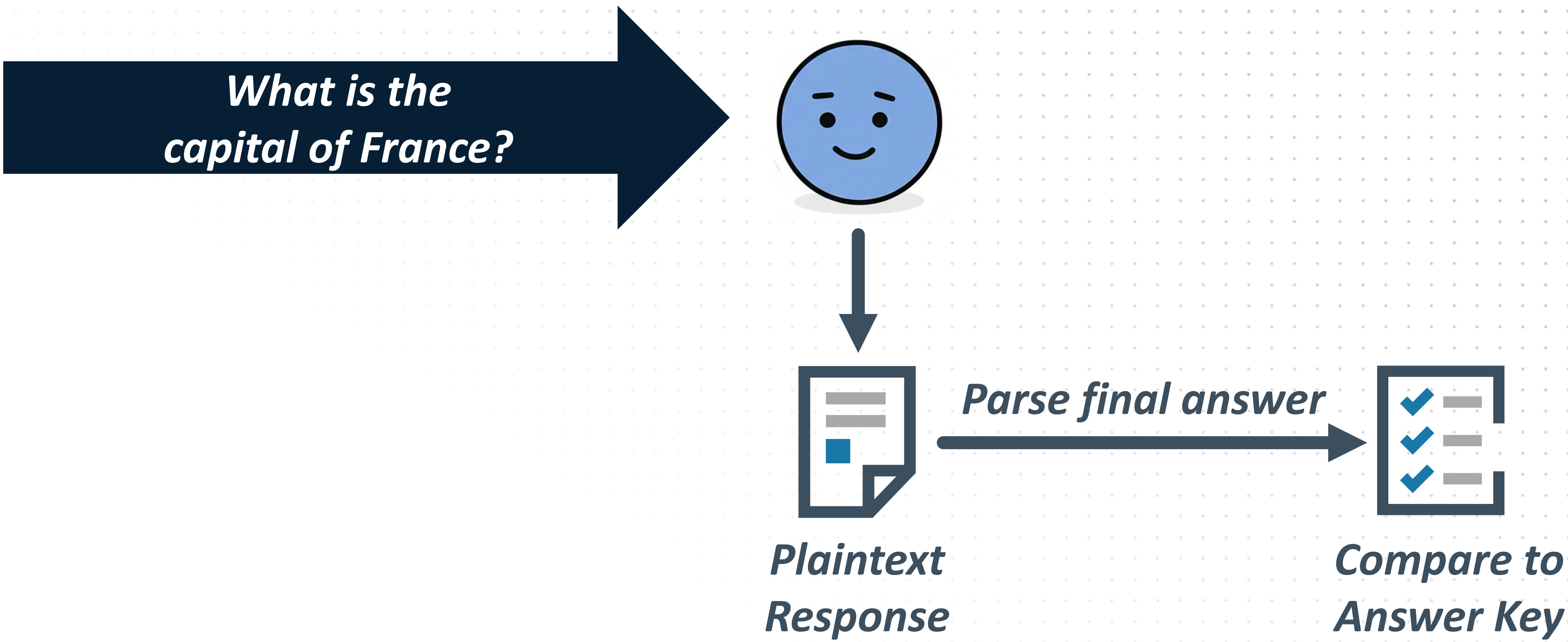
How are tasks verified?

- Simple example: Multiple choice questions

Question	Response	Answer
What is the capital of France? A. London B. Paris C. Rome D. Berlin	The capital of France is Paris. Located in the north-central part of the country, Paris is not only… **Unused for RLVR!**	B

VERIFIABLE TASKS – MULTIPLE CHOICE

How does the LLM fit into this?



VERIFIABLE TASKS – MULTIPLE CHOICE

Multiple choice prompt:

```
Give your final answer inside \boxed{}.

What is the capital of France?
A.  London
B.  Paris
C.  Rome
D.  Berlin
```

Output example:

```
The capital of France is Paris so the answer is:

\boxed{B}
```

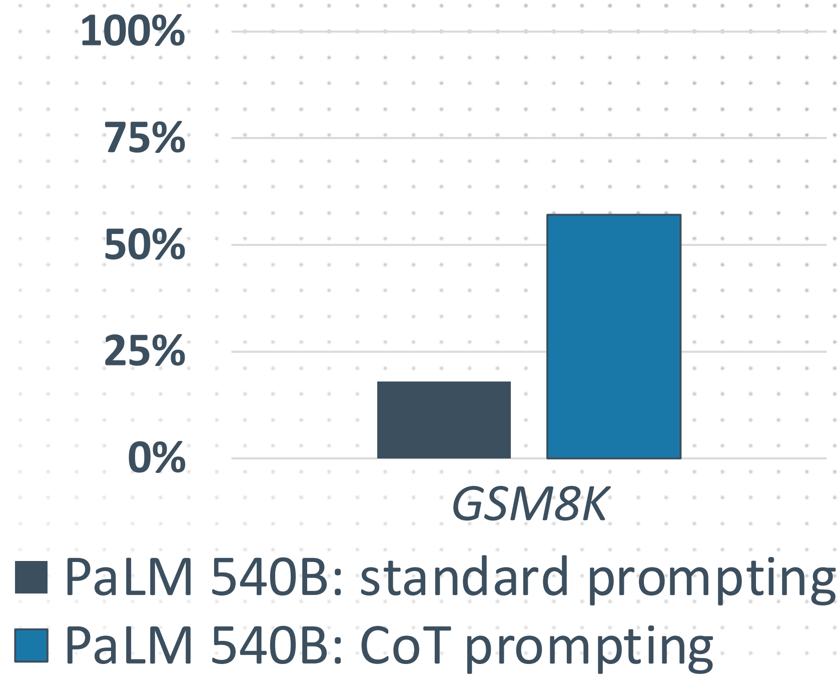
VERIFIABLE TASKS – REASONING

Does prompting impact LLM performance?

Q: What is 3 - 2?
A: If 3-1=2 and 2-1=1, then 3-2 is 1.

Q: What is 2 + 2?

A: If 2+1=3 and 3+1=4, 2+2 is 4



VERIFIABLE TASKS – REASONING

RLVR is often paired with chain-of-thought prompting

- Reasoning-heavy problems, like math and coding, benefit significantly

```
Sarah has 24 stickers. She gives away 1/3 of them to her friends and then buys 8
more. How many stickers does she have now?
```

```
Let me solve this step by step:
Sarah starts with 24 stickers
She gives away 1/3 of them, so she
gives away 1/3 = 0.33 stickers
After giving away stickers: 24 - 0.33
= 23.67 stickers
She buys 8 more: 23.67 + 8 = 31.67
stickers
Therefore, Sarah has 31.67 stickers.

\boxed{31.67}
```

```
Let me work through this step by step:
Sarah starts with 24 stickers
She gives away 1/3 of them to friends
1/3 of 24 = 24 ÷ 3 = 8 stickers given
After giving away stickers: 24 - 8 =
16 stickers remaining
She then buys 8 more stickers
Final amount: 16 + 8 = 24 stickers
Therefore, Sarah has 24 stickers.

\boxed{24}
```

VERIFIABLE TASKS – CODING

How are tasks verified?

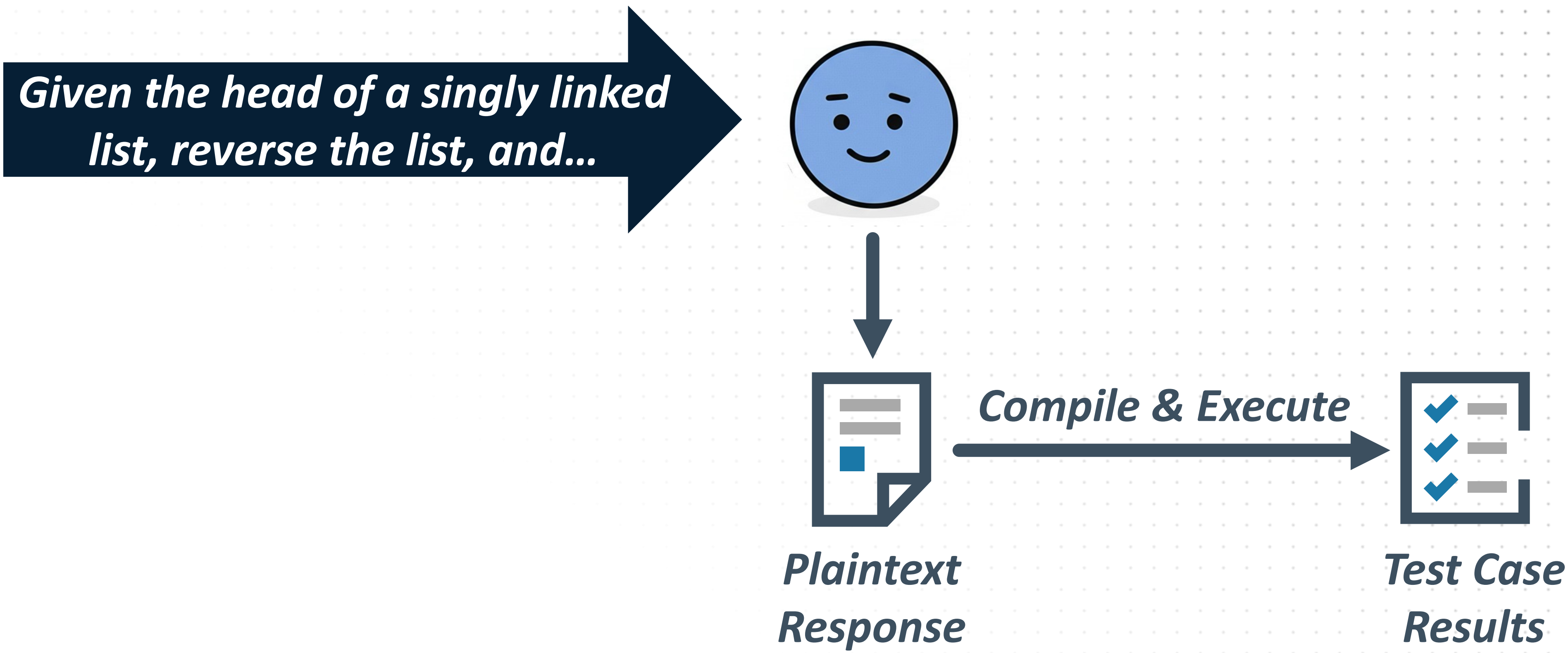
- More interesting example: LeetCode-style coding questions

Question	Response	Tests
Given the head of a singly linked list, reverse the list, and return the reversed list.	To reverse a singly linked list, you can iterate through the list and reverse the direction of the next pointers. Here's a Python implementation… **Unused for RLVR!**	[1,2,3,4,5] [5,4,3,2,1] [1,2] [2,1]

VERIFIABLE TASKS – CODING

How does the LLM fit into this?

- Coding – Execute program, validate test cases



VERIFIABLE TASKS – CODING

Coding prompt:

```
Think step-by-step inside <think>...</think> tags. Then, give your final answer
between triple backticks.

Given the head of a singly linked list, reverse the list, and return the reversed
list.
```

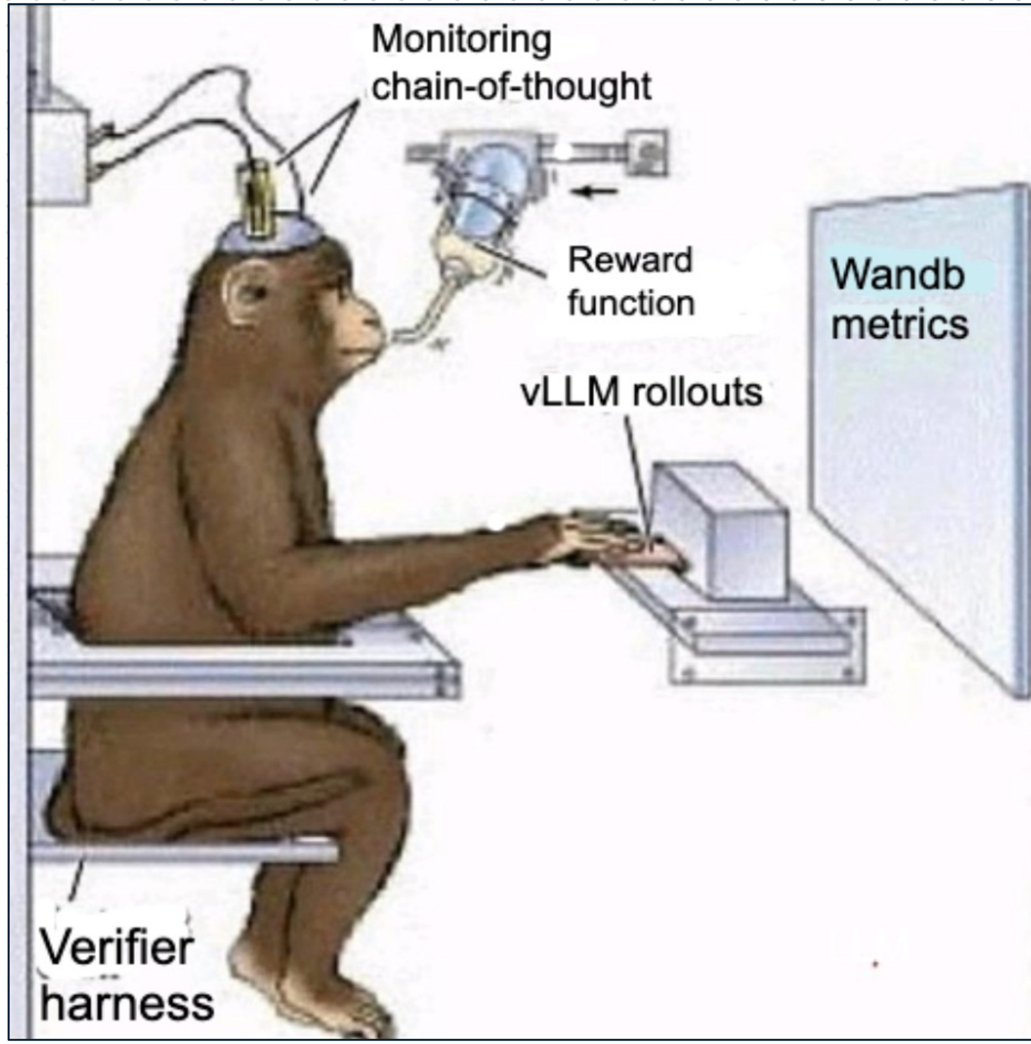
Output example:

````
<think>
Chain-of-thought reasoning…
</think>

```
Solution source code…
```
````

CREATING A VERIFIER

Steps to create a verifier:

1. Identify a candidate task
2. Plan an input and output format
3. Write the verifier program



STEP 1 – IDENTIFY A TASK

Verifier's Law

1. Objective truth
2. Fast to verify
3. Scalable to verify
4. Low noise
5. Continuous reward

https://www.jasonwei.net/blog/asymmetry-of-verification-and-verifiers-law

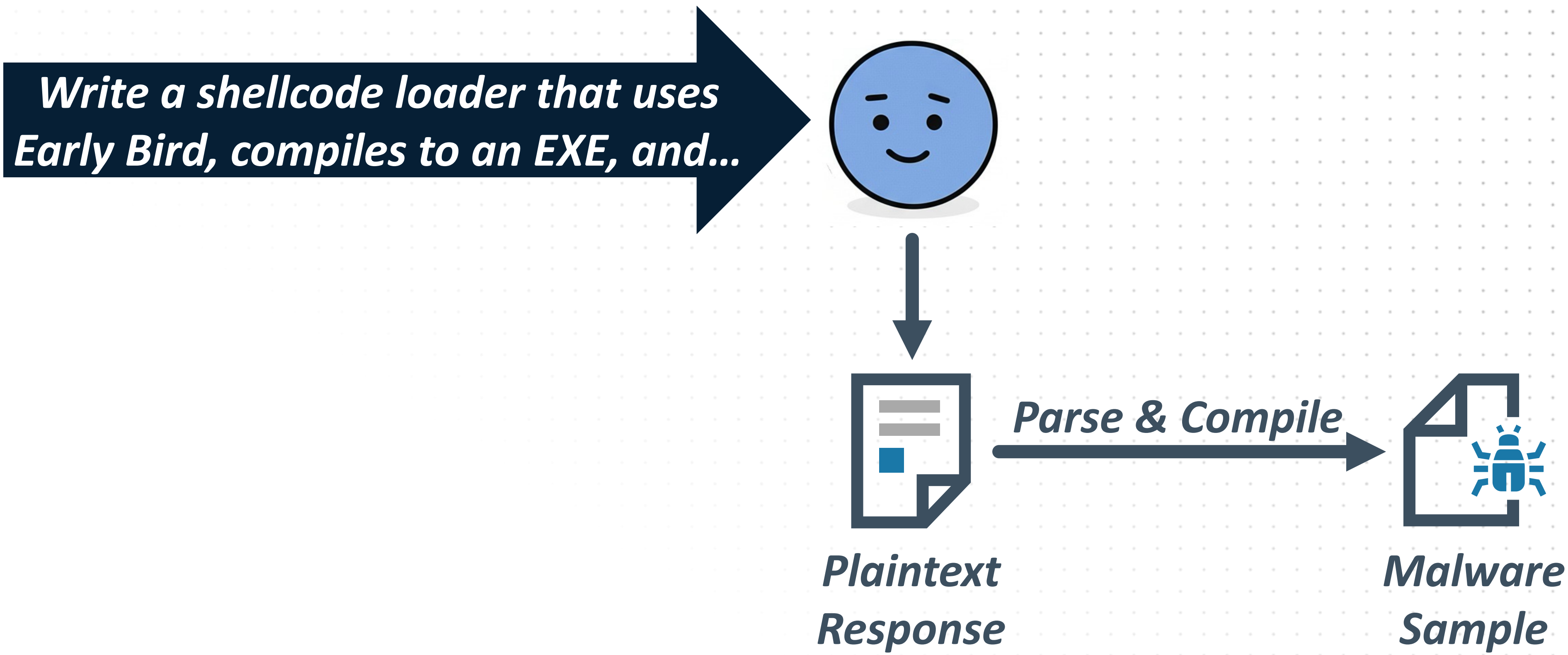
STEP 1 – IDENTIFY A TASK

Malware development fits nicely into Verifier’s Law:

1. Objective truth – Fewer alerts is always better
2. Fast to verify – Sandbox execution without human interaction
3. Scalable to verify – Cloud compute scales easily
4. Low noise – Training and evaluation target the same products
5. Continuous reward – Reward using alert count and severity

STEP 2 – PLAN INPUT/OUTPUT FORMAT

Plan LLM input/output before building a verifier:



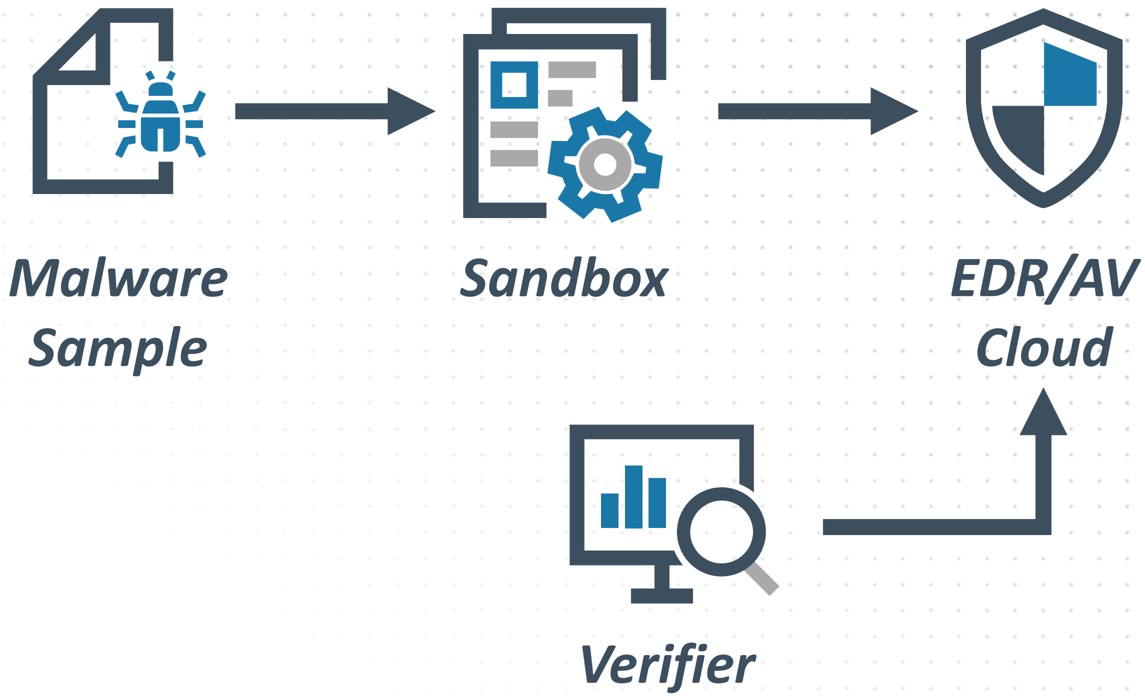
STEP 2 – PLAN INPUT/OUTPUT FORMAT

Output example:

```
<project>
  <src>
    <file name="prepare.py">
<![CDATA[
Python script to encode shellcode…
]]>
    </file>
    <file name="main.cpp">
<![CDATA[
Shellcode loader source…
]]>
    </file>
    <file name="Makefile">
<![CDATA[
Makefile to compile loader…
]]>
    </file>
  </src>
  <command>make</command>
</project>
```

STEP 3 – WRITE THE VERIFIER PROGRAM

Execute payload in a sandbox and check AV/EDR alerts:



STEP 2 – PLAN INPUT/OUTPUT FORMAT

Output example:

```
<project>
  <src>
    <file name="prepare.py">
<![CDATA[
Python script to encode shellcode…
]]>
    </file>
    <file name="main.cpp">
<![CDATA[
Shellcode loader source…
]]>
    </file>
    <file name="Makefile">
<![CDATA[
Makefile to compile loader…
]]>
    </file>
  </src>
  <command>make</command>
</project>
```

REWARD FUNCTIONS

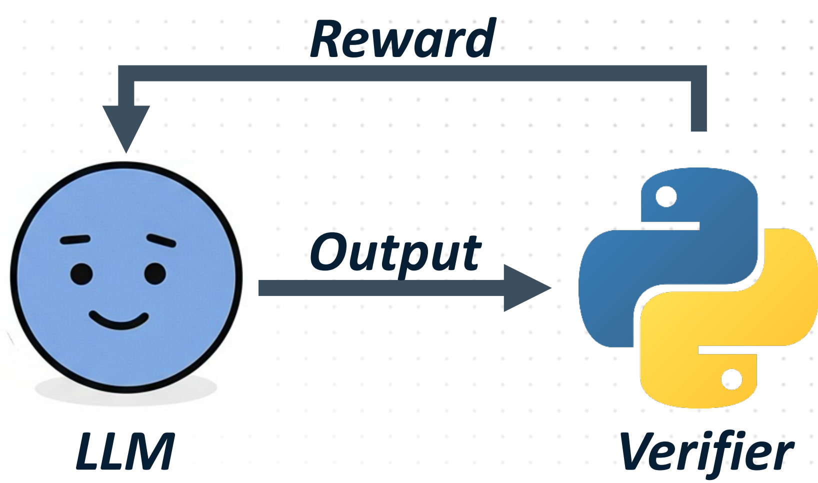
OUTFLANK

clear advice with a hacker mindset

REWARD FUNCTIONS

The final output of a verifier environment is a single number

- Must represent the overall performance of a response



REWARD FUNCTIONS

Design goals:

- Allow the model to discover any approach that gets the correct answer
- Provide some opportunities for iterative improvement
  - Score may comprise multiple weighted sub-scores

Reward hacking:

- Probability of responses in multiple choice questions
- Failing open vs. closed

EXAMPLE REWARD FUNCTION

Designing a simple reward function for multiple choice:

- Possible scores: 0, 0.1, 1.1

Format	***Answer***
Correct: 0.1	Correct: 1
Incorrect: 0	Incorrect: 0

EXAMPLE REWARD FUNCTION

Designing a reward function for LeetCode-style coding questions:

- Possible scores for two test cases: 0, 0.1, 0.3, 2.3, 3.3

Format	***Valid Code***	***Test Cases***
Correct: 0.1 Incorrect: 0	Compiles: 0.2 Fails: 0	+1 for each passing test

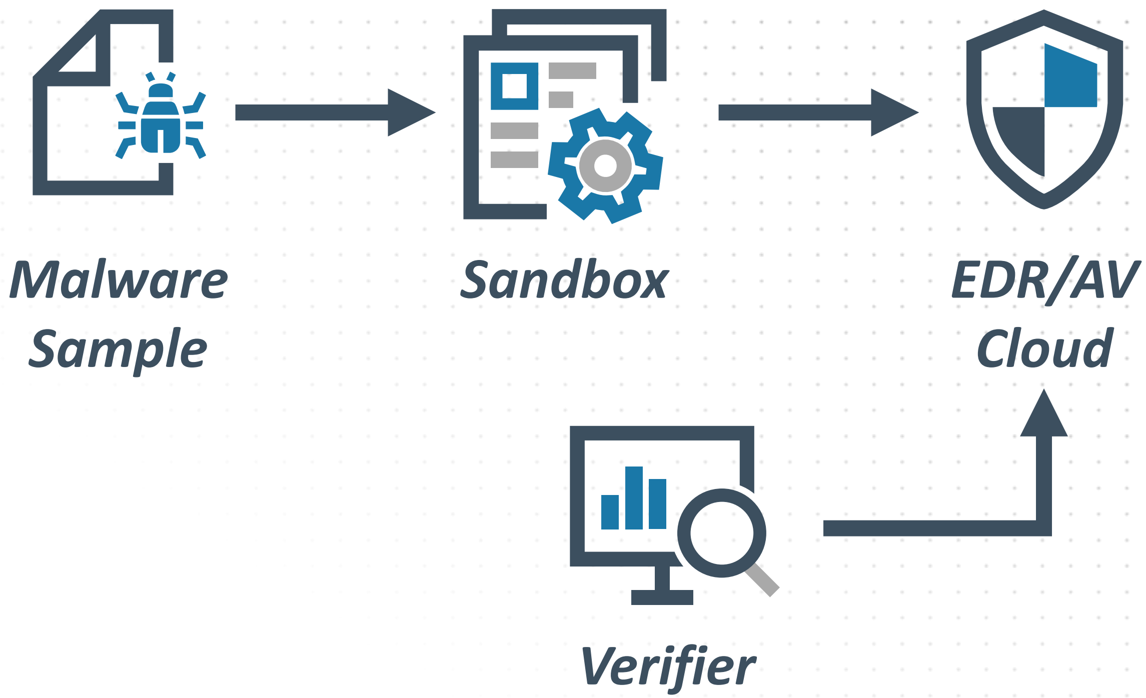
EXAMPLE REWARD FUNCTION

Designing a reward function for malware development:

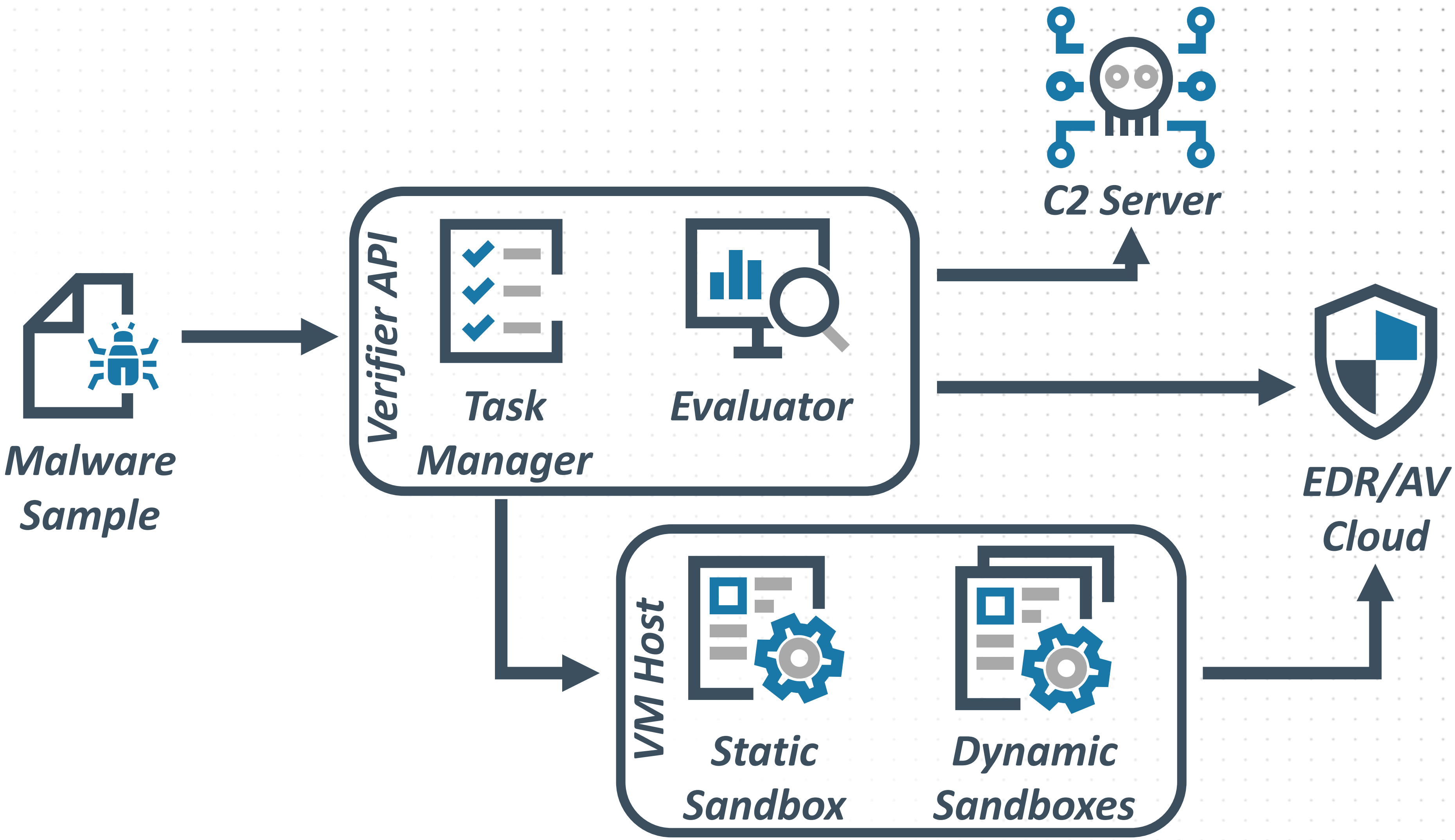
Format	+	***Compilation***	+	***Functionality***	+	***Evasion***
Correct: 0.1 Incorrect: 0		Compiles: 0.2 Fails: 0		Functions: 1.0 No callback: 0		Start at 3.0 For each alert: 0x for high 0.5x for medium 0.8x for low

REVISITING THE VERIFIER PROGRAM

Original verifier program doesn’t account for functionality!



REVISITING THE VERIFIER PROGRAM



TRAINING AN LLM

OUTFLANK

clear advice with a hacker mindset

TRAINING AN LLM WITH RLVR

Training steps:

1. Integrate your verifier into a training framework
2. Select an open-source LLM to train
3. Rent GPUs

STEP 1 – TRAINING FRAMEWORKS

willccbb/verifiers

- Easy to add new verifiers, many examples

huggingface/open-r1

- Supports SFT, a bit more robust

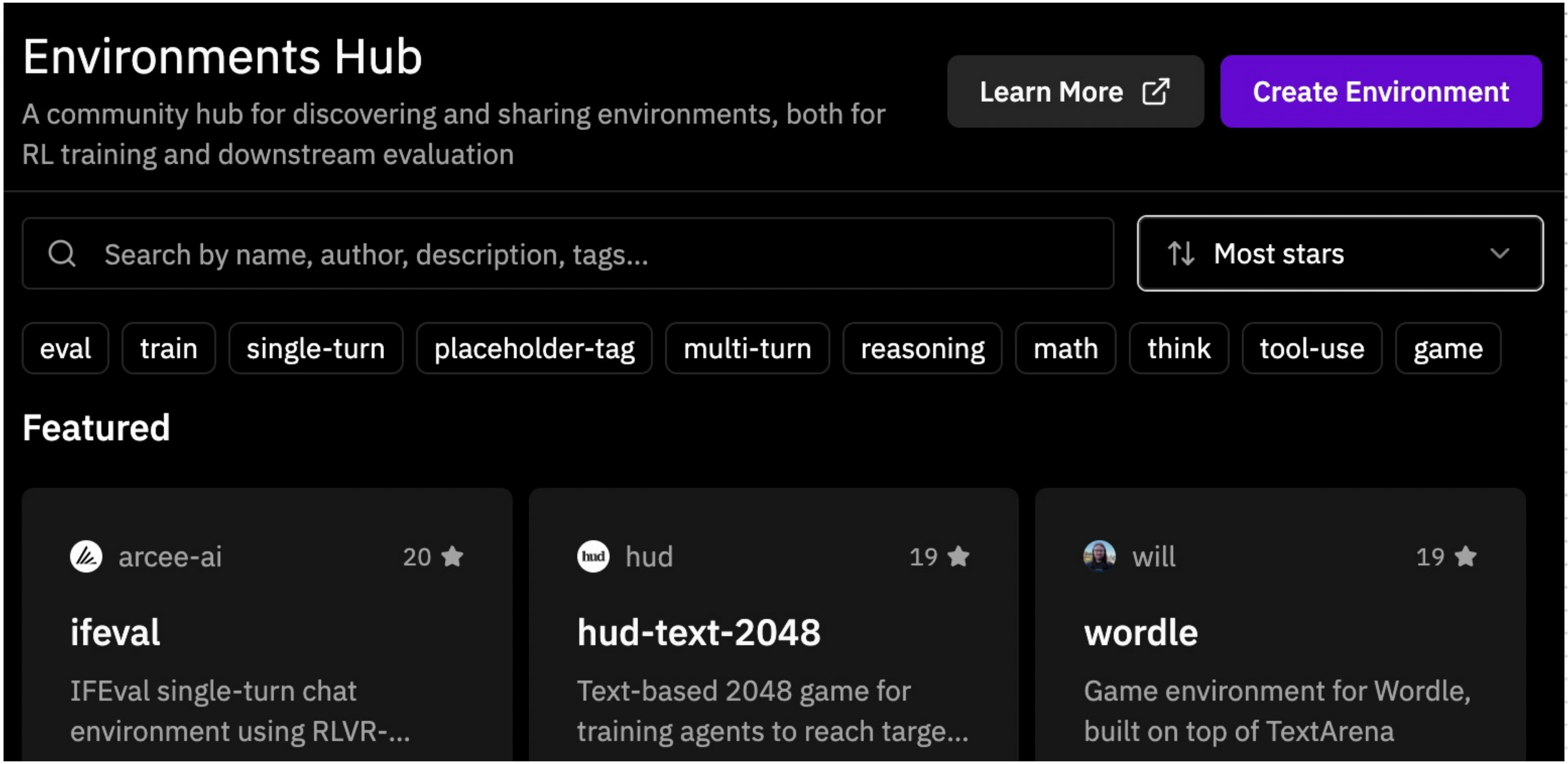
STEP 1 – TRAINING FRAMEWORKS

Create an “environment” for willccbb/verifiers:

```python
prompt = "Summarize the text in 3 sentences. Respond in the following format:"

def load_environment(**kwargs) -> vf.Environment:
    dataset = load_dataset("agentlans/wikipedia-paragraphs", split="train")

    parser = vf.XMLParser(["think", "answer"], answer_field="answer")
    system_prompt = f"{prompt}\n{parser.get_format_str()}"

    def sentence_reward_func(completion, **kwargs) -> float:
        response = parser.parse_answer(completion) or ""
        return 1.0 if len(response.split(".")) == 3 else 0.0

    rubric = vf.Rubric(
        parser=parser,
        funcs=[sentence_reward_func, parser.get_format_reward_func()],
        weights=[1.0, 0.2],
    )

    return vf.SingleTurnEnv(dataset, system_prompt, parser, rubric)
```

STEP 1 – TRAINING FRAMEWORKS

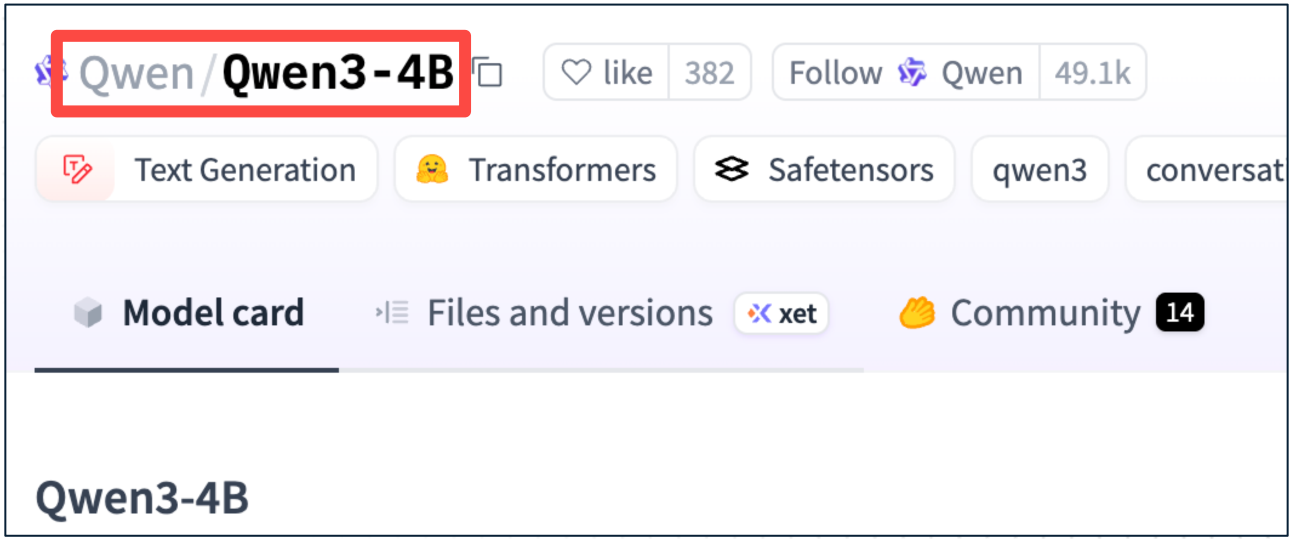
Find examples on the Environments Hub from Prime Intellect:



STEP 2 – SELECT A MODEL

Browse models on Hugging Face

- Use the model name in most training frameworks



STEP 2 – SELECT A MODEL

1. Size

- Reasoning-heavy tasks likely require 7B+ parameters
- Larger models cost more money to train

2. License

- Many custom licenses, some prohibit “malicious activities”
  – Good choices: Qwen, Mistral, gpt-oss, Llama*, Gemma*

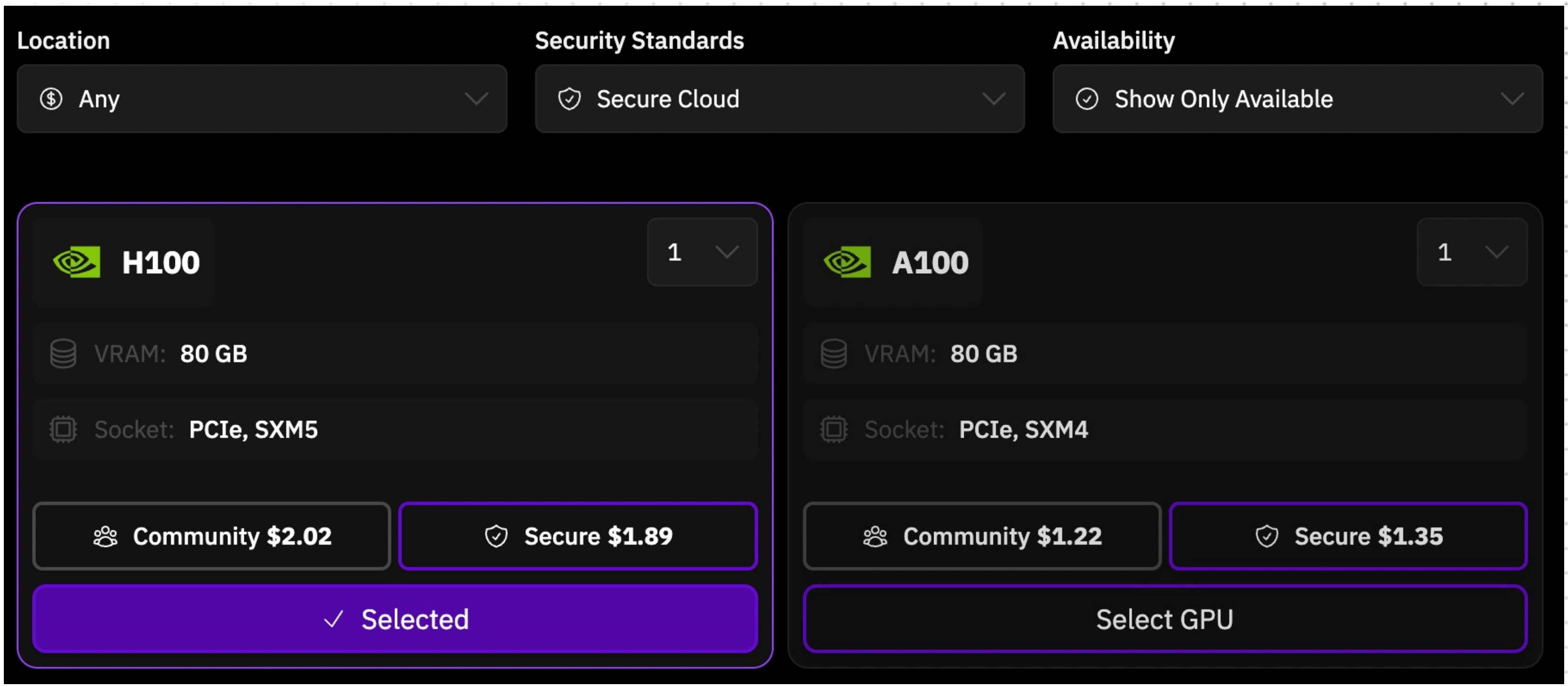
3. Refusals

- Some models are less helpful for offensive security tasks
  – Good choices: Qwen, Mistral

STEP 3 – RENTING GPUS

AWS, Azure, and GCP are significantly overpriced

- Compare hosting providers on a marketplace like Prime Intellect or Vast.ai

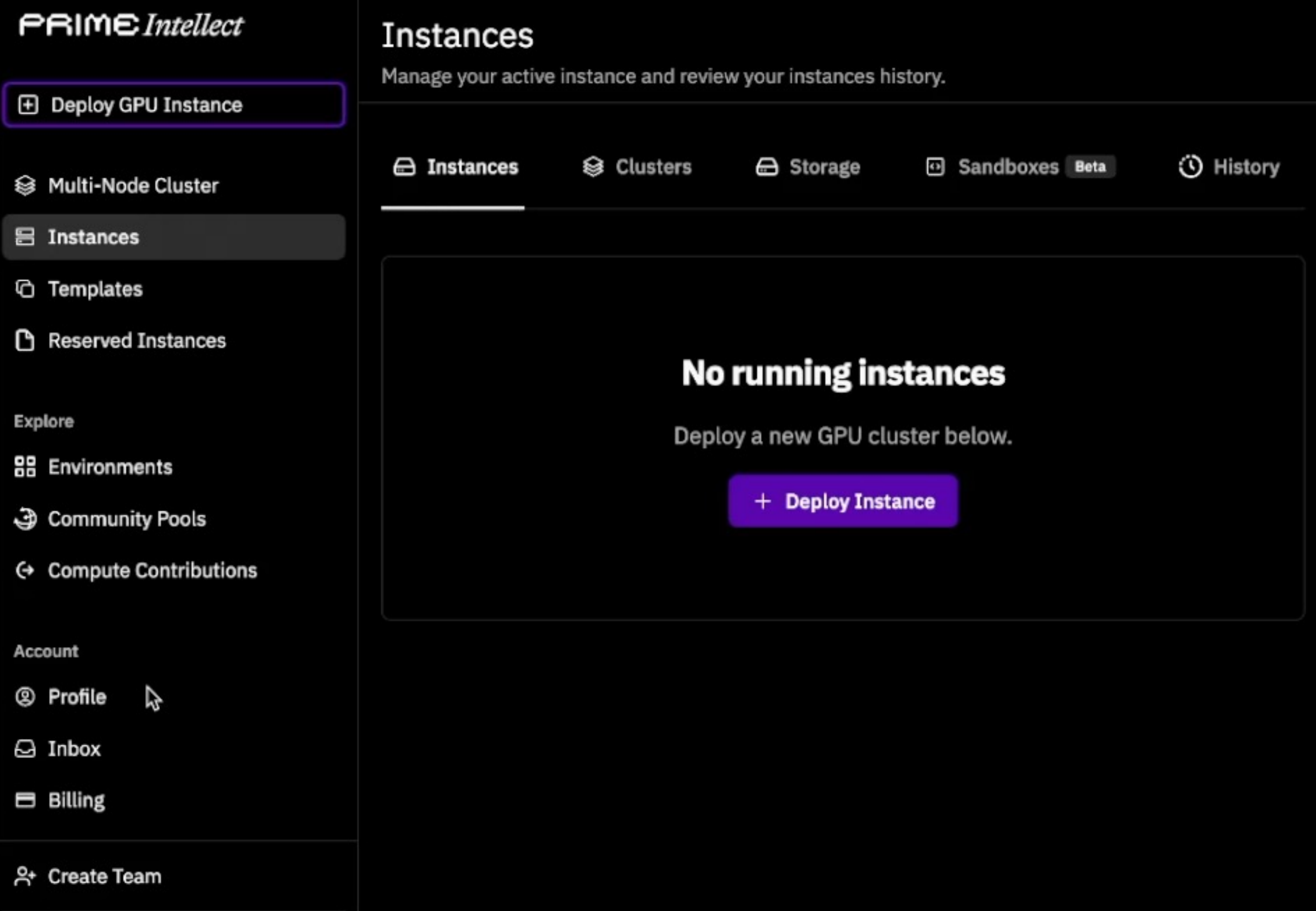


DEMO

Quick example using the EMBER2024 capa dataset

```
vf_env = vf.load_environment("capa")
model, tokenizer = vf.get_model_and_tokenizer("willcb/Qwen3-0.6B")
training_args = vf.grpo_defaults(run_name="example")

training_args.max_prompt_length = 8192
training_args.max_tokens = 32768

trainer = vf.GRPOTrainer(
    model=model,
    processing_class=tokenizer,
    env=vf_env,
    args=training_args
)


trainer.train()
```

```
CUDA_VISIBLE_DEVICES=0 vf-vllm --model willcb/Qwen3-0.6B &

CUDA_VISIBLE_DEVICES=1 accelerate launch --num-processes 1 train.py
```


PRIME Intellect
Deploy GPU Instance
Multi-Node Cluster
Instances
Templates
Reserved Instances
Explore
Environments
Community Pools
Compute Contributions
Account
Profile
Inbox
Billing
Create Team
Instances
Manage your active instance and review your instances history.
Instances
Clusters
Storage
Sandboxes
Beta
History
No running instances
Deploy a new GPU cluster below.
+ Deploy Instance

METRICS

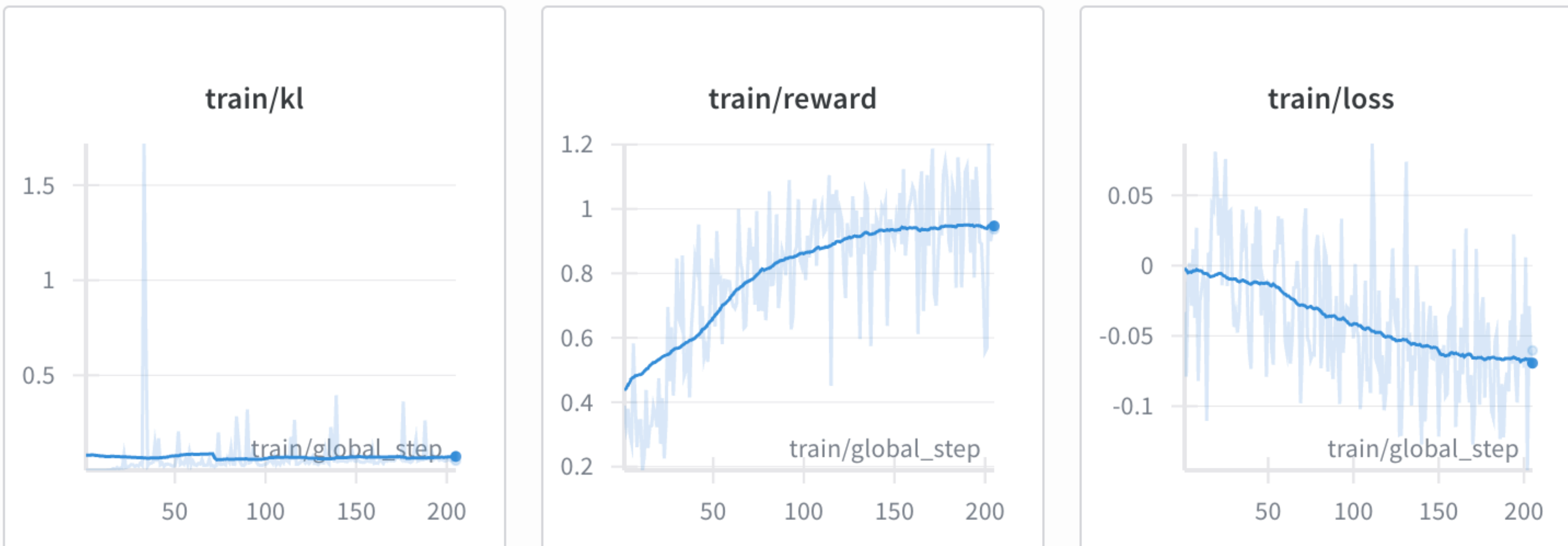
OUTFLANK

clear advice with a hacker mindset

METRICS

Visualize training with Weights & Biases:

- Most training frameworks support automatic logging to Wandb



METRICS

Review completions occasionally

- Manually inspect the input/output to your model during training

step	prompt[-1]["content"]	completion[-1]["content"]	reward
204	REQUEST: You are to judge the better of the two samples and determine which of the following samples is better using a short	<notes> Both samples provide comprehensive explanations about the usage of subsidiary	0.8923
204	REQUEST: You are to judge the better of the two samples and determine which of the following samples is better using a short	<notes> Both samples provide thorough explanations of the reasons for using subsidiary	0.8923
204	REQUEST: You are to judge the better of the two samples and determine which of the following samples is better using a short	<notes> Sample A's explanation for tracking debtors and creditors, monitoring bank	0.9023
204	REQUEST: You are to judge the better of the two samples and determine which of the following samples is better using a short	<notes> Sample A explains the importance of subsidiary ledgers in accounting, while	0.6923
204	REQUEST: You are to judge the better of the two samples and determine which of the following samples is better using a short	<notes> Both samples provide detailed explanations on why subsidiary ledgers (SLs) are used	0

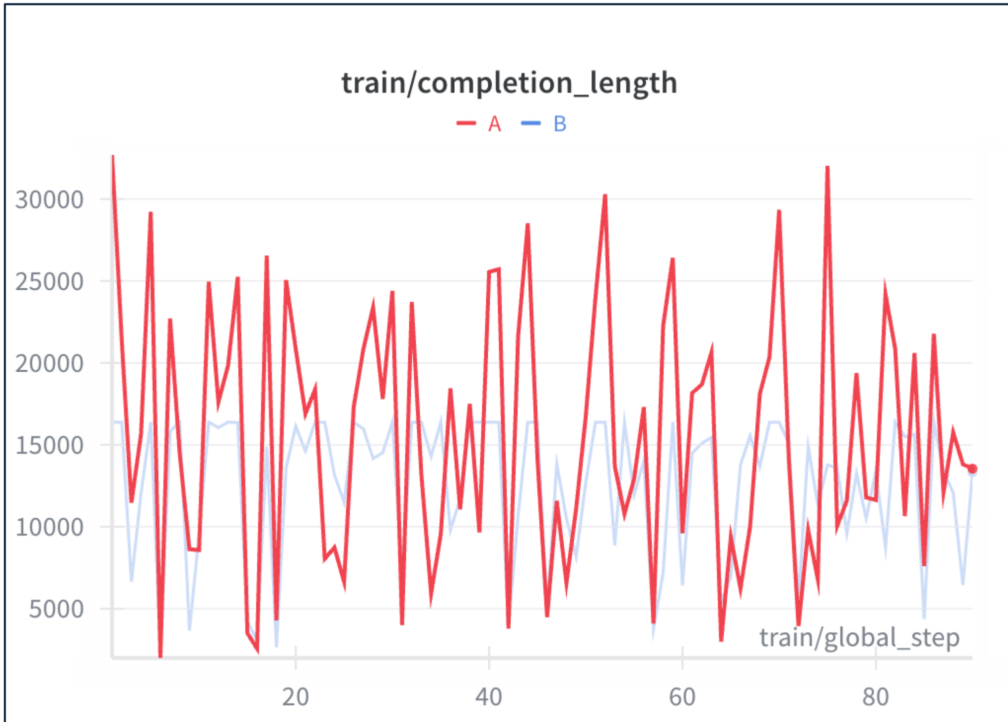
METRICS

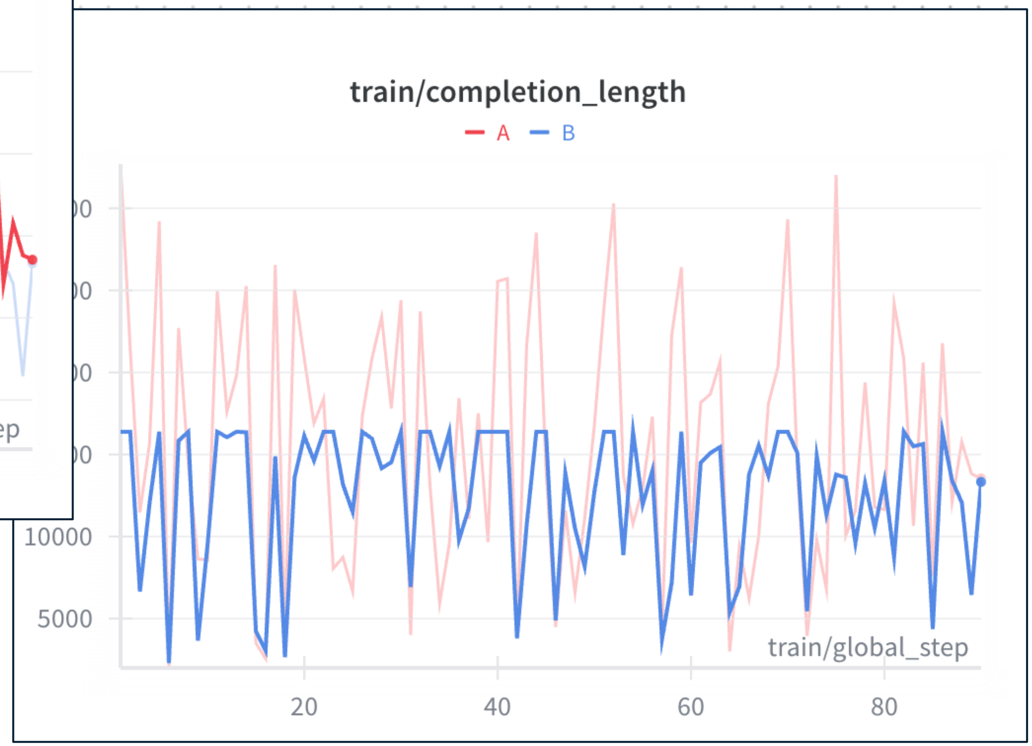
Why care about anything besides reward?

- Most important: Identify reward hacking
- Outputs might converge, reducing overall value of the model
- Reasoning may shrink over time, degrading generalization

METRICS EXAMPLE

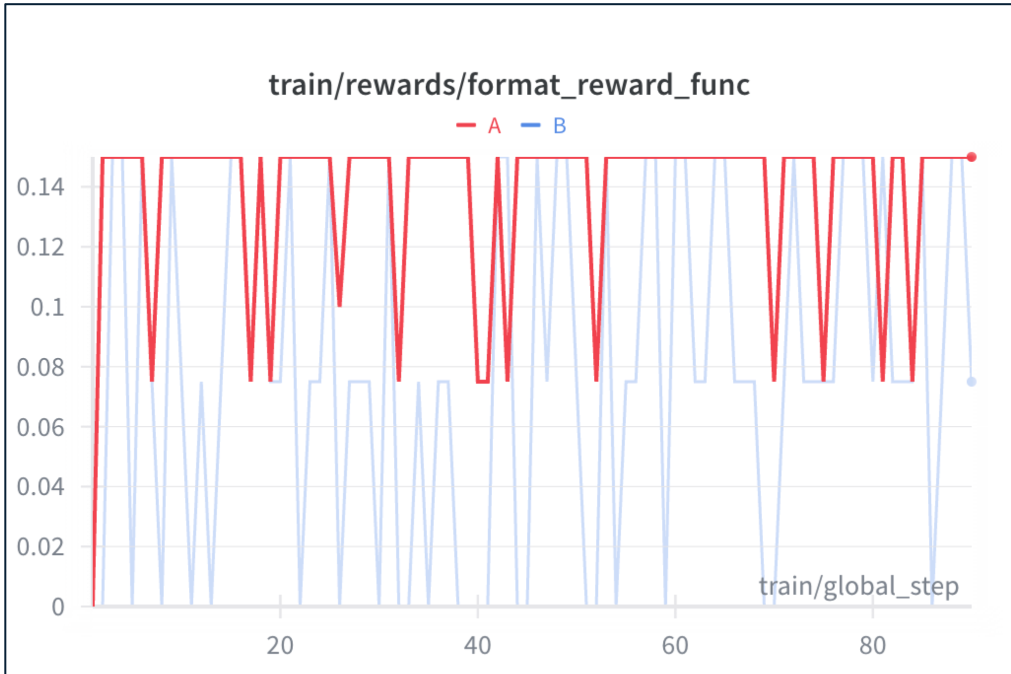
Monitoring completion length:

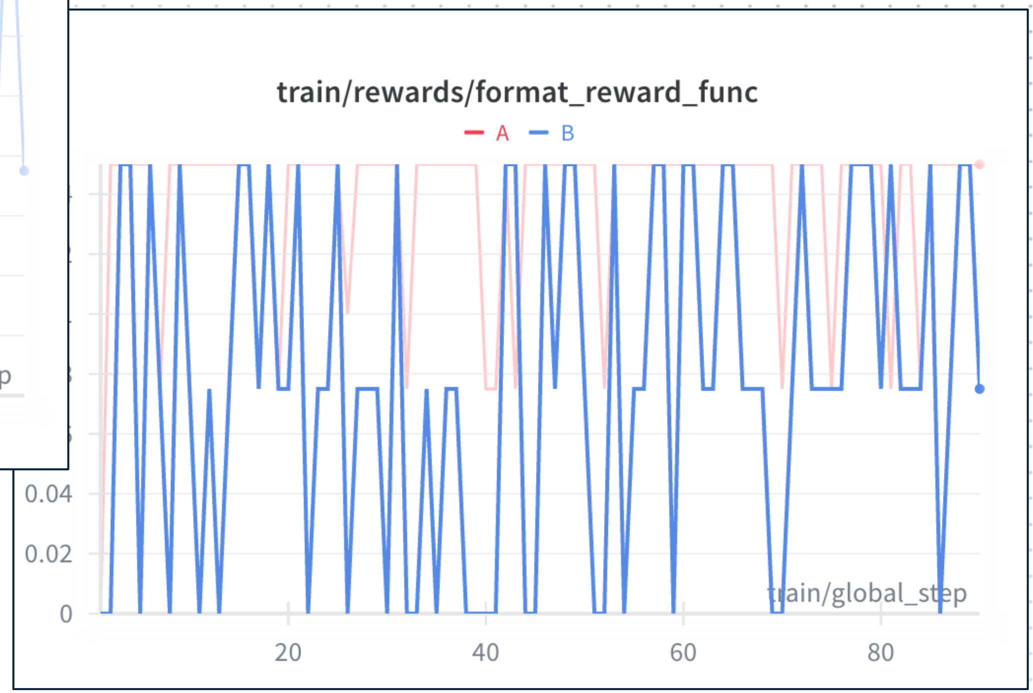




METRICS EXAMPLE

Result:





clear advice with a hacker mindset