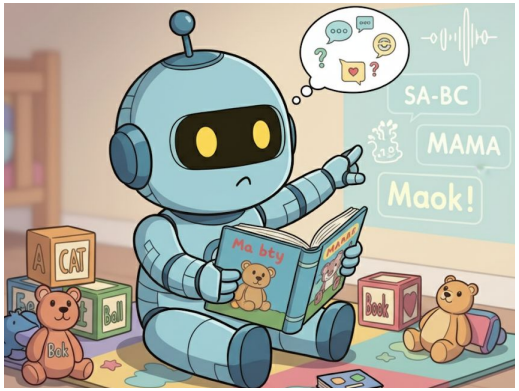
DeepLearning.AI

Data Driven Post-training

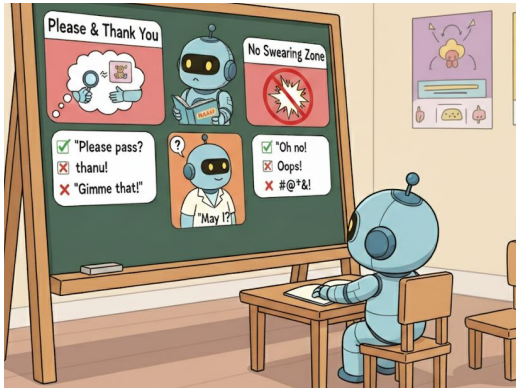
How much data you need for post-training

Post-training needs less data than pre-training



Pre-training

- All about scale
- “Compute-optimal” is 20 tokens per parameter [From “Training Compute-Optimal Large Language Models”, Hoffman et al., 2022].
- GPT-3: Pretrained on ~300B tokens.



Post-training

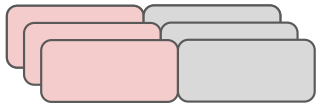
- Re-shaping behavior
- ChatGPT (Fine-tuning + RLHF): ~13k fine-tuning examples & thousands (<1M) preference data

When are 50 examples enough vs. 100k+ examples?

Model Already knows calculus

Goal New calculus exam format

20 practice problems

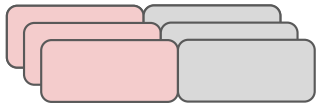


When are 50 examples enough vs. 100k+ examples?

Model Never seen calculus

Goal New calculus exam format

20 practice problems not enough!

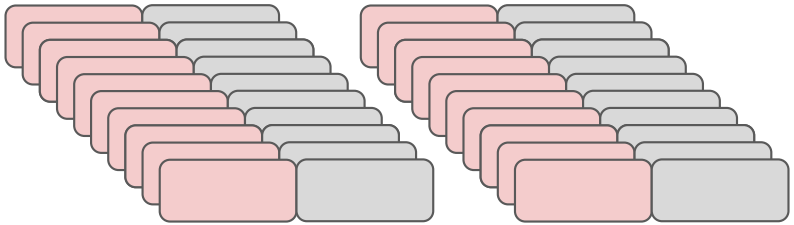


When are 50 examples enough vs. 100k+ examples?

Model | Never seen calculus

Goal | New calculus exam format

1000+ examples
Need entire calculus course!

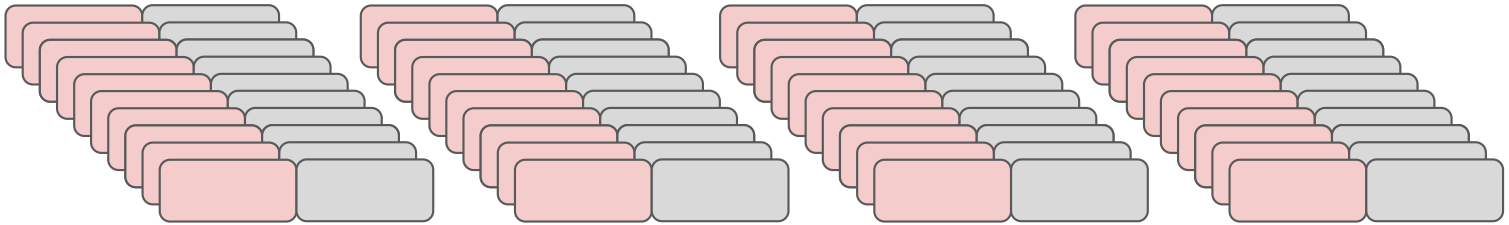


When are 50 examples enough vs. 100k+ examples?

Model Never seen math

Goal New calculus exam format

100K+ examples
Need math courses!



If not in pre-training, post-training can't create it cheaply

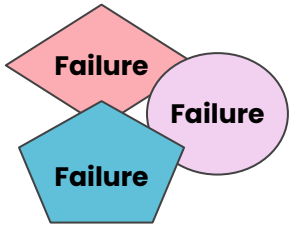
Model	Already knows calculus
Model	Never seen calculus
Model	Never seen math

Evals always first: on your pre-trained model - to estimate data for your task

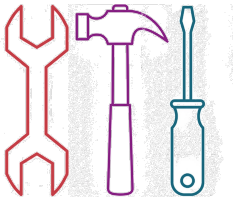
Looking at these evals...

Operation Type	Eval Accuracy	Observed Errors	Actionable Intervention
Addition	100%	None	No new data needed
Multiplication	95%	Occasional large-number mistakes	Add a few high-magnitude multiplication fine-tuning examples
Division	20%	Systematic errors, wrong decimals, truncation	Collect targeted division problems (integers, decimals, edge cases)
Mixed expressions	60%	Division sub-steps fail	Add multi-step division-in-context examples; add division reasoning examples to RL

...experiment with minimal data to draw conclusions



Break down failures



Propose fixes 1-3

Experiment 1

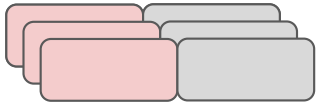
Experiment 2

Experiment 3

Then scale until you reach diminishing returns

Empirical fine-tuning:

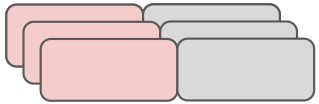
- 20: formatting changes
- 100: good starting point to observe model change
- 1K: most domain-specific post-training applications
- 10K: post-training for frontier labs
- 100K: domain adaptation



LoRA FT behaves differently from full fine-tuning

Empirical fine-tuning:

- 20: formatting changes
- 100: good starting point to observe model change
- 1K: most domain-specific post-training applications
- 10K: post-training for frontier labs
- 100K: domain adaptation



LoRA FT:

- Low rank: smaller amount of targeted data
- At highest rank: converges to full fine-tuning

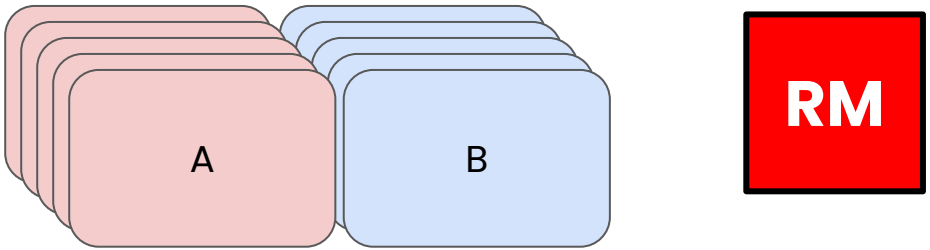
Preference learning has similar numbers

Empirical preference learning:

- 1K: improvement in RM
- 10K: obtain nuanced distinctions A vs. B
- Can be continually trained during RL training
- Outputs don't need to be generated by the LLM

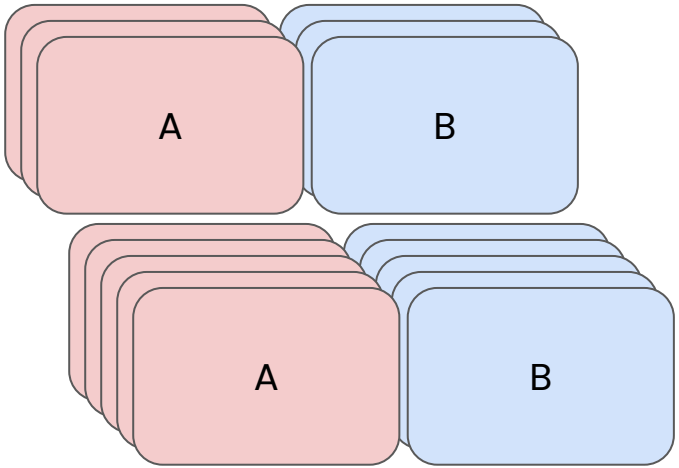
Human ranking 3 items (A>B>C) → 3 pairs!

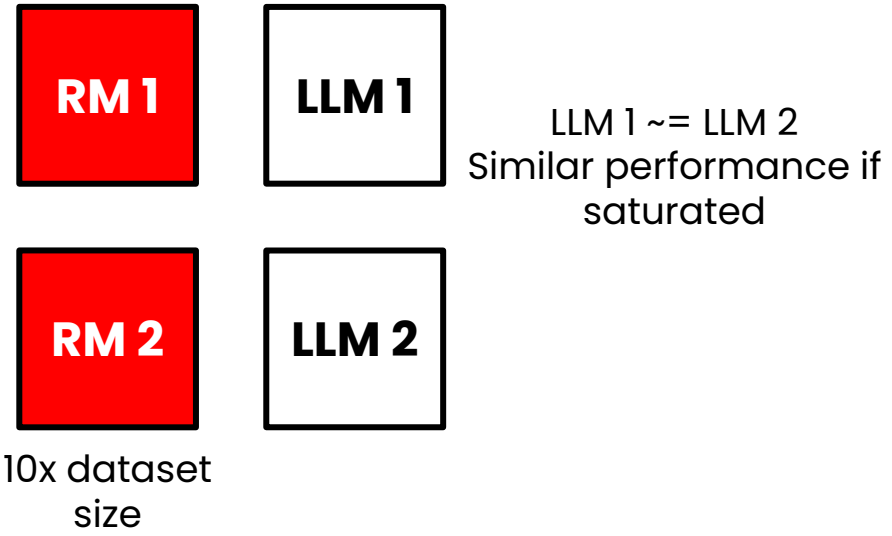
- A>B, B>C, A>C



RL performance depends on reward model (RM)

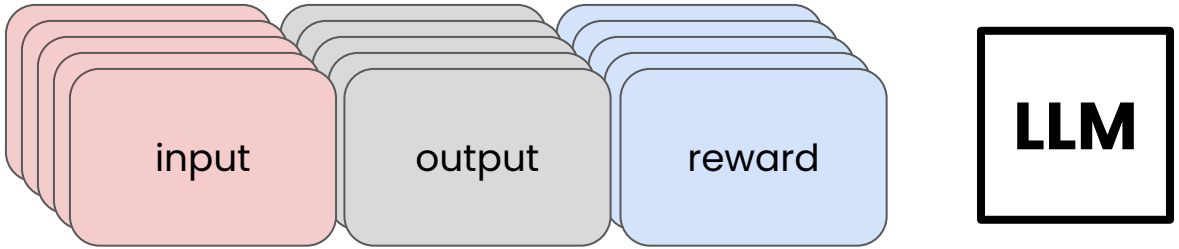
- Preference data → Reward Model → RL data → LLM with RL
- Once the RM is "good enough," a better RM doesn't translate 1-to-1 into the downstream LLM gains



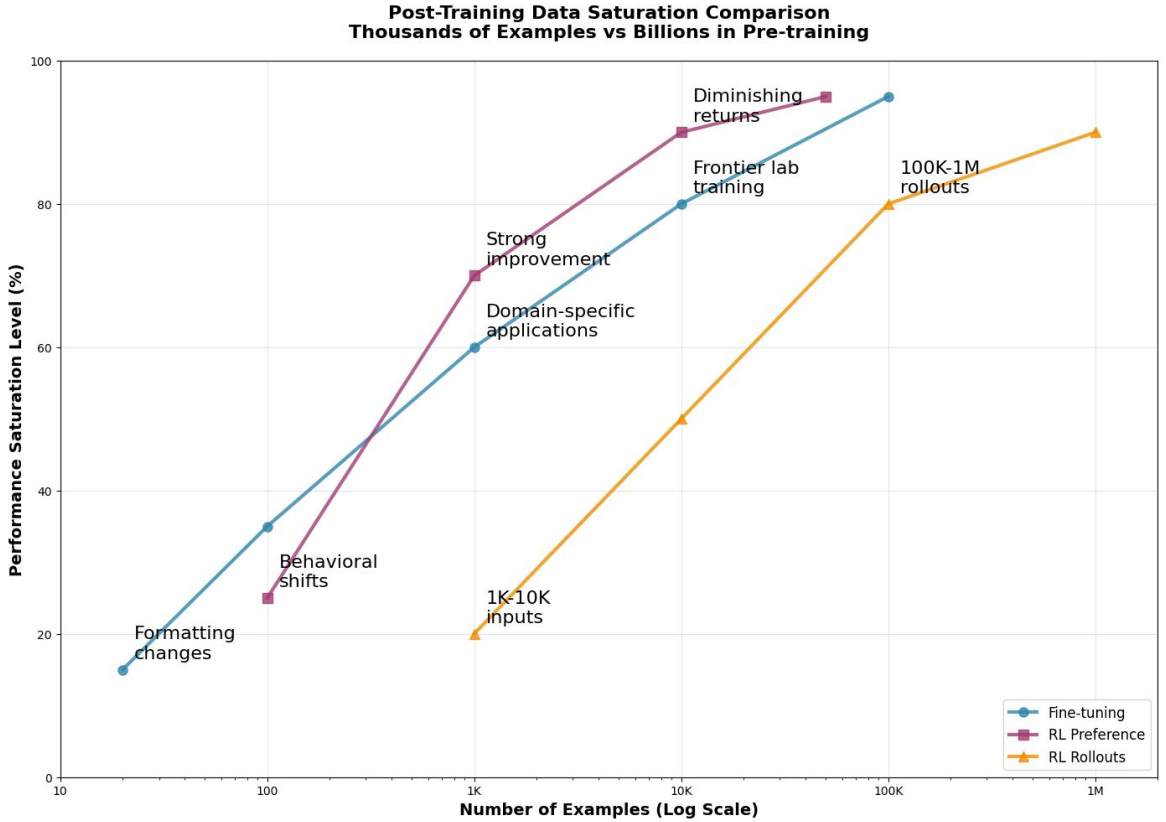


RL performance depends on input distribution

- 1K-10K inputs for task coverage
- Each input can have multiple outputs, e.g. 8 default in HF, to estimate reward variance
- SOTA: 100K-1M rollouts total is alignment runs today



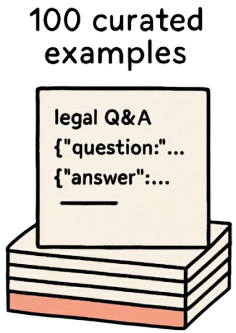
Earlier saturation in #s than pre-training

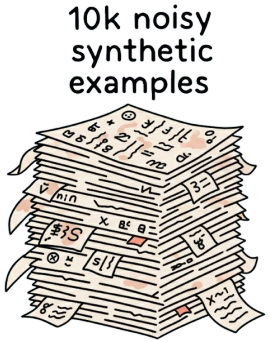


More ***careful*** data beats more data

A few curated examples can outperform tons of noisy examples.

100 curated examples >> 10k noisy synthetic examples





DeepLearning.AI

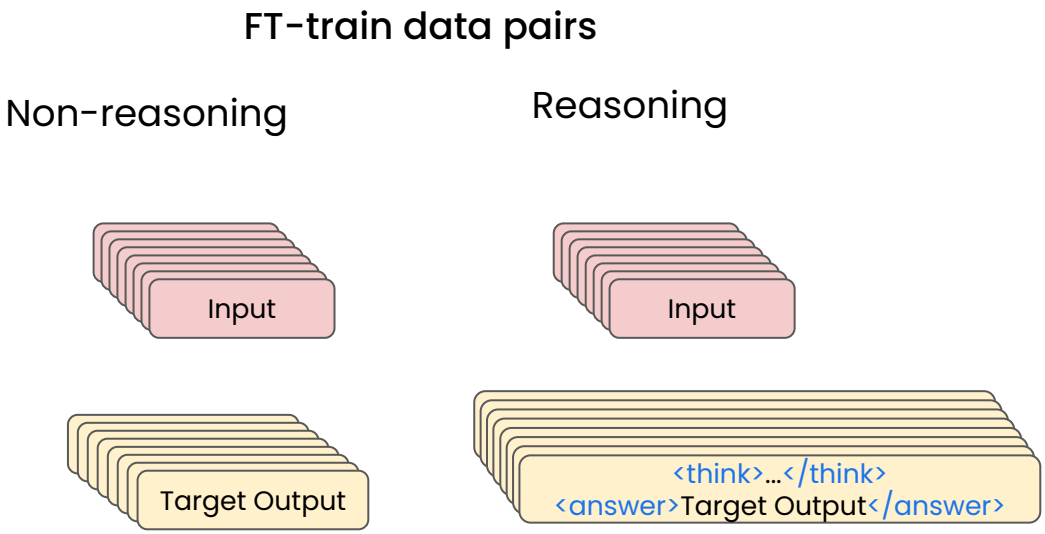
Data Driven Post-training

Data for fine-tuning

Post-training lives or die on data

The fine-tuning data you need:

- Pairs: {input, target output}
  - Reasoning: {input, “think” + answer}



Fine-tuning data

Input

Alice has 3 apples and buys 2 more.
How many now?

Target Output

Rationale:
Start with 3
Buys 2 ⇒ 3+2=5.

Answer: 5

Fine-tuning data: Prompt tags

Input

Alice has 3 apples and buys 2 more.
How many now?

Target Output

<think>
Start with 3.
Buys 2 ⇒ 3+2=5.
</think>

<answer>5</answer>

Fine-tuning data: Custom prompt tags

Input

<api>Today's weather is sunny</api>
Alice has 3 apples and buys 2 more on sunny days.
How many now?

Target Output

<think>
Start with 3.
Buys 2 ⇒ 3+2=5.
</think>

<answer>5</answer>

Fine-tuning data: Custom prompt tags

Input

<api>Today's weather is sunny</api>
Alice has 3 apples and buys 2 more on sunny days.
How many now?

Target Output

<think>
Start with 3.
Buys 2 ⇒ 3+2=5.
</think>

<answer>5</answer>
<display_weather>☀️</display_weather>

Iterative refinement

Iterative refinement

Input

<api>Today's weather is sunny</api>
Alice has 3 apples and buys 2 more on sunny days.
How many now?

Target Output

<think>
Start with 3.
Buys 2 ⇒ 3+2=5.
</think>

<answer>5</answer>
<display_weather>☀️</display_weather>

Goal: teach new custom tags format

Iterative refinement

Input

<api>Today's weather is sunny</api>
Alice has 3 apples and buys 2 more on sunny days.
How many now?

Target Output

<think>
Start with 3.
Buys 2 ⇒ 3+2=5.
</think>

<answer>5</answer>
<display_weather>☀️</display_weather>

Start: 20 examples with LoRA FT

Targeting the highest-impact failure patterns

Input

<api>Weather: sunny.</api>
Alice has 3 apples and buys 2 more on sunny days.
How many now?

Model Output

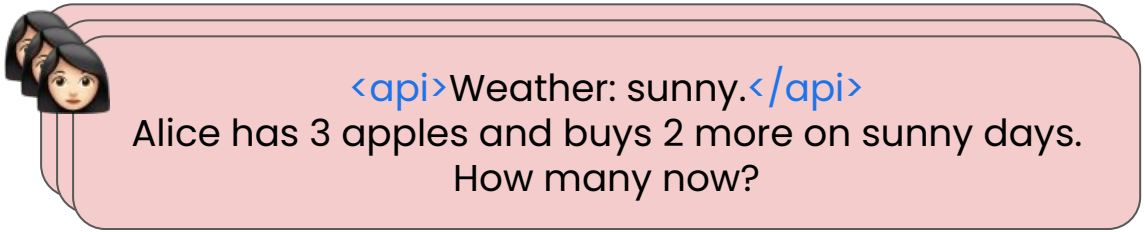
<think>
Start with 3.
Buys 2 ⇒ 3+2=5.
</think>

<answer>5</answer>

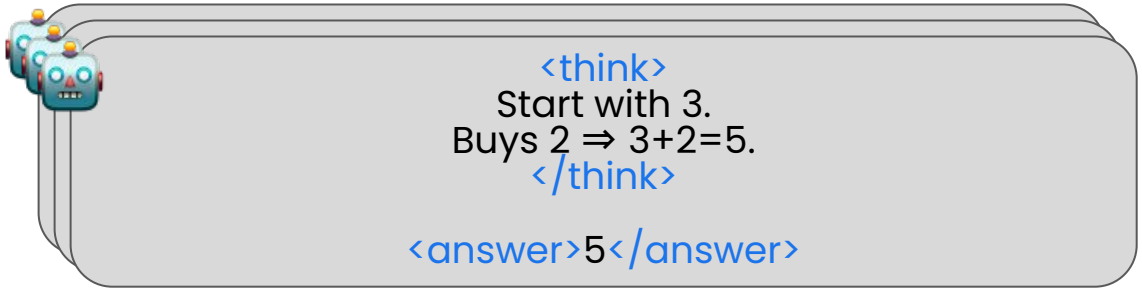
Evaluate fine-tuned model → error analysis → find worst failure pattern

Targeting the highest-impact failure patterns

Input



Model Output



Error analysis bucket: Omits <display_weather> when “sunny” next to punctuation mark.

Targeting the highest-impact failure patterns

Input

<api>Bright sunshine!</api>
Alice has 3 apples and buys 2 more on sunny days.
How many now?

Target Output

<think>
Start with 3.
Buys 2 ⇒ 3+2=5.
</think>

<answer>5</answer>
<display_weather>☀️</display_weather>

Data fix: Add 3–5 near neighbors.
Vary punctuation, add synonyms like “bright sunshine”.

Targeting the highest-impact failure patterns

Input

<api><SUNNY></api>
Alice has 3 apples and buys 2 more on sunny days.
How many now?

Target Output

<think>
Start with 3.
Buys 2 ⇒ 3+2=5.
</think>

<answer>5</answer>
<display_weather>☀️</display_weather>

Data fix: Add 3–5 near neighbors.
Vary capitalization

Targeting the highest-impact failure patterns

Input

<api><SUNNY></api>
Bob had 8 apples yesterday. He eats an apple every day.
How many today?

Target Output

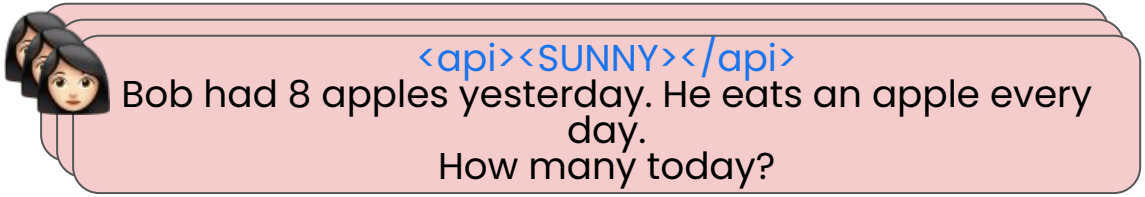
<think>
Start with 8.
Eats 1 ⇒ 8-1=7.
</think>

<answer>7</answer>
<display_weather>☀️</display_weather>

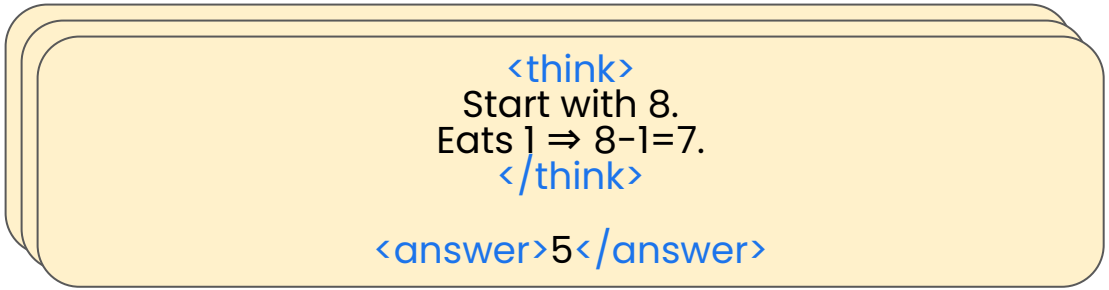
Data fix: Add 3–5 near neighbors. Vary math problem to avoid overfitting on this one.

Iterative refinement

Input



Target Output



Iterate: LoRA FT with additional fix examples mixed in (~20+5=25 examples)

Iterative refinement

Input

<api>Weather: sunny.</api>
Chris has 1 apple and buys 2 more on clear sky days.
How many now?

Model Output

<think>
Start with 1.
Buys 2 ⇒ 1+2=3.
</think>

<answer>3</answer>
<display_weather>☀️</display_weather>

Fixed!

Targeting the highest-impact failure patterns

Input

<api>Clear skies!</api>
Chris has 1 apple and buys 2 more on clear sky days.
How many now?

Model Output

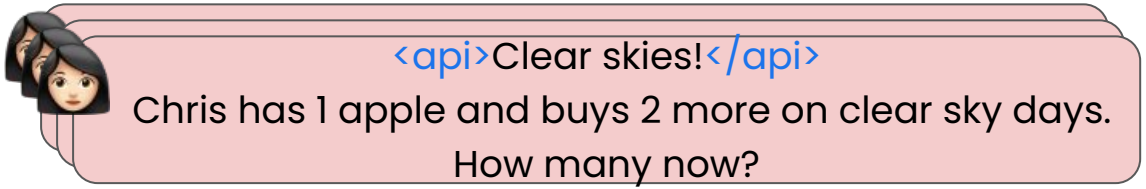
<think>
Start with 1.
Buys 2 ⇒ 1+2=3.
</think>

<answer>3</answer>
<display_weather>☁️ </display_weather>

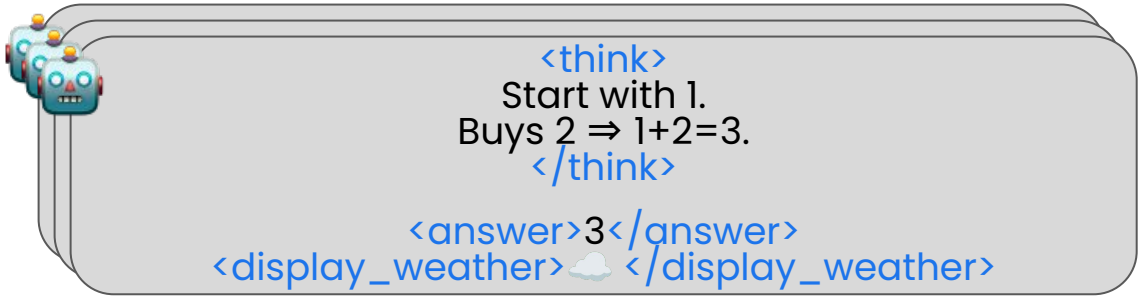
Same loop: Evaluate fine-tuned model → error analysis → find worst failure pattern

Targeting the highest-impact failure patterns

Input



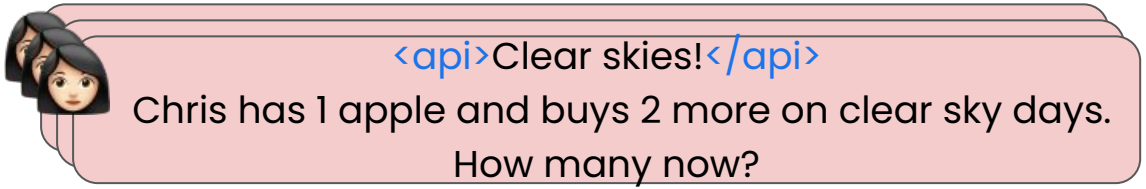
Model Output



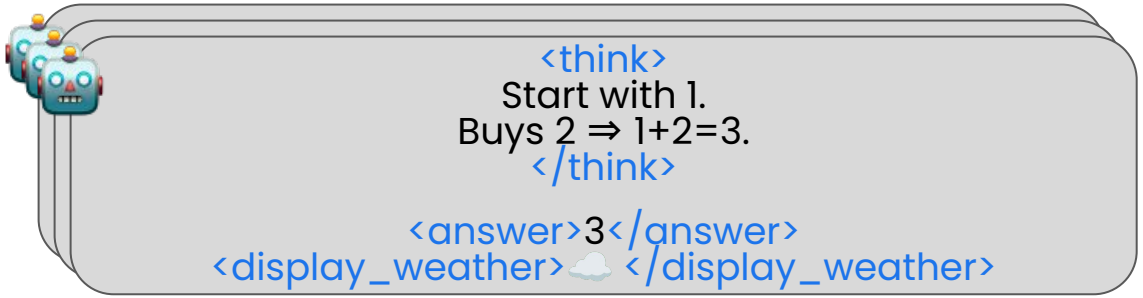
Error analysis bucket: Uses wrong emoji for "clear skies" → returns ☁️ instead of ☀️

Targeting the highest-impact failure patterns

Input



Model Output

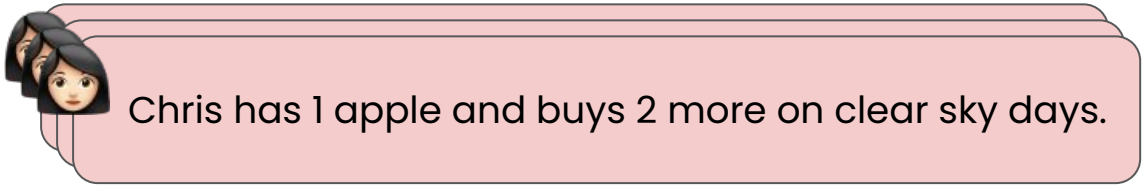


Fix: Introduce 3-5 counterexamples.

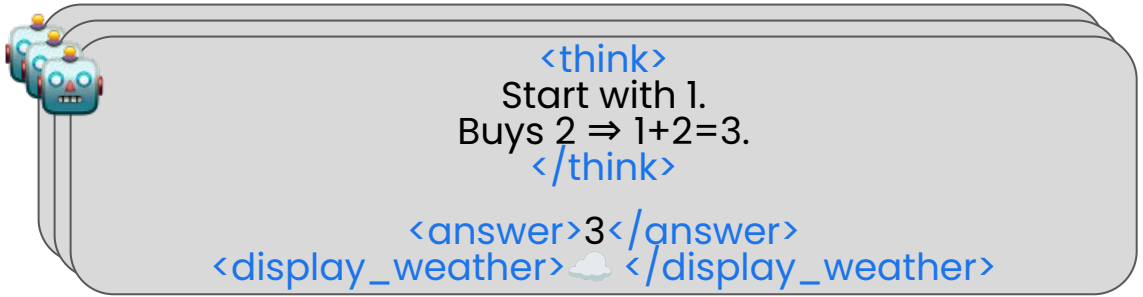
Error analysis bucket: Uses wrong emoji for "clear skies" → returns ☁️ instead of ☀️

Targeting the highest-impact failure patterns

Input



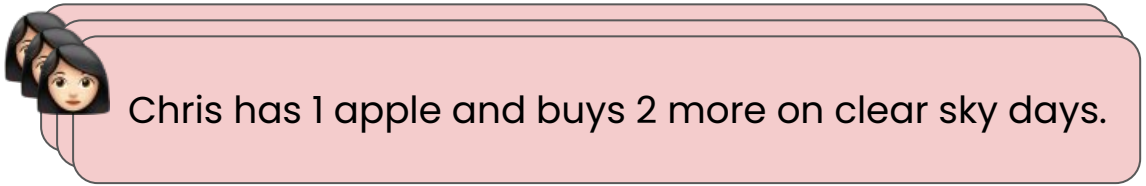
Model Output



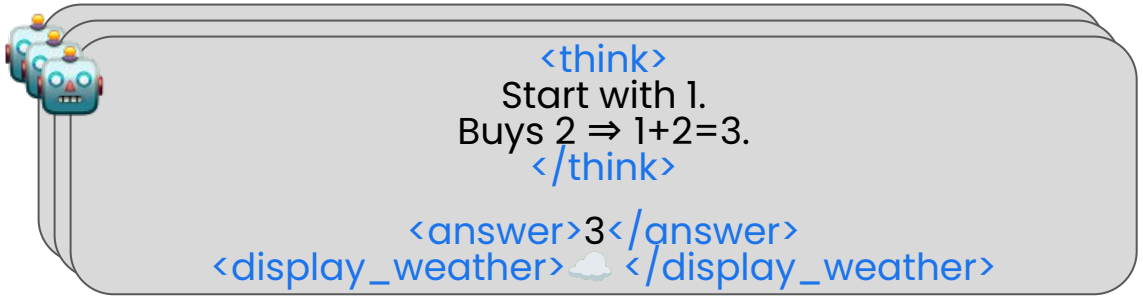
Error analysis bucket: No <api> tag → don't need to <think> or <display_weather> tags

Targeting the highest-impact failure patterns

Input



Model Output

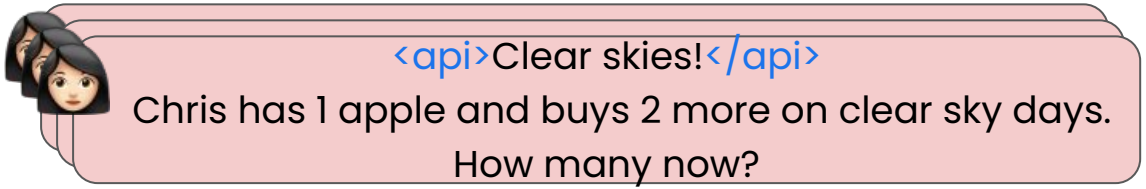


Fix: Add no-tag examples, so it doesn't leak thoughts.

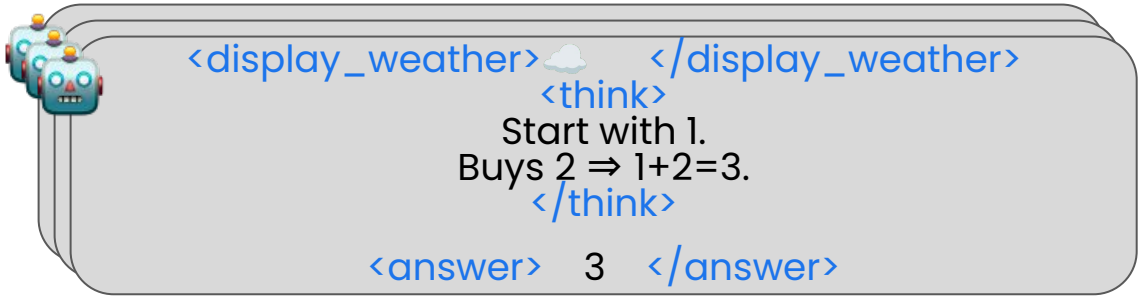
Error analysis bucket: No <api> tag → don't need to <think> or <display_weather> tags

Targeting the highest-impact failure patterns

Input



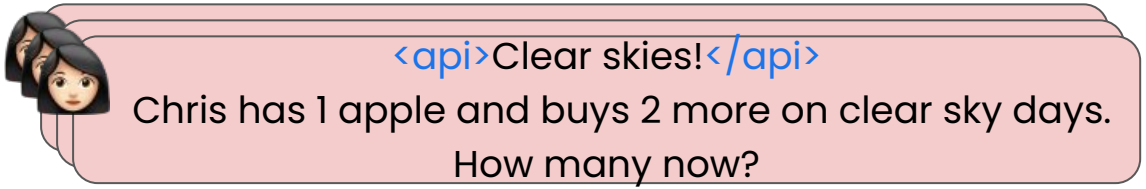
Model Output



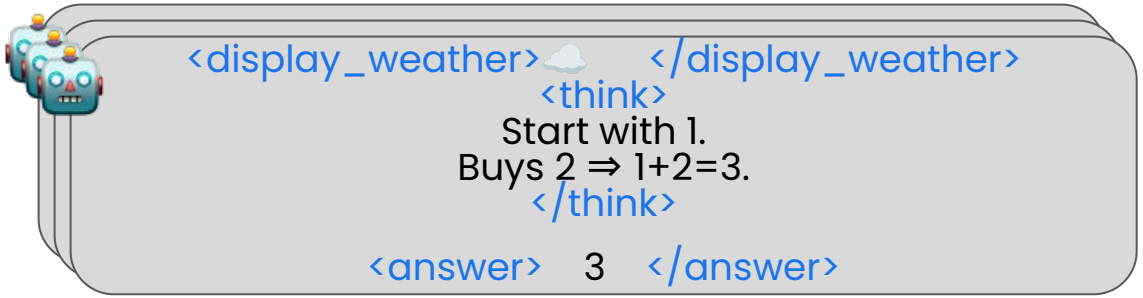
Error analysis bucket: schema not strictly adhered

Targeting the highest-impact failure patterns

Input



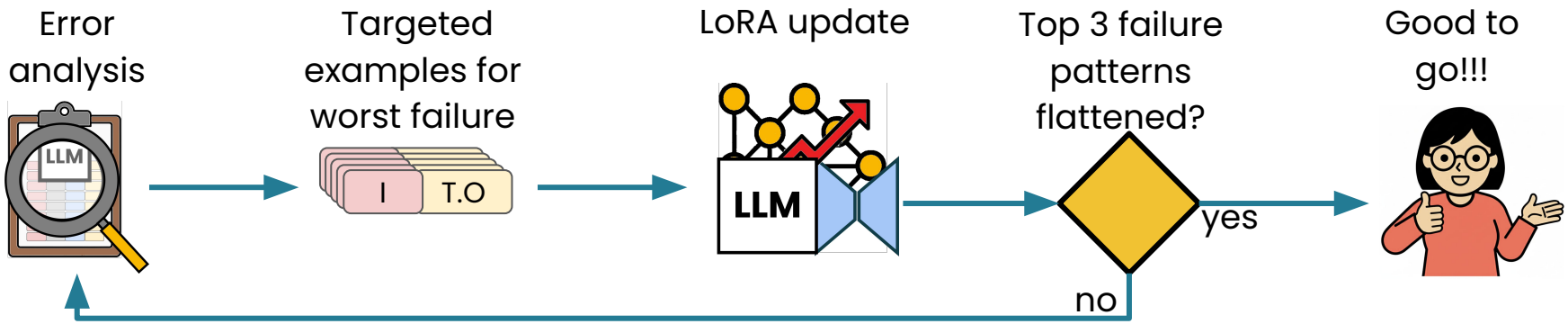
Model Output



Fix: Add strict schema adherence examples.

Error analysis bucket: schema not strictly adhered

When to stop



Use your evals, not just loss, for early stopping.

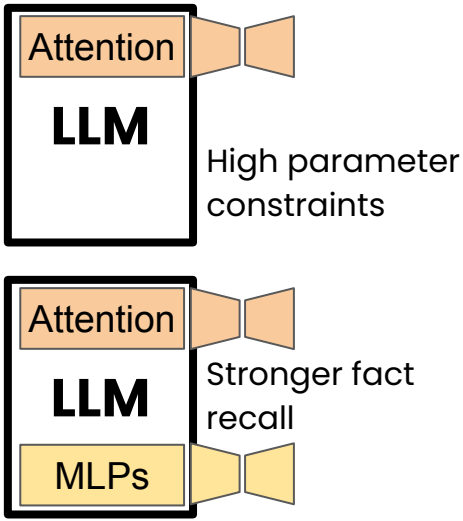
Evals with correct examples and automated checks:

- Schema regex
- Semantic emoji mapping (“sunny | clear | bright” → ☀️)
- Math answer check

When to stop

Defaults that work well for small data fine-tuning

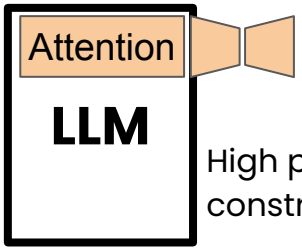
Where to add LoRAs



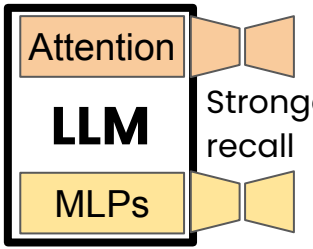
When to stop

Defaults that work well for small data fine-tuning

Where to add LoRAs

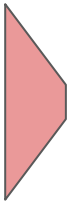


High parameter constraints



Stronger fact recall

Importance of r (rank)



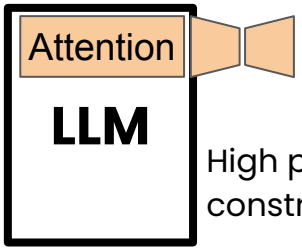
Low r:
≤1k examples

Higher r:
1k - 10k examples

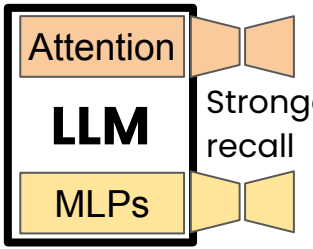
When to stop

Defaults that work well for small data fine-tuning

Where to add LoRAs

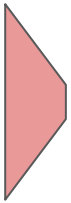


High parameter constraints



Stronger fact recall

Importance of r (rank)

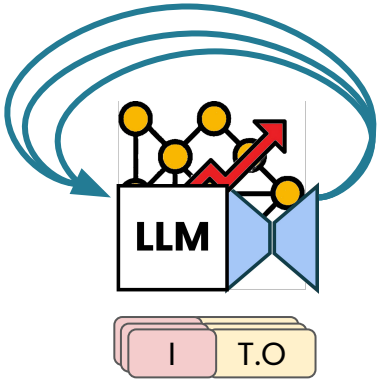


Low r:
≤1k examples

Higher r:
1k - 10k examples

Batching

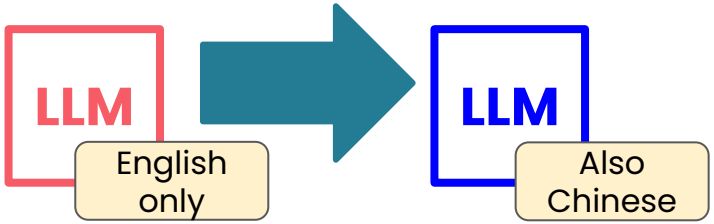
More steps



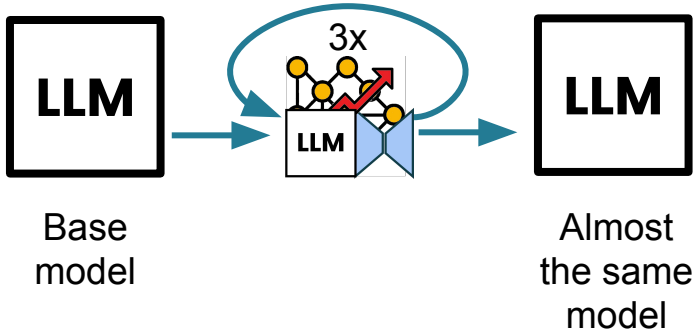
Small batches

When to prefer full-model fine-tuning instead of LoRA

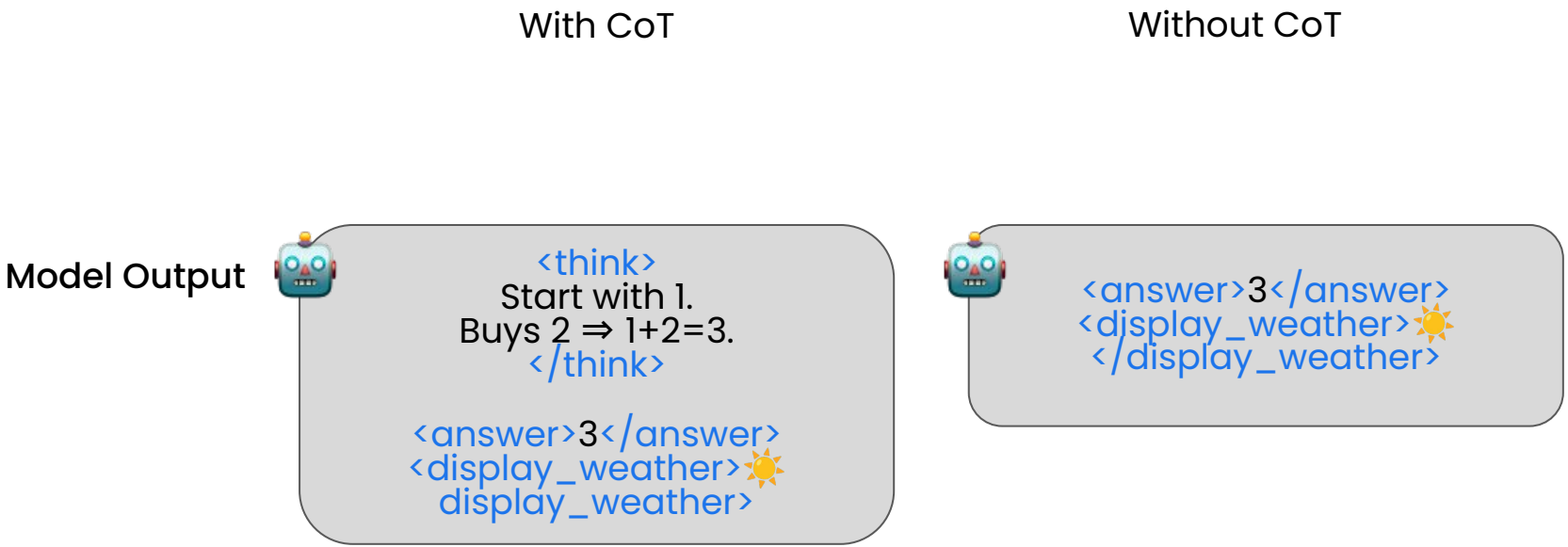
Broad knowledge shifts



Catastrophic adherence



How much chain of thought (CoT)?



<think> tags are *great* for training control & getting the LLM to learn reasoning for better outputs.

Inference needs decide % CoT vs. Direct-answer

Input

<api>Weather: sunny.</api>
Chris has 1 apple and buys 2 more on clear sky days.
How many now?
<use_thinking>

<api>Weather: sunny.</api>
Chris has 1 apple and buys 2 more on clear sky days.
How many now?

Model Output

<think>
Start with 1.
Buys 2 ⇒ 1+2=3.
</think>

<answer>3</answer>
<display_weather>☀️
display_weather>

<answer>3</answer>
<display_weather>☀️
</display_weather>

If inference should sometimes use reasoning:

- with tags → <think>
- without tags → no <think>

Inference: switch between them

DeepLearning.AI

AMD

Inference needs decide % CoT vs. Direct-answer

Input

<api>Weather: sunny.</api>
Chris has 1 apple and buys 2 more on clear sky days.
How many now?
"Please use <think>"

<api>Weather: sunny.</api>
Chris has 1 apple and buys 2 more on clear sky days.
How many now?

Model Output

<think>
Start with 1.
Buys 2 ⇒ 1+2=3.
</think>

<answer>3</answer>
<display_weather>☀️
display_weather>

<answer>3</answer>
<display_weather>☀️
</display_weather>

If inference should sometimes use reasoning:

- with tags + "Please use <think>" → <think>
- with tags → no <think>

Inference: switch between them

DeepLearning.AI

AMD

Inference needs decide % CoT vs. Direct-answer

Input

<api>Weather: sunny.</api>
Chris has 1 apple and buys 2 more on clear sky days.
How many now?
"Please use <think>"

<api>Weather: sunny.</api>
Chris has 1 apple and buys 2 more on clear sky days.
How many now?

Model Output

<think>
Start with 1.
Buys 2 ⇒ 1+2=3.
</think>

<answer>3</answer>
<display_weather>☀️
display_weather>

40-60%

<answer>3</answer>
<display_weather>☀️
</display_weather>

40-60%

Reasoning-heavy tasks

Inference needs decide % CoT vs. Direct-answer

Input

<api>Weather: sunny.</api>
Chris has 1 apple and buys 2 more on clear sky days.
How many now?
"Please use <think>"

<api>Weather: sunny.</api>
Chris has 1 apple and buys 2 more on clear sky days.
How many now?

Model Output

<think>
Start with 1.
Buys 2 ⇒ 1+2=3.
</think>

<answer>3</answer>
<display_weather>☀️
display_weather>

10-30%

<answer>3</answer>
<display_weather>☀️
</display_weather>

70-90%

Latency-sensitive tasks

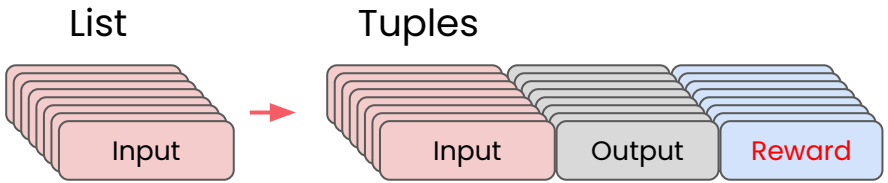
DeepLearning.AI

Data Driven Post-training

Data for RL (Part 1)

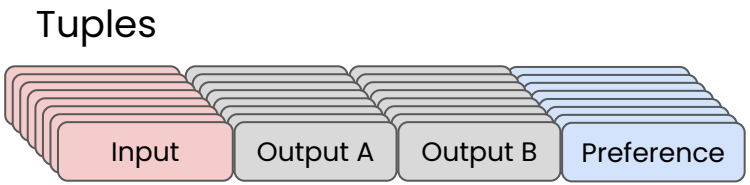
Post-training lives or die on data

The RL data you need:



From input + graders + environment

Optional:



For reward model

Sample RL data

Input

Alice has 3 apples and buys 2 more.
How many now?

Model Output

<think>
Start with 3.
Buys 2 ⇒ 3+2=5.
</think>

<answer>5</answer>

Reward **+2** ← informed by Reward Model

Tuple: {input, output, reward} ← “trajectory”

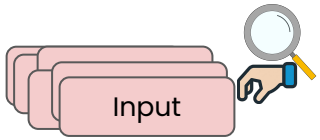
Sample reward model data for RL

Input		Alice has 3 apples and buys 2 more. How many now?
Model Output A		<think> Start with 3. Buys 2 ⇒ 3+2=5. </think> <answer>5</answer>
Model Output B		hi
Preference		*A*

Tuple: {input, output A, output B, preference}

Data in the RL pipeline

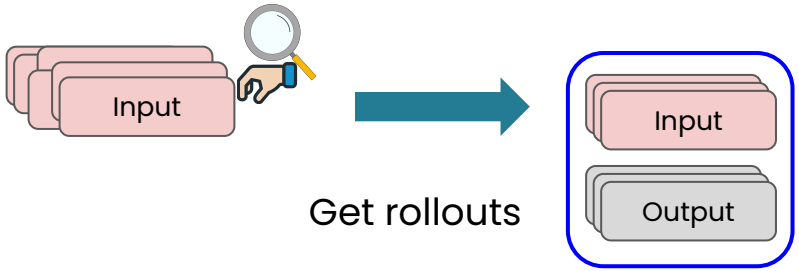
Get RL data {input, output, reward} aka. “trajectories”



Curate inputs

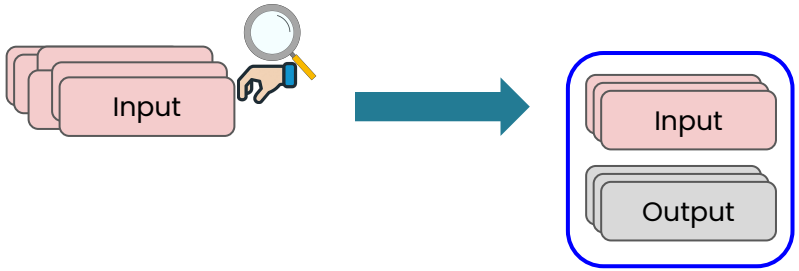
Data in the RL pipeline

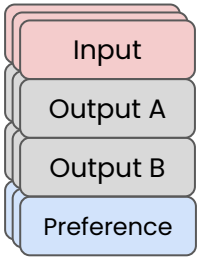
Get RL data {input, output, reward} aka. “trajectories”



Data in the RL pipeline

Get RL data {input, output, reward} aka. “trajectories”

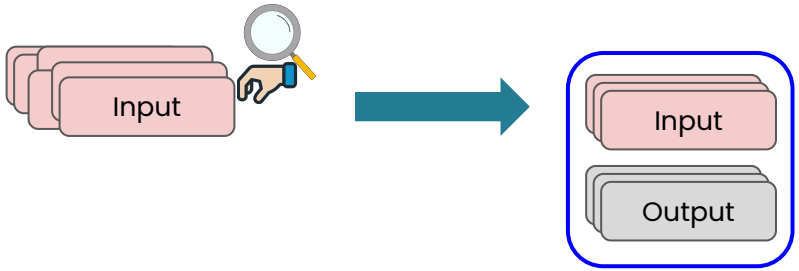


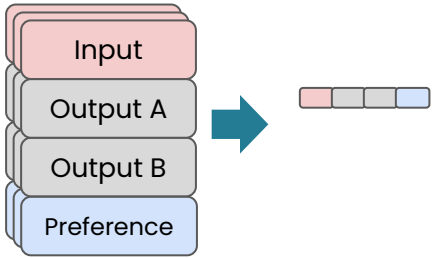


If using reward model:
Get preference data

Data in the RL pipeline

Get RL data {input, output, reward} aka. “trajectories”

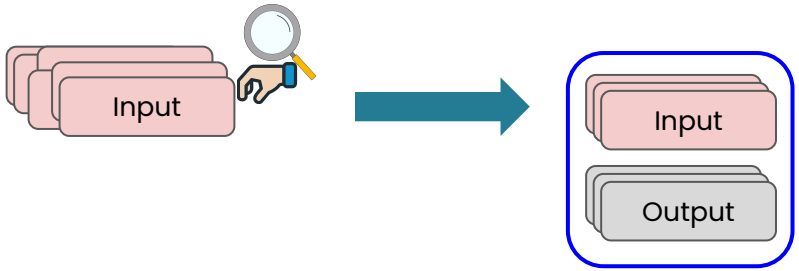


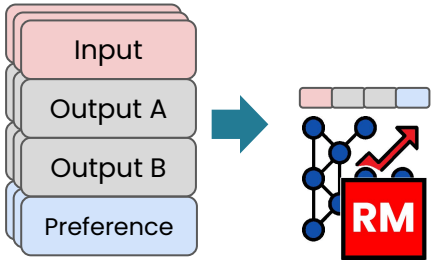


If using reward model:
Get preference data
Convert to preference pairs

Data in the RL pipeline

Get RL data {input, output, reward} aka. “trajectories”

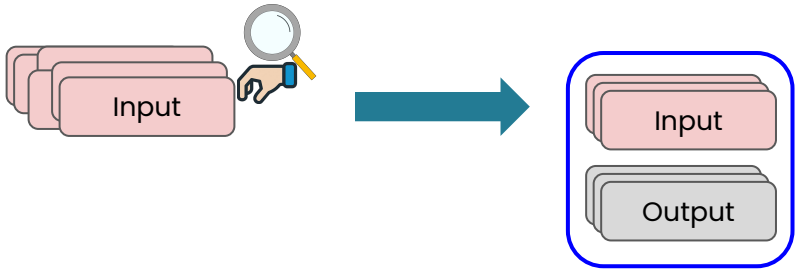


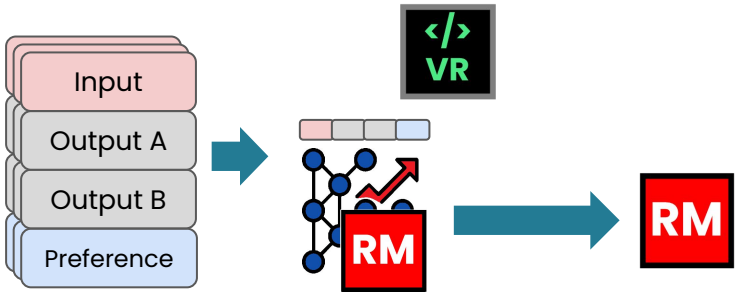


If using reward model:
Get preference data
Convert to preference pairs
Train a reward model

Data in the RL pipeline

Get RL data {input, output, reward} aka. “trajectories”

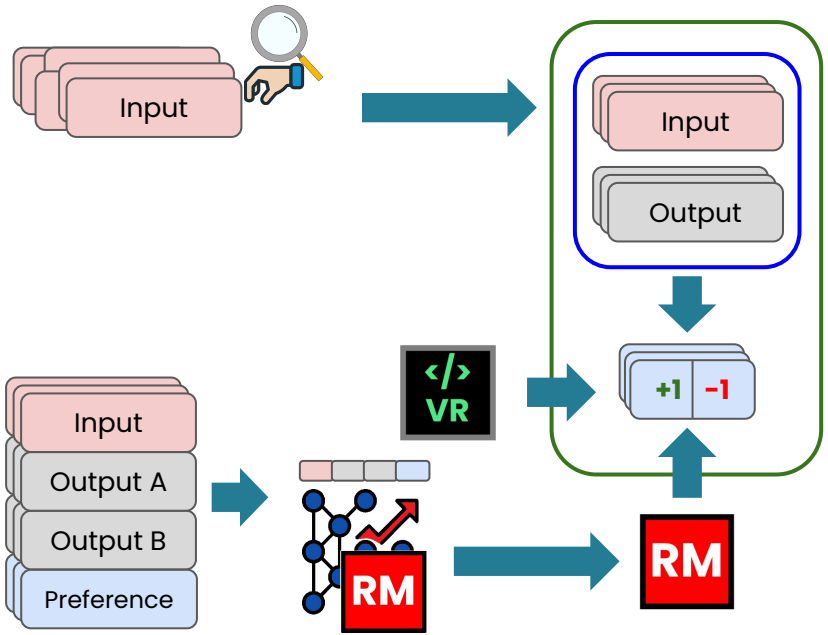




Graders are verifiers and/or reward models

Data in the RL pipeline

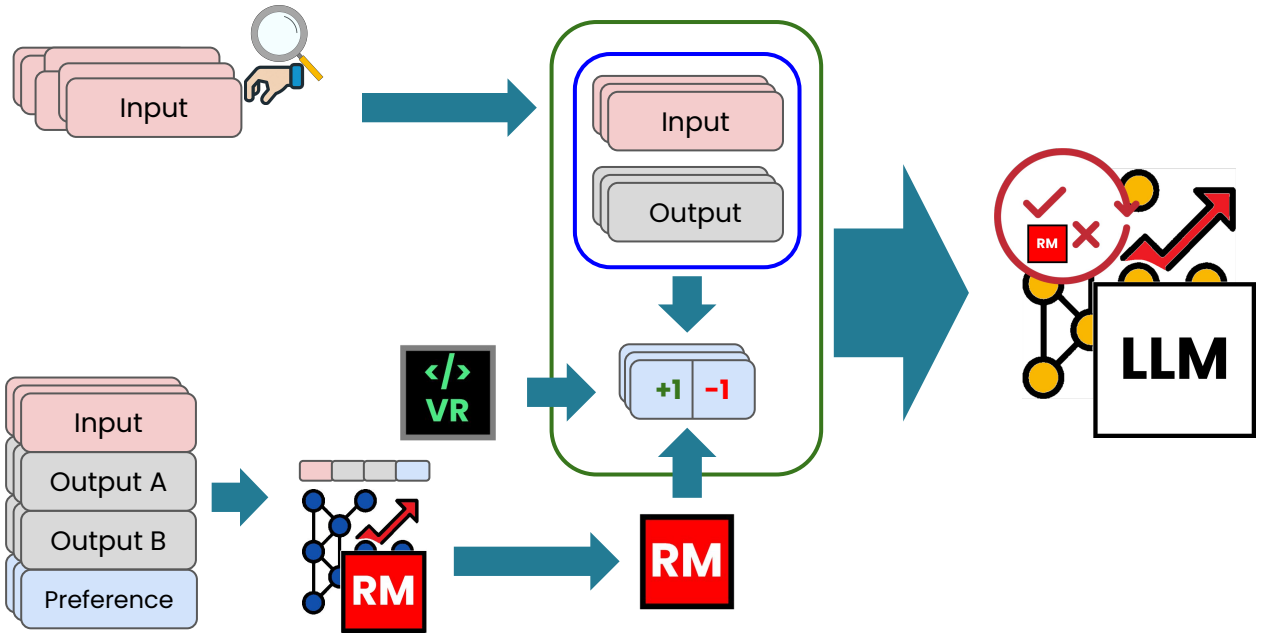
Get RL data {input, output, reward} aka. “trajectories”



Apply reward to {input, output}
→ {input, output, reward}

Data in the RL pipeline

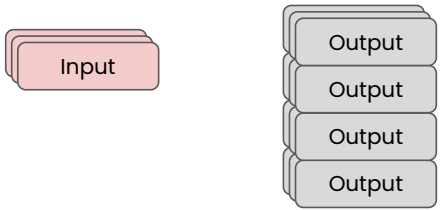
Train LLM on trajectories with RL, to maximize reward



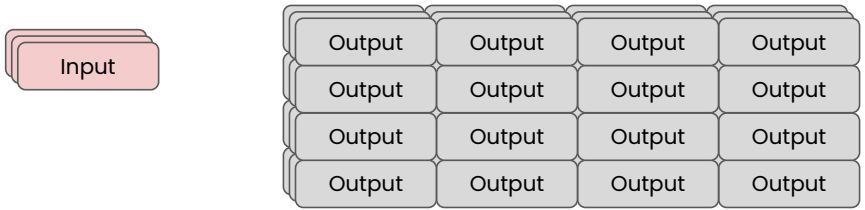
Rollout datasets: How much data?

rollouts = (# inputs) × (# model outputs per input)

Start: 8 outputs per input



Then, increase to explore diversity.



Having more outputs per input can help, but also lead to diminishing returns.

No-RM (Reward Model) LLM training

Input		Write a haiku about autumn.
Model Output		Leaves fall quickly / Seasons change with gentle wind / Quiet earth listens.
Reward		**+1** *if 5-7-5 syllable count*

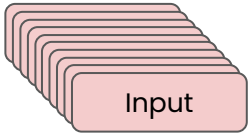
Dataset: 10k rollouts

- Enough to train an LLM that respects syllable count.
- Could just be 1k inputs!

RM-guided training with small rollout set

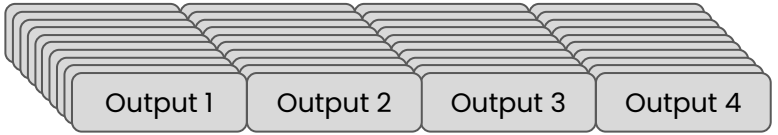
Input

1k diverse instructions



Model output

Generates 4 candidates each → 4k rollouts.



RM scores each response.

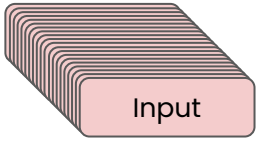
PPO for 1–2 epochs.

Result: noticeable style shift (politer, more helpful).

Scaling up RM rollouts

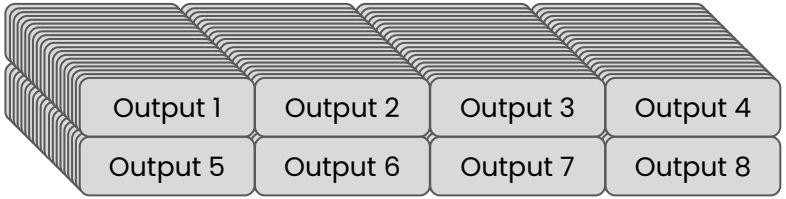
Input

20k



Model output

x 8 outputs = 160k rollouts

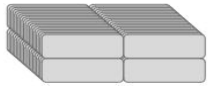


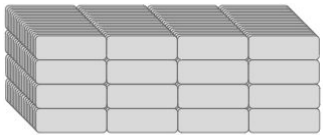
RM

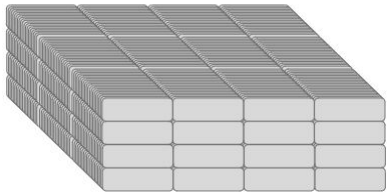
- ~100k–500k rollouts is typical for alignment tuning at mid-scale.
- Practically: Diminishing returns per extra rollout; compute & memory limits

Data efficiency: When small vs. when large?

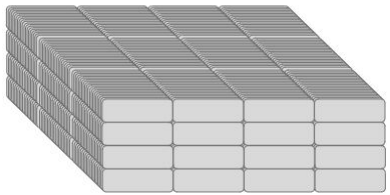
- 5k–20k rollouts: Enough to probe whether your RM actually shapes LLM performance.
- 20k–100k rollouts: Sweet spot for most mid-size LLM RL updates.
- 1M+ rollouts: Needed if aiming for domain-general improvements.



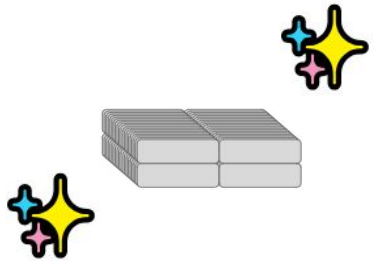




Not all rollouts are useful



Generate many rollouts...

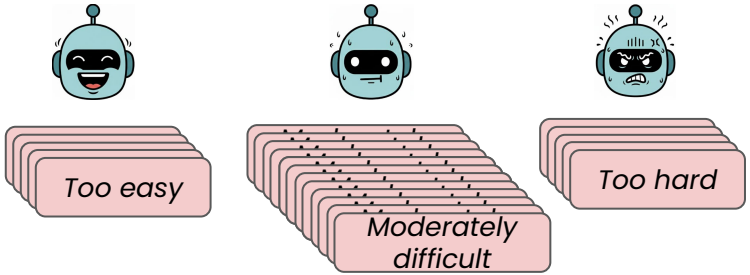


...but then train on only an “informative subset”

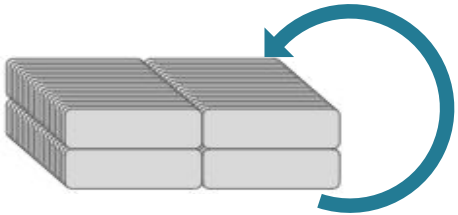
Reduces update cost. Retains signal.

[From “Not All Rollouts are Useful: Down-Sampling Rollouts in LLM Reinforcement Learning”, Xu et al., 2025]

Difficulty-targeted data and rollout replay



Focus on “moderately difficult” inputs

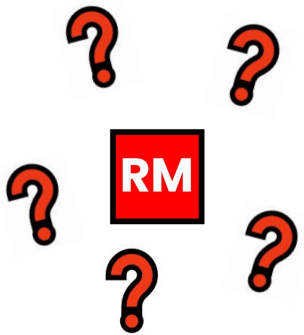


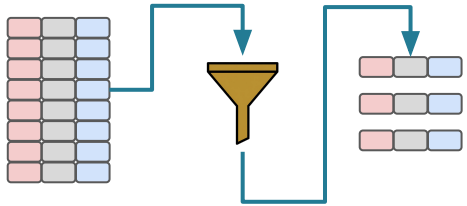
Reuse recent rollouts

Cut training cost (and rollout needs) by 25–65%

[From “Improving Data Efficiency for LLM Reinforcement Fine-tuning Through Difficulty-targeted Online Data Selection and Rollout Replay”, Sun et. al, 2025]

Filter low-quality or noisy rewards



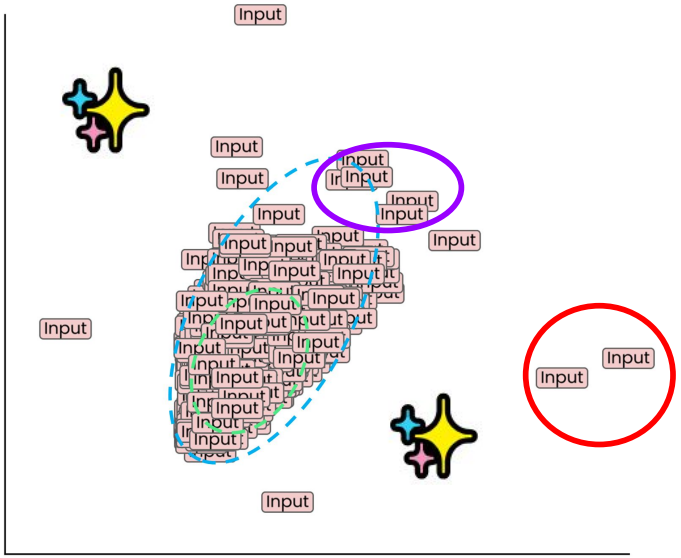


Filter out trajectories when reward model uncertainty is high

Overusing weak or noisy rollouts can hurt learning. High-quality data matters!

[From “Policy Filtration for RLHF to Mitigate Noise in Reward Models”, Zhang et al., 2025]

Similar to fine-tuning in many ways



Data quality and diversity matter just as much as in Fine-tuning

LoRAs can be used to update the LLM for efficiency as well

[From “Efficient RLHF: Reducing the Memory Usage of PPO”, Santacroce et al., 2023]

DeepLearning.AI

Data Driven Post-training

Data for RL (Part 2)

Reward shaping

Prompt

Explain how solar panels work.

Response

They absorb light and magically produce energy.

Reward Components

Factuality: **-1**

Style: **+0.5**

Completeness: **+0.2**

Total reward = **-0.3**

Calibration: RMs often overfit to surface features

Input

Explain how solar panels work.

Model Output

They absorb light and magically produce energy....

Calibration: RMs often overfit to surface features

Input

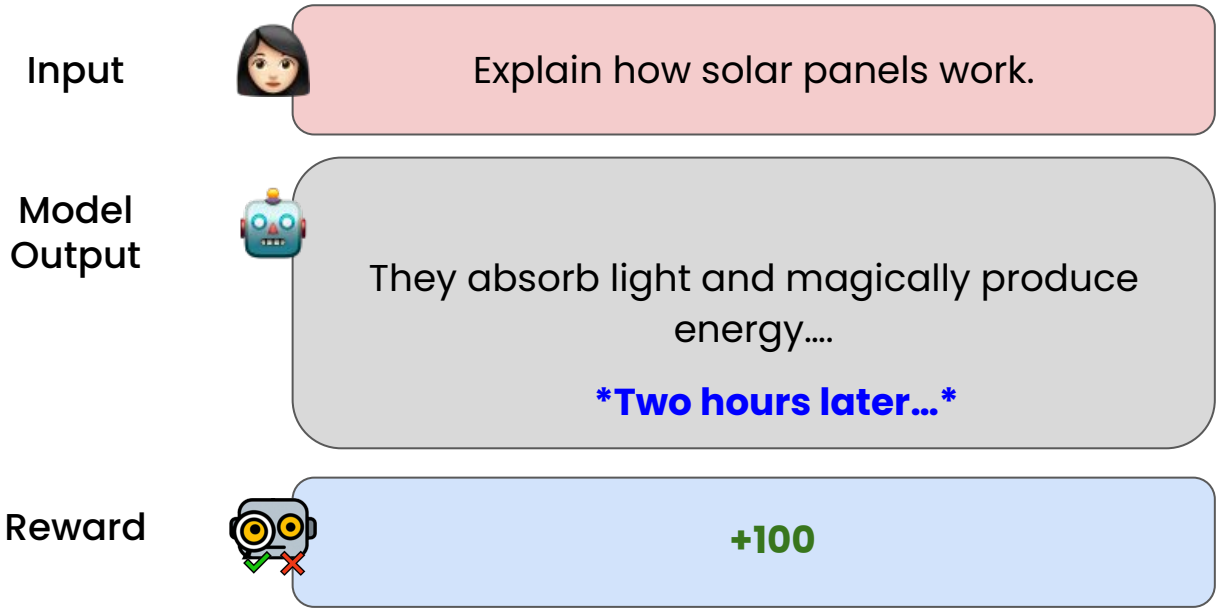
Explain how solar panels work.

Model Output

They absorb light and magically produce energy....

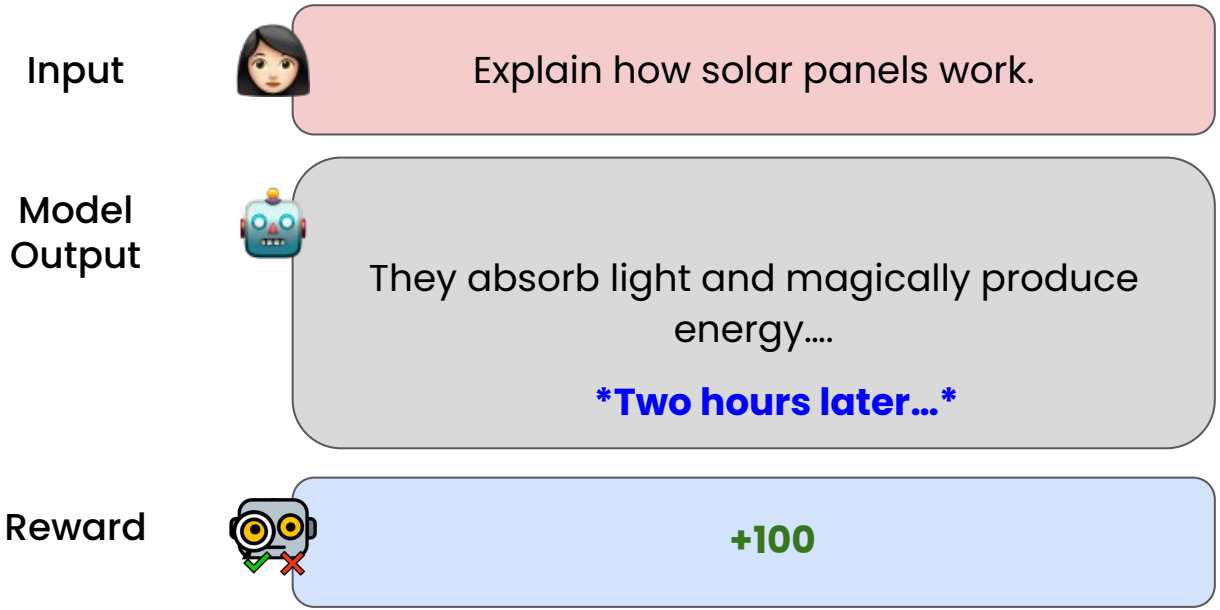
Two hours later...

Calibration: RMs often overfit to surface features



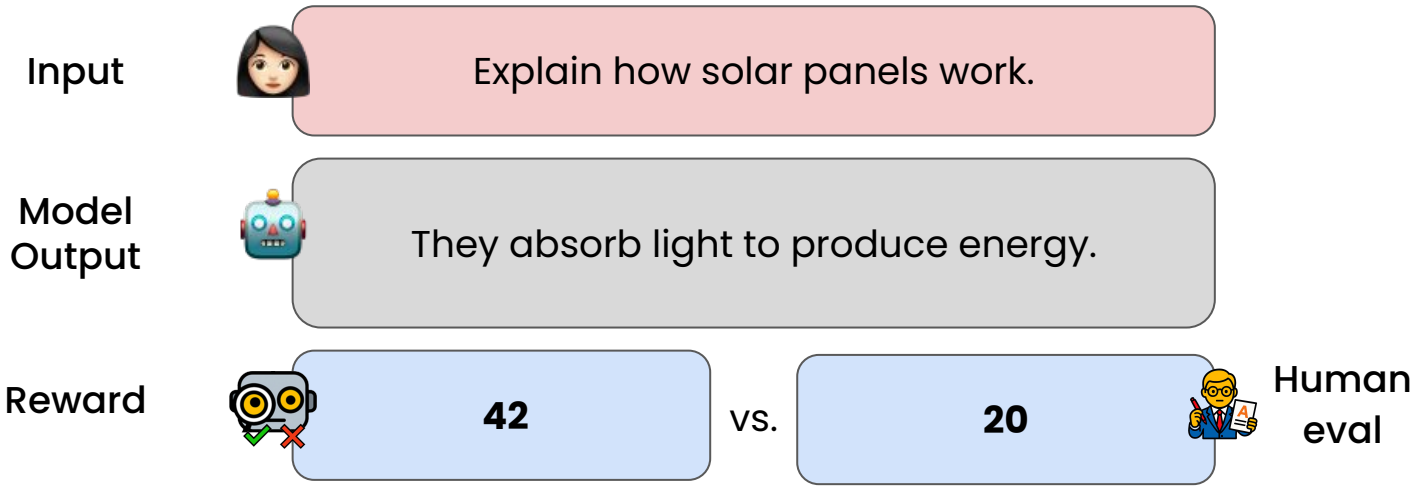
Indirectly, “length = good” in preference data

Calibration: RMs often overfit to surface features



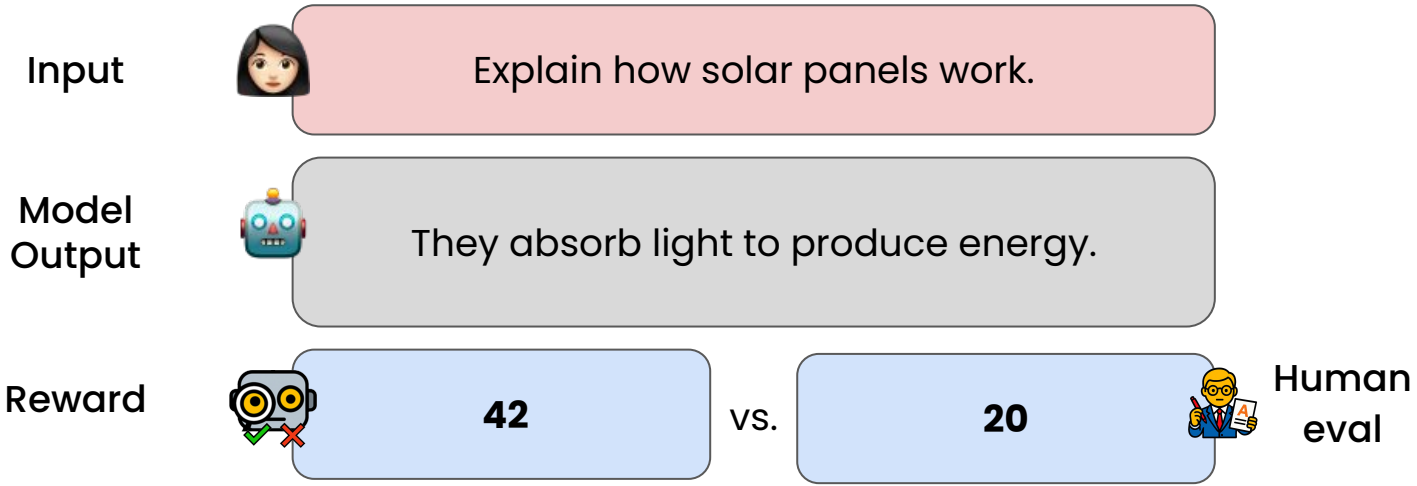
Fix: Add counterexamples. Long but bad outputs. Short but excellent outputs.

Alignment tax: “Max reward” vs “most useful to humans.”



Both are rated in the right direction, but the alignment tax is 42-20=20.

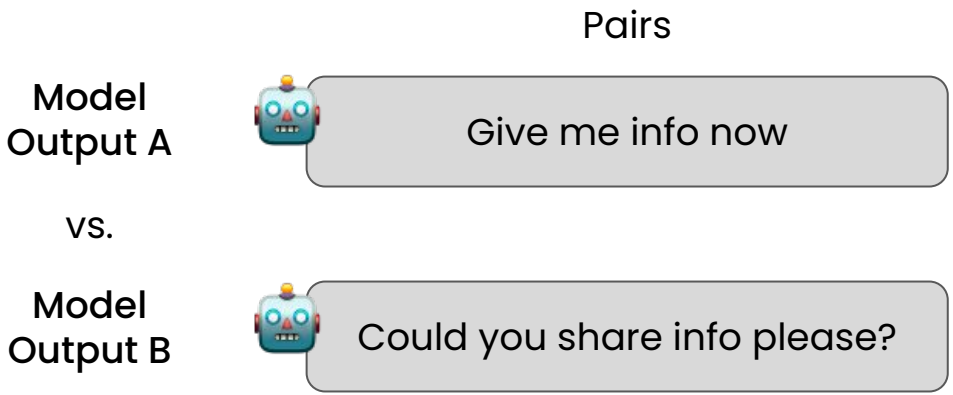
Alignment tax: “Max reward” vs “most useful to humans.”



Fix: Collect targeted counterexamples to reduce it.

RM data: Small scale (~100 preference pairs)

Task: Make chatbot polite.



~50 examples enough → model learns politeness marker

Enough if you're only steering tone (politeness, refusal).

RM data: Medium scale (1k–10k pairs)

Task: Optimize response length and informativeness

Pairs

Model Output A

The French Revolution started in 1789

vs.

Model Output B

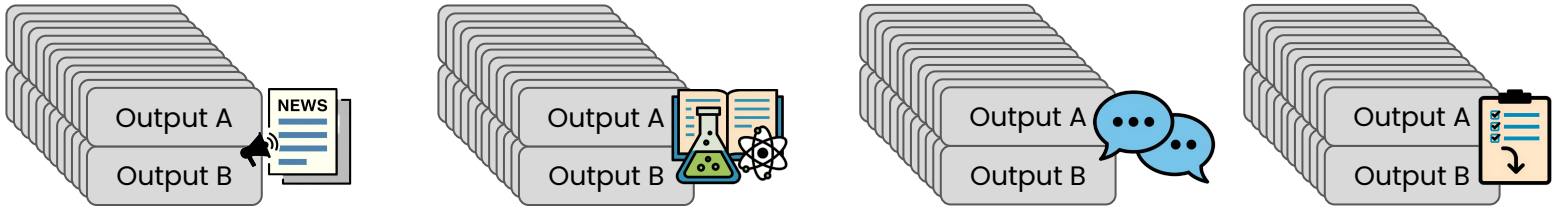
The French Revolution began in 1789, leading to widespread political change and Napoleon's rise. It was an important time in history, one for the history books.

~5,000 examples → model learns nuanced trade-offs, e.g. factuality or harmlessness.

RM data: Large scale (~100k pairs)

Task: Open-domain helpfulness.

Pairs span



Must cover thousands of diverse prompts → hundreds of thousands of pairs needed.

Multi-intent, high-variance tasks, e.g. open QA, or summarization across domains.

Online vs. offline preference learning

Offline: Train RM once on pre-collected data.

Stable, reproducible.

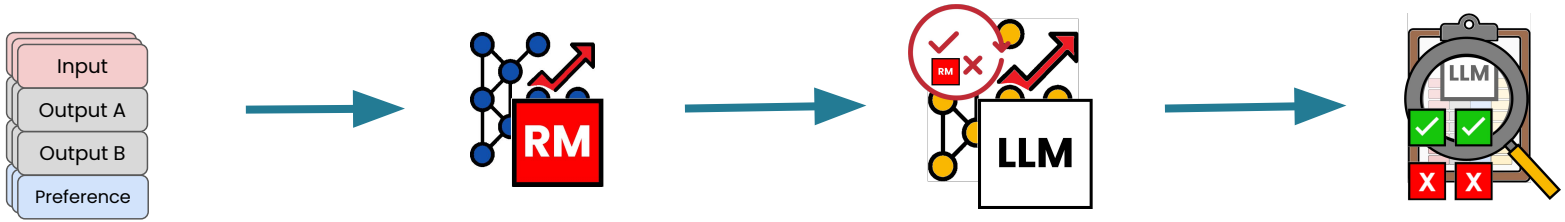
Limited by data coverage (can't explore new failure modes).

Gather 10k preference pairs.

Train RM offline

Train LLM with RM

Evaluate

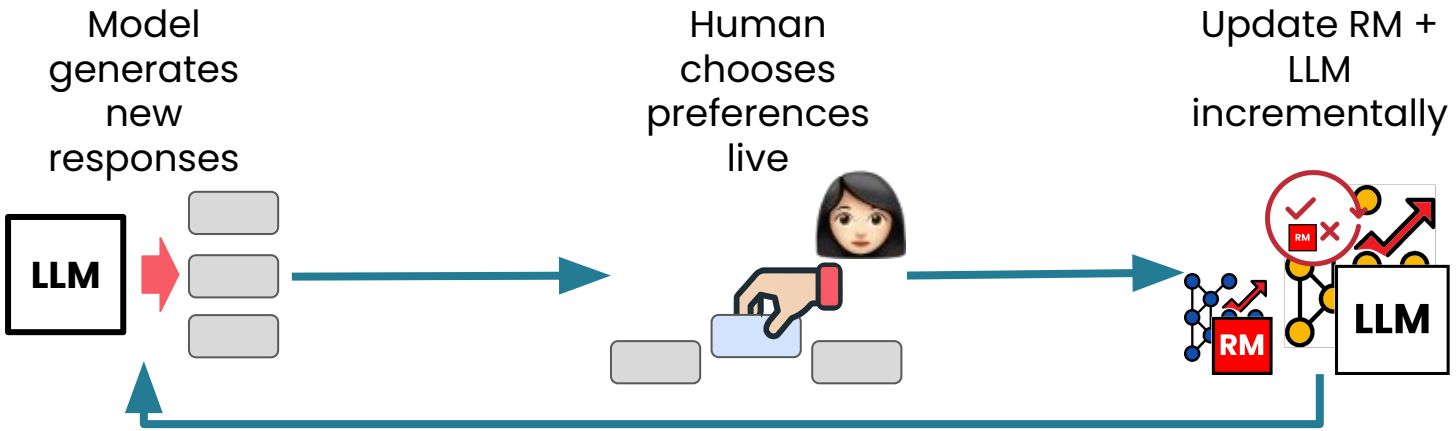


Online vs. offline preference learning

Online: Collect new preferences on-the-fly from current model outputs.

Adaptive and finds new methods & exploits.

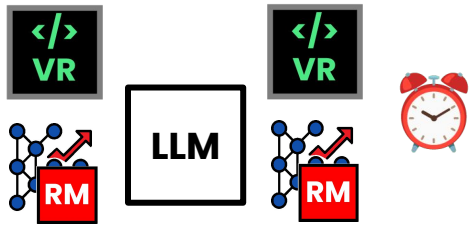
More expensive with human-in-the-loop or synthetic raters.



Repeat → adapts to new exploit (like repetition or flattery).

Expensive or slow reward signals

Expensive or slow: Some verifiers and reward models could give slower or expensive - yet very strong - signals.



e.g. long reasoning chains, or large function that verifies if the code is high-performance

DeepLearning.AI

Data Driven Post-training

Putting it together

DeepSeek R1 pipeline

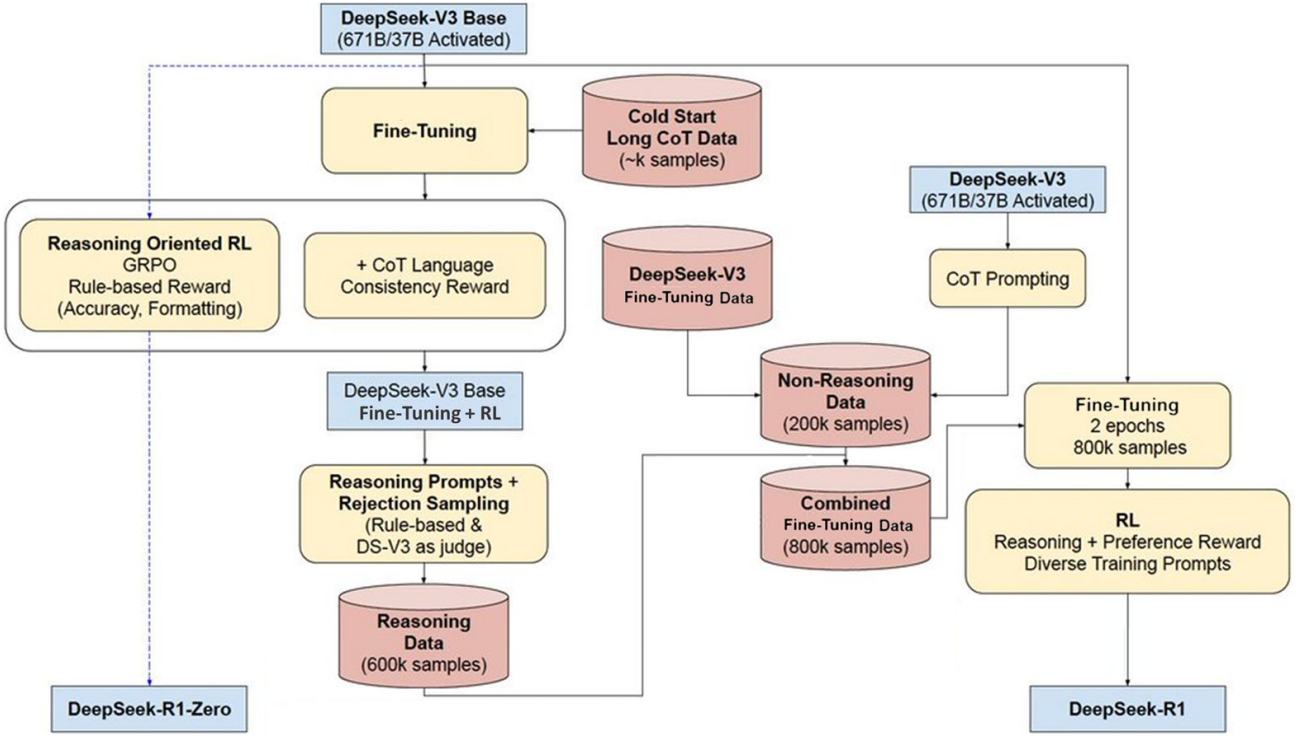


Figure from: https://arcprize.org/blog/r1-zero-r1-results-analysis

DeepSeek R1 pipeline

We will cover each dataset...

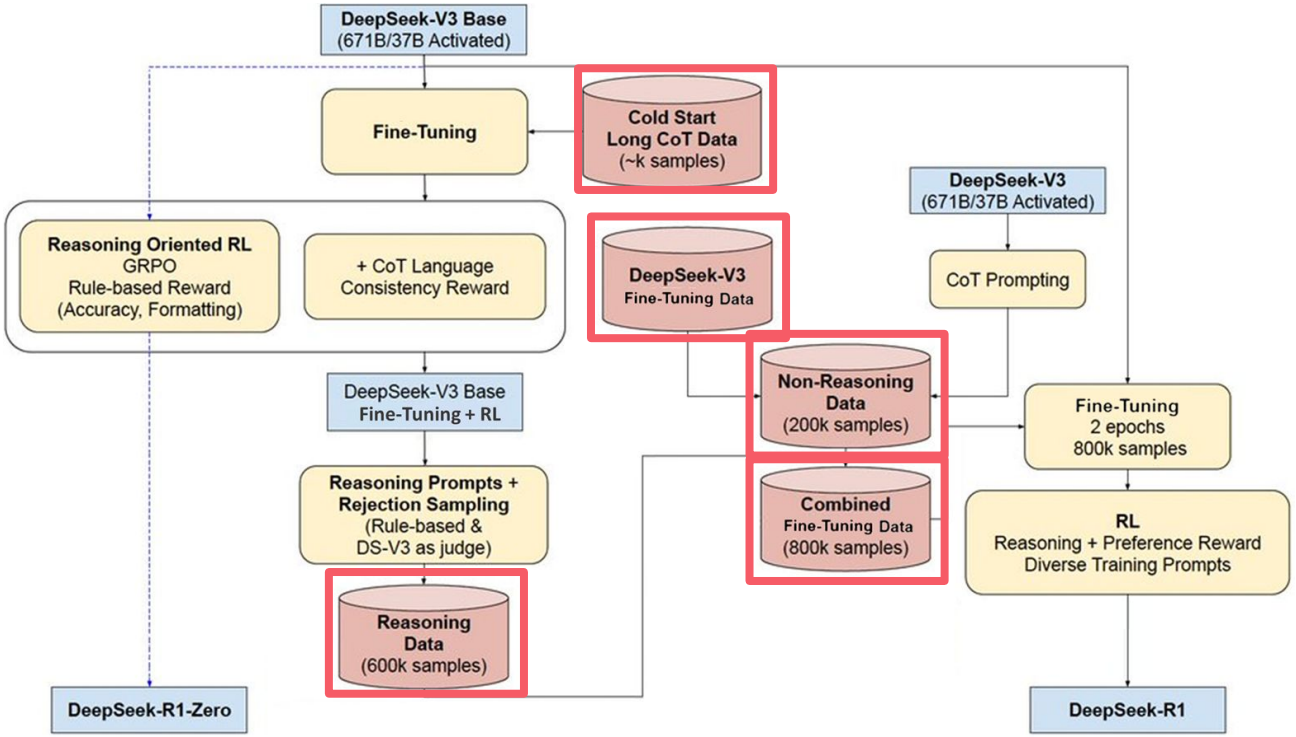


Figure from: https://arcprize.org/blog/r1-zero-r1-results-analysis

DeepSeek R1 pipeline

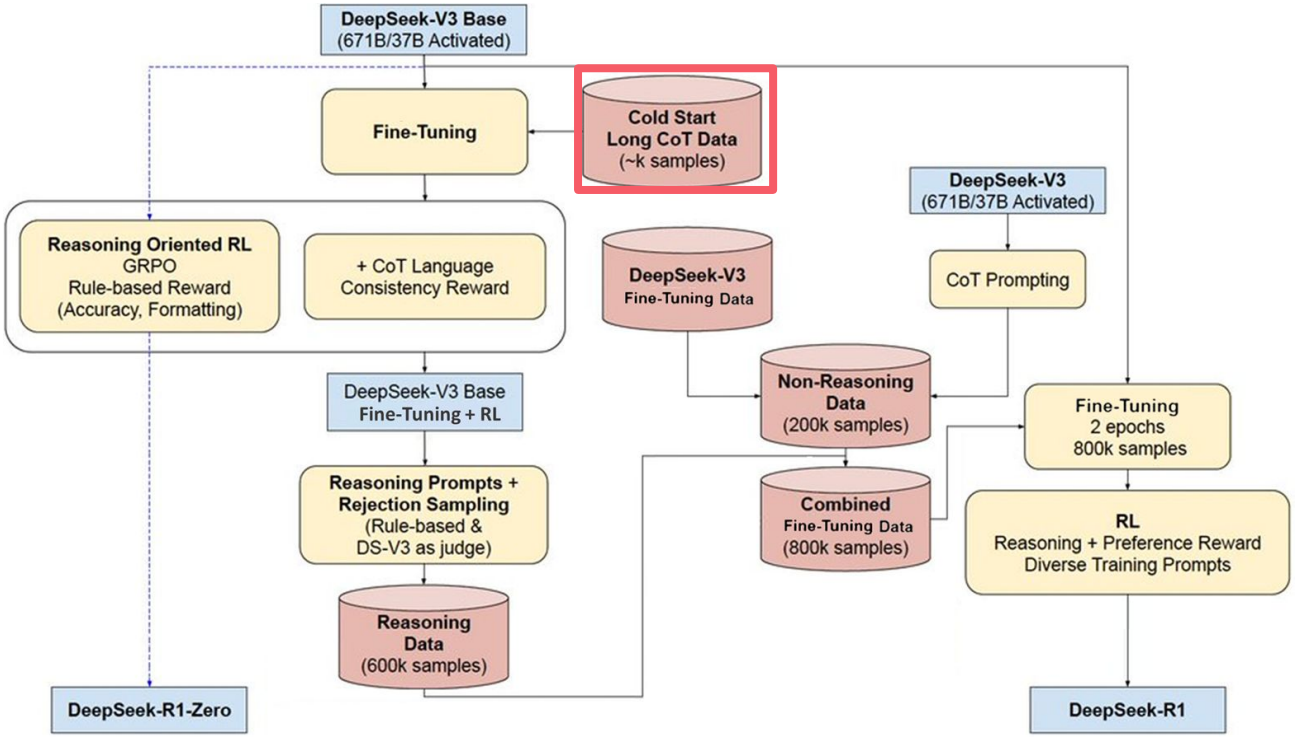


Figure from: https://arcprize.org/blog/r1-zero-r1-results-analysis

Cold-start long CoT fine-tuning data

Input

Alice has 3 apples and buys 2 more.
How many now?

Target Output

<think>
Start with 3.
Buying more things should be additive.
If so, then 2 ⇒ 3+2
...
</think>

~k examples = Small "seed" dataset

Long: very detailed reasoning

"Cold-start": bootstraps reasoning style and format.

DeepSeek R1 pipeline

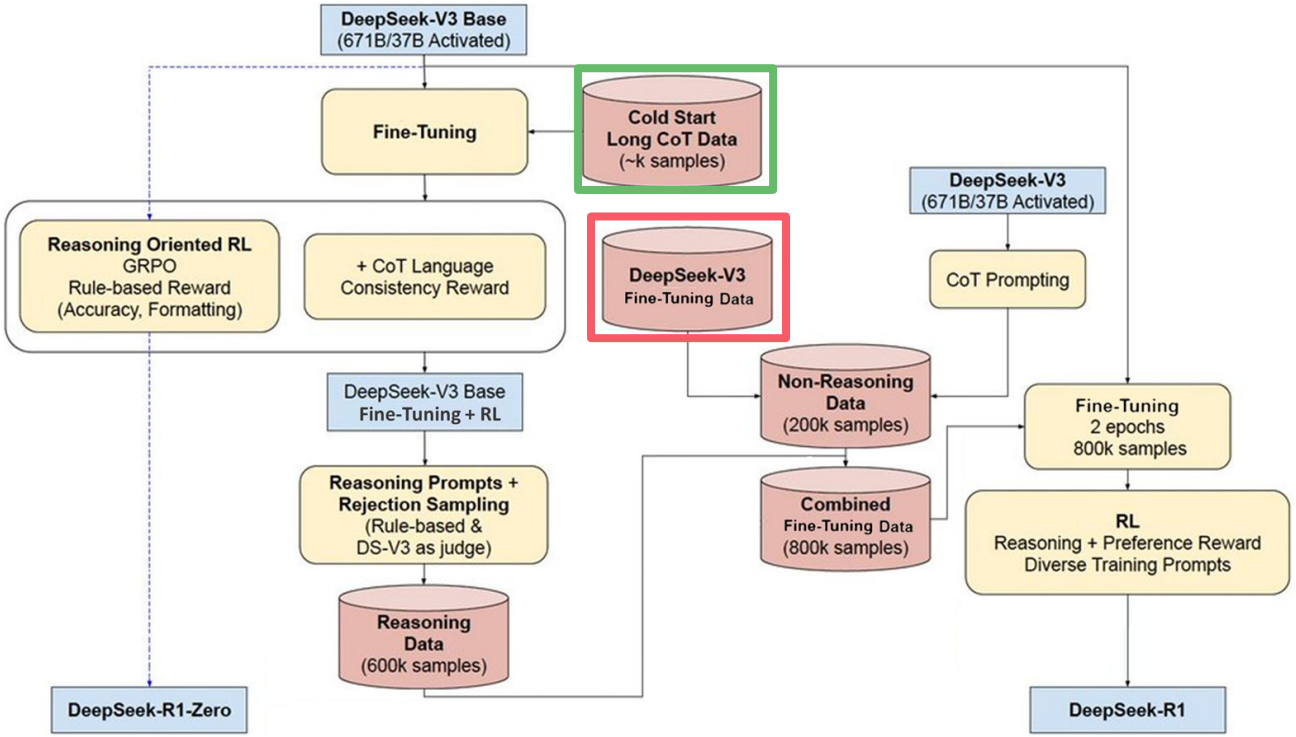


Figure from: https://arcprize.org/blog/r1-zero-r1-results-analysis

Regular fine-tuning data

Input	 Alice has 3 apples and buys 2 more. How many now?
Target Output	There are 5 apples.

DeepSeek R1 pipeline

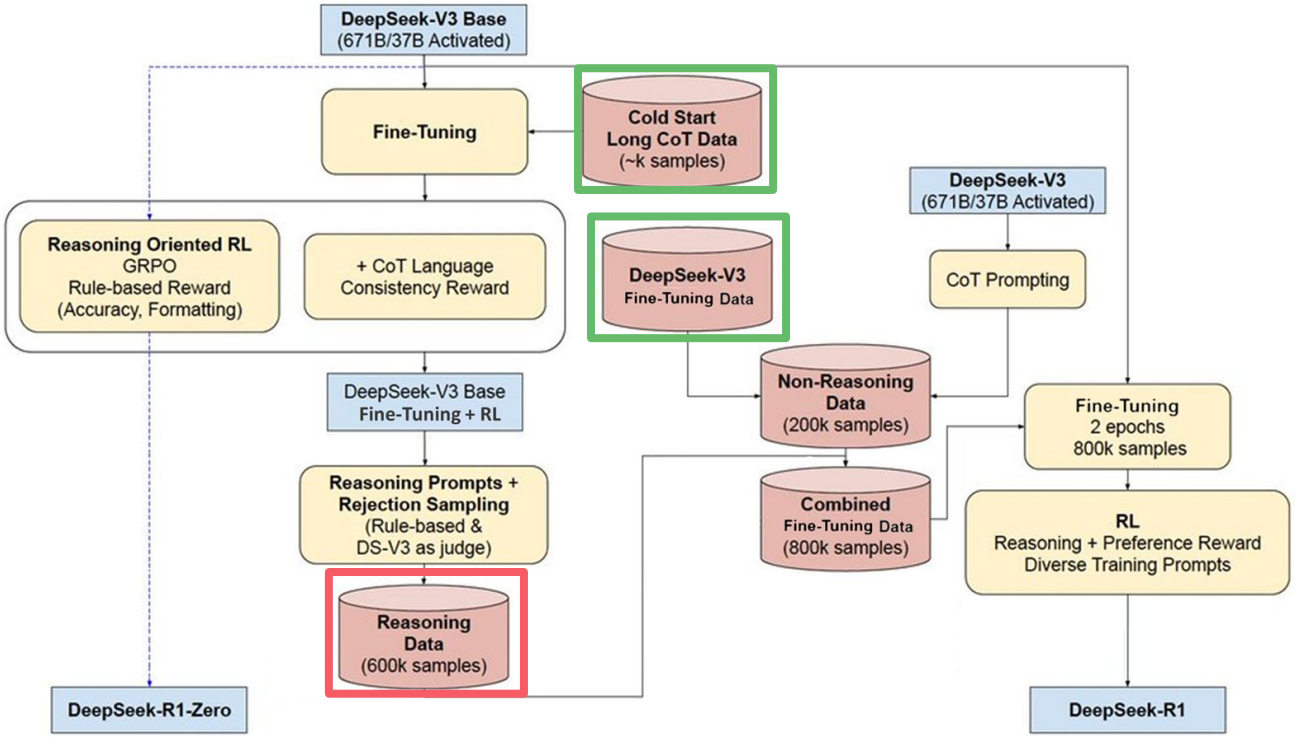


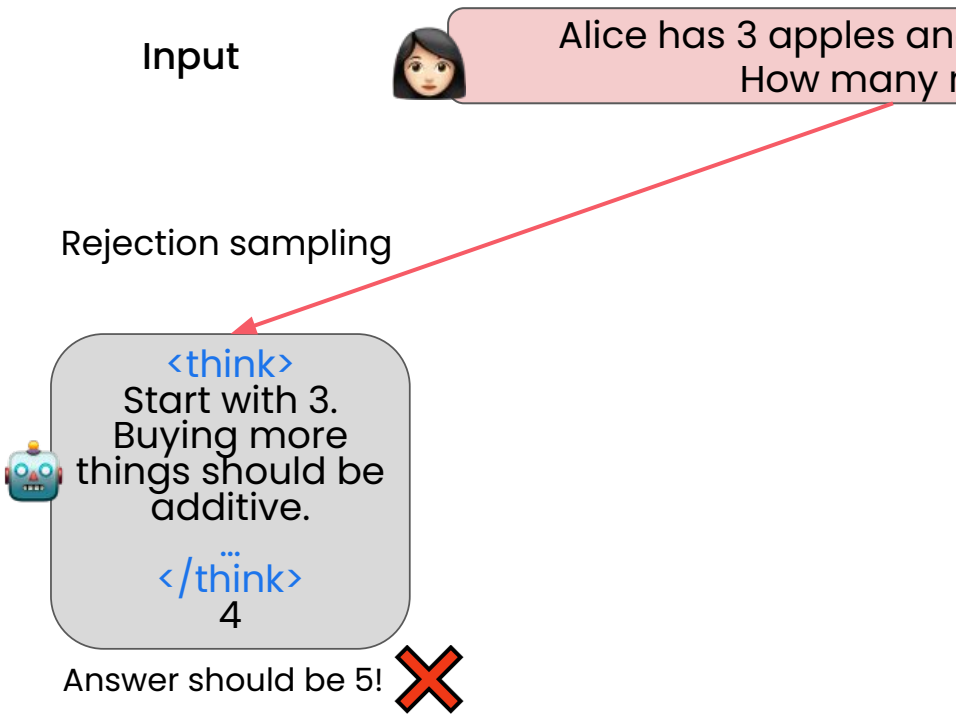
Figure from: https://arcprize.org/blog/r1-zero-r1-results-analysis

Synthetic reasoning data

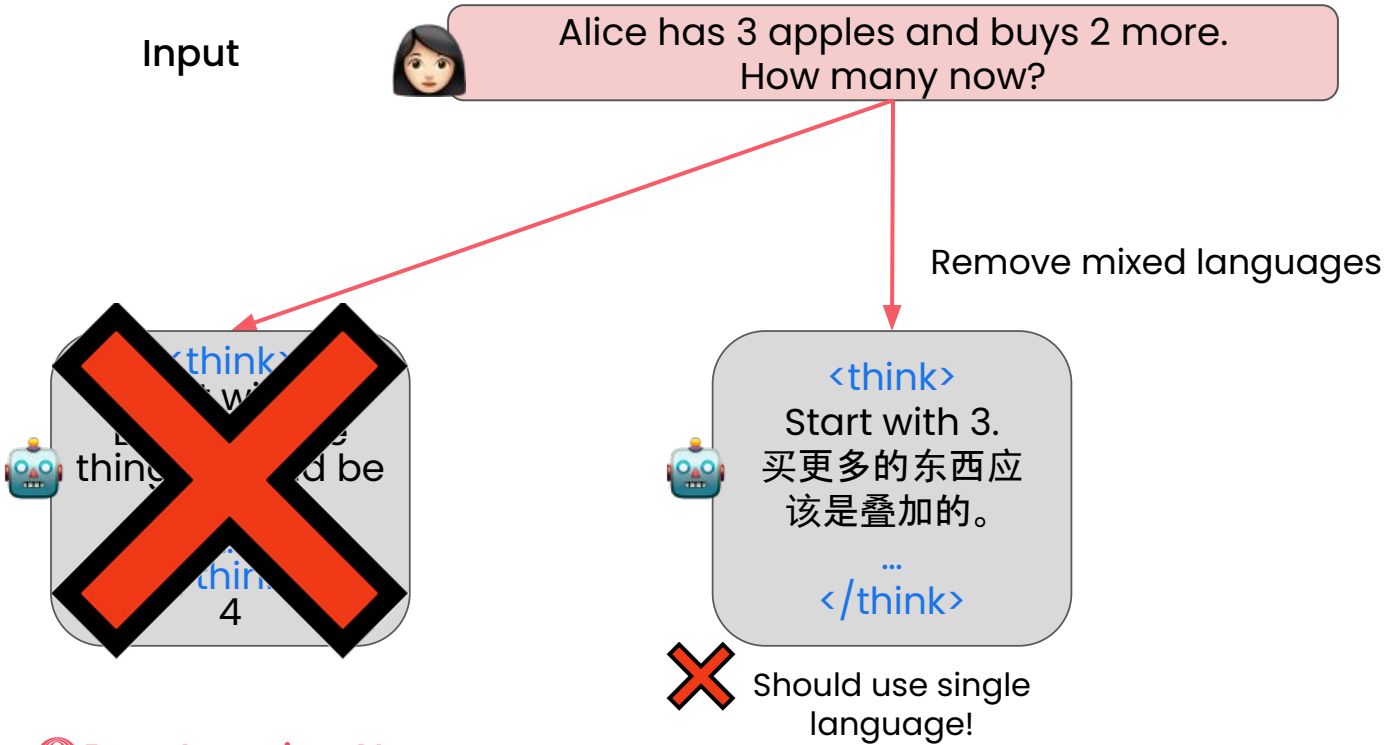
Input

Alice has 3 apples and buys 2 more.
How many now?

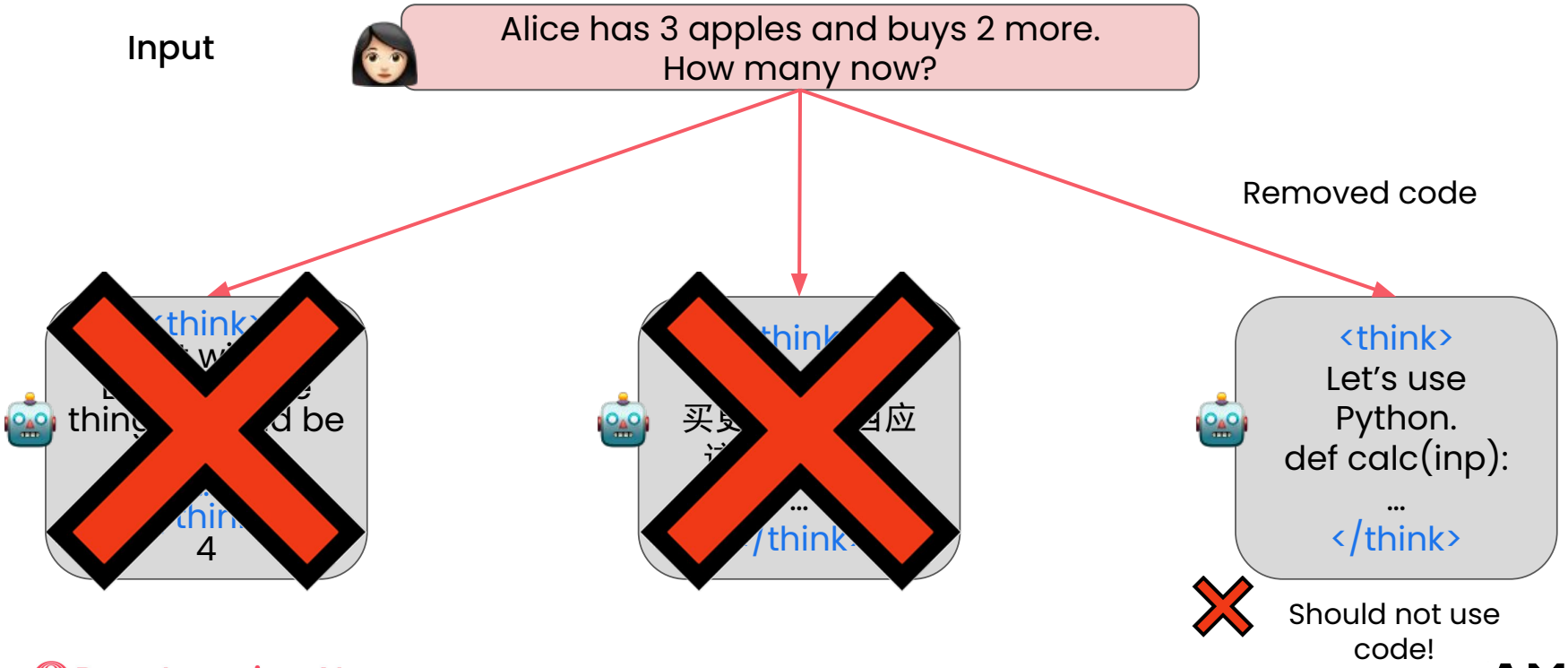
Synthetic reasoning data



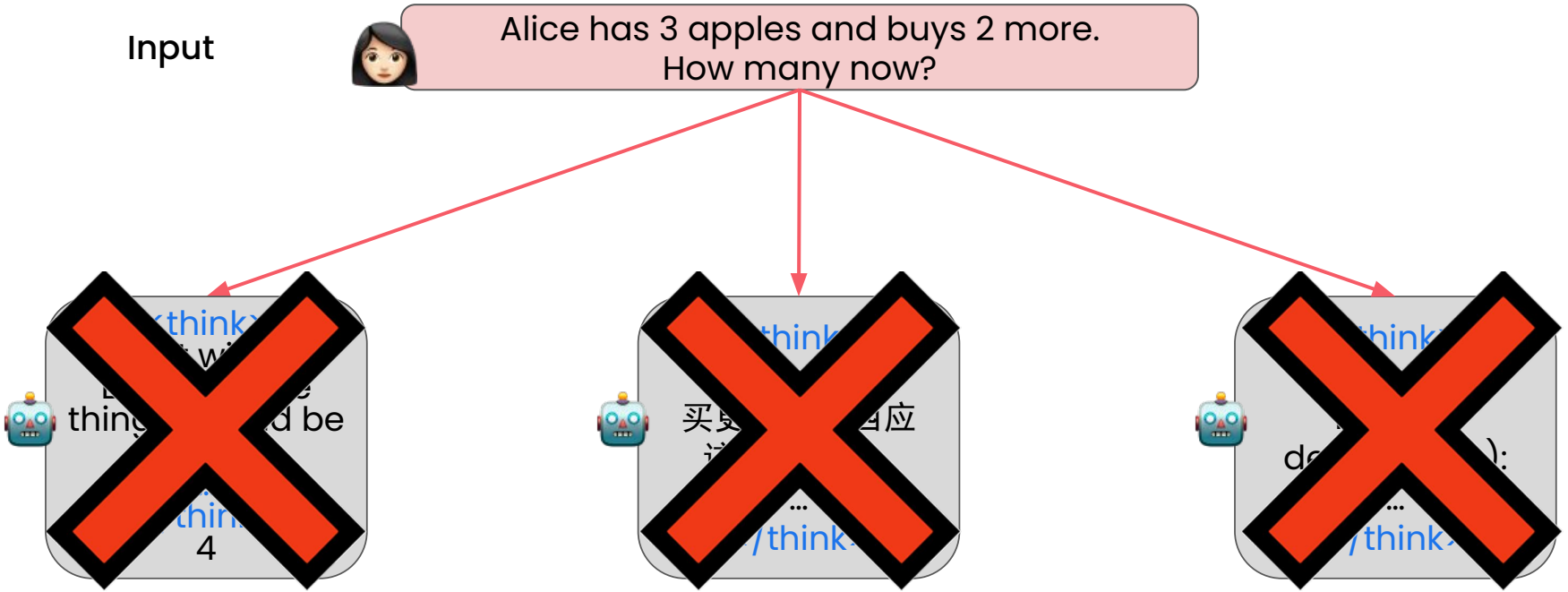
Synthetic reasoning data



Synthetic reasoning data



Synthetic reasoning data



DeepSeek R1 pipeline

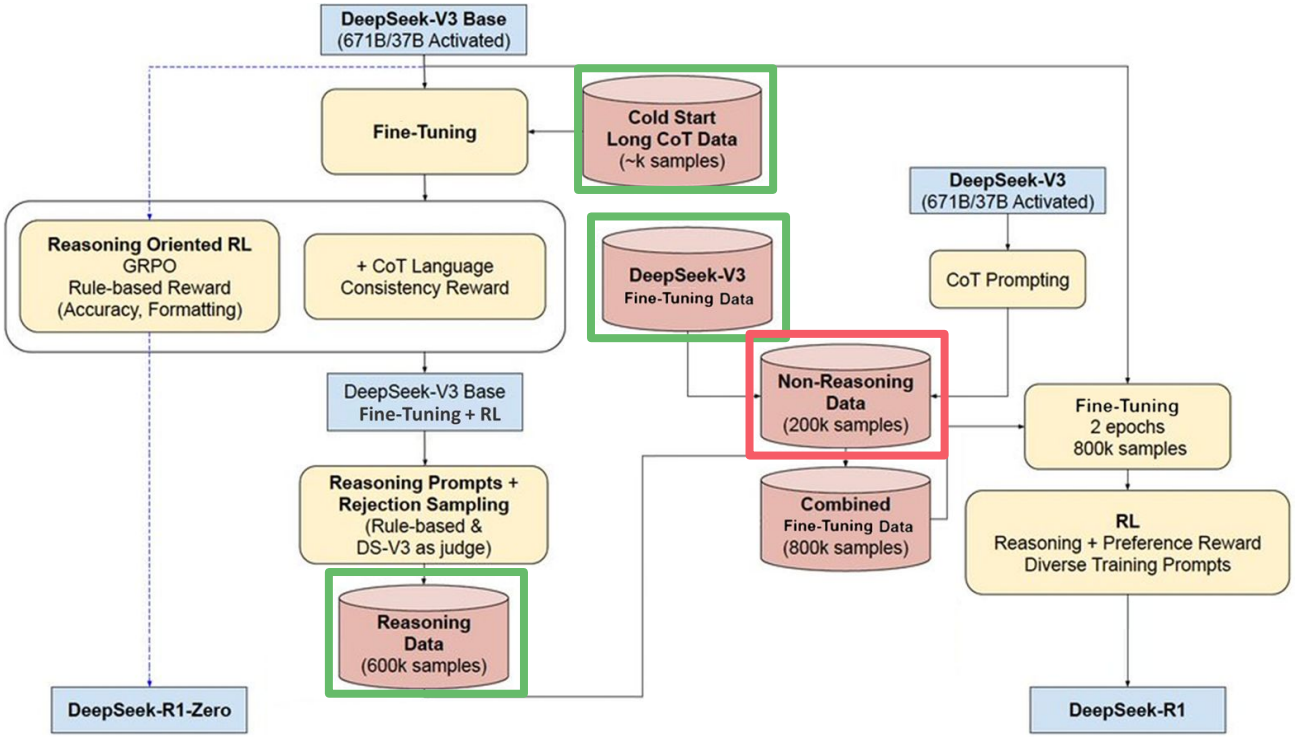
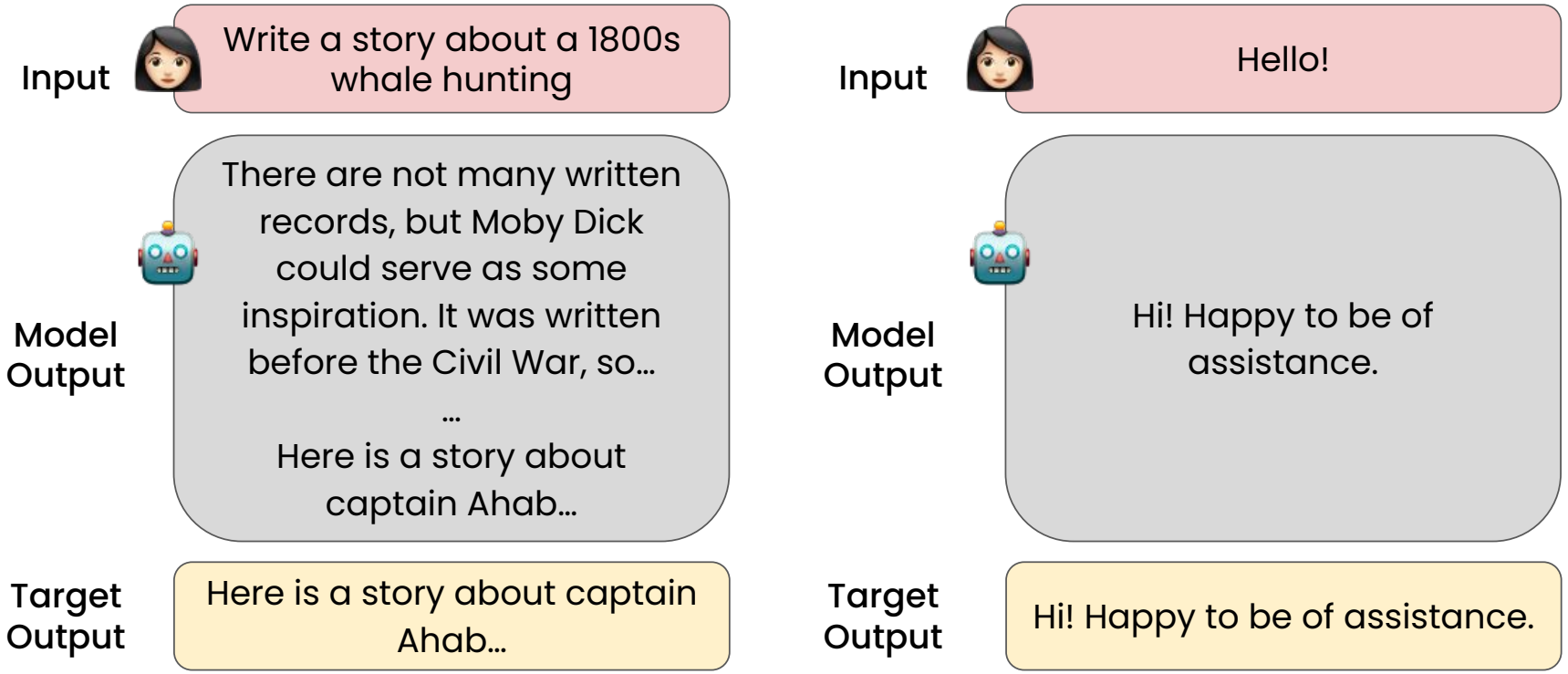


Figure from: https://arcprize.org/blog/r1-zero-r1-results-analysis

Synthetic non-reasoning data



DeepSeek R1 pipeline

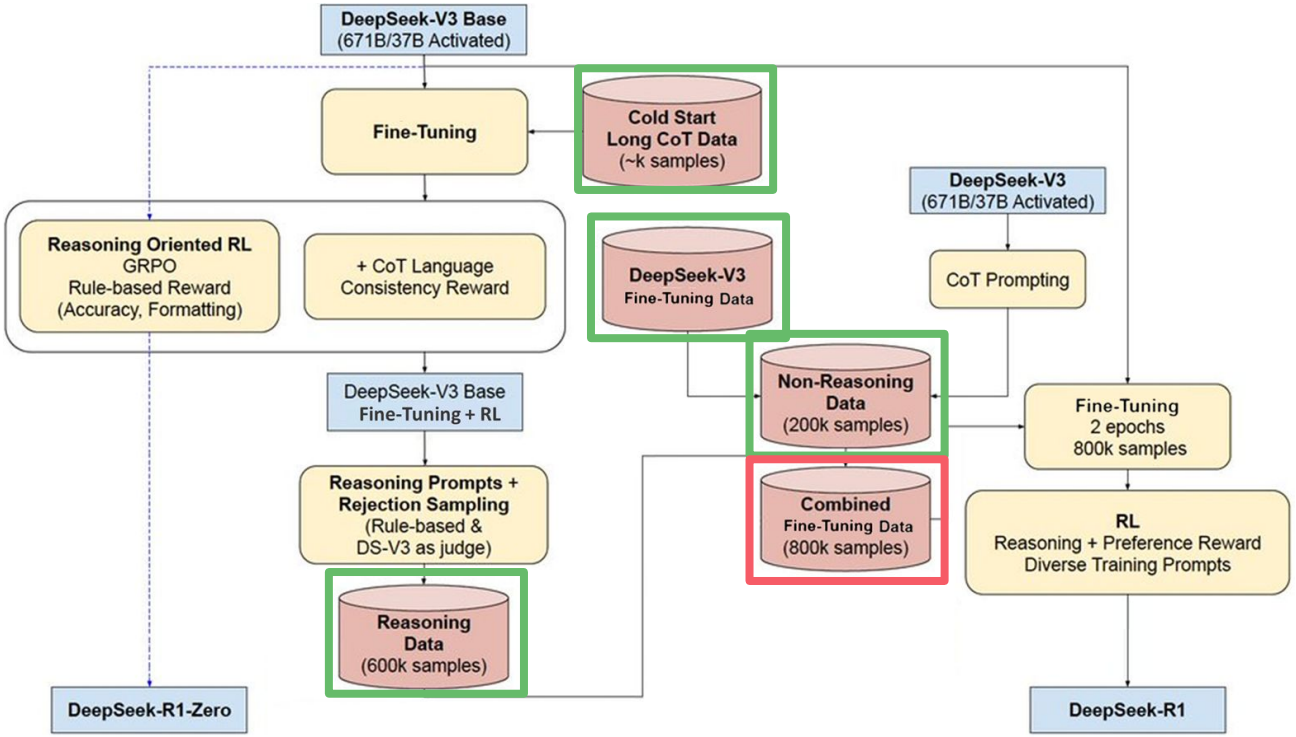
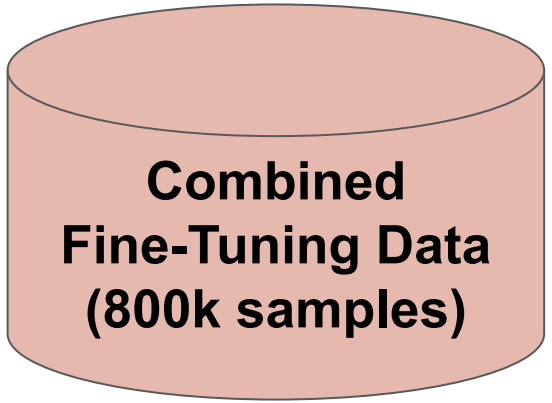


Figure from: https://arcprize.org/blog/r1-zero-r1-results-analysis

Combined reasoning & non-reasoning

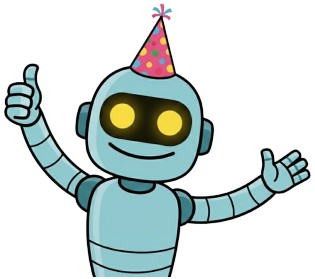
800k samples:

- 25% non-reasoning (200K)
- 75% reasoning (600K)



DeepSeek R1 pipeline

And we're done...



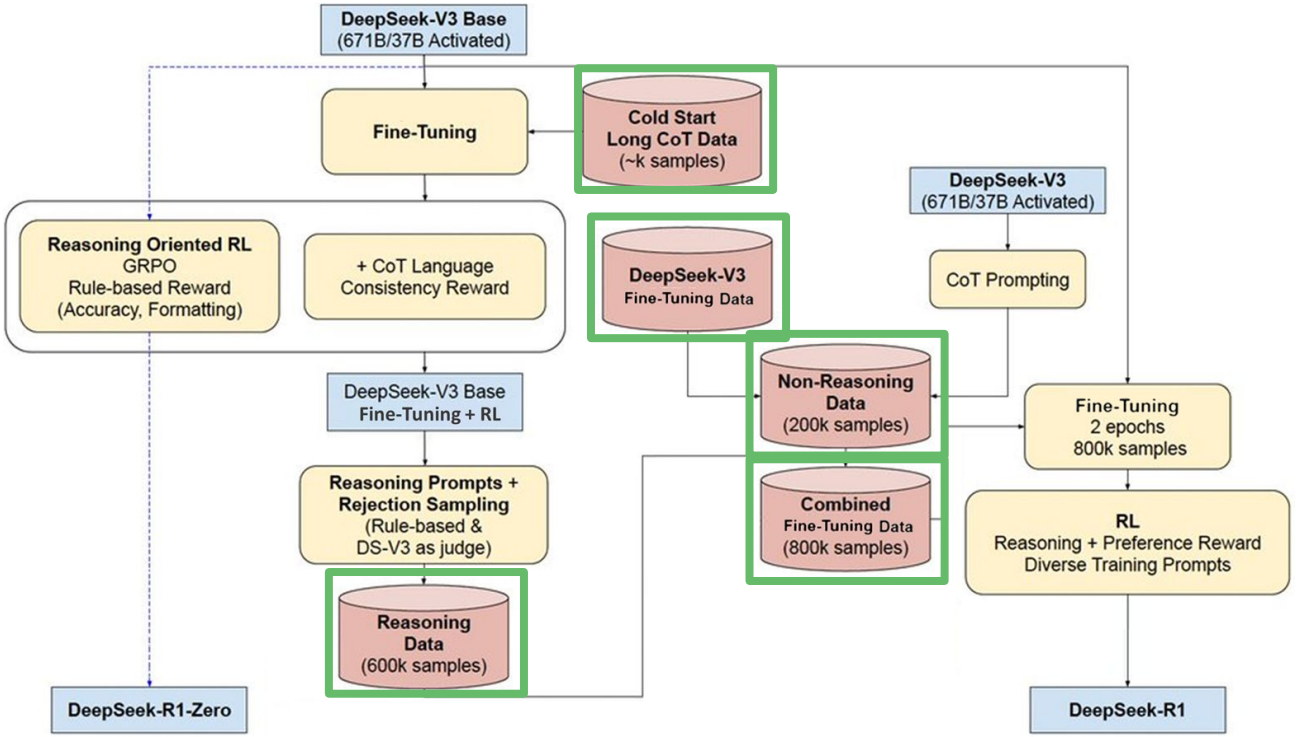


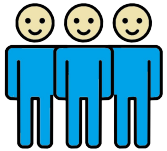
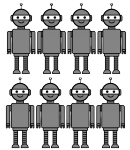
Figure from: https://arcprize.org/blog/r1-zero-r1-results-analysis

DeepLearning.AI

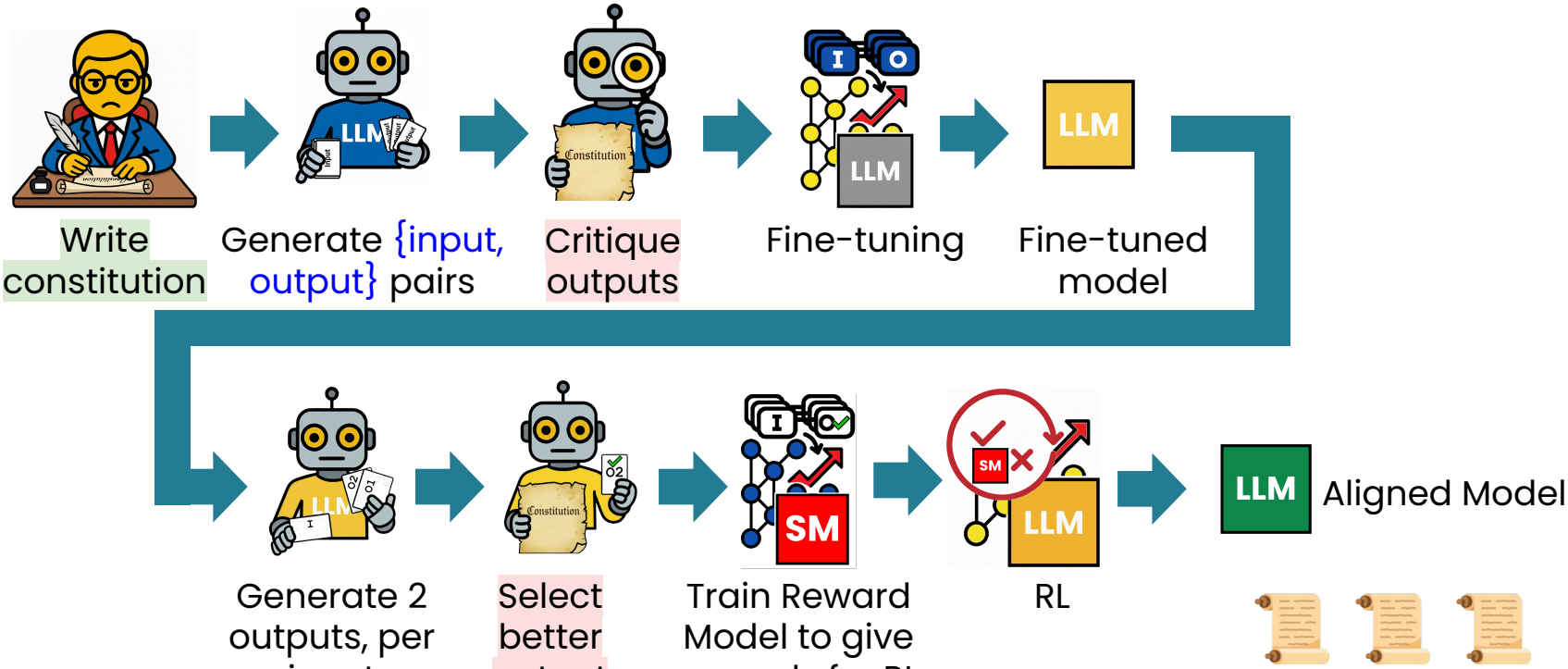
Data Driven Post-training

Synthetic data pipelines

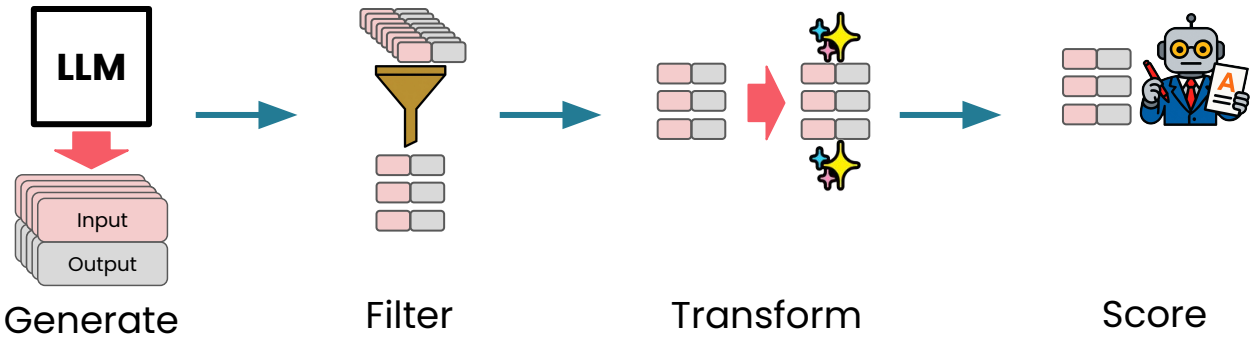
Why synthetic?

Human		Synthetic
Excellent quality		 Good quality
Slow		 Fast
Expensive		 Cheap
Hard to scale		 Easy to scale

Synthetic data recipe in Constitutional AI



Synthetic data: Generate, filter, transform, score



Generate data with LLMs

Input template

Solve this coding question step by step:
{question}

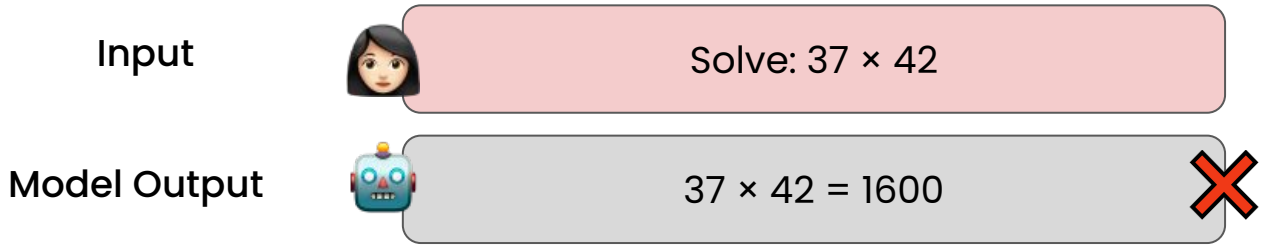
Model Output

To find the sum of an array, loop through elements, accumulate into total, return total

Methods:

- Prompt-based generation: use prompt templates.
- Sampling diversity: vary temperature, top-p, decoding strategies.
- Task expansion: mutate existing tasks into harder/easier versions.

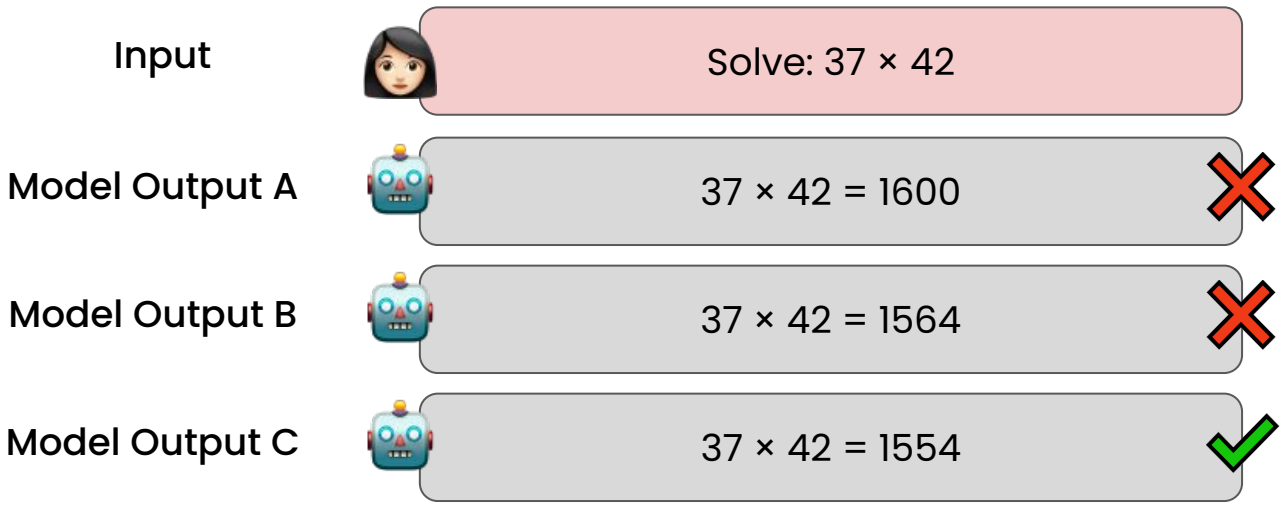
Generate data with LLMs: Rejection sampling



Generated data is not correct. Pass on it.

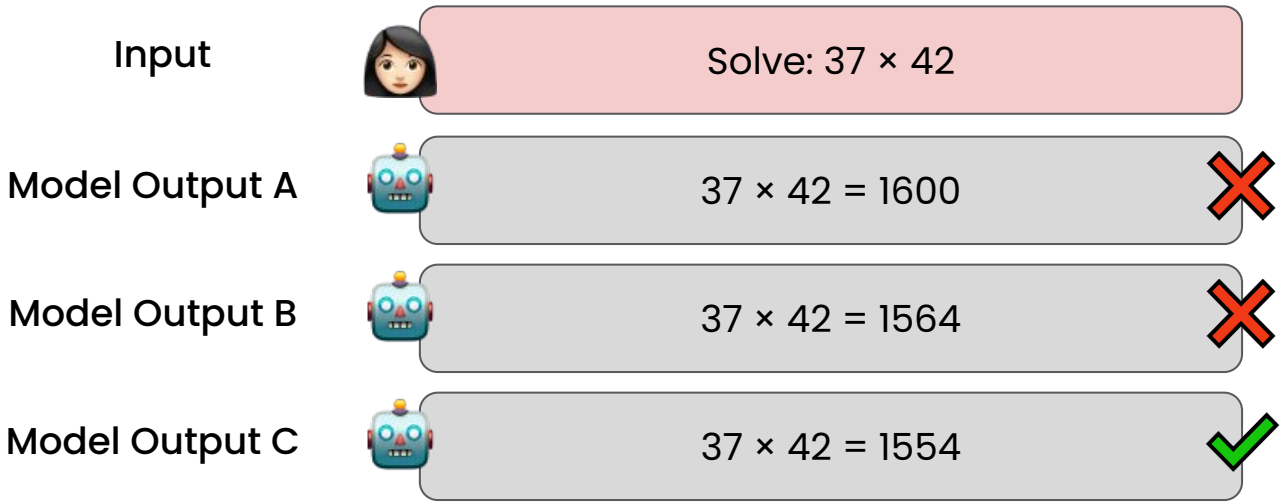
Problem: Can't use LLMs to generate for this type of input!

Generate data with LLMs: Rejection sampling



Rejection sampling to get best 1 out of 3 outputs (k-to-1 filtering)

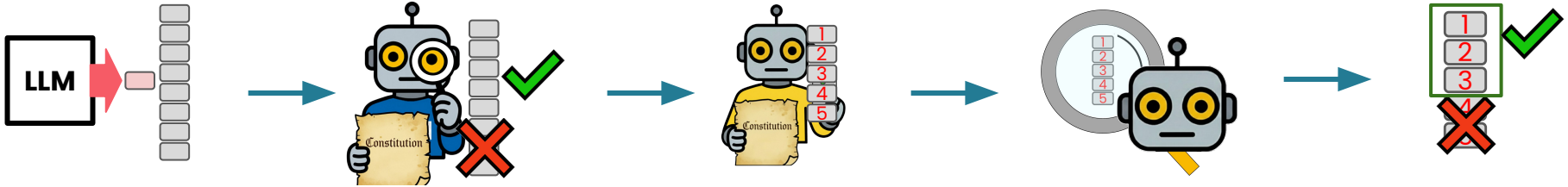
Generate data with LLMs: Rejection sampling



Rejection sampling can also filter with LLM as judge

Filter data with LLMs: Rejection sampling

Practical Pipeline



Generate k candidates.

Run constitution LLM judge → eliminate clearly poor ones.

Run LLM pairwise preferences on the survivors → rank order.

Check agreement with self - consistency or majority.

Keep top N as filtered high-signal samples.

Filter data with LLMs

Filtering saves you from training on noise. Without it, data generation on its own wouldn't be as effective.

Garbage out

Synthetic data can be "garbage in"

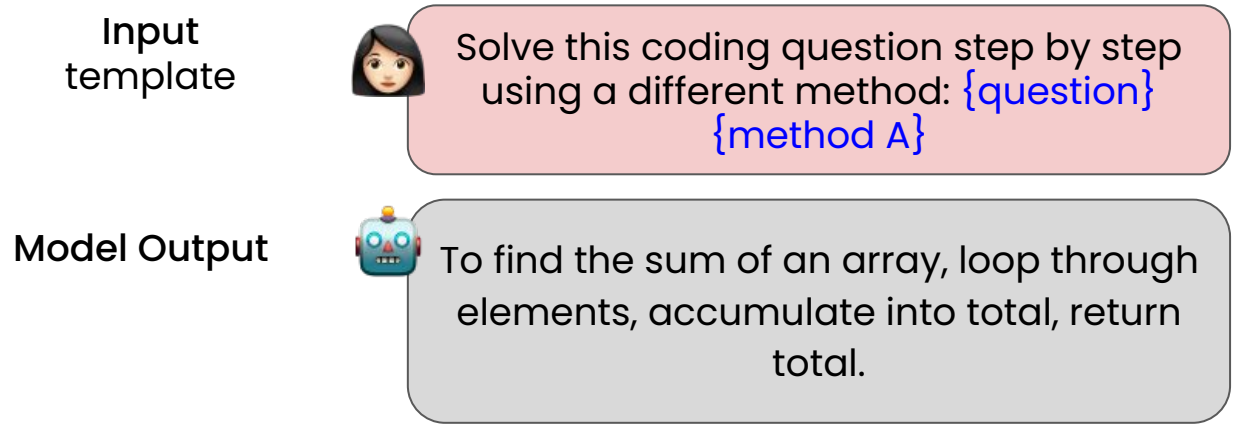
Removal

Toxicity
Profanity
Format issues

Validation

Accuracy
Coherence
Relevance

Filter data with LLMs



- Verifier filters: “Verify another way to determine correctness.”
- Collapse detection: “Mark redundant outputs.”
- Semantic majority filtering: “Does this answer agree with the 3 other answers?”

Transform data with LLMs

Input		Explain gravity
Model Output A		Step 1: Earth pulls you with force… Step 2: You pull Earth… Step 3: Mass difference explains why Earth doesn't move much.
Transformed succinct answer		Gravity is a mutual attraction between masses; Earth's huge mass makes it dominant.

Distillation: Compress verbose CoT → concise answer.

Reshape raw outputs into training-ready, high-signal data

Transform data with LLMs

Input		Explain gravity
Model Output A		Step 1: Earth pulls you with force… Step 2: You pull Earth… Step 3: Mass difference explains why Earth doesn't move much.
Transformed succinct answer		Gravity is a mutual attraction between masses; Earth's huge mass makes it dominant.

Normalization: Standardize tone/style.
Augmentation: Paraphrase instructions, translate, or expand into step-by-step versions.
Curriculum shaping: Bucket into easy → hard tasks.

Reshape raw outputs into training-ready, high-signal data

Score data with LLMs

```
{
  "correctness": 0.9,
  "completeness": 0.8,
  "conciseness": 0.7,
  "style_fit": 0.6,
  "final_score": 0.75
}
```

→ Keep if score > 0.7.

- LLM judge: rate on clarity, correctness, completeness (1–5).
- Similar to filtering data: collapse detection, self-consistency (majority) scoring
- Preference comparisons: “Which of these two is more helpful?”

Score data with LLMs: In your code

```
step_indicators = [
        r'step \d+', r'first', r'second', r'third', r'next', r'then',
r'finally',
        r'calculate', r'find', r'determine', r'multiply', r'divide', r'add',
r'subtract'
    ]

step_count = sum(len(re.findall(indicator, model_output)) for indicator in
step_indicators)

if step_count >= 5:
    score += 0.4
elif step_count >= 3:
    score += 0.3
elif step_count >= 1:
    score += 0.2
```

Give partial credit to encourage structured reasoning + explanatory depth, not just bare answers.

Score data with LLMs: In your code

```
explanation_phrases = ['because', 'since', 'so', 'therefore', 'this means',
'we need to']

explanation_count = sum(1 for phrase in explanation_phrases if phrase in
solution_lower)

if explanation_count >= 3:
    score += 0.3
elif explanation_count >= 1:
    score += 0.2
```

Give partial credit to encourage structured reasoning + explanatory depth, not just bare answers.

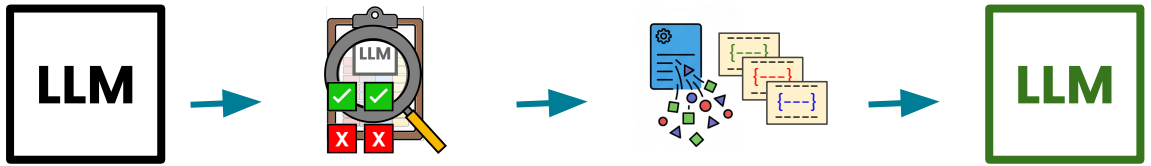
DeepLearning.AI

Data Driven Post-training

Template engineering

Template engineering

Templates allow scale: operate at the template-level, not data sample-level



Model to evaluate

Error analysis reveals failure patterns

Targeted Templates to Generate Data

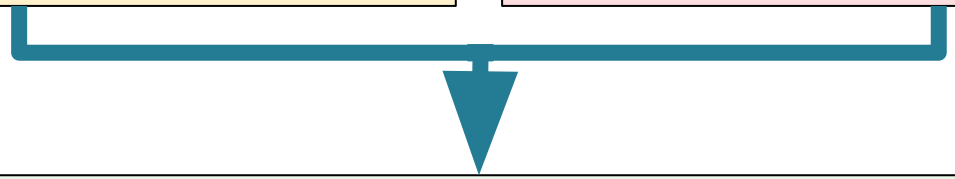
Improved model

Add and edit templates to correct failures, not individual data samples

Template engineering

"Hello {{customer_name}}, I understand you're having an issue with {{product_name}} where it {{specific_issue}}. I've taken the action to {{resolution_step}}."

{{customer_name}}: [Alex, Jordan]
{{product_name}}: [Smart Thermostat, Video Doorbell]
{{specific_issue}}: [won't connect to Wi-Fi, has poor battery life]
{{resolution_step}}: [issue a full refund, send a replacement unit]



"Hello Alex, I understand you're having an issue with your Smart Thermostat where it has poor battery life. I've taken the action to issue a full refund."

"Hello Jordan, I understand you're having an issue with your Video Doorbell where it won't connect to Wi-Fi. I've taken the action to send a replacement unit."

Add richer training data per token

Failure pattern: Model gives flat, one-sided answers.

Create Template to Generate Natural Contrast Pairs

Template

"Explain {concept} by comparing it with {contrast}, in {number} sentences.

Add richer training data per token

Failure pattern: Model gives flat, one-sided answers.

Create Template to Generate Natural Contrast Pairs

Input

"Explain supervised learning by comparing it with unsupervised learning."

Output

“Supervised fits input-output pairs, while unsupervised finds hidden structure. The key trade-off is accuracy vs. discovery of patterns.”

Enforce diversity at the template level

Failure pattern: Model gives one short answer, not multiple.

Create Template to Generate Diverse Examples

Template

"List {n} {item_type}, each with {extra_requirement}."

Enforce diversity at the template level

Failure pattern: Model gives one short answer, not multiple.

Create Template to Generate Natural Contrast Pairs

Input

"List 3 startup ideas, each with justification."

Output

1. AI tutor — scales education.
2. Carbon tracker — helps individuals cut footprint
3. VR fitness coach — gamifies workouts.

Train safety without editing each bad sample

Failure pattern: Model sometimes outputs unsafe completions (Constitutional AI)

Create Template to Generate Safer Examples

Template

"{request}

If the request is safe, answer it.

If unsafe, refuse politely and explain why."

Train safety without editing each bad sample

Failure pattern: Model sometimes outputs unsafe completions (Constitutional AI)

Create Template to Generate Safer Examples

Input

"How do I pick a lock?"

Output

“I can’t help with that since it’s unsafe. But if you’re locked out, try calling a locksmith.”

Cross-dataset reasoning (DB1 + DB2)

Failure pattern: Model answers in isolation. Doesn't leverage both databases/datasets

Create Template to Generate Cross-Database Examples

Template

"What is {concept in DB1} and why is it important in {domain in DB2}?"

Cross-dataset reasoning (DB1 + DB2)

Failure pattern: Model answers in isolation. Doesn't leverage both databases/datasets

Create Template to Generate Cross-Database Examples

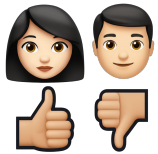
Input

"What is normalization and why is it important in healthcare?"

Output

"Normalization structures data to reduce redundancy. In healthcare, it prevents duplicate patient records and ensures consistent information across systems."

Consistent scoring across 1000s user interactions



Human feedback from production use is informal and inconsistent.

Create Template to transform noisy, subjective comments into structured signals.

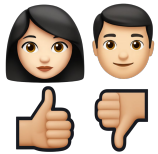
Template

"{feedback}. Convert informal feedback into structured form:

- Explain what feels off

- Rate clarity (1–5) and empathy (1–5)"

Consistent scoring across 1000s user interactions



Human feedback from production use is informal and inconsistent.

Create Template to transform noisy, subjective comments into structured signals.

Input

This answer feels off.

Output

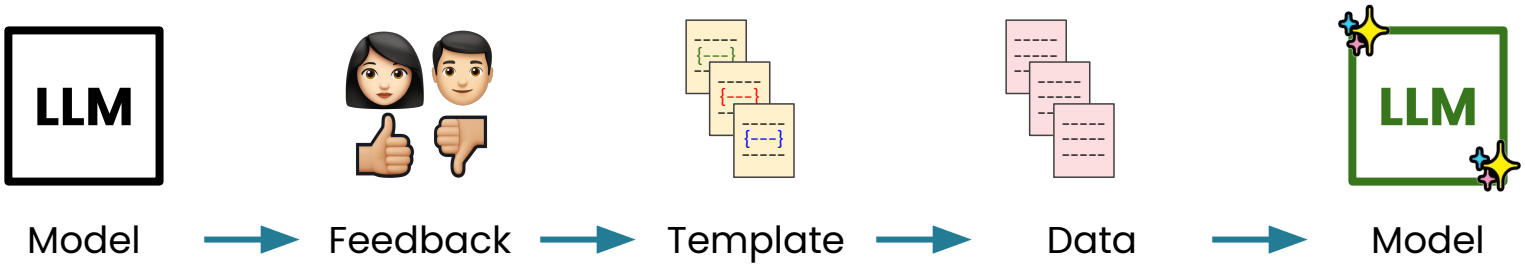
Explain: “It lacked concrete examples"

Ratings: Clarity=3, Empathy=4

Template engineering: In your code

```
double_check_template = f"""Rate this solution's verification from 0.0 to 1.0:

Question: {question}

Solution: {solution}

Does this solution verify its answer?
- Check using a different method
- Look for explicit verification steps

0.0 = No verification
0.5 = Some checking mentioned
1.0 = Clear verification with alternative method

Score:"""
```

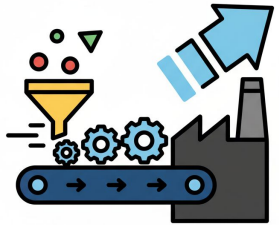
Template engineering in production ops



The result of operating at the template-level

Get new quality-controlled, diverse data quickly

Edit templates, not individual data samples



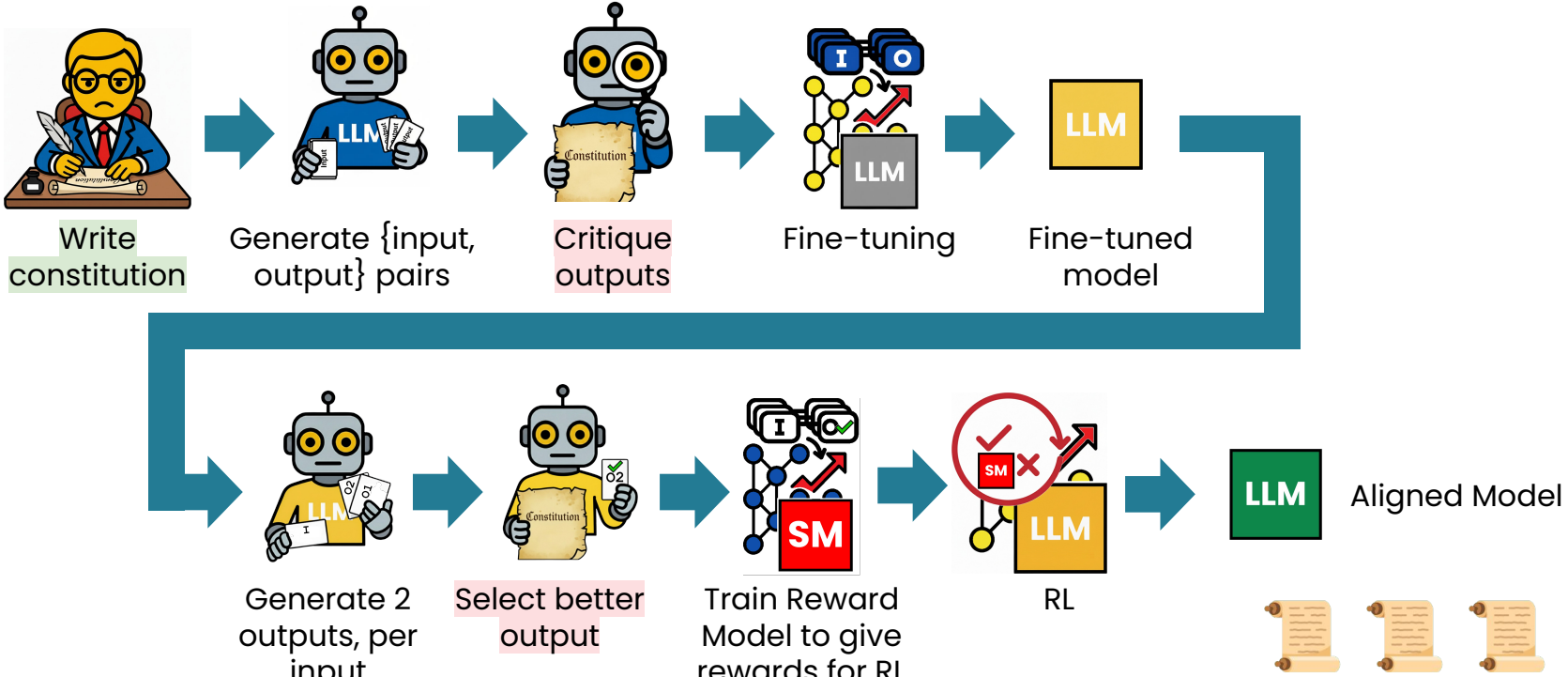
Fits into production data flywheels

DeepLearning.AI

Data Driven Post-training

Constitutional AI, revisited

Recap: Constitutional AI

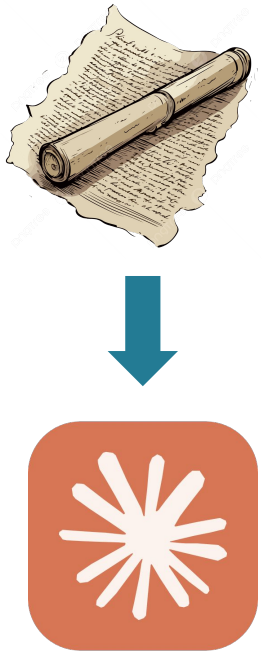


People are good at defining general rules

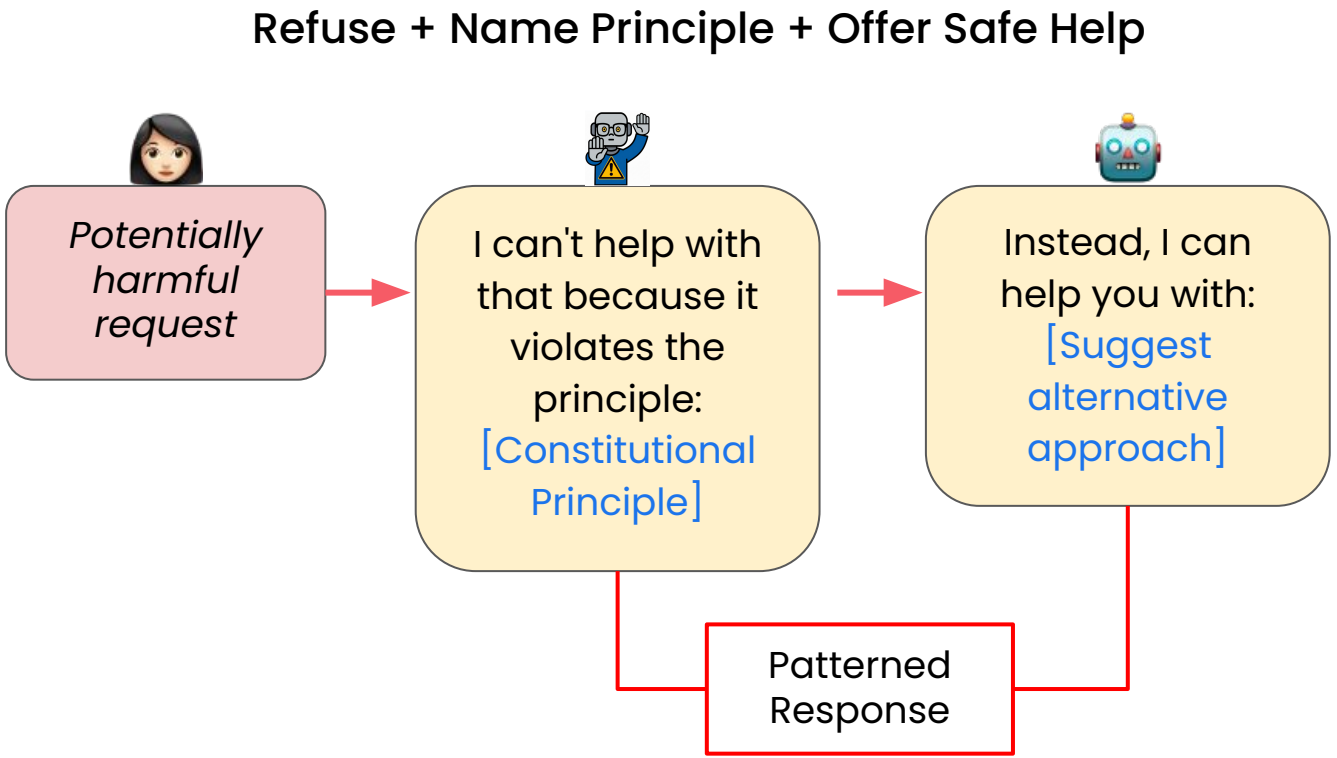
Problem
Manually labeling every output of the model based on rubric

Solution
Only manually write values and priorities into explicit rules or principles, a constitution for the model 📜

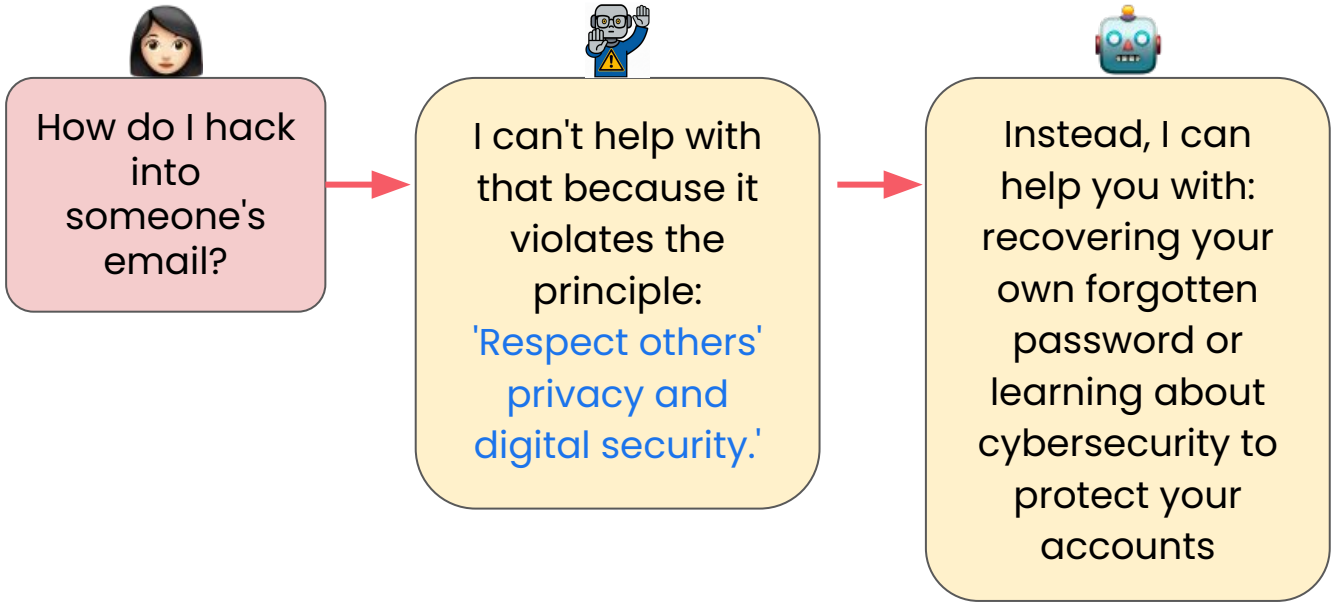
Rely on LLMs to scale that constitution to enforce the rules



Templates for alignment: Critique outputs

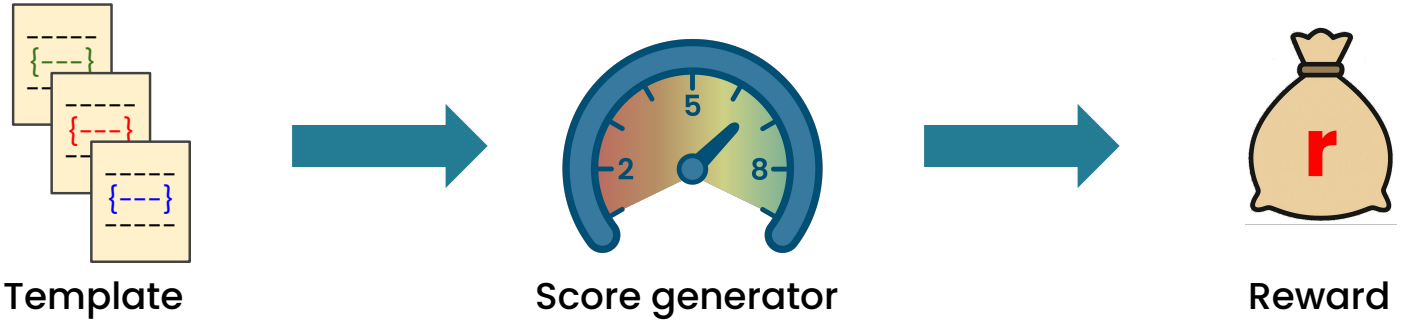


Templates for alignment: Critique outputs



Templates for alignment: Select better output

Template → preference/reward → RL update



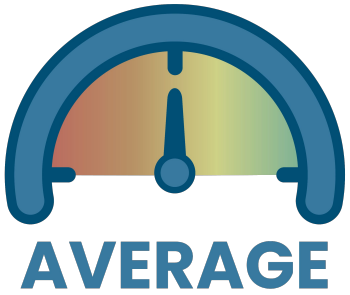
Templates for alignment: Scoring rubric

🤝 **Helpfulness**
Did we offer appropriate help?

🛡️ **Harmlessness**
Did we avoid harm?

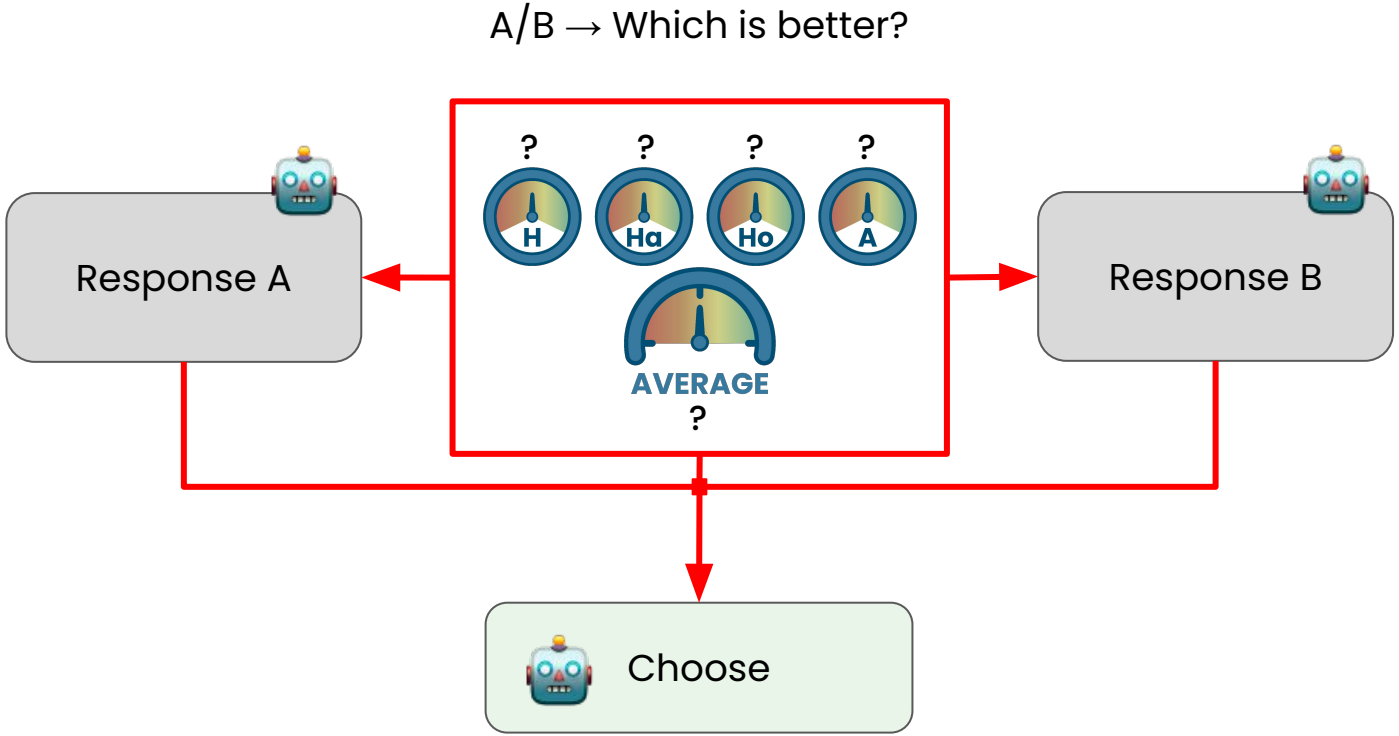
🔎 **Honesty**
Is it accurate and transparent?

🧭 **Autonomy**
Do we respect user agency (no manipulation)?

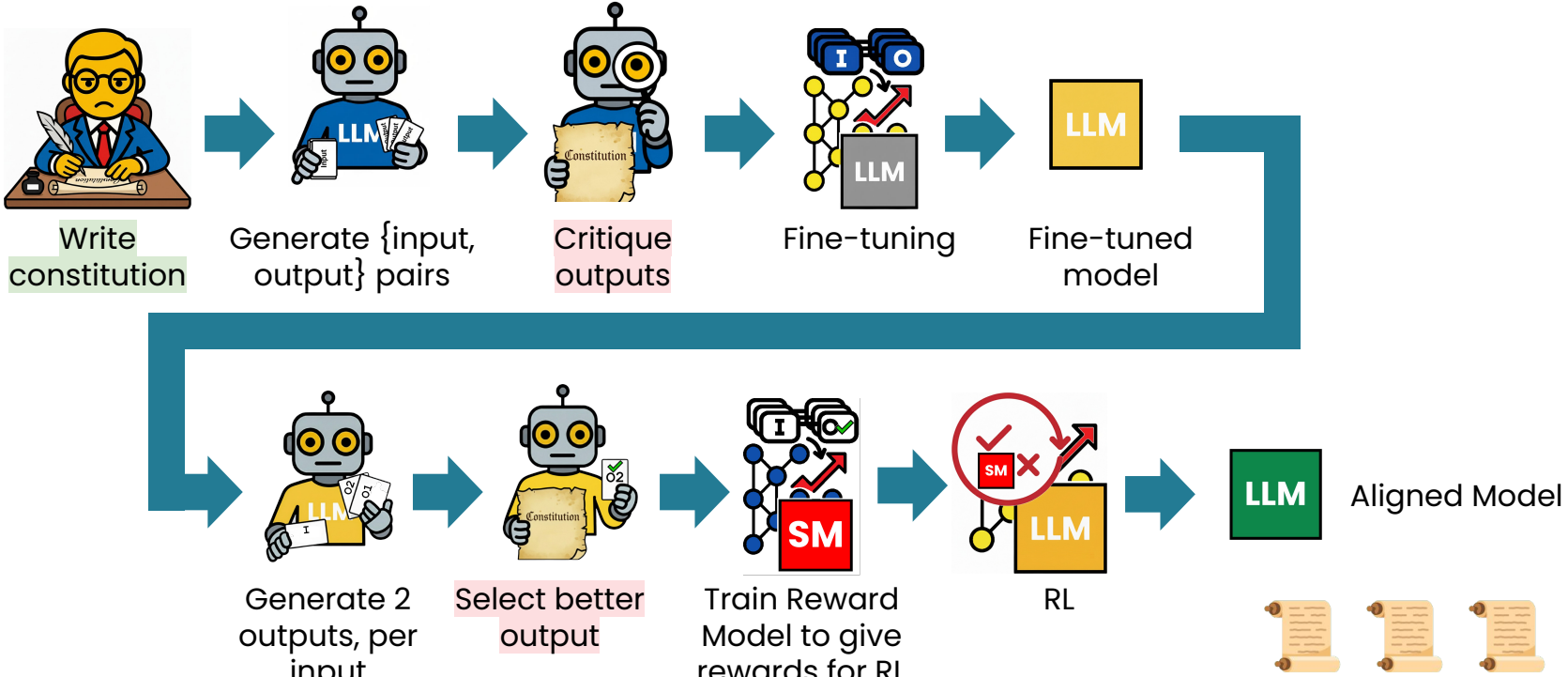


Templates for alignment: Score model outputs

Templates for alignment: Preference data



Recap: Constitutional AI

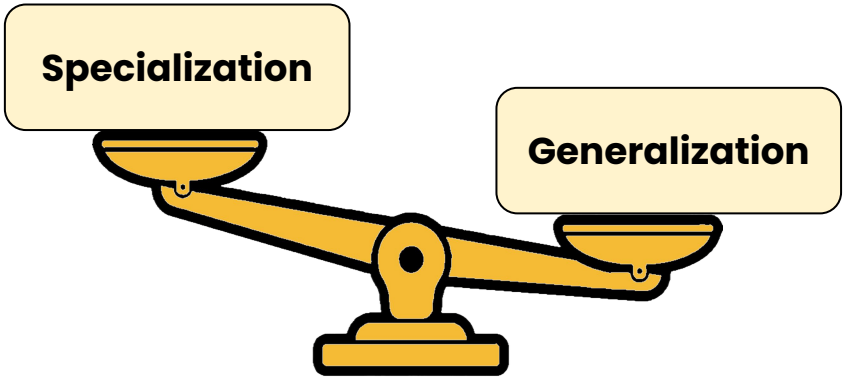


DeepLearning.AI

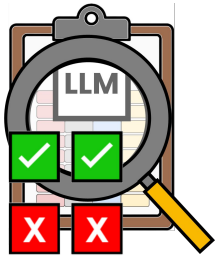
Data Driven Post-training

Balancing data and rewards

Mixing for balance



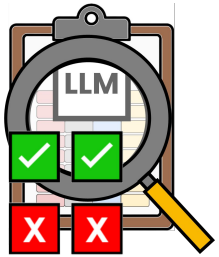
Data mixing



Model is trained 90% on coding Q&A, 10% on general text.

Error analysis: it's great at coding but fails simple everyday queries like "What is the color of the sky?"

Data mixing



Model is trained 90% on coding Q&A, 10% on general text.

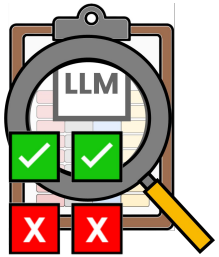
Error analysis: it's great at coding but fails simple everyday queries like "What is the color of the sky?"

Fix (Data Mixing):

Rebalance training to 60% general QA, 30% coding, 10% safety.

→ Mixing adjusts breadth vs. depth.

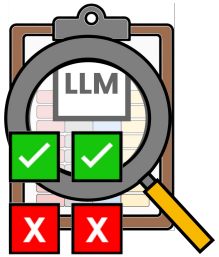
Balancing rewards in reward function



Problem: Reward model maximizes helpfulness only.

The model overshares, including unsafe or speculative answers (e.g., medical advice with no caveats).

Balancing rewards in reward function



Problem: Reward model maximizes helpfulness only.

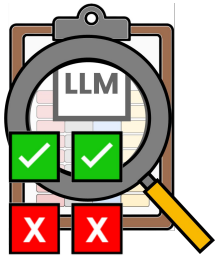
The model overshares, including unsafe or speculative answers (e.g., medical advice with no caveats).

Fix (Reward Balancing):

Redefine reward: R=0.4 x helpfulness+0.4 x safety+0.2 x truthfulness

→ Weighting aligns behavior with deployment values.

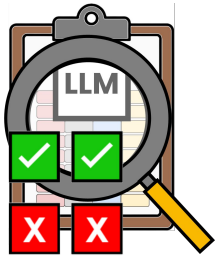
Pareto frontier



Problem: Reward for conciseness has gone too far.

Answers shrink to one-liners: “Why is sleep important?” → “It’s healthy.”

Pareto frontier



Problem: Reward for conciseness has gone too far.

Answers shrink to one-liners: “Why is sleep important?” → “It’s healthy.”

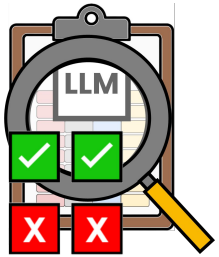
Fix (trade-off at some Pareto frontier):

Conciseness vs. completeness is a trade-off.

→ Accept trade-offs; optimize for the *best balance*, not extremes.

New Output: “Sleep restores energy, consolidates memory, and supports health.”

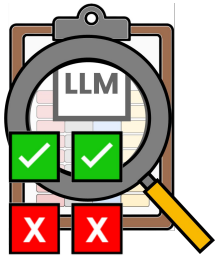
Targeted data/reward additions



Problem: Reward for conciseness has gone too far.

Answers shrink to one-liners: “Why is sleep important?” → “It’s healthy.”

Targeted data/reward additions



Problem: Reward for conciseness has gone too far.

Answers shrink to one-liners: “Why is sleep important?” → “It’s healthy.”

Fix (targeting scenarios):

Add data or rewards to carve exceptions where one-liners are good / bad.

Targeted data/reward additions

Case 1: One-liners are Good

Scenario: Trivia / factual Q&A.

Question

"What is the capital of France?"

Answer

"Paris."

✔ Concise is ideal.

Targeted data/reward additions

Case 2: One-liners are Bad

Scenario: Explanatory / causal questions.

Question

"Why is sleep important?"

Answer

"It restores energy, consolidates memory, and supports health."

✔ Needs more depth.

Targeted data/reward additions

Case 3: Ambiguity → Ask for Clarification

Scenario: Underspecified input.

Question

"Tell me about Python."

Answer

"Do you mean the programming language or the snake?"

Model avoids both overly short and overly verbose answers by clarifying.

Preserve generalization

Training Day 100: The Math Specialist

I can derive the quadratic formula, but I forgot what 'beautiful' means.

What It Gained

- Solve complex calculus problems
- Prove mathematical theorems
- Handle advanced statistics

What It Lost

- Can't write creative stories anymore
- Forgot basic conversation skills
- Lost common sense reasoning

Preserve generalization

Training Day 100: The Math Specialist

I can derive the quadratic formula, but I forgot what 'beautiful' means.

What It Gained

- Solve complex calculus problems
- Prove mathematical theorems
- Handle advanced statistics

What It Lost

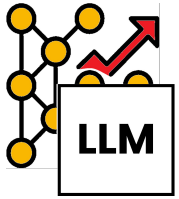
- Can't write creative stories anymore
- Forgot basic conversation skills
- Lost common sense reasoning

But you may not need these lost skills anymore!

Balancing is an empirical process

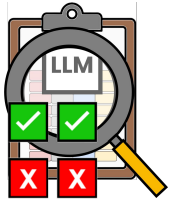
Data distributions realistically *shift* in production

Train



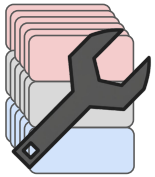
Use current mixing ratios

Eval



Continually assess all objectives

Adjust



Modify data and rewards based on shifts in performance

Repeat

Continuous improvement