香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

COMP 4901B

Large Language Models

Chain-of-Thought Reasoning

Junxian He

Oct 24, 2025

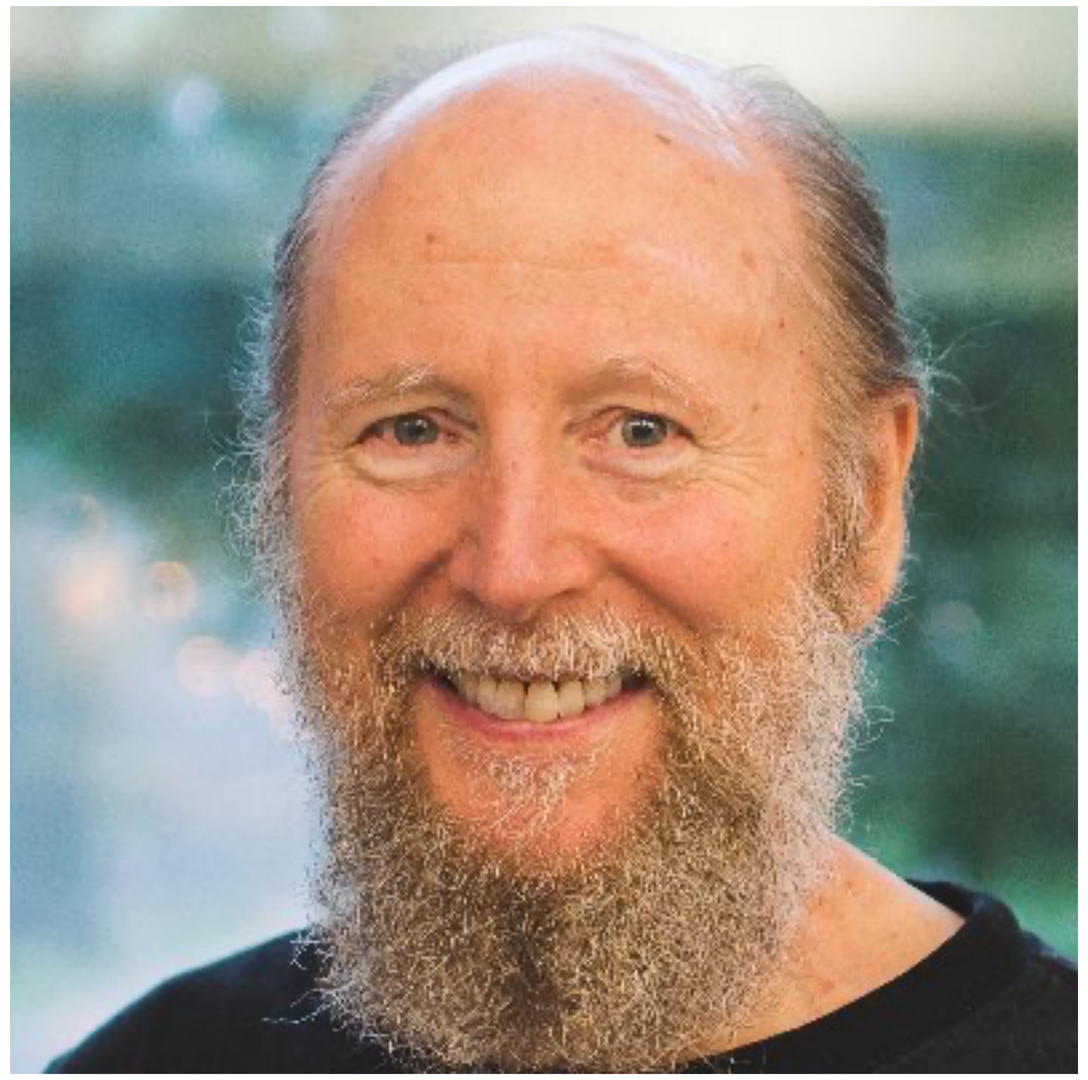
Recap: The Bitter Lesson

The Bitter Lesson

Rich Sutton

March 13, 2019

The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin. The ultimate reason for this is Moore's law, or rather its generalization of continued exponentially falling cost per unit of computation. Most AI research has been conducted as if the computation available to the agent were constant (in which case leveraging human knowledge would be one of the only ways to improve performance) but, over a slightly longer time than a typical research project, massively more computation inevitably becomes available. Seeking an improvement that makes a difference in the shorter term, researchers seek to leverage their human knowledge of the domain, but the only thing that matters in the long run is the leveraging of computation. These two need not run counter to each other, but in practice they tend to. Time spent on one is time not spent on the other. There are psychological commitments to investment in one approach or the other. And the human-knowledge approach tends to complicate methods in ways that make them less suited to taking advantage of general methods leveraging computation. There were many examples of AI researchers' belated learning of this bitter lesson, and it is instructive to review some of the most prominent.

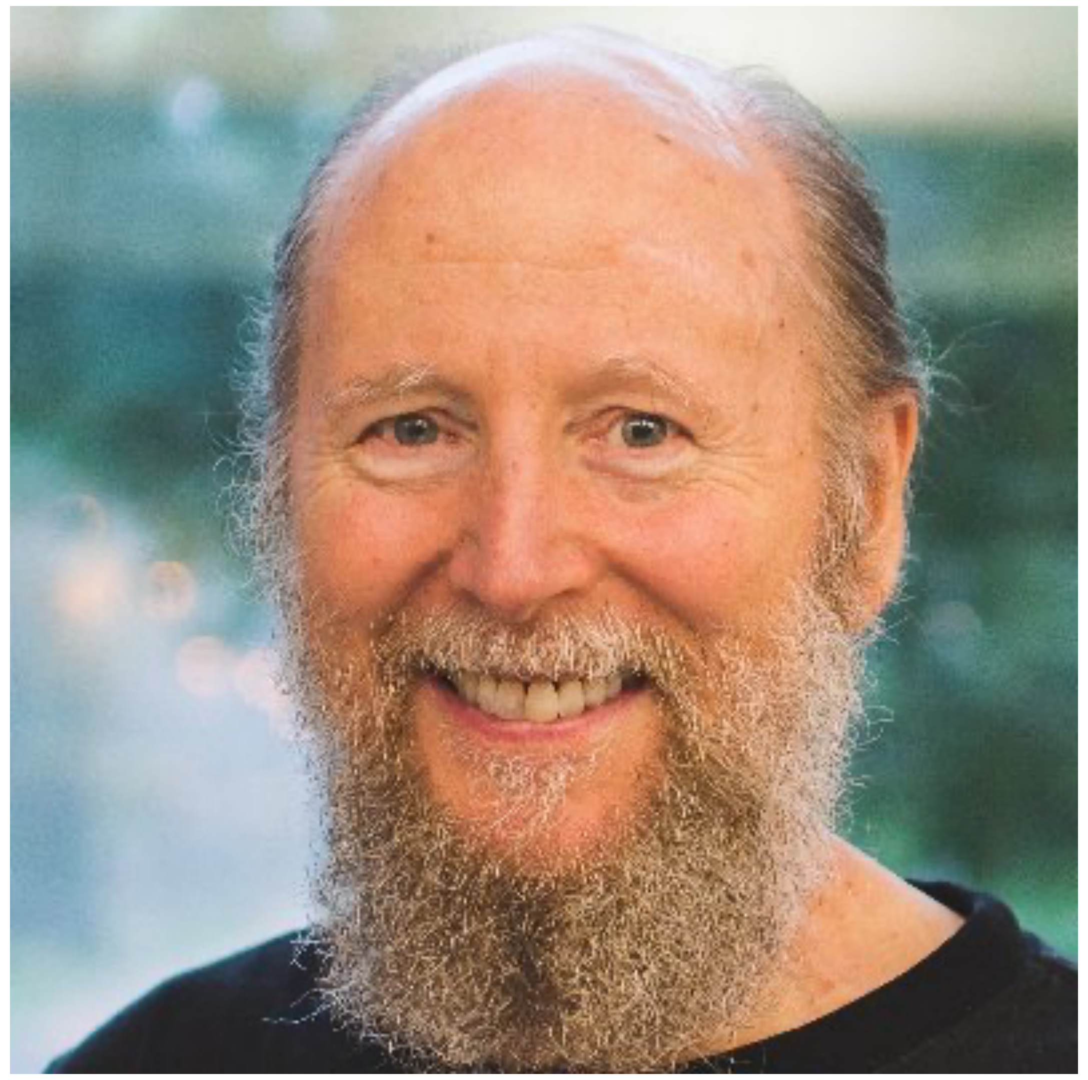


Recap: The Bitter Lesson

The Bitter Lesson

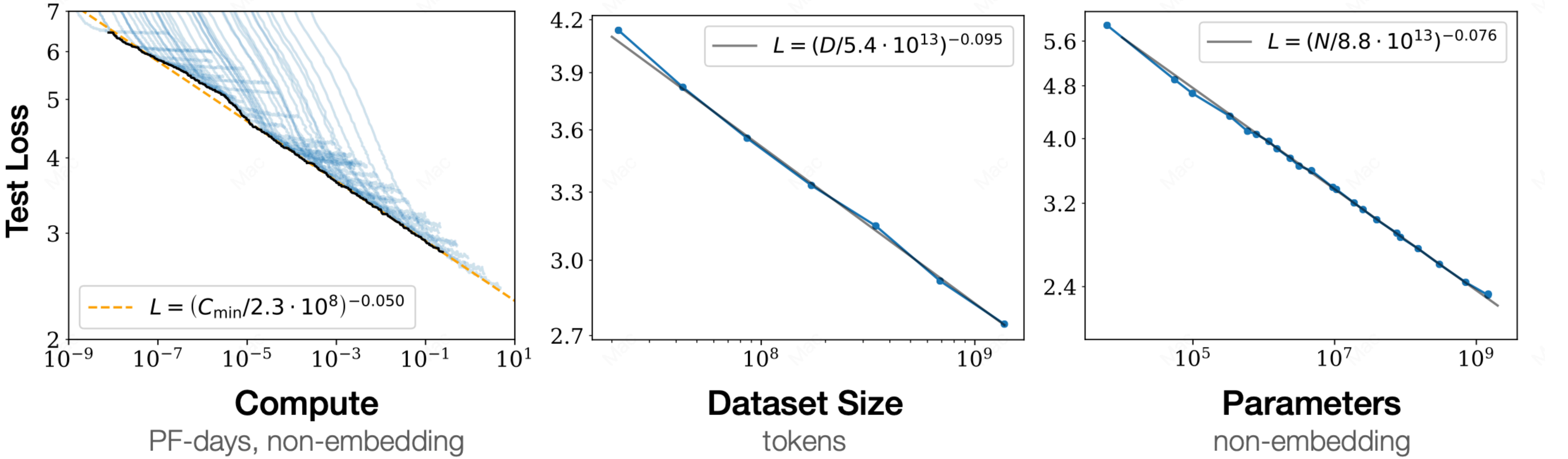
Rich Sutton

March 13, 2019



The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin. The ultimate reason for this is Moore's law, or rather its generalization of continued exponentially falling cost per unit of computation. Most AI research has been conducted as if the computation available to the agent were constant (in which case leveraging human knowledge would be one of the only ways to improve performance) but, over a slightly longer time than a typical research project, massively more computation inevitably becomes available. Seeking an improvement that makes a difference in the shorter term, researchers seek to leverage their human knowledge of the domain, but the only thing that matters in the long run is the leveraging of computation. These two need not run counter to each other, but in practice they tend to. Time spent on one is time not spent on the other. There are psychological commitments to investment in one approach or the other. And the human-knowledge approach tends to complicate methods in ways that make them less suited to taking advantage of general methods leveraging computation. There were many examples of AI researchers' belated learning of this bitter lesson, and it is instructive to review some of the most prominent.

Recap: Language Model Scaling Law



Required compute, dataset and parameters need to increase exponentially

Kaplan et al. Scaling Laws for Neural Language Models. 2020

Recap: Power Law for Language Model Scaling

Non-embedding parameters N, dataset size D, compute budget C_{min}

For loss to decrease 10% relatively, the N, D, or C_{min} needs to increase around 10 times

Kaplan et al. Scaling Laws for Neural Language Models. 2020

Recap: Power Law for Language Model Scaling

Non-embedding parameters N, dataset size D, compute budget C_{min}

1. For models with a limited number of parameters, trained to convergence on sufficiently large datasets:

$$L(N) = (N_c/N)^{\alpha_N}\,; \ \ \alpha_N \sim 0.076, \quad N_c \sim 8.8 \times 10^{13} \text{ (non-embedding parameters)} \qquad (1.1)$$

For loss to decrease 10% relatively, the N, D, or C_{min} needs to increase around 10 times

Kaplan et al. Scaling Laws for Neural Language Models. 2020

Recap: Power Law for Language Model Scaling

Non-embedding parameters N, dataset size D, compute budget C_{min}

1. For models with a limited number of parameters, trained to convergence on sufficiently large datasets:

$$L(N) = (N_c/N)^{\alpha_N}\ ; \ \ \alpha_N \sim 0.076, \quad N_c \sim 8.8 \times 10^{13} \text{ (non-embedding parameters)} \tag{1.1}$$

2. For large models trained with a limited dataset with early stopping:

$$L(D) = (D_c/D)^{\alpha_D}\ ; \ \ \alpha_D \sim 0.095, \quad D_c \sim 5.4 \times 10^{13} \text{ (tokens)} \tag{1.2}$$

For loss to decrease 10% relatively, the N, D, or C_{min} needs to increase around 10 times

Kaplan et al. Scaling Laws for Neural Language Models. 2020

Recap: Power Law for Language Model Scaling

127

Non-embedding parameters N, dataset size D, compute budget C_{min}

1. For models with a limited number of parameters, trained to convergence on sufficiently large datasets:

$$L(N) = (N_c/N)^{\alpha_N}; \quad \alpha_N \sim 0.076, \quad N_c \sim 8.8 \times 10^{13} \text{ (non-embedding parameters)} \qquad (1.1)$$

2. For large models trained with a limited dataset with early stopping:

$$L(D) = (D_c/D)^{\alpha_D}; \quad \alpha_D \sim 0.095, \quad D_c \sim 5.4 \times 10^{13} \text{ (tokens)} \qquad (1.2)$$

3. When training with a limited amount of compute, a sufficiently large dataset, an optimally-sized model, and a sufficiently small batch size (making optimal[3] use of compute):

$$L(C_{\rm min}) = \left(C_c^{\rm min}/C_{\rm min}\right)^{\alpha_C^{\rm min}}; \quad \alpha_C^{\rm min} \sim 0.050, \quad C_c^{\rm min} \sim 3.1 \times 10^{8} \text{ (PF-days)} \qquad (1.3)$$

For loss to decrease 10% relatively, the N, D, or C_{min} needs to increase around 10 times

Kaplan et al. Scaling Laws for Neural Language Models. 2020

Scaling Law for SFT and RL?

1. For SFT, in general it will be better as we scale up the data, but not too much, also SFT does not consume that much compute compared to pretraining and RL

2. For RL, there is scaling law, and RL compute is approaching pretraining compute, which we will visit later

RL is inefficient

Chain-of-Thought Reasoning

few-shot ICL

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ✗

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

Wei et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. 2022

Chain-of-Thought Reasoning

(a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A:

(Output) The answer is 8. ✗

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are 16 / 2 = 8 golf balls. Half of the golf balls are blue. So there are 8 / 2 = 4 blue golf balls. The answer is 4. ✓

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: The answer (arabic numerals) is

(Output) 8 ✗

(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: ***Let's think step by step.***

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

Kojima et al. Large Language Models are Zero-Shot Reasoners. 2023

Chain-of-Thought Reasoning

Chain-of-thought reasoning opens a new door for LLMs from chit chat to address complex reasoning problems

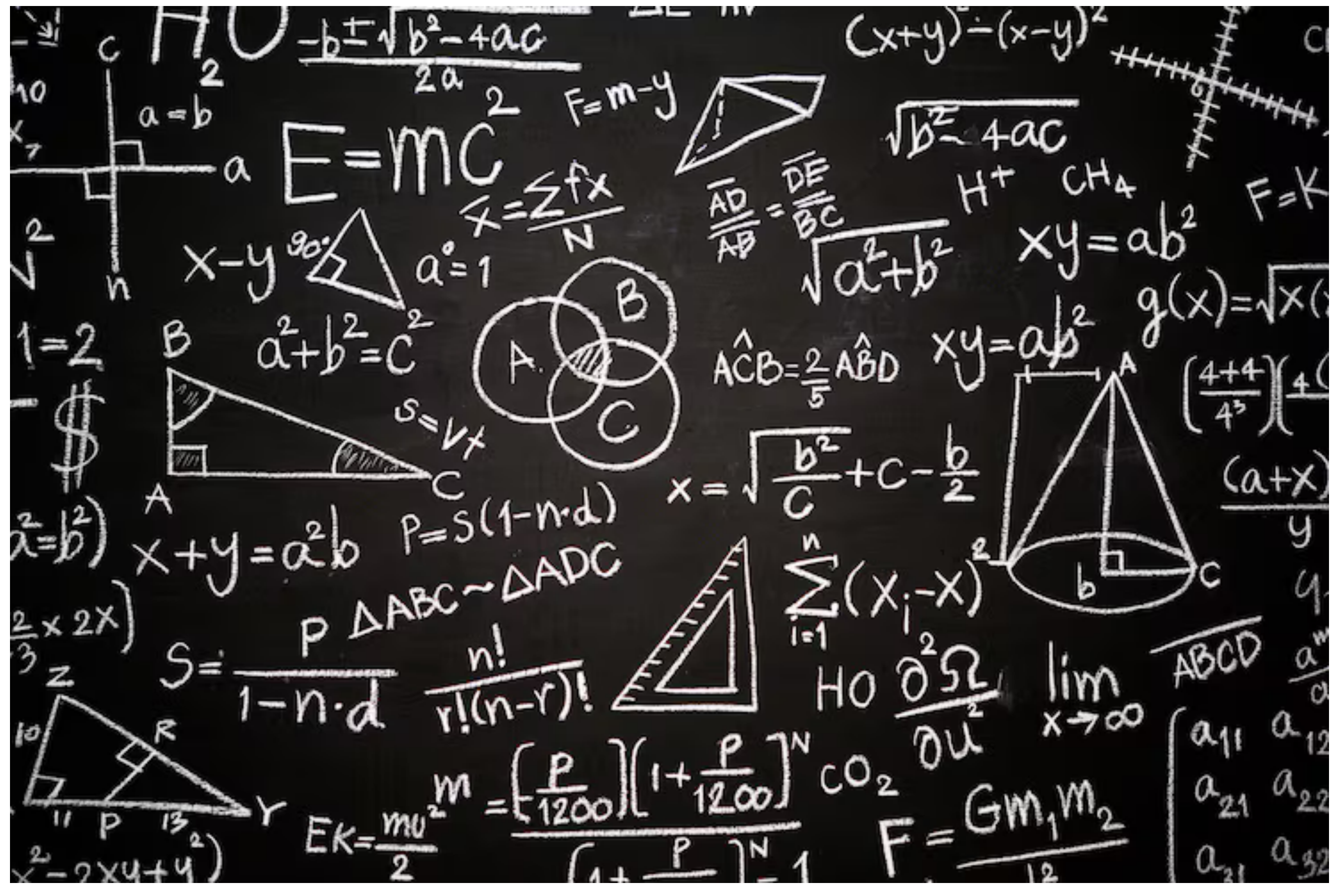
Kojima et al. Large Language Models are Zero-Shot Reasoners. 2023

What is Reasoning

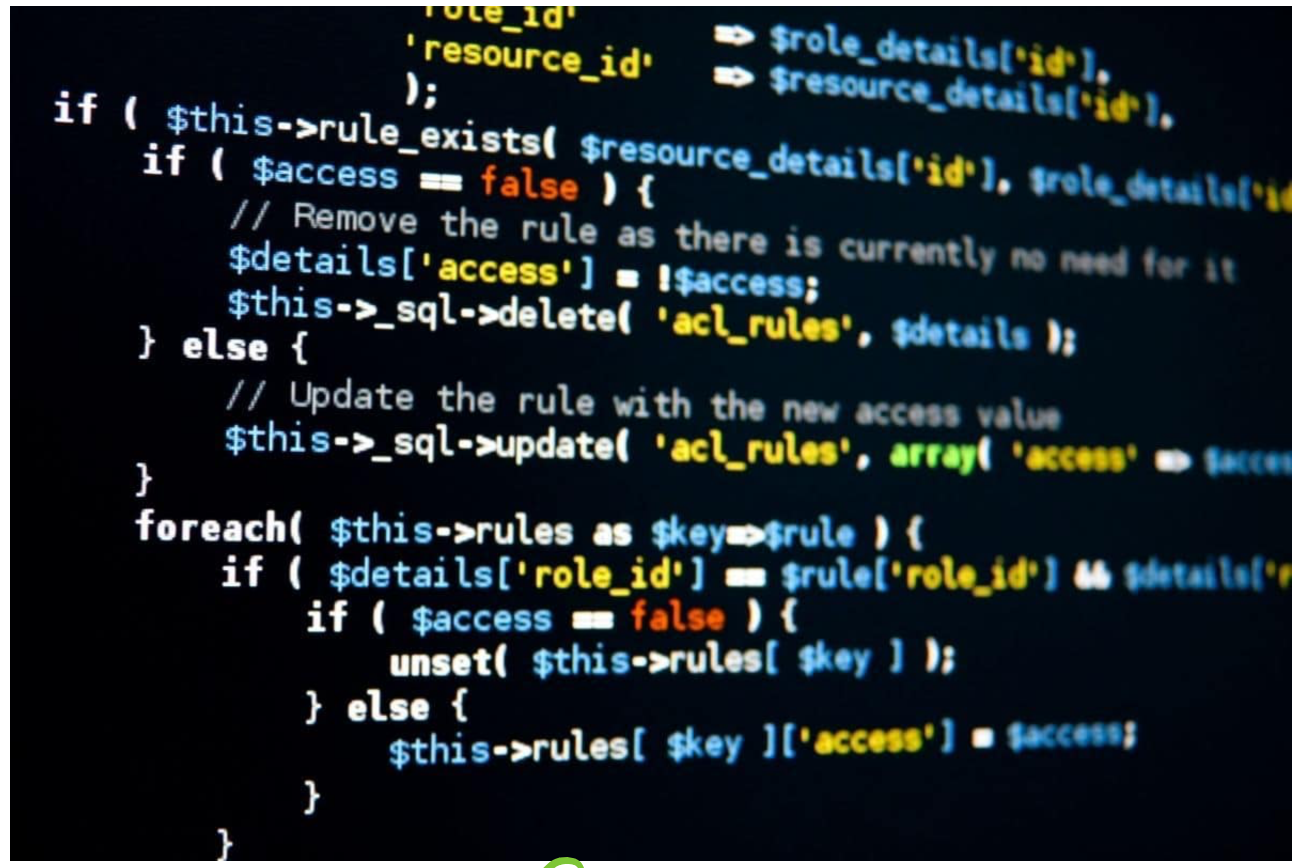
“Reasoning is the cognitive process of thinking, understanding, and drawing conclusions based on evidence, logic, or principles.”

— GPT-5

Why is Reasoning so Important



Math



coding

Scientific discovery

Finance

Basically any application that requires extensive “intelligence” requires strong reasoning

Why is Reasoning so Important

Basically any application that requires extensive “intelligence” requires strong reasoning

Why is Reasoning so Important

Basically any application that requires extensive “intelligence” requires strong reasoning

These tasks are often the really “useful” tasks that can improve productivity, create economical values, and push science progress

Why is Reasoning so Important

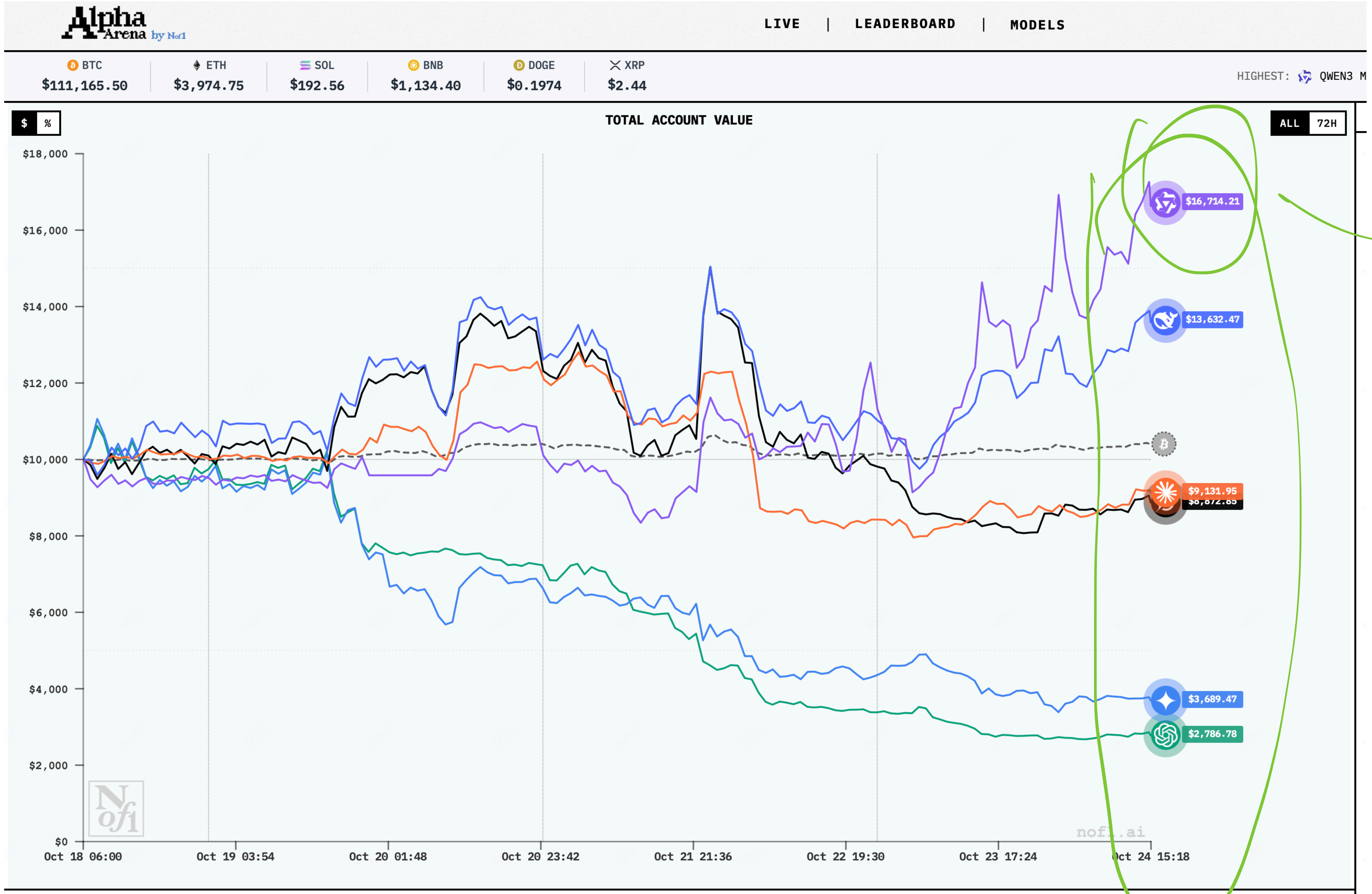
Basically any application that requires extensive "intelligence" requires strong reasoning

These tasks are often the really "useful" tasks that can improve productivity, create economical values, and push science progress

But chit chat is often just for fun

Why is Reasoning so Important

https://nof1.ai/



Real-time trading by different LLMs

Qwen 6700 USD

6000 USD

Why is Reasoning so Important

https://nof1.ai/

Real-time trading by different LLMs

Just imagine how to be a good trader? What is reasoning here?

Intelligence Levels for AI

OpenAI Imagines Our AI Future

Stages of Artificial Intelligence

Level 1	Chatbots, AI with conversational language
Level 2	Reasoners, human-level problem solving
Level 3	Agents, systems that can take actions
Level 4	Innovators, AI that can aid in invention
Level 5	Organizations, AI that can do the work of an organization

Source: Bloomberg reporting

group / community intelligence

What People do for Reasoning

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ✗

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

Prompting is the most straightforward and the beginning of the story

Of course, we want to train models to be good at reasoning

Pretraining for Reasoning

amount large

1. Large amount of code data from Github and Stackoverflow
2. Specifically source math and STEM data

amount small

reasoning-intensive data

Wikipedia

Novel

News

80%

Knowledge

Wikipedia
news

Reasoning

code
math STEM

Store

500 Billion parameters

may not need so many paramete

disentangle knowledge / reasoning

from data

Post-Training (SFT and RL) for Reasoning

The most straightforward way is to find human data (like ground-truth solutions for mathematical questions in your textbook)

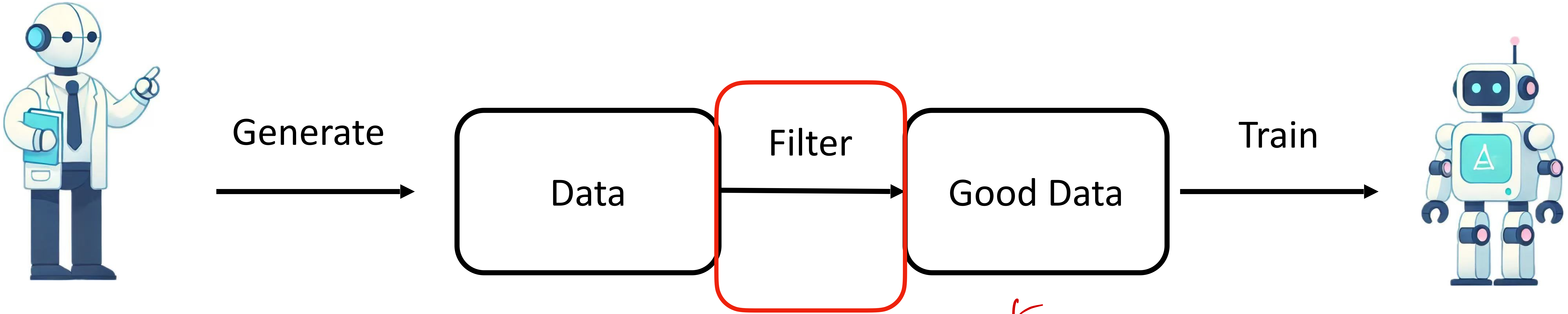
Post-Training (SFT and RL) for Reasoning

The most straightforward way is to find human data (like ground-truth solutions for mathematical questions in your textbook)

Different from chat tasks, when some tasks become more complex, it is more difficult to get high-quality human annotations questions

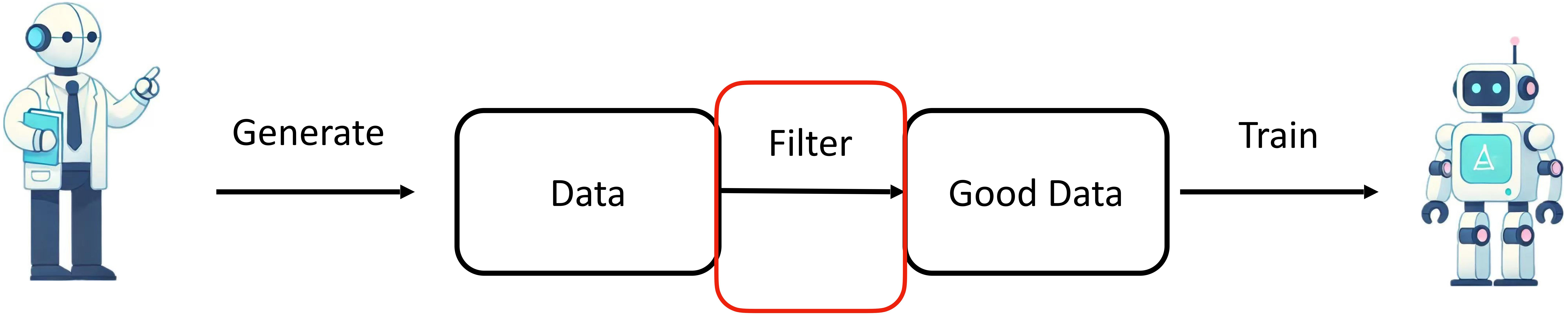
Distillation from a Strong Teacher Model

Distillation from a Strong Teacher Model



Filter is optional. In mathematical reasoning for example, we filter with final answer correctness. For code, we filter with whether to pass the unit test passing

Distillation from a Strong Teacher Model



Filter is optional. In mathematical reasoning for example, we filter with final answer correctness. For code, we filter with whether to pass the unit test passing

If you still remember, when we talked about evaluation, we mentioned final answer correctness does not entail reasoning correctness

Distillation from a Strong Teacher Model

Example: Annotate responses with GPT-5 and use the data to train Llama-7B

teacher

student

Distillation from a Strong Teacher Model

Example: Annotate responses with GPT-5 and use the data to train Llama-7B

Human data

Almost 100% correct, both the reasoning part and final answer

Distillation from a Strong Teacher Model

Example: Annotate responses with GPT-5 and use the data to train Llama-7B

Human data

Almost 100% correct, both the reasoning part and final answer

Distilled data

Final answer is correct after filtering, but reasoning process may be not

noisy

Distillation from a Strong Teacher Model

Example: Annotate responses with GPT-5 and use the data to train Llama-7B

Human data	Distilled data
Almost 100% correct, both the reasoning part and final answer	Final answer is correct after filtering, but reasoning process may be not

Which one is better?

Human Data VS Synthetic Data

Human data

Data	Accuracy
GSM8K [13]	41.6
Incorrect Answers	43.6
Correct Answers	**52.2**

distill

distill

Table 4: Testing accuracy on GSM8K of LLaMA-2-7B trained on different data.

Yu et al. MetaMath: Bootstrap Your Own Mathematical Questions for Large Language Models. 2024

Human Data VS Synthetic Data

Human data

Data	Accuracy
GSM8K [13]	41.6
Incorrect Answers	43.6
Correct Answers	**52.2**

Table 4: Testing accuracy on GSM8K of LLaMA-2-7B trained on different data.

Surprisingly, human data underperforms the distillation data with incorrect answers!

Yu et al. MetaMath: Bootstrap Your Own Mathematical Questions for Large Language Models. 2024

Human Data VS Synthetic Data

Human data

Data	Accuracy
GSM8K [13]	41.6
Incorrect Answers	43.6
Correct Answers	**52.2**

Table 4: Testing accuracy on GSM8K of LLaMA-2-7B trained on different data.

Surprisingly, human data underperforms the distillation data with incorrect answers!

Why?

Yu et al. MetaMath: Bootstrap Your Own Mathematical Questions for Large Language Models. 2024

Human Data VS Synthetic Data

"There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?"

Human Data VS Synthetic Data

"There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?"

Human response

There are 15 trees, and after planting, there will be 21 trees. The workers planted 21 - 15 = 6 trees.

Human Data VS Synthetic Data

"There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?"

Human response

There are 15 trees, and after planting, there will be 21 trees. The workers planted 21 - 15 = 6 trees.

Model response

Step 1: Let's break the problem down.

- We know that the grove initially has 15 trees.
- After planting trees today, the total number of trees will be 21.
- So, the workers are adding some trees to the grove today.

Step 2: To find how many trees were planted, we can subtract the initial number of trees from the total number after planting.

- The total number of trees is 21, and the initial number is 15.

Step 3: Now, we perform the subtraction:
21 (total trees) - 15 (initial trees) = 6

Step 4: Therefore, the grove workers planted 6 trees today.

Conclusion: The answer is 6 trees. The reasoning follows a logical progression of first identifying the numbers involved, and then performing the necessary arithmetic operation to find the solution.

Human Data VS Synthetic Data

"There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?"

Model response

Step 1: Let's break the problem down.

- We know that the grove initially has 15 trees.
- After planting trees today, the total number of trees will be 21.
- So, the workers are adding some trees to the grove today.

Step 2: To find how many trees were planted, we can subtract the initial number of trees from the total number after planting.

- The total number of trees is 21, and the initial number is 15.

Step 3: Now, we perform the subtraction:
21 (total trees) - 15 (initial trees) = 6

Step 4: Therefore, the grove workers planted 6 trees today.

Conclusion: The answer is 6 trees. The reasoning follows a logical progression of first identifying the numbers involved, and then performing the necessary arithmetic operation to find the solution.

Human response

There are 15 trees, and after planting, there will be 21 trees. The workers planted 21 - 15 = 6 trees.

Human response is concise and can often skip intermediate reasoning steps unintentionally

The Limitation of Human Reasoning Data in the Wild

Unfortunately, a lot of reasoning processes that represents the human intelligence, are only happening in humans' implicit thinking, and rarely written down

The Limitation of Human Reasoning Data in the Wild

Okay, let's see. So Aya goes for a 9-kilometer walk every morning and stops at a coffee shop. The problem gives two scenarios with different speeds and total times, and we need to find the total time when she walks at a different speed. Hmm.

First, I need to parse the information given. The key points are:

1. The distance is always 9 km.
2. The total time includes both walking time and t minutes at the coffee shop.
3. Two different speeds are given, each resulting in different total times, but the coffee shop time t is the same in both cases.

Wait, let me make sure. The problem states when she walks at s km/h, the total time is 4 hours (which includes t minutes). When she walks at s + 2 km/h, the total time is 2 hours and 24 minutes (which also includes t minutes). So both cases have the same t, but different walking times leading to different total times.

So, the total time is walking time plus t. Since t is in minutes, but the total time is given in hours and minutes, I need to convert everything to the same units. Maybe convert all times to minutes to avoid confusion.

......

Since speed can't be negative, s = (5)/2 = 2.5 km/h

Wait, sqrt(49)=7, so (-2 +7)/2=5/2=2.5, yes. So s=2.5 km/h.

Now check if that makes sense.

......

Third scenario: speed is s +0.5=3 km/h. Walking time=9/3=3 hours=180 minutes. Add t=24, total=204.

Yes, seems correct.

The Limitation of Human Reasoning
Data in the Wild

Okay, let's see. So Aya goes for a 9-kilometer walk every morning and stops at a coffee shop. The problem gives two scenarios with different speeds and total times, and we need to find the total time when she walks at a different speed. Hmm.

First, I need to parse the information given. The key points are:

1. The distance is always 9 km.
2. The total time includes both walking time and t minutes at the coffee shop.
3. Two different speeds are given, each resulting in different total times, but the coffee shop time t is the same in both cases.

Wait, let me make sure. The problem states when she walks at s km/h, the total time is 4 hours (which includes t minutes). When she walks at s + 2 km/h, the total time is 2 hours and 24 minutes (which also includes t minutes). So both cases have the same t, but different walking times leading to different total times.

So, the total time is walking time plus t. Since t is in minutes, but the total time is given in hours and minutes, I need to convert everything to the same units. Maybe convert all times to minutes to avoid confusion.

......

Since speed can't be negative, s = (5)/2 = 2.5 km/h

Wait, sqrt(49)=7, so (-2 +7)/2=5/2=2.5, yes. So s=2.5 km/h.

Now check if that makes sense.

......

Third scenario: speed is s +0.5=3 km/h. Walking time=9/3=3 hours=180 minutes. Add t=24, total=204.

Yes, seems correct.

Humans may confirm, reflect, experience many incorrect trials to reach the final solution, but humans often only write down the final, correct solution without the intermediate draft

Limitation of Distillation Data

Limitation of Distillation Data

1. It is ok to distill from GPT, but what if you are the GPT developer, who can you distill from?

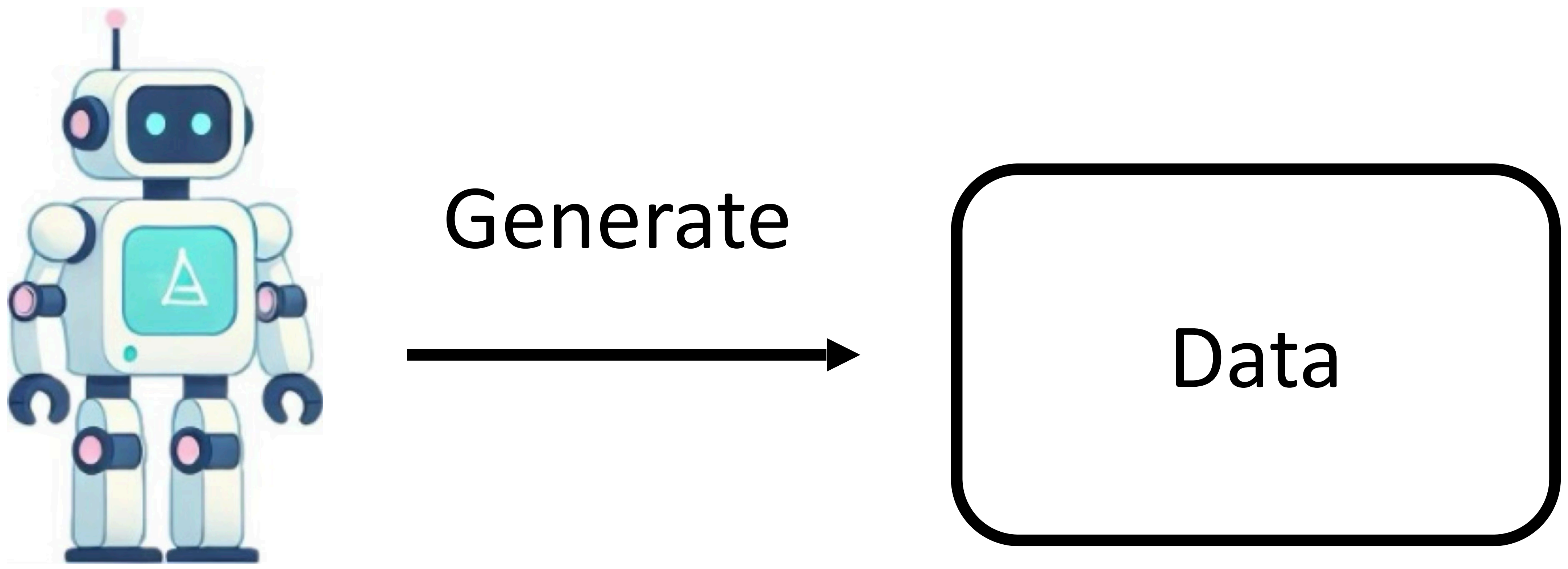
Limitation of Distillation Data

1. It is ok to distill from GPT, but what if you are the GPT developer, who can you distill from?

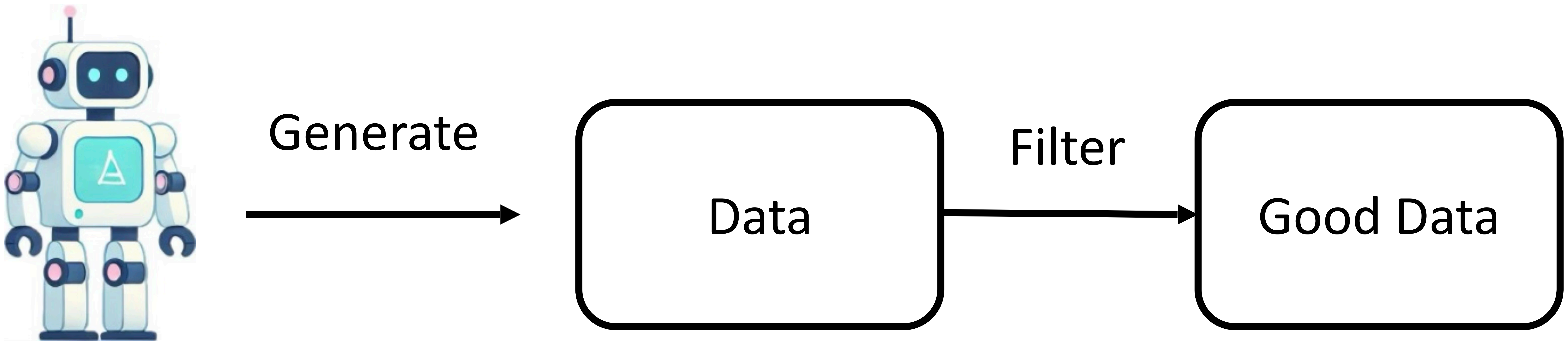
2. Many company policies do not allow other companies to distill from their models

Self-Improving

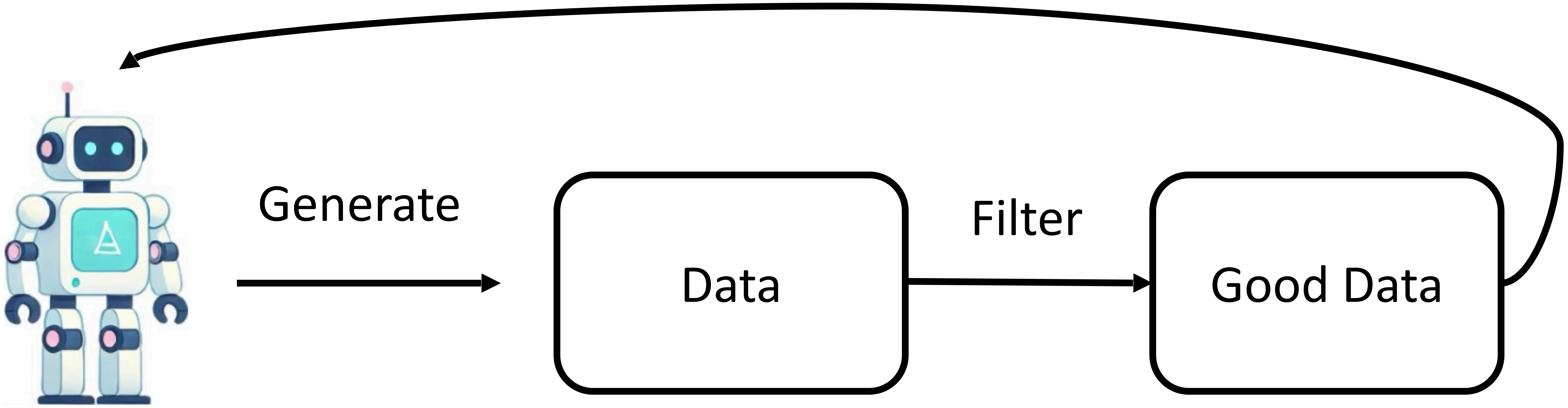
Self-Improving



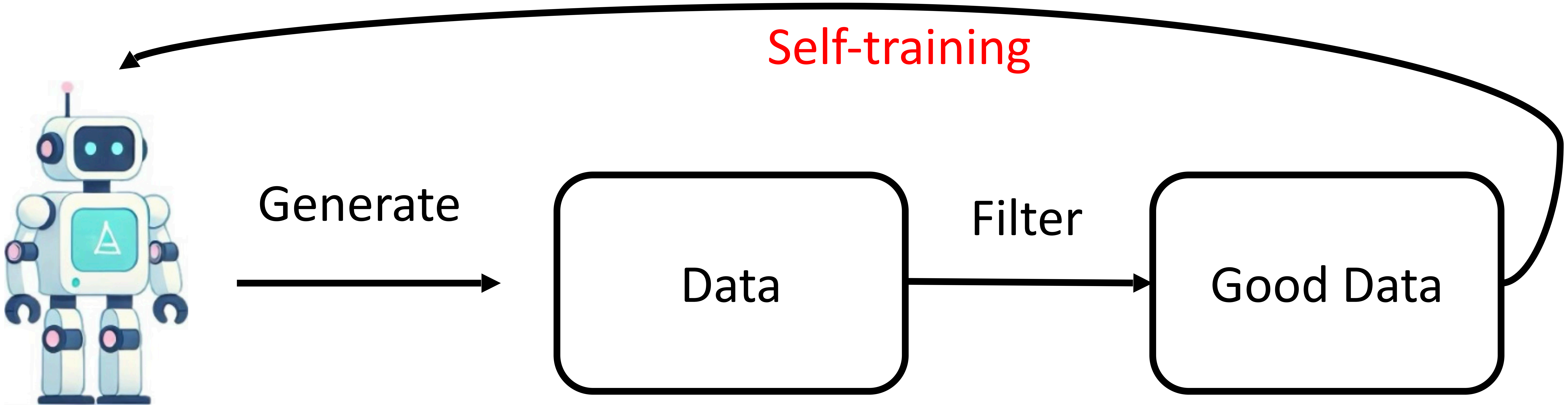
Self-Improving



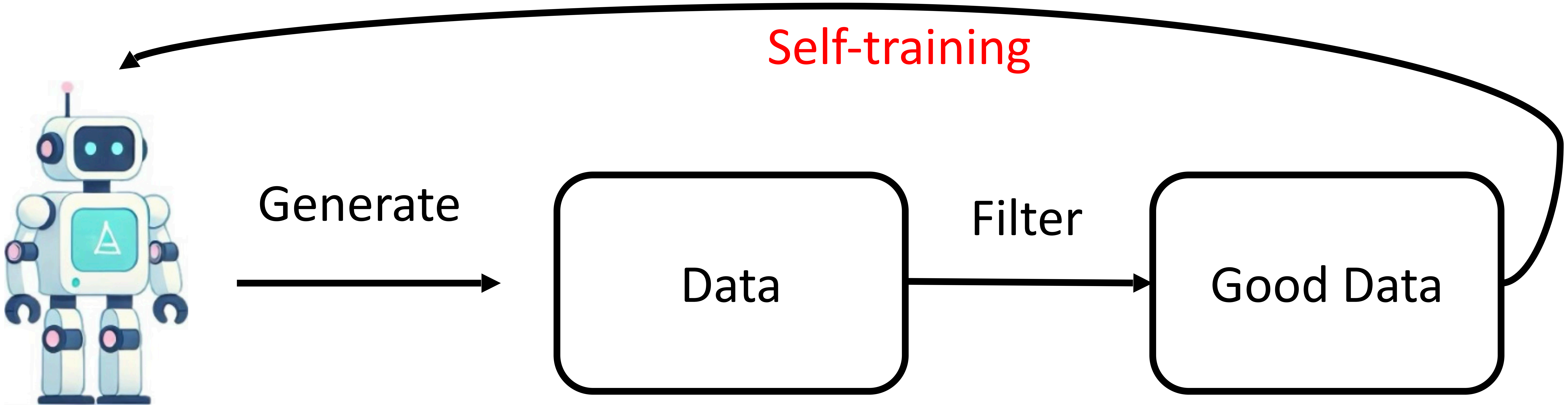
Self-Improving



Self-Improving



Self-Improving



Self-Improving

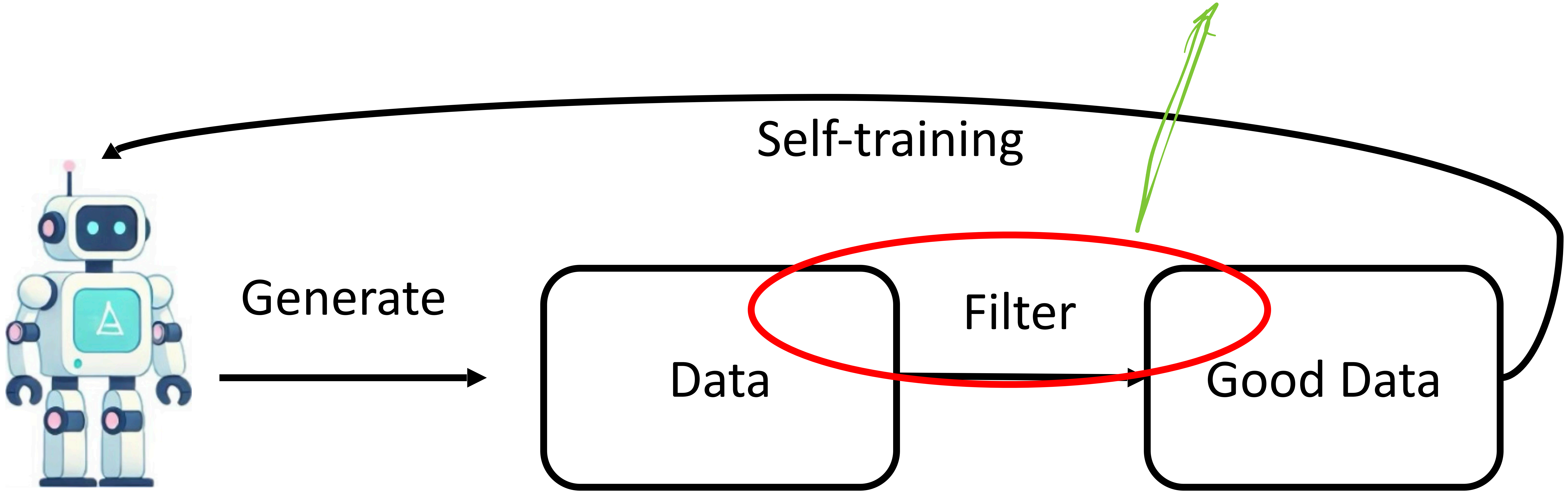
Reinforcement Learning

Reinforce

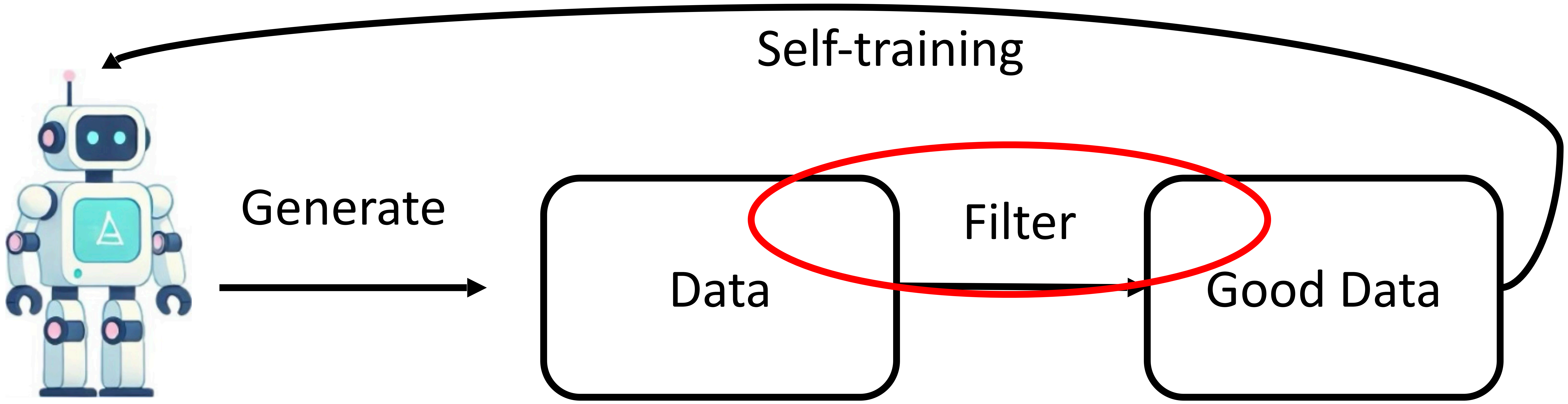
inference

self-generate data

Why can Self-Training Work?

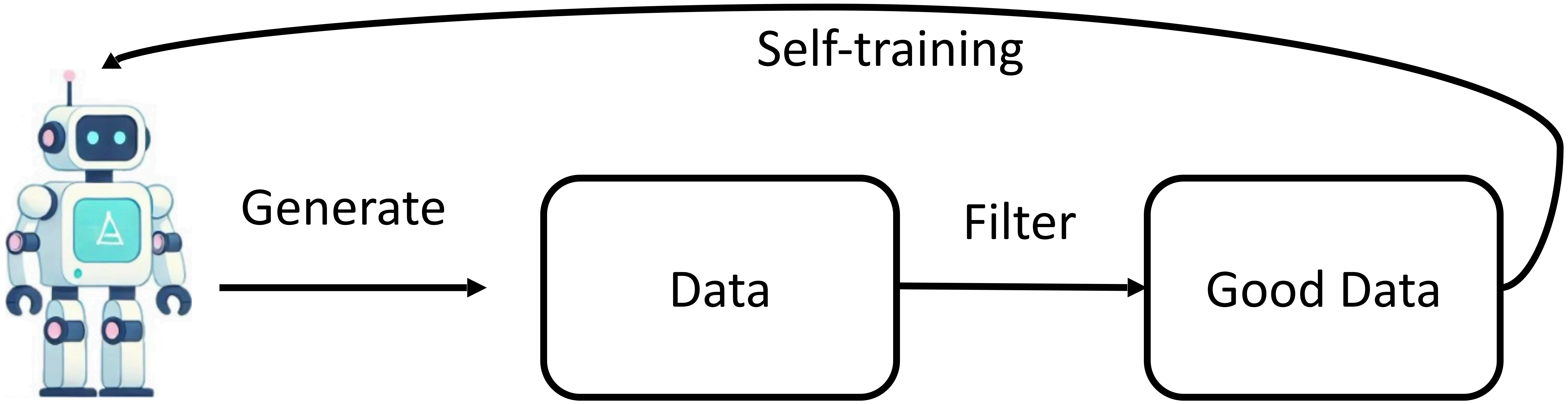


Why can Self-Training Work?

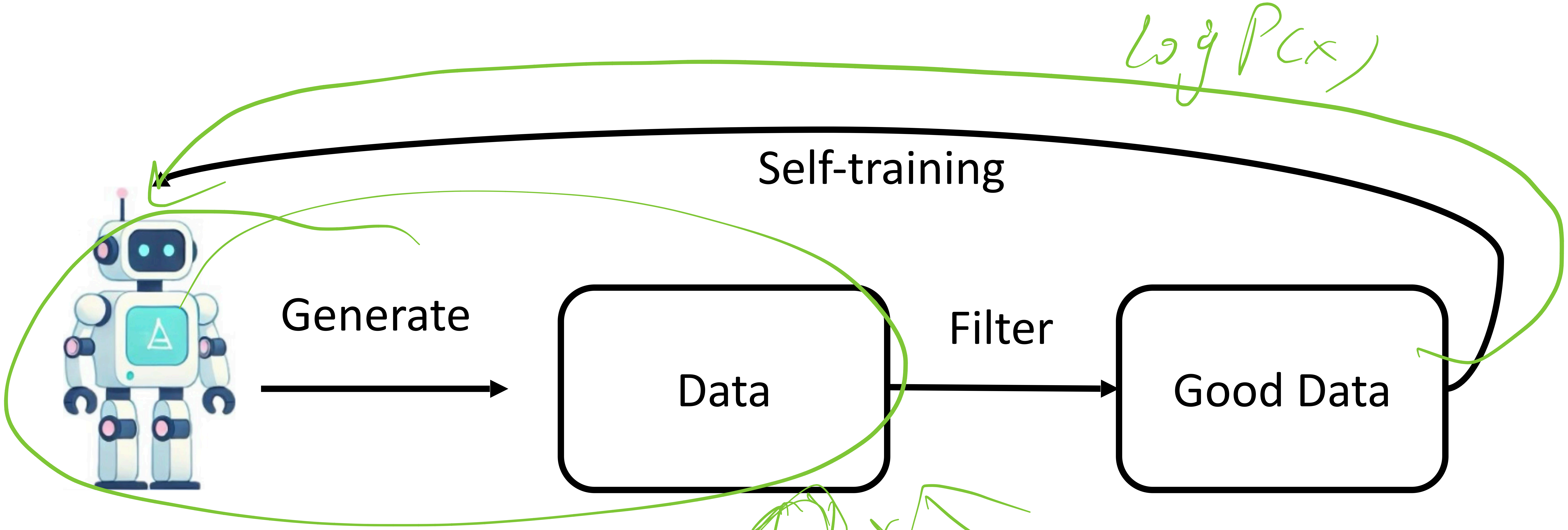


The filter step provides external signal/supervision

Self-Improving and Reinforcement Learning



Self-Improving and Reinforcement Learning

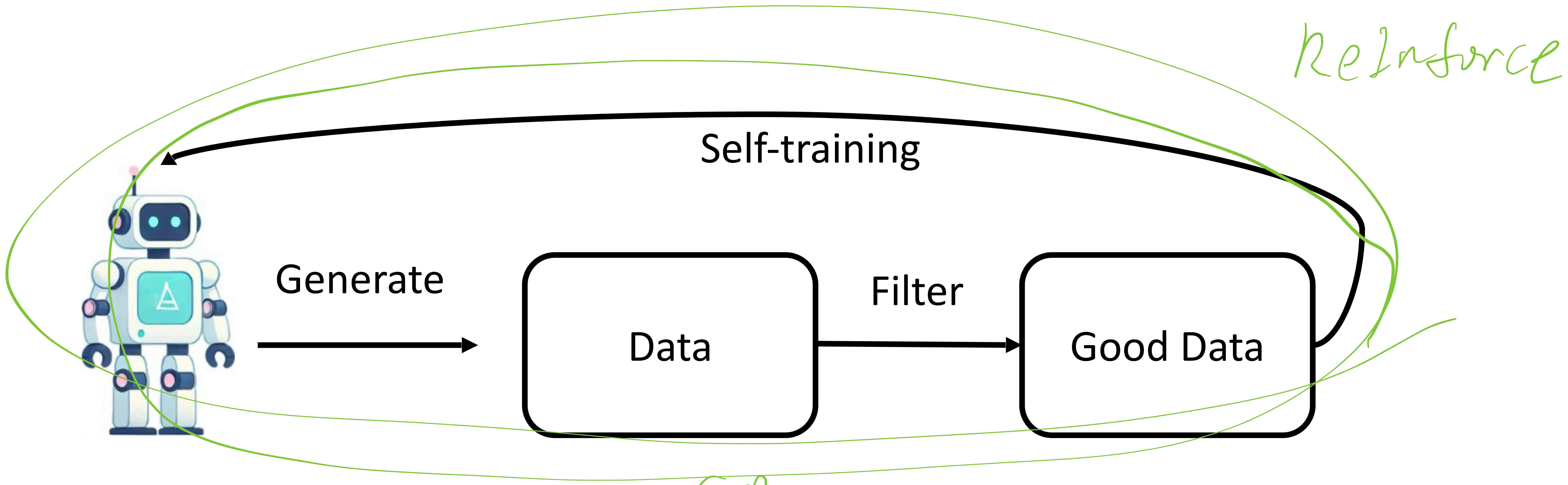


Policy Gradient:

$$\text{Objective} = \sum_{i=1}^{n} \frac{1}{n} R(x^{(i)}) \log p_\theta(x^{(i)}) \qquad x^{(1)}, \ldots, x^{(n)} \sim p_\theta(x)$$

$\log p_\theta(x)$

Self-Improving and Reinforcement Learning



$R = \begin{cases} 0 \\ 1 \end{cases}$

Policy Gradient:

$$\text{Objective} = \sum_{i=1}^{n} \frac{1}{n} R(x^{(i)}) \log p_\theta(x^{(i)}) \qquad x^{(1)}, \ldots, x^{(n)} \sim p_\theta(x)$$

Filtering is binary reward R

Test-Time Scaling

The Bitter Lesson

Rich Sutton

March 13, 2019

The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin. The ultimate reason for this is Moore's law, or rather its generalization of continued exponentially falling cost per unit of computation. Most AI research has been conducted as if the computation available to the agent were constant (in which case leveraging human knowledge would be one of the only ways to improve performance) but, over a slightly longer time than a typical research project, massively more computation inevitably becomes available. Seeking an improvement that makes a difference in the shorter term, researchers seek to leverage their human knowledge of the domain, but the only thing that matters in the long run is the leveraging of computation. These two need not run counter to each other, but in practice they tend to. Time spent on one is time not spent on the other. There are psychological commitments to investment in one approach or the other. And the human-knowledge approach tends to complicate methods in ways that make them less suited to taking advantage of general methods leveraging computation. There were many examples of AI researchers' belated learning of this bitter lesson, and it is instructive to review some of the most prominent.

Test-Time Scaling

o1

The Bitter Lesson

Rich Sutton

March 13, 2019

The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin. The ultimate reason for this is Moore's law, or rather its generalization of continued exponentially falling cost per unit of computation. Most AI research has been conducted as if the computation available to the agent were constant (in which case leveraging human knowledge would be one of the only ways to improve performance) but, over a slightly longer time than a typical research project, massively more computation inevitably becomes available. Seeking an improvement that makes a difference in the shorter term, researchers seek to leverage their human knowledge of the domain, but the only thing that matters in the long run is the leveraging of computation. These two need not run counter to each other, but in practice they tend to. Time spent on one is time not spent on the other. There are psychological commitments to investment in one approach or the other. And the human-knowledge approach tends to complicate methods in ways that make them less suited to taking advantage of general methods leveraging computation. There were many examples of AI researchers' belated learning of this bitter lesson, and it is instructive to review some of the most prominent.

computation

Test-Time Scaling

2024 Nov.

The most important part about CoT, is that itself can scale, providing a new scaling dimension.

By reasoning longer and longer, the performance gets better and better, this is one way of test-time scaling

This is very natural, because humans think longer when dealing with more complex problems. Before CoT, transformers didn't have such a mechanism!

before CoT:

Q: mathe met question

A: [shot anwser] not spend more compu- during test-time

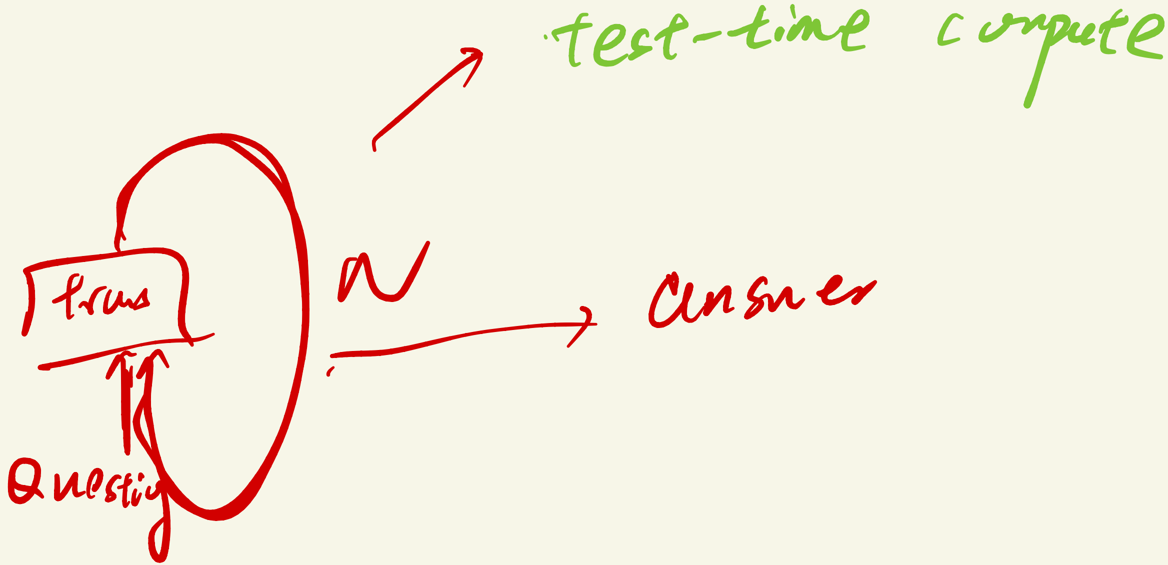
after CoT.

Q:

CoT

A: [thinking] answer

continuous CoT


test-time compute
N
answer
Question

Test-Time Scaling

Long CoT opens a new era and paradigm shift (e.g., o1, o3, GPT-5, DeepSeek-R1)

The Evolution of “LLMs Reasoning”

Wei et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. 2022.
Yu et al. MetaMath: Bootstrap Your Own Mathematical Questions for Large Language Models. 2023
Yue et al. MAmmoTH: Building Math Generalist Models through Hybrid Instruction Tuning. 2023
Zelikman et al. STaR: Bootstrapping Reasoning With Reasoning. 2022
Singh et al. Beyond Human Data: Scaling Self-Training for Problem-Solving with Language Models. 2023
Lightman et al. Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations. 2023
Wang et al. Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations. 2023
Shao et al. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. 2024
OpenAI. Introducing OpenAI o1. 2024
DeepSeek-AI. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. 2025
Kimi Team. Kimi k1.5: Scaling Reinforcement Learning with LLMs. 2025

The Evolution of “LLMs Reasoning”

Wei et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. 2022.
Yu et al. MetaMath: Bootstrap Your Own Mathematical Questions for Large Language Models. 2023
Yue et al. MAmmoTH: Building Math Generalist Models through Hybrid Instruction Tuning. 2023
Zelikman et al. STaR: Bootstrapping Reasoning With Reasoning. 2022
Singh et al. Beyond Human Data: Scaling Self-Training for Problem-Solving with Language Models. 2023
Lightman et al. Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations. 2023
Wang et al. Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations. 2023
Shao et al. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. 2024
OpenAI. Introducing OpenAI o1. 2024
DeepSeek-AI. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. 2025
Kimi Team. Kimi k1.5: Scaling Reinforcement Learning with LLMs. 2025

The Evolution of “LLMs Reasoning”

Wei et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. 2022.
Yu et al. MetaMath: Bootstrap Your Own Mathematical Questions for Large Language Models. 2023
Yue et al. MAmmoTH: Building Math Generalist Models through Hybrid Instruction Tuning. 2023
Zelikman et al. STaR: Bootstrapping Reasoning With Reasoning. 2022
Singh et al. Beyond Human Data: Scaling Self-Training for Problem-Solving with Language Models. 2023
Lightman et al. Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations. 2023
Wang et al. Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations. 2023
Shao et al. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. 2024
OpenAI. Introducing OpenAI o1. 2024
DeepSeek-AI. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. 2025
Kimi Team. Kimi k1.5: Scaling Reinforcement Learning with LLMs. 2025

The Evolution of "LLMs Reasoning"

Prompting CoT Reasoning

CoT [Wei et al. 2022]

Distilling from (Stronger)Teachers

MetaMath [Yu et al. 2023]

MAmmoTH [Yue et al. 2023]

Self-improving reasoning

STaR [Eric et al. 2022]
ReST-EM [Singh et al. 2023]

Wei et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. 2022.
Yu et al. MetaMath: Bootstrap Your Own Mathematical Questions for Large Language Models. 2023
Yue et al. MAmmoTH: Building Math Generalist Models through Hybrid Instruction Tuning. 2023
Zelikman et al. STaR: Bootstrapping Reasoning With Reasoning. 2022
Singh et al. Beyond Human Data: Scaling Self-Training for Problem-Solving with Language Models. 2023
Lightman et al. Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations. 2023
Wang et al. Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations. 2023
Shao et al. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. 2024
OpenAI. Introducing OpenAI o1. 2024
DeepSeek-AI. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. 2025
Kimi Team. Kimi k1.5: Scaling Reinforcement Learning with LLMs. 2025

The Evolution of "LLMs Reasoning"

Reward Model

Prompting CoT Reasoning

CoT [Wei et al. 2022]

Distilling from (Stronger)Teachers

MetaMath [Yu et al. 2023]

MAmmoTH [Yue et al. 2023]

Self-improving reasoning

STaR [Eric et al. 2022]

ReST-EM [Singh et al. 2023]

(RL with) Model-based Process Reward

Verify-step [Lightman et al. 2023]

Math-Shepherd [Singh et al. 2023]

DeepSeek-Math [Shao et al. 2023]

Wei et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. 2022.
Yu et al. MetaMath: Bootstrap Your Own Mathematical Questions for Large Language Models. 2023
Yue et al. MAmmoTH: Building Math Generalist Models through Hybrid Instruction Tuning. 2023
Zelikman et al. STaR: Bootstrapping Reasoning With Reasoning. 2022
Singh et al. Beyond Human Data: Scaling Self-Training for Problem-Solving with Language Models. 2023
Lightman et al. Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations. 2023
Wang et al. Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations. 2023
Shao et al. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. 2024
OpenAI. Introducing OpenAI o1. 2024
DeepSeek-AI. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. 2025
Kimi Team. Kimi k1.5: Scaling Reinforcement Learning with LLMs. 2025

The Evolution of “LLMs Reasoning”

Prompting CoT Reasoning

CoT [Wei et al. 2022]

Distilling from (Stronger)Teachers

MetaMath [Yu et al. 2023]

MAmmoTH [Yue et al. 2023]

Self-improving reasoning

STaR [Eric et al. 2022]
ReST-EM [Singh et al. 2023]

(RL with) Model-based Process Reward

Verify-step [Lightman et al. 2023]
Math-Shepherd [Singh et al. 2023]
DeepSeek-Math [Shao et al. 2023]

Reinforcement Learning with Verifiable Reward (RLVR)

o1 [OpenAI 2024]
DeepSeek-R1 [DeepSeek, 2025]
Kimi k1.5 [Kimi Team, 2025]

Wei et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. 2022.
Yu et al. MetaMath: Bootstrap Your Own Mathematical Questions for Large Language Models. 2023
Yue et al. MAmmoTH: Building Math Generalist Models through Hybrid Instruction Tuning. 2023
Zelikman et al. STaR: Bootstrapping Reasoning With Reasoning. 2022
Singh et al. Beyond Human Data: Scaling Self-Training for Problem-Solving with Language Models. 2023
Lightman et al. Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations. 2023
Wang et al. Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations. 2023
Shao et al. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. 2024
OpenAI. Introducing OpenAI o1. 2024
DeepSeek-AI. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. 2025
Kimi Team. Kimi k1.5: Scaling Reinforcement Learning with LLMs. 2025

The Evolution of "LLMs Reasoning"

Prompting CoT Reasoning

CoT [Wei et al. 2022]

Distilling from (Stronger)Teachers

MetaMath [Yu et al. 2023]

MAmmoTH [Yue et al. 2023]

Self-improving reasoning

STaR [Eric et al. 2022]

ReST-EM [Singh et al. 2023]

(RL with) Model-based Process Reward

Verify-step [Lightman et al. 2023]

Math-Shepherd [Singh et al. 2023]

DeepSeek-Math [Shao et al. 2023]

Reinforcement Learning with Verifiable Reward (RLVR)

o1 [OpenAI 2024]

DeepSeek-R1 [DeepSeek, 2025]

Kimi k1.5 [Kimi Team, 2025]

Test-Time Compute Scaling

Wei et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. 2022.
Yu et al. MetaMath: Bootstrap Your Own Mathematical Questions for Large Language Models. 2023
Yue et al. MAmmoTH: Building Math Generalist Models through Hybrid Instruction Tuning. 2023
Zelikman et al. STaR: Bootstrapping Reasoning With Reasoning. 2022
Singh et al. Beyond Human Data: Scaling Self-Training for Problem-Solving with Language Models. 2023
Lightman et al. Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations. 2023
Wang et al. Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations. 2023
Shao et al. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. 2024
OpenAI. Introducing OpenAI o1. 2024
DeepSeek-AI. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. 2025
Kimi Team. Kimi k1.5: Scaling Reinforcement Learning with LLMs. 2025

Thank You!