Training Language Models to Reason - I

Advances in Large Language Models

ELL8299 · AIL861

Gaurav Pandey
Senior Research Scientist, IBM Research

What is reasoning in LLMs?

- The intermediate sequence of tokens/phrases and sentences that the LLM generates before it generates the final solution.
- It can simply be the sequence of intermediate steps that lead to the solution.
- Or it can be a sequence of planning, reflection and backtracking steps that allow the model to explore multiple possible solutions and verify them.

π_θ , $y \rightarrow$ answer x - question

$z \rightarrow$ $x \rightarrow z \rightarrow y$ $\pi_\theta(z|x)\ \pi_\theta(y|z,x)$

From Prompting → Self-Training → Preferences → RL

Chain-of-Thought (CoT) prompting - inference time only

↓

STaR (Self-Taught Reasoner) – SFT on self-generated thoughts

↓

TPO (Thought Preference Optimization) – DPO on self-generated thoughts

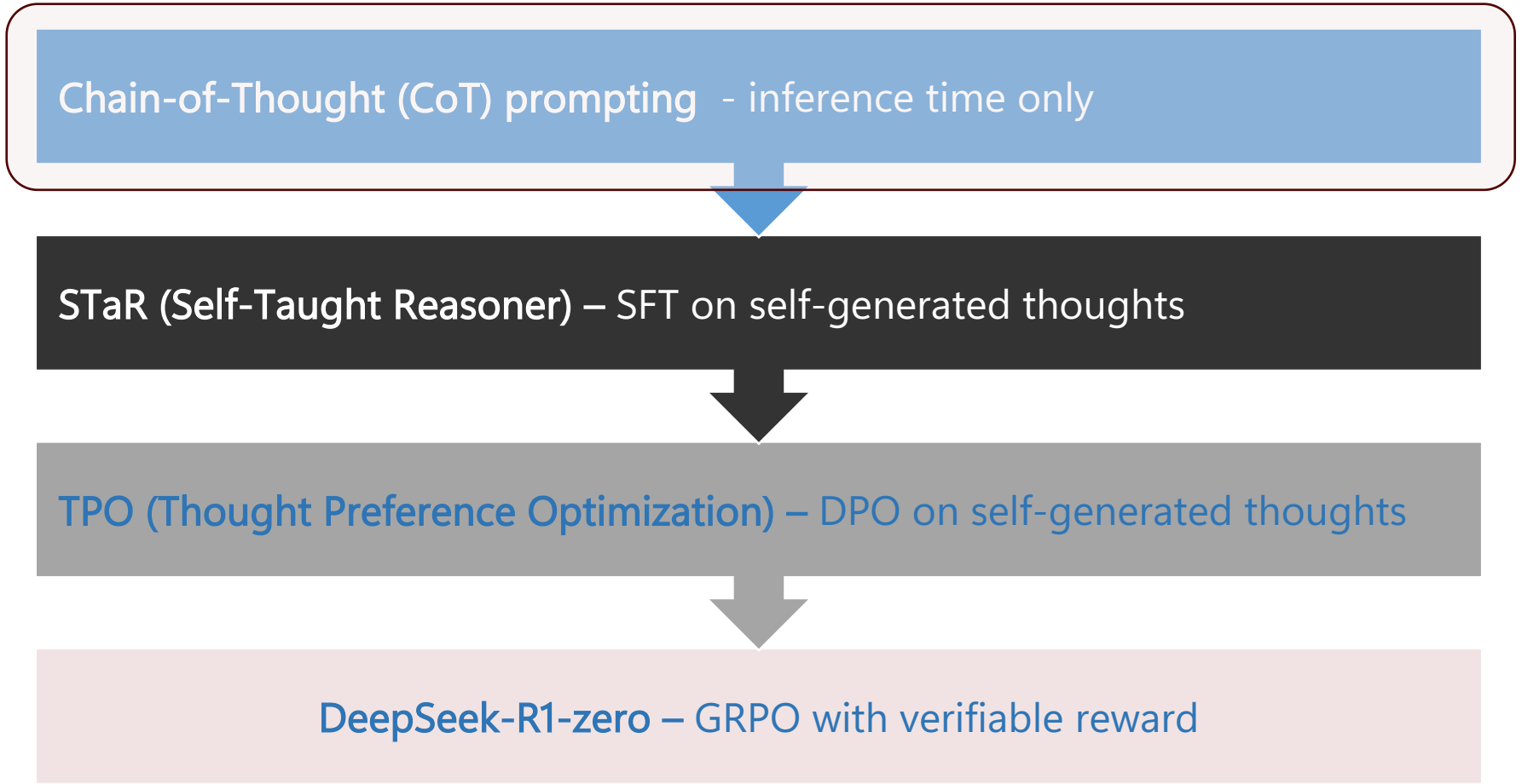
↓

DeepSeek-R1-zero – GRPO with verifiable reward

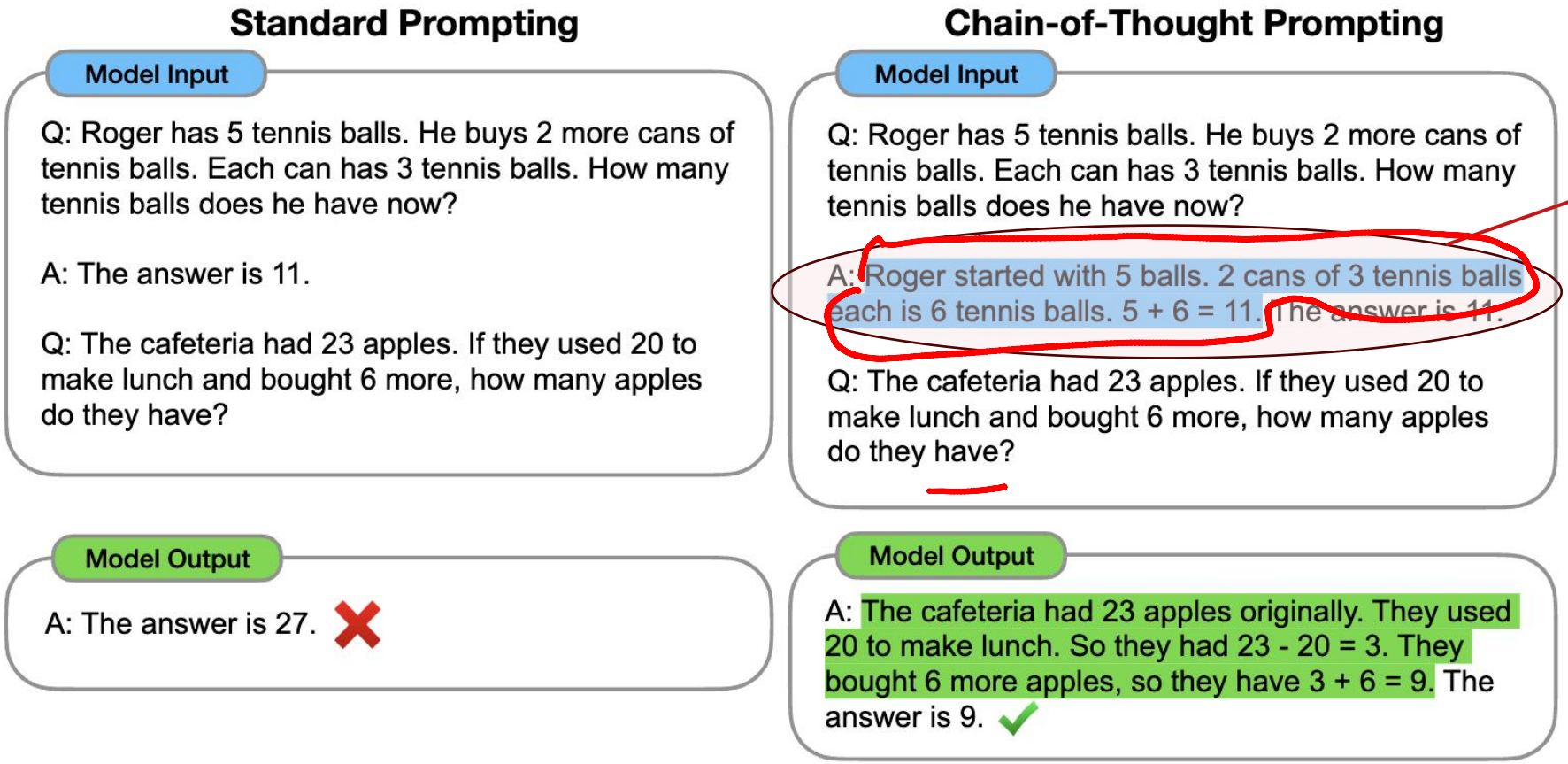
Quick recap

Train the model to generate better thoughts

From Prompting → Self-Training → Preferences → RL



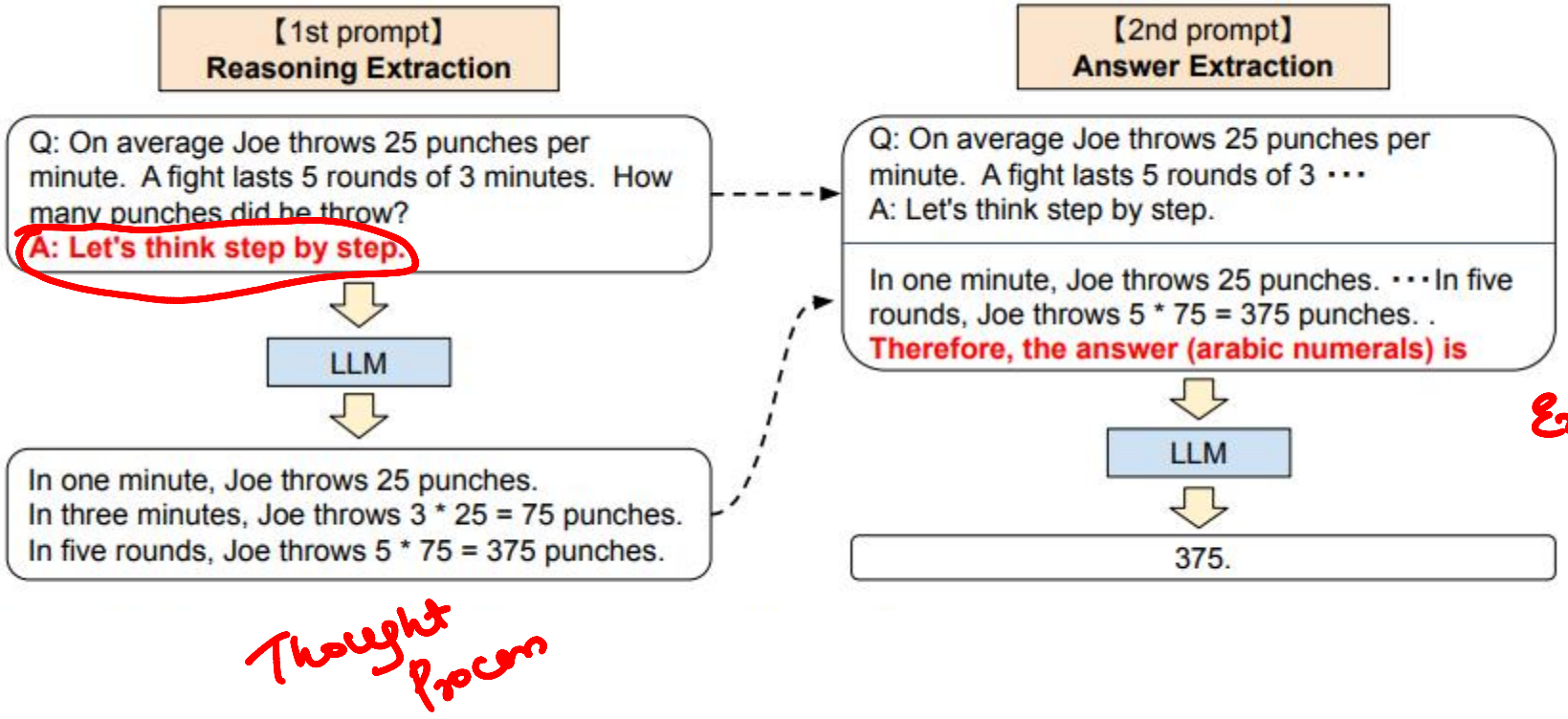
Chain-of-thought (CoT) prompting – A quick recap



In-context examples with intermediate steps

Chain-of-Thought Prompting Elicits Reasoning in Large Language Models

Zero-shot Chain-of-Thought



Put the final answer within \boxed{ }

Extract

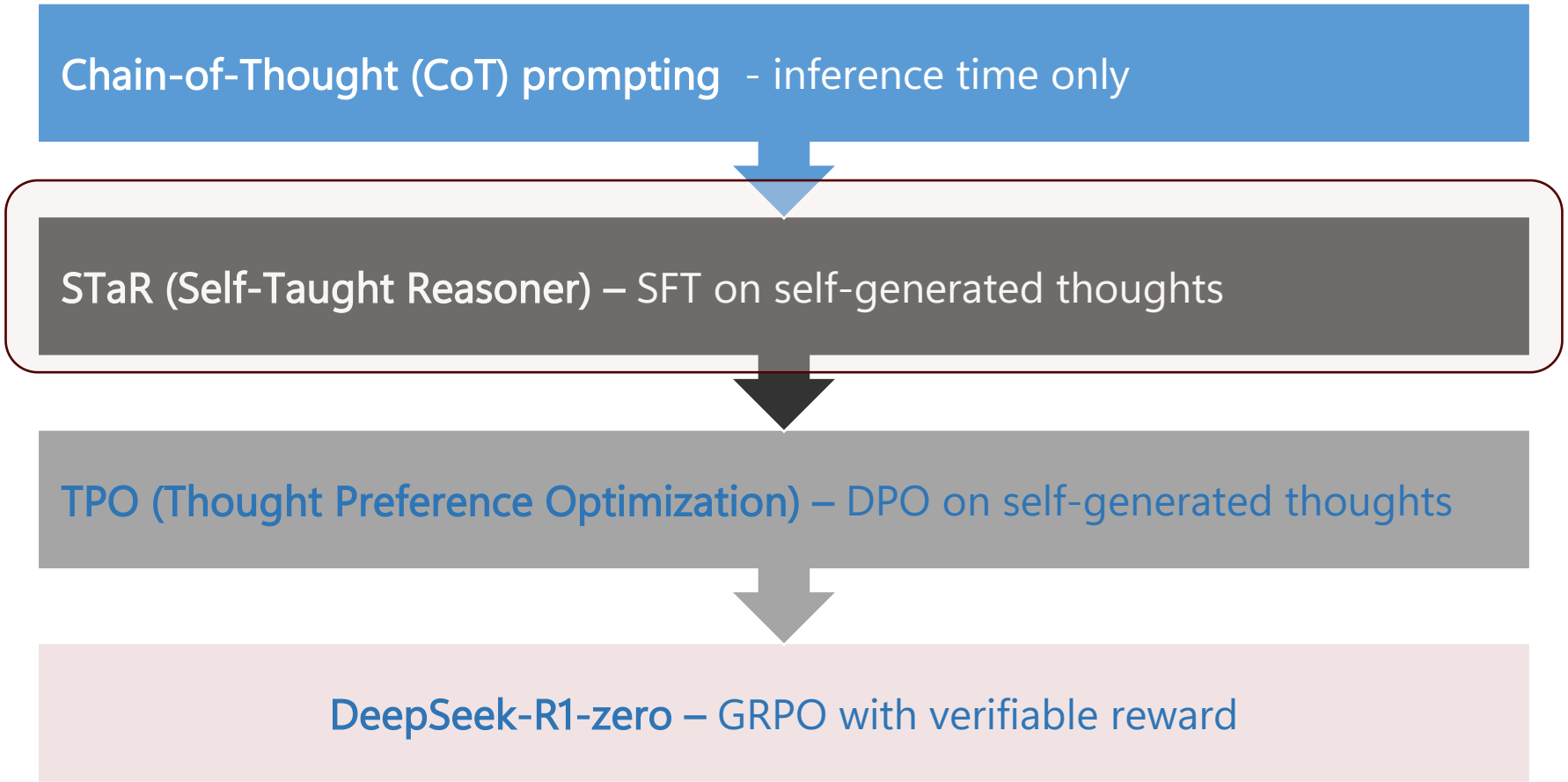
Thought Process

Finetuning for CoT

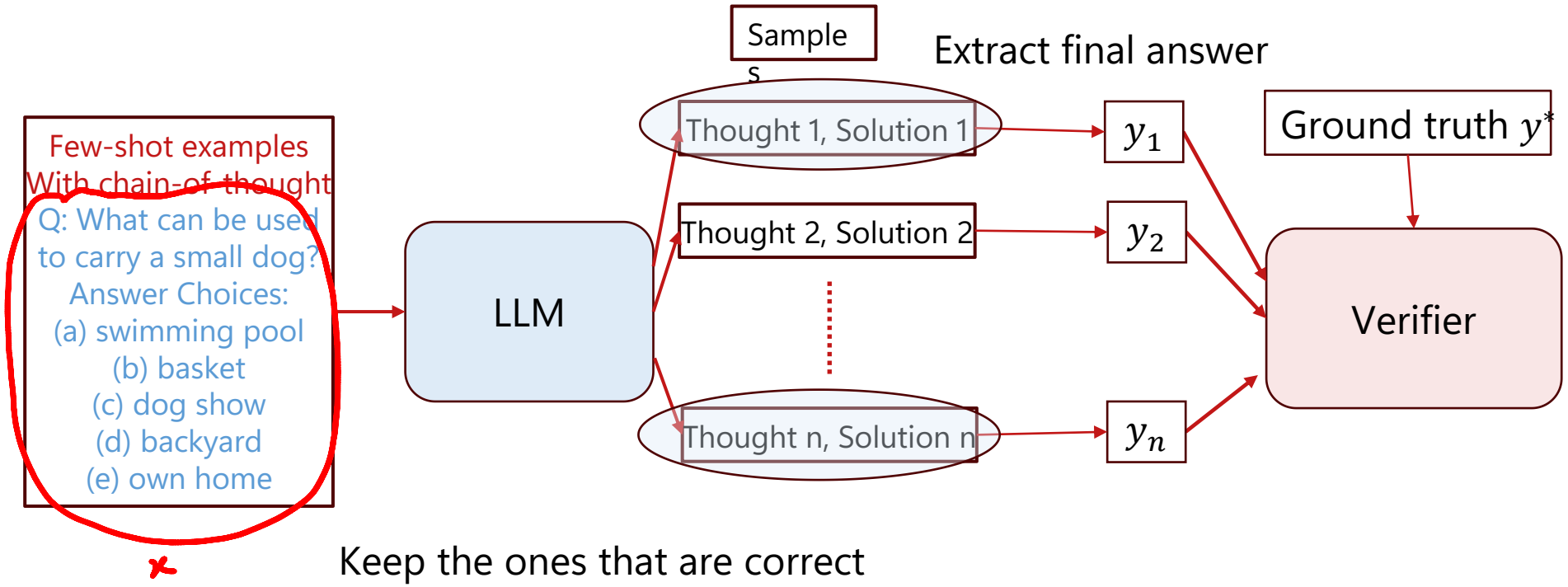
- Construct a dataset of (Question, Thought, Answer)
- Train the model to predict the Thought and Answer given the Question.

- Where can we get the Thought data from?
  - Human annotation – too expensive
  - Self generated
    - STaR – SFT on generated (thought, response) pairs
    - TPO – DPO on generated (thought, response) pairs
    - Deepseek-R1-Zero – GRPO with verifiable reward on generated (thought, response) pairs
  - From a teacher
    - Distillation

From Prompting → Self-Training → Preferences → RL



Self-Taught Reasoner without rationalization



Keep the ones that are correct

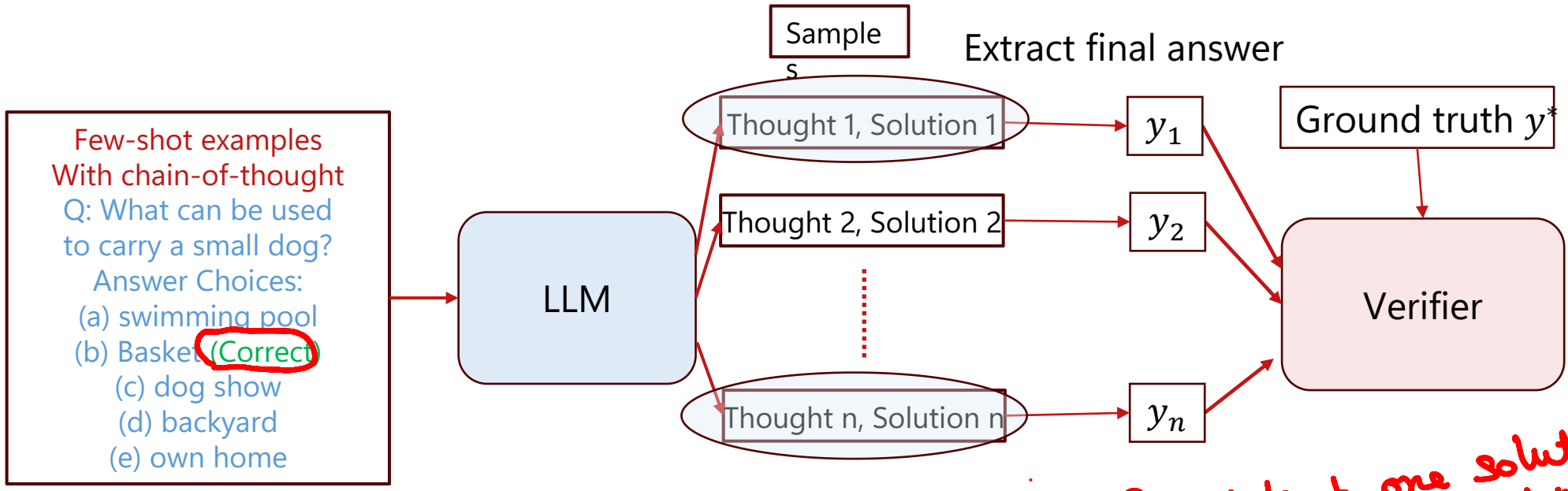
Repeat for all the queries

Perform SFT on accepted (query, thought, solution) triplets and repeat.

Limitations

- Model is only trained on examples on which it is already correct.
- Improvement ends when the model fails to solve new problems in training set.
- There is no training signal from failed examples.
- Solution:
  - Provide the correct answer as a hint.
  - Allows the model to reason backwards

Self-Taught Reasoner with rationalization



Keep the ones that are correct

Repeat for all the queries

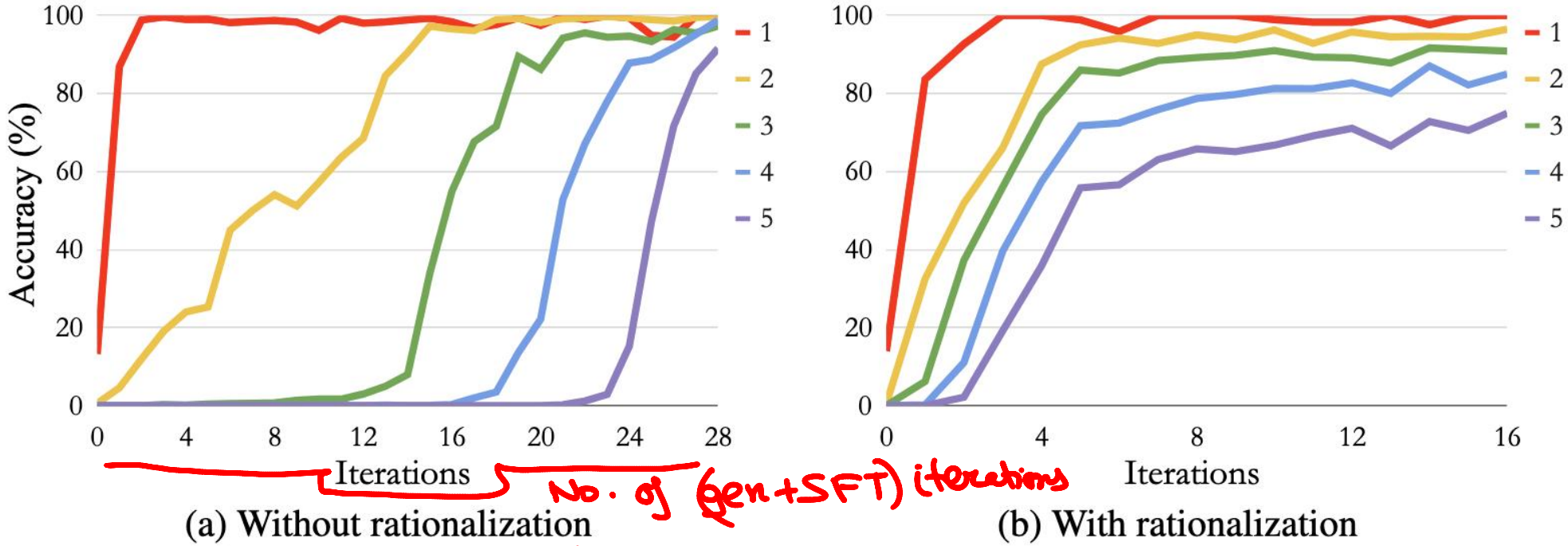
Perform SFT on accepted (query (without hint), thought, solution) triplets and repeat.

→ If at least one solution is correct, don't use hint

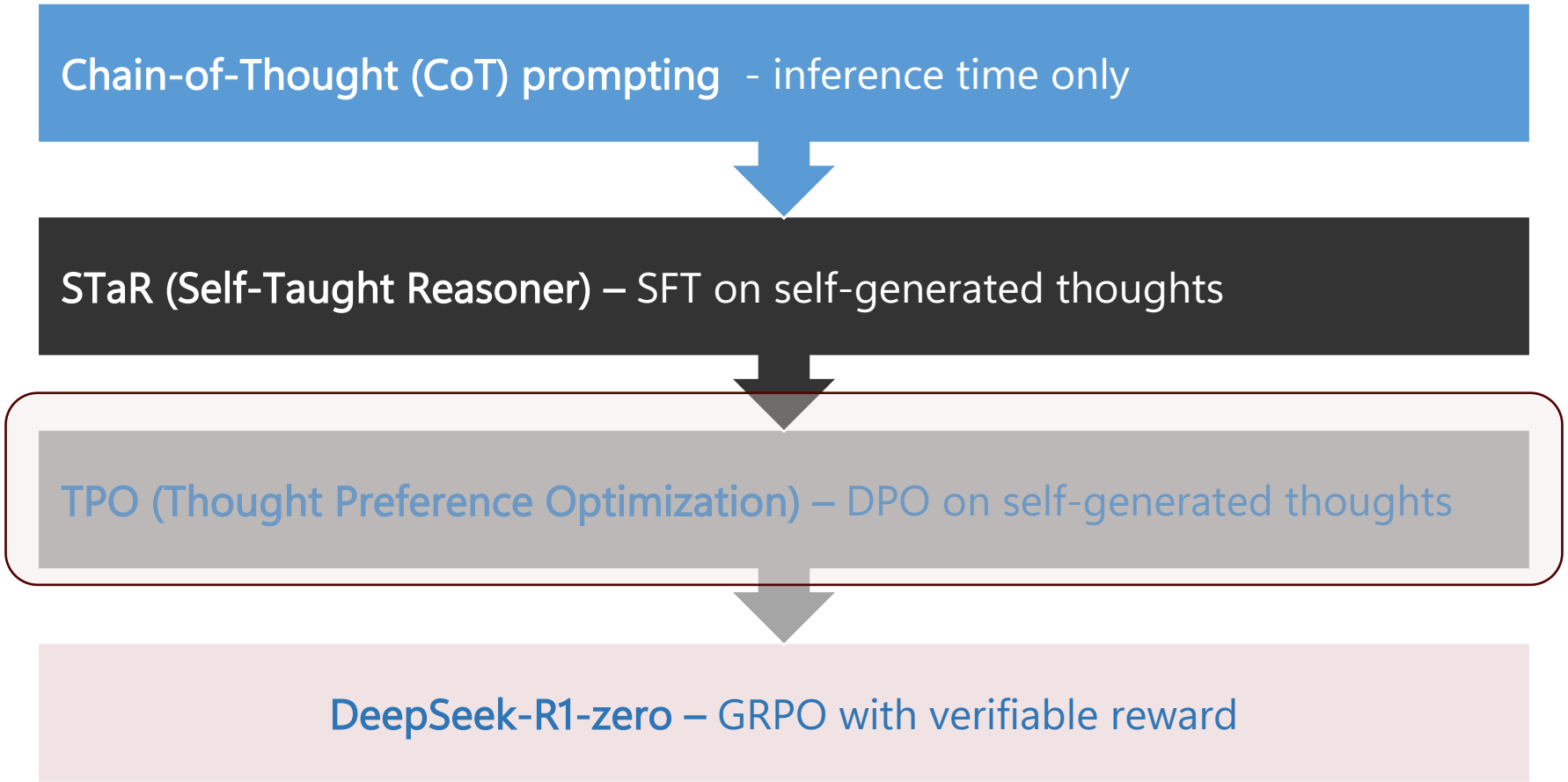
→ If none are correct, use hint

Performance with & without rationalization

- Problem – n digit summation
- Base Model – GPT-J

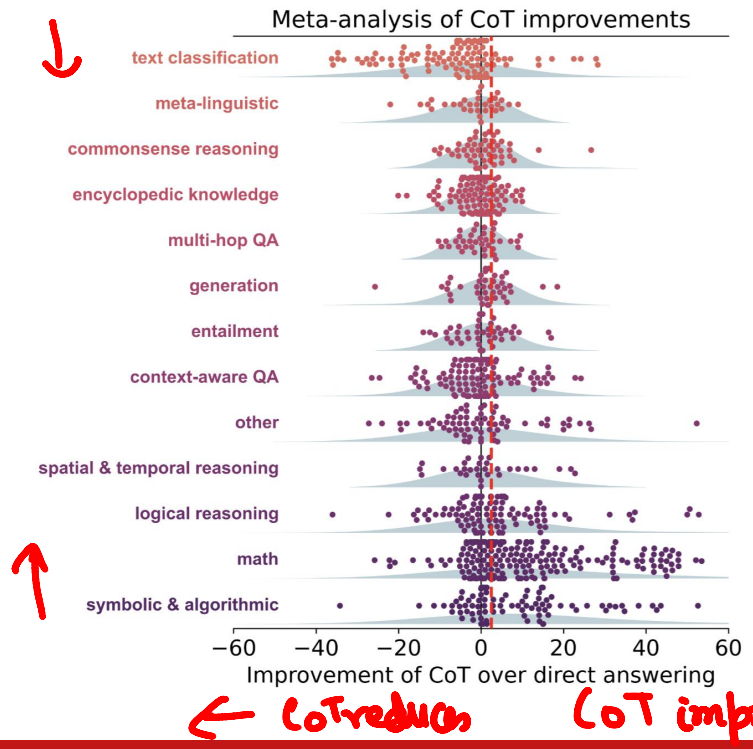


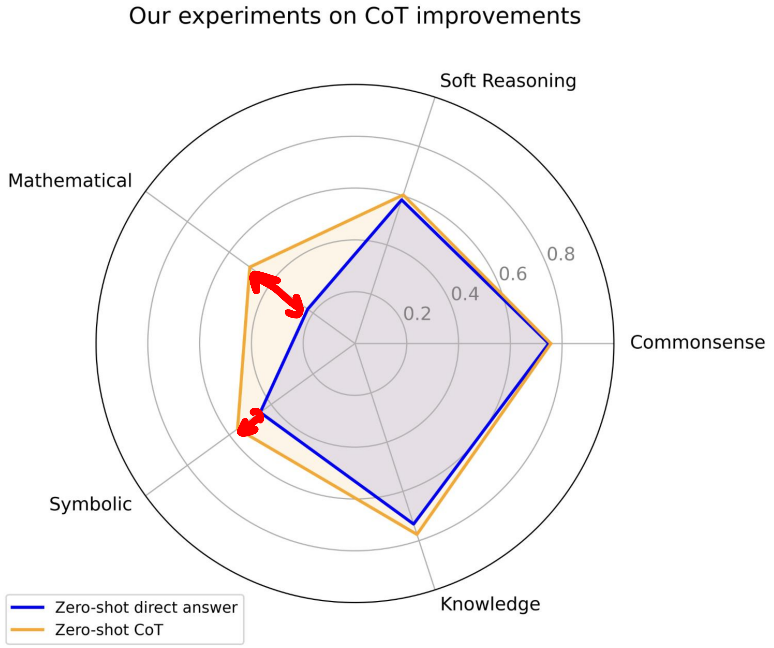
From Prompting → Self-Training → Preferences → RL



Limitations of CoT

- Chain-of-Thought works well on symbolic tasks such as Maths & Logic.
  - But not so much on other tasks





https://arxiv.org/pdf/2409.12183x

Beyond CoT

Can we do more with thoughts than just maths and logic?

For instance:

- Creative writing
  - ⓾ Thoughts can help plan overall structure and characters.
- Instruction following
  - ⓾ Thoughts can help understand the user instruction better

A non-reasoning example

Example of Non-Reasoning Instruction (generic thought prompt)

Instruction: Write me a poem in the style of neruda

Here is my thought process:

- Neruda's style is known for being passionate, sensual, and evocative. I need to tap into that emotional depth.
- His poetry often explores themes of love, politics, and social justice. I should choose a theme that resonates with Neruda's concerns.
- Neruda's writing is also infused with a sense of mysticism and the mysterious. I'll try to incorporate some enigmatic elements.

Here is my response:

Whispers in the Rainy Night
In the city's secret heart, where shadows play
Like lovers' fingers tracing the city's gray
The rain whispers secrets, a nocturnal refrain
As I stand here, lost in your eyes' sweet pain *(continued ...)*

latent thought

https://arxiv.org/pdf/2410.10630

Thinking LLMs

- Always generate a thought before the response.
  - Prompt the model to generate the thought before generating the response.

Generic Thought Prompt

Respond to the following user query in a comprehensive and detailed way. You can write down your thought process before responding. Write your thoughts after "Here is my thought process:" and write your response after "Here is my response:".

User query: {user instruction}

Specific Thought Prompt

Respond to the following user query in a comprehensive and detailed way. But first write down your internal thoughts. This must include your draft response and its evaluation. After this, write your final response after "<R>".

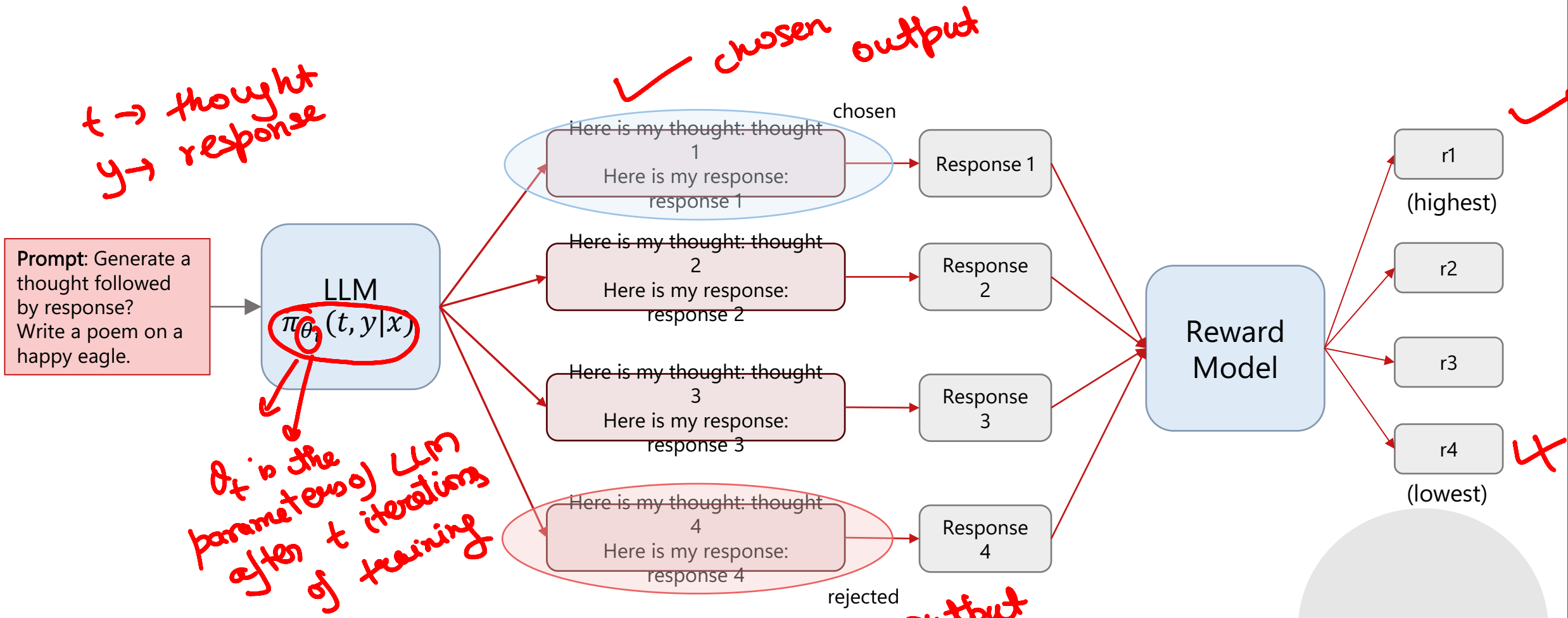
User query: {user instruction}

https://arxiv.org/pdf/2410.10630

Does it help?

- Not in the beginning.
  - The performance always drops in the beginning.
  - The model has never been trained on such data.
  - Hence, the responses initially are worse with the thoughts.
- So, how do we improve the thoughts and responses?
- Idea:
  - Lets use a reward model to judge the quality of the response.
  - Sample multiple (thought, response) pairs from the model
  - Train it to generate those pairs where the final response has high reward.

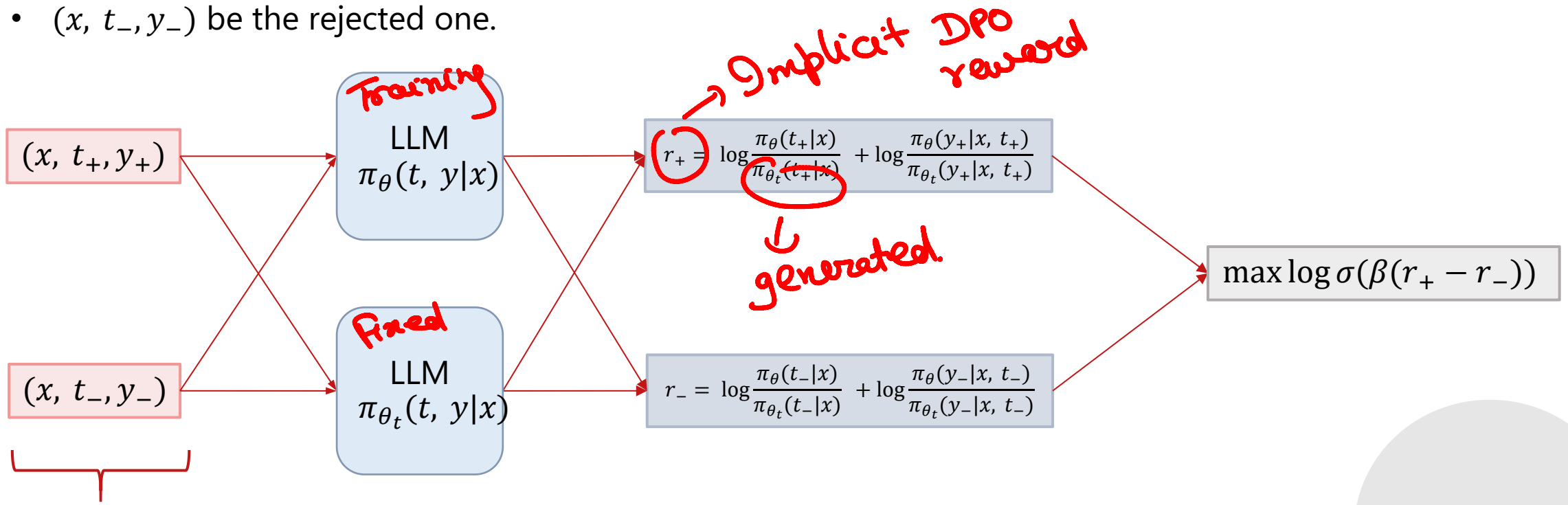
Thought Process Optimization – Preference Pair Creation



Repeat this process for all queries/instructions in the training data to create a dataset of preference pairs

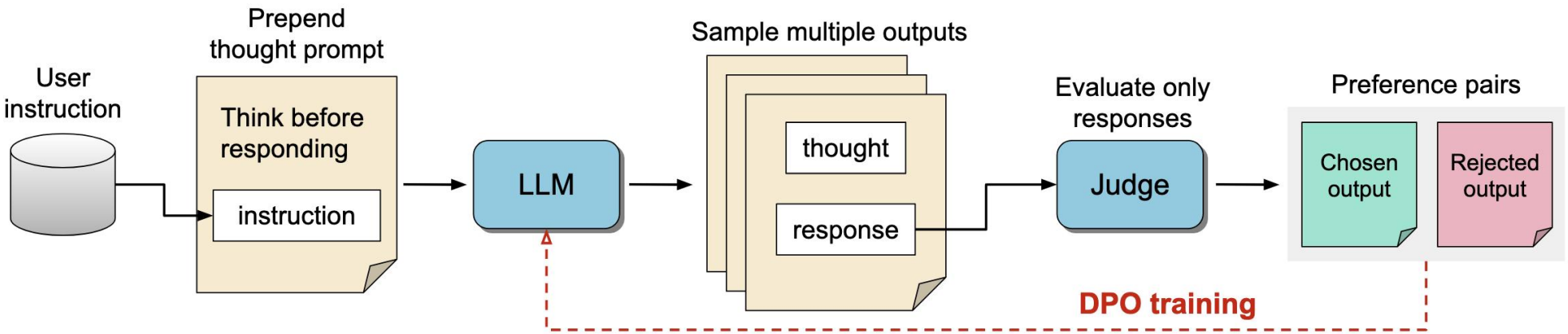
Thought Process Optimization – Train Step

- Let $(x,\ t_+, y_+)$ be the triplet containing the prompt + instruction, the chosen thought and response.
- $(x,\ t_-, y_-)$ be the rejected one.



Thought Process Optimization - Full picture

- The preference pairs creation and DPO training happen alternatively for a few iterations



https://arxiv.org/pdf/2410.10630

Length control

- Some judge models tend to favor long responses.
  - Response length grows with each iteration
- Solution: Use length-control while selecting preference pairs.
- Define a normalization function. Apply it to the reward of the samples
  - $N(r; r_1, \ldots, r_n) = \frac{r - mean(r_1, \ldots, r_n)}{\sigma(r_1, \ldots, r_n)}$
- Apply it to the length of the samples
  - $N(\ell; \ell_1, \ldots, \ell_n) = \frac{\ell - mean(\ell_1, \ldots, \ell_n)}{\sigma(\ell_1, \ldots, \ell_n)}$
- Combine the two
  - $r \leftarrow N(r; r_1, \ldots, r_n) - \beta N(\ell; \ell_1, \ldots, \ell_n)$
- Use this new reward for creating preference pairs

scaling [0, 0.04]

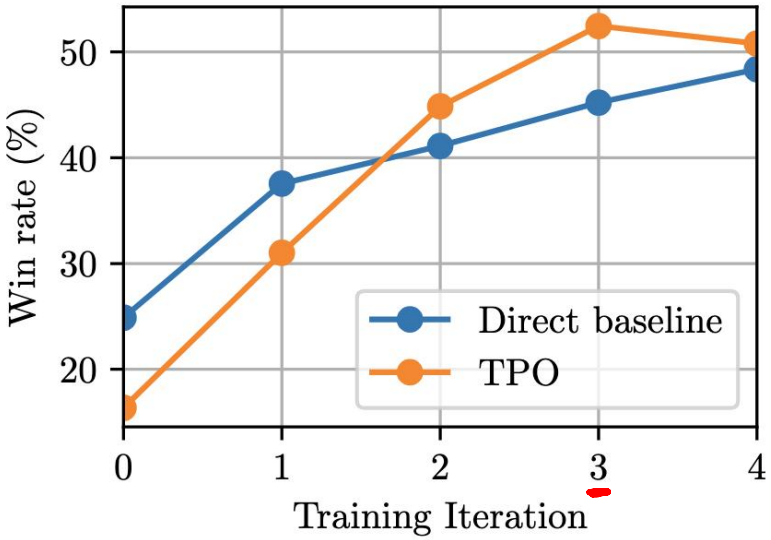
Does it work?

Method	AlpacaEval (LC)	Arena-Hard
Llama-3-8B-Instruct-based		
Llama-3-8B-Instruct	24.9	20.6
Llama-3-8B-Instruct + Thought prompt	17.3	14.1
Direct response baseline	48.4	33.0
TPO	**52.5**	**37.3**

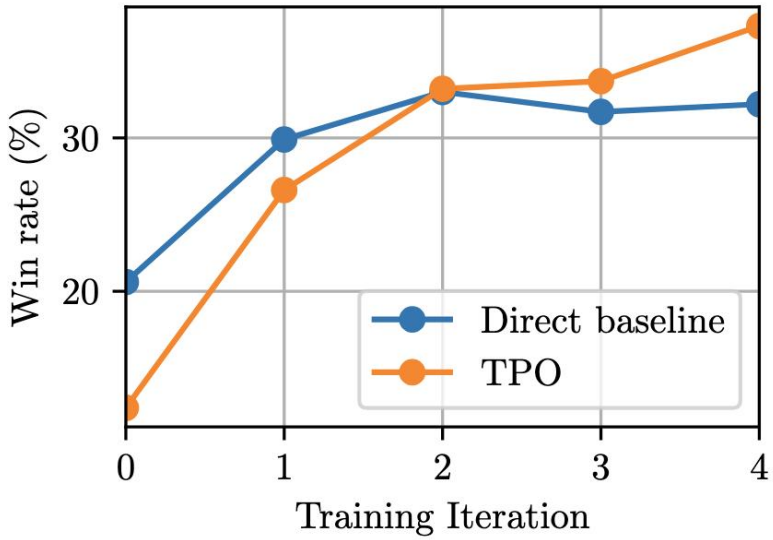
Trained on the open-assistant dataset

SFT on response only

Performance improvement with training iterations



Alpaca-eval



Arena-hard

Category-wise breakdown

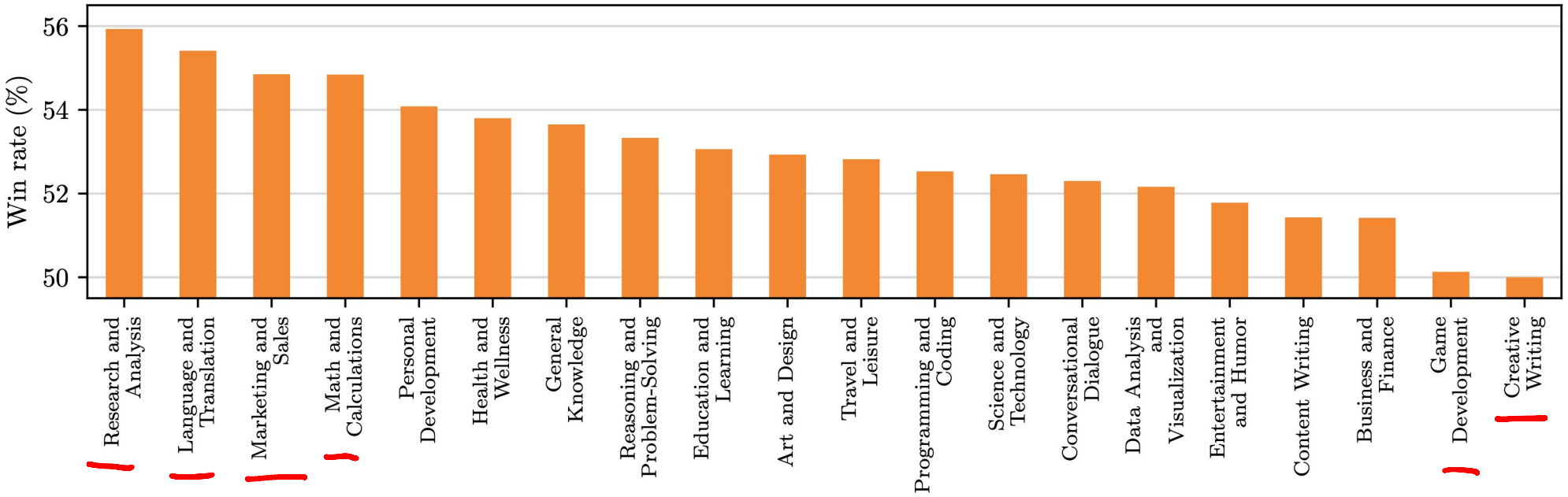
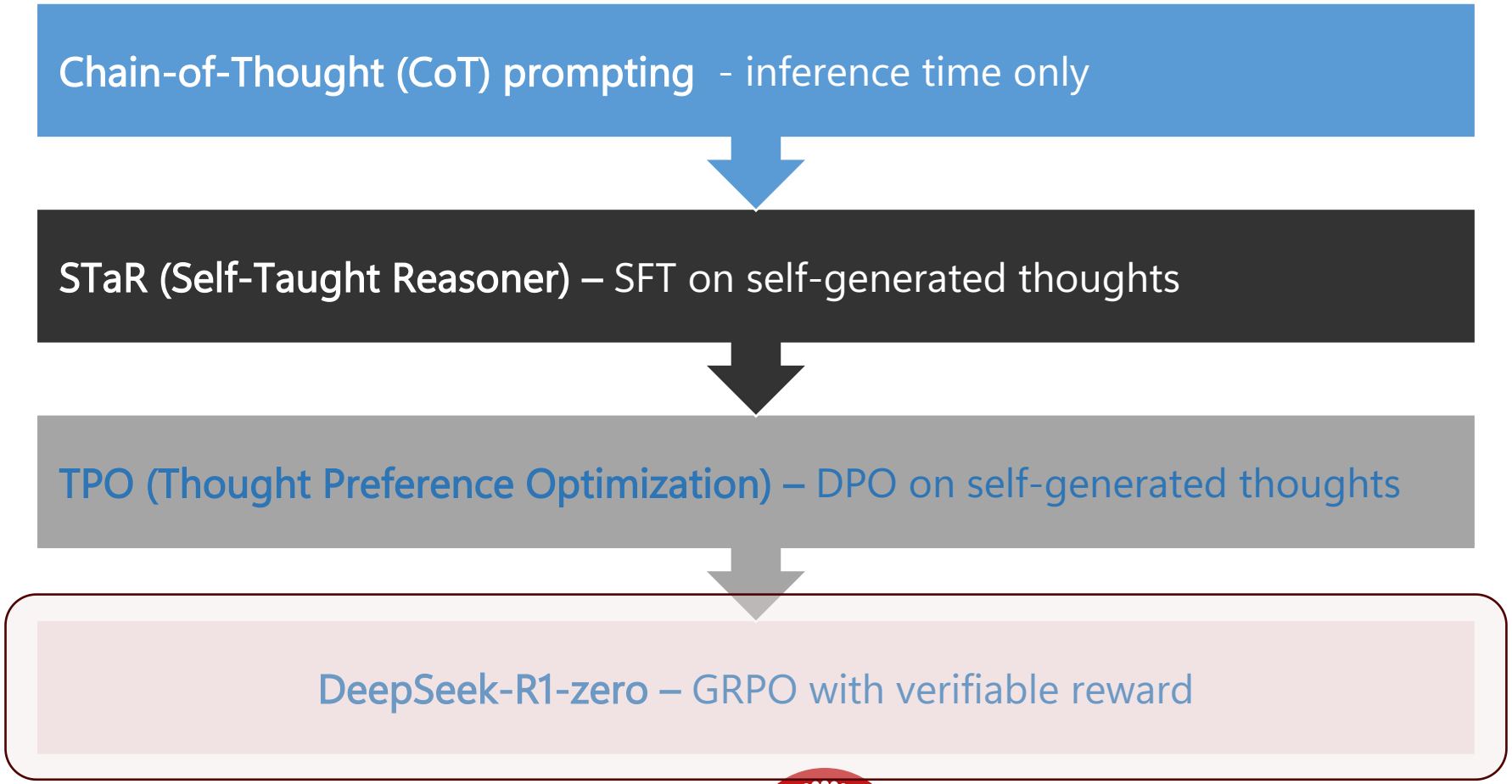


Figure 4: **Fine-Grained evaluation on unseen instructions from UltraFeedback, broken down by category.** We measure the win rate of TPO against the direct baseline as judged by GPT4.

What happens if we train forever?

- Reward models are not perfect
  - The LLM will learn to hack the reward models.
  - The average reward will keep on improving.
  - The performance across benchmarks will decline.
- The training data remains the same
  - The LLM will start overfitting to the training data.
- The DPO approach wastes a lot of generated samples
- We need
  - A method that rewards perfectly.
  - A dataset that keeps getting harder to prevent overfitting.
  - An RL method that can use all the samples.

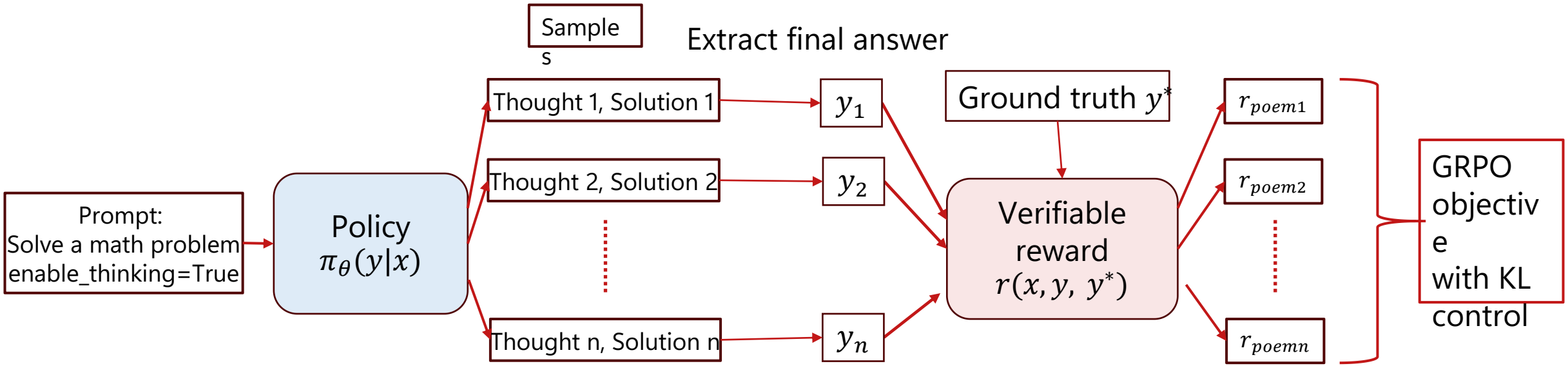
From Prompting → Self-Training → Preferences → RL



Verifiable rewards

- Rewards that can be computed objectively and reproducibly from a ground truth.
- Examples of Verifiable Reward Functions
  - **Math:** Exact numerical answer match
  - **Code:** Passes all test cases
  - **QA:** String match or F1-score over entities
  - **Formal logic tasks:** Correct proof sequence
  - **Chemistry:** Exact Match in Reaction Prediction
  - **Biology:** RMSD for Protein structure prediction
- Does not depend on noisy human or AI preferences.
- Responsible for the latest revolution in reasoning in AI (Deepseek-R1, OpenAI o1, Kimi K1.5 and Kimi K2, Qwen3)

RL with verifiable rewards



The GRPO objective – a quick recap

$$L^{GRPO}(\theta) = E\left[\min\left(r_t(\theta)\widehat{A}(y_i),\ clip(r_t(\theta),\ 1-\epsilon,\ 1+\epsilon)\widehat{A}(y_i)\right)\right]$$

- $r_t(\theta)$ is the probability ratio $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$
  - a_t is the t^{th} token of the generated output
  - s_t is the tokens before a_t
  - $\pi_{\theta_{old}}$ is the distribution from which y_i was sampled.
- Key trick of GRPO - Advantage is relative

$$A(y_i) = \frac{r(y_i) - mean\left(r(y_1), \ldots, r(y_K)\right)}{stddev\left(r(y_1), \ldots, r(y_K)\right)}$$

Deepseek-R1-zero

- Used a strong base model – Deepseek-V3-base
  - Extensive Math and Code data – sources not stated
- **Prompt Template**

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

- GRPO is used for reinforcement learning

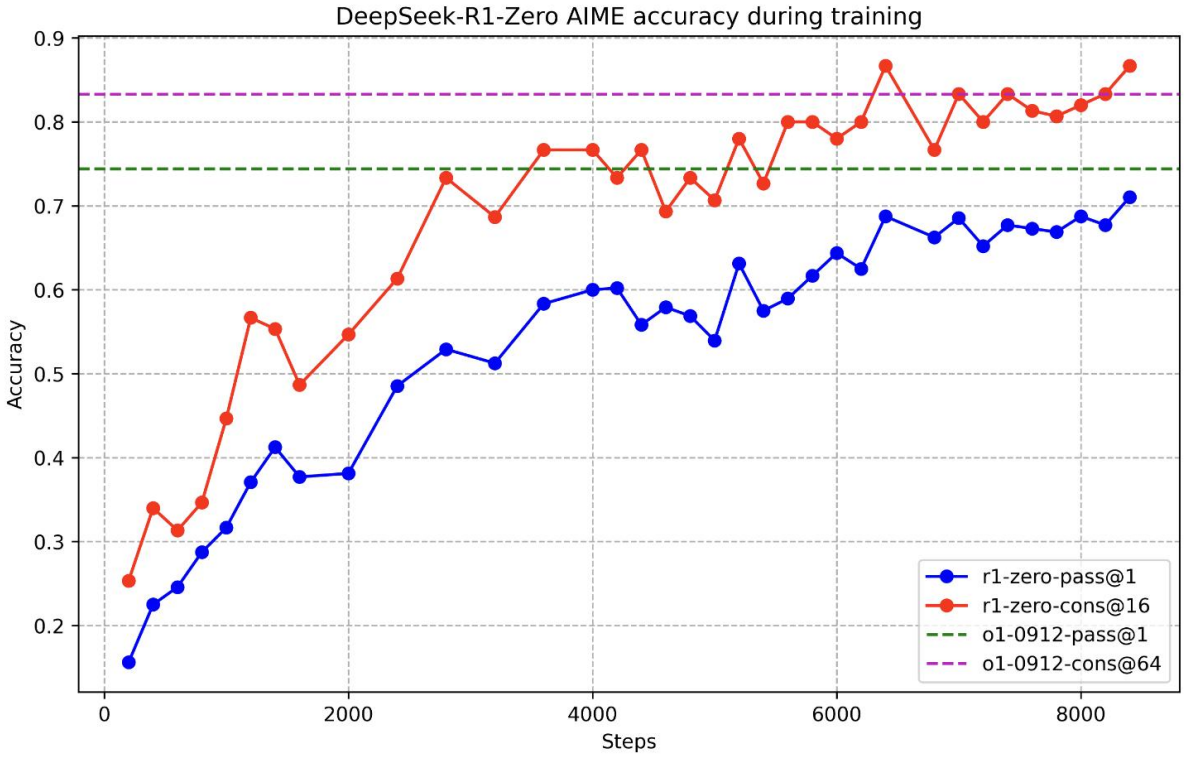
Reward Modelling

- Accuracy reward for Math
  - Instruct the model to put the final output in a specific format
  - Extract the output from the generated response
  - Verify against the ground truth answer.
- Accuracy reward for Code
  - Run unit tests on the generated code
- Format reward
  - The format should look like

$$< think > Thought\ process </think >< response > Response </response >$$

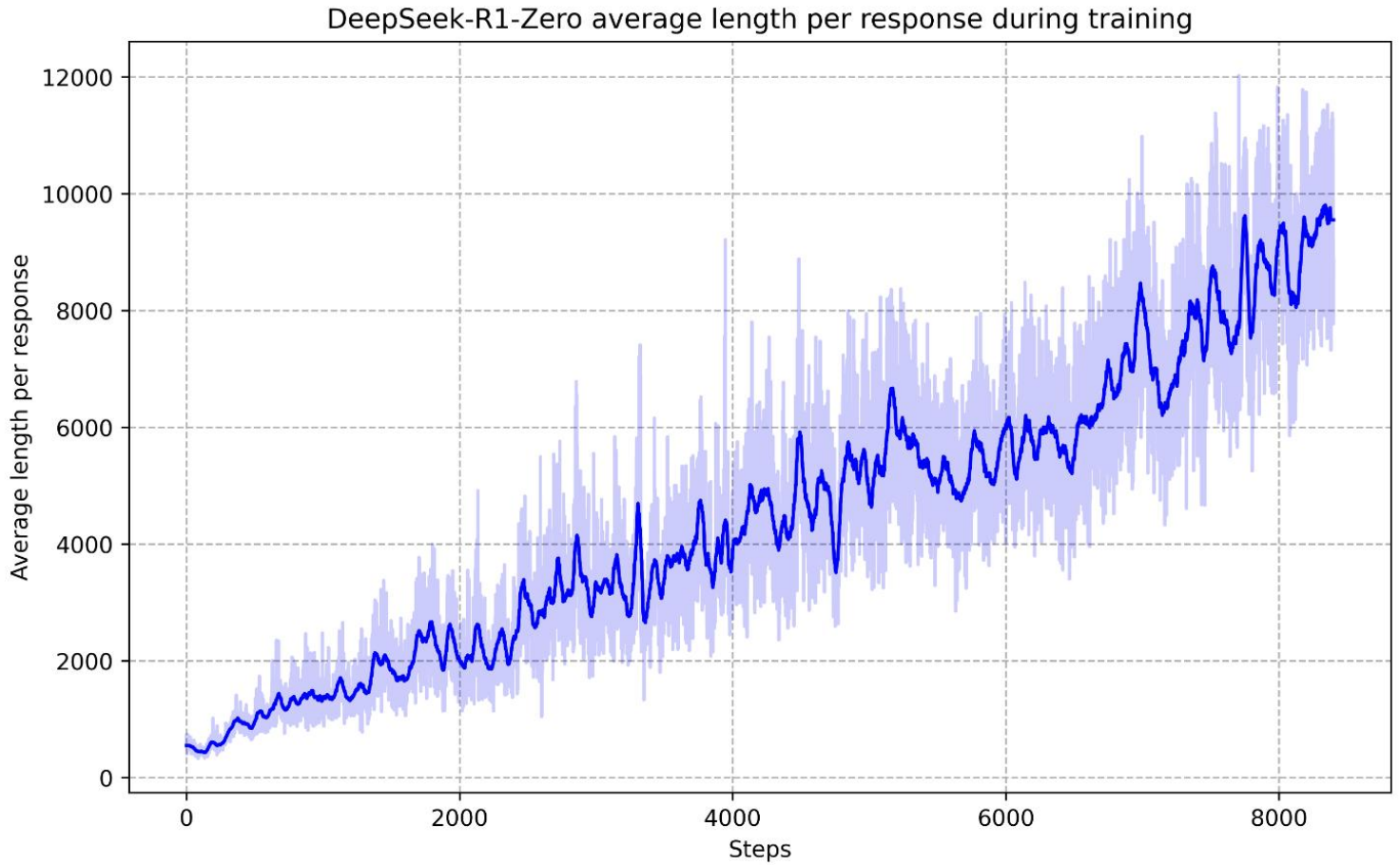
RL training - performance

- GRPO on top of math and code data



AIME accuracy of Deepseek-R1-Zero increases with the number of RL steps

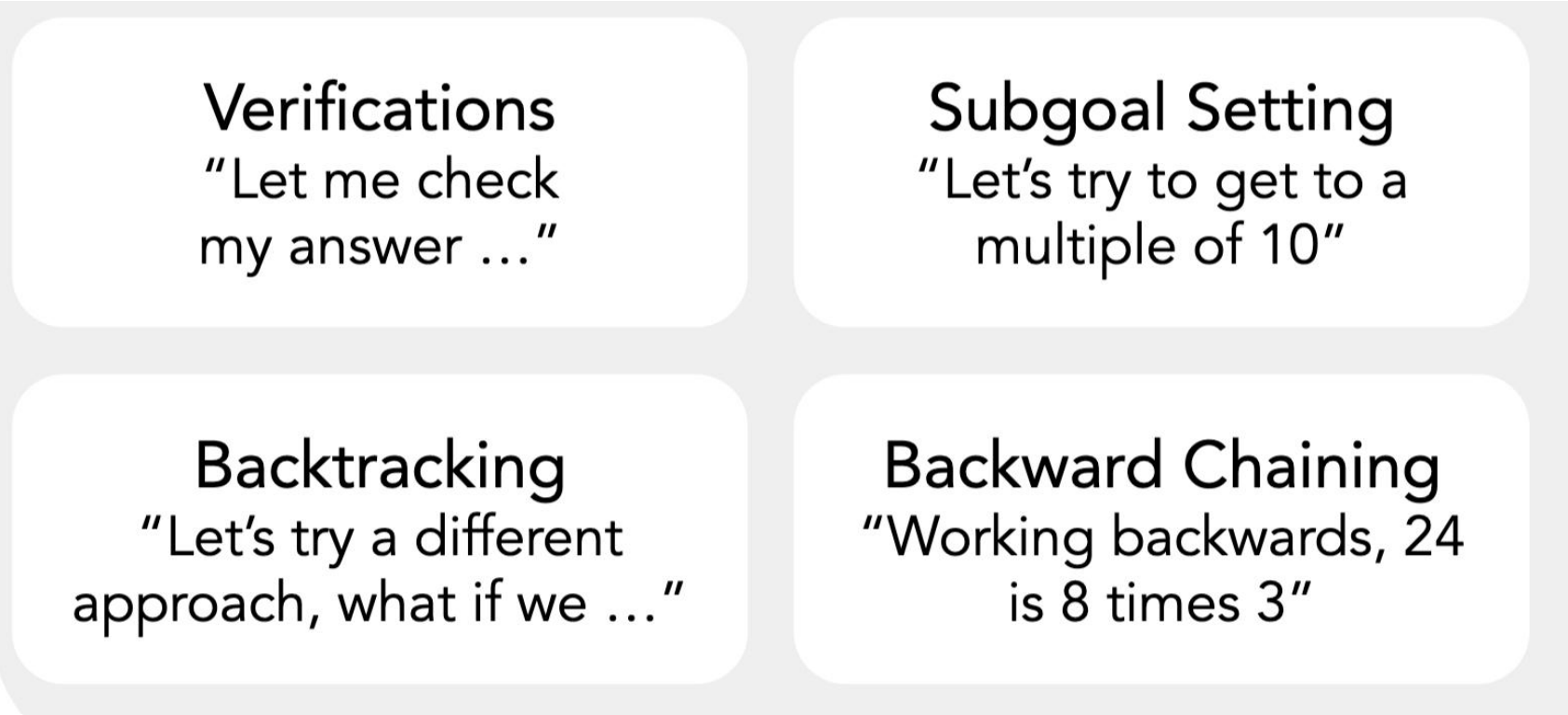
RL training – length of responses



Length of responses increases with the number of RL steps

Why do the lengths increase?

- Emergence of new reasoning patterns – the aha moment



Gandhi et al. Cognitive Behaviors that Enable Self-Improving Reasoners. arXiv 2025

An example

Problem:

James writes a 3-page letter to 2 different friends twice a week. How many pages does he write a year?

Thinking Process

<think>
Okay, let me try to figure out this problem. So, James is writing a 3-page letter to two different friends twice a week. We need to find out how many pages he writes in a year. **Hmm**, let's break this down step by step.
First, **let me understand the details**. [...]
Wait, if he writes to 2 friends twice a week, does that mean he writes 2 letters each time, and he does this twice? [...] So per week, he writes 2 friends * 3 pages * 2 times. **Let me check that.** [...]
Alternatively, maybe it's 3-page letter to 2 different friends each time he writes, and he does that twice a week. So each time he writes, he's writing two letters (each 3 pages), so that's 2 * 3 = 6 pages per writing session. Then, twice a week would be 6 * 2 = 12 pages per week. Then, multiply that by the number of weeks in a year. But how many weeks are in a year? Typically, 52 weeks. So 12 pages/week * 52 weeks = 624 pages. **Is that right?**
Wait, let me confirm. So, per week: [...] **That makes sense.**
But maybe **I need to double-check** if the problem is interpreted differently. [...] That seems consistent.
Wait, but wait a second. Is there any chance that the problem might be considering a year as 52 weeks, **but some people might account for leap years or something else?** But generally, in problems like this, unless specified, we take 52 weeks per year. So that's standard.
So, conclusion: [...] So the answer is 624. So I think that's correct.
</think>

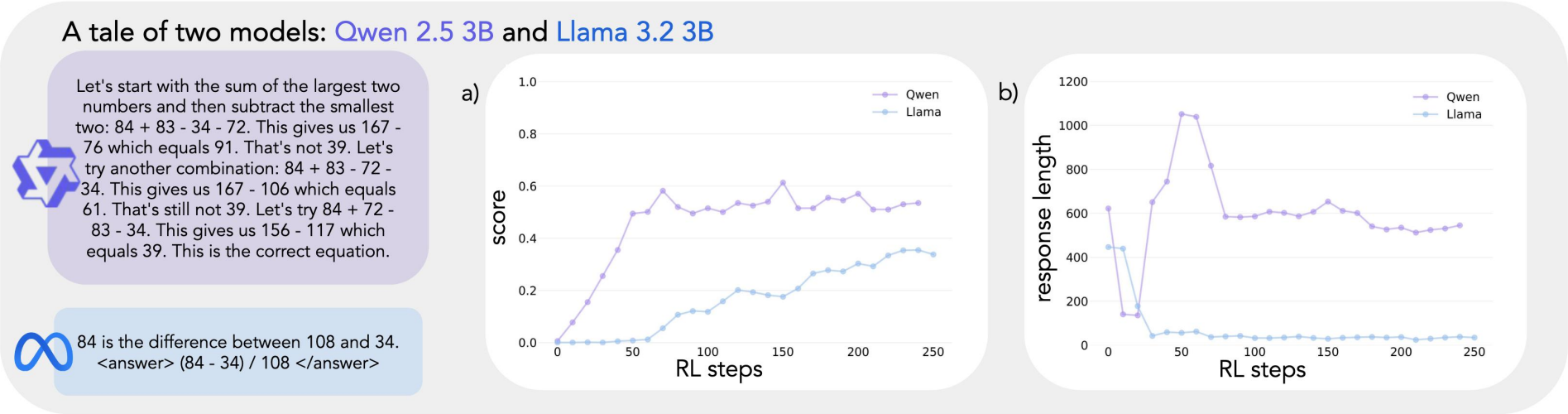
Answer

James writes 3-page letters to 2 friends twice a week. Here's the breakdown:
[...]
Answer: James writes 624 pages in a year. .

Figure from: Marjanovi´c et al. DeepSeek-R1 Thoughtolgy. arXiv 2025.

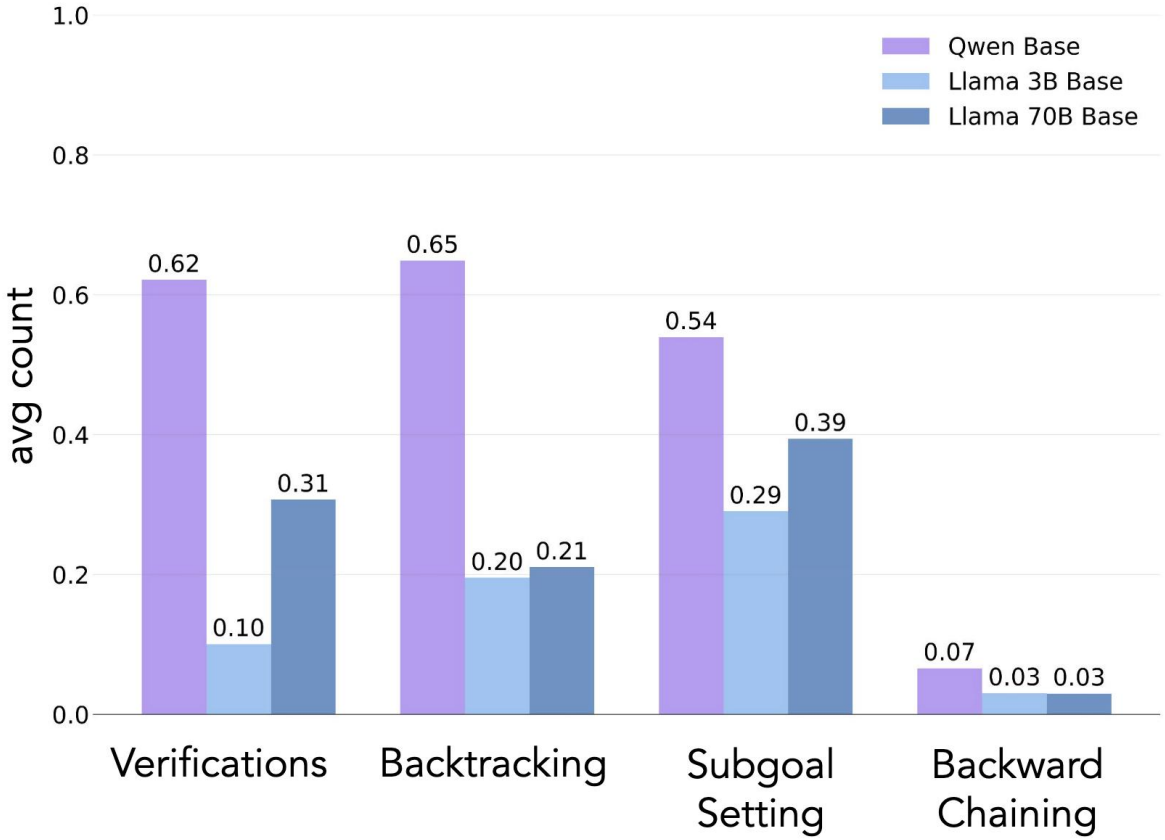
Will this work on all models?

- Short answer, no
- The model must seen have such patterns during its pretraining.
- Countdown game — a mathematical puzzle where players must combine a set of input numbers using the four basic arithmetic operations (+, −, ×, ÷) to reach a target number



Gandhi et al. Cognitive Behaviors that Enable Self-Improving Reasoners. arXiv 2025

The role of initial behavior

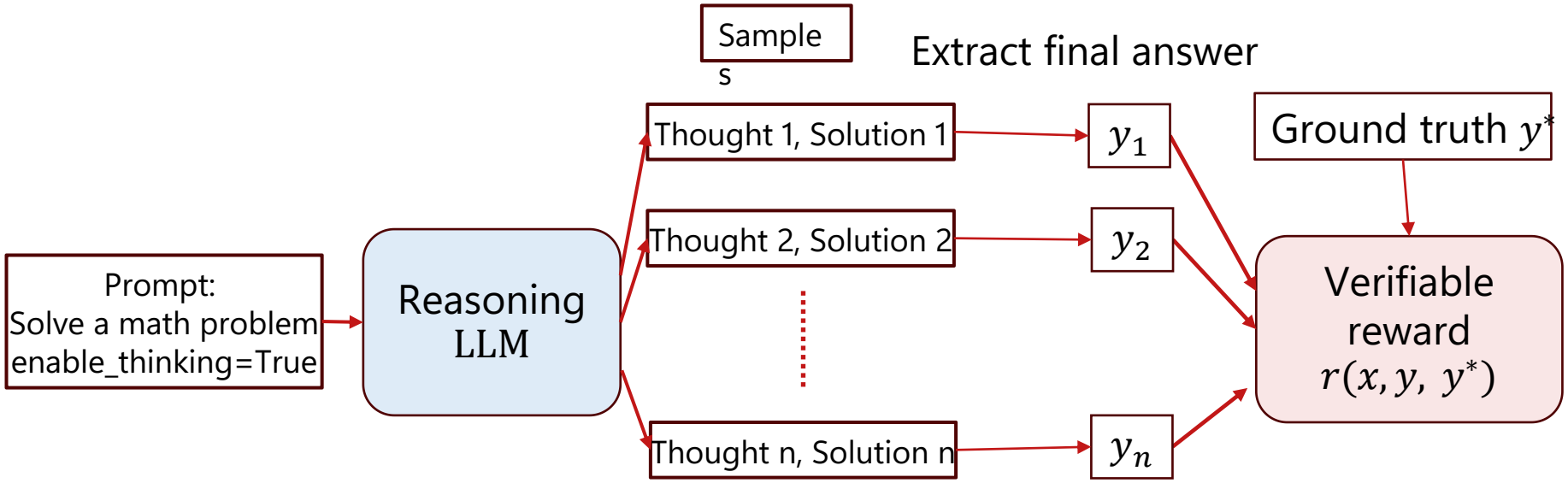


- Qwen-2.5-3B models already exhibit all the 4 behaviors at a much higher rate than Llama-70B.
- The initial policy must show the cognitive behavior for RL to exploit it.

Gandhi et al. Cognitive Behaviors that Enable Self-Improving Reasoners. arXiv 2025

Inducing reasoning patterns in weaker models

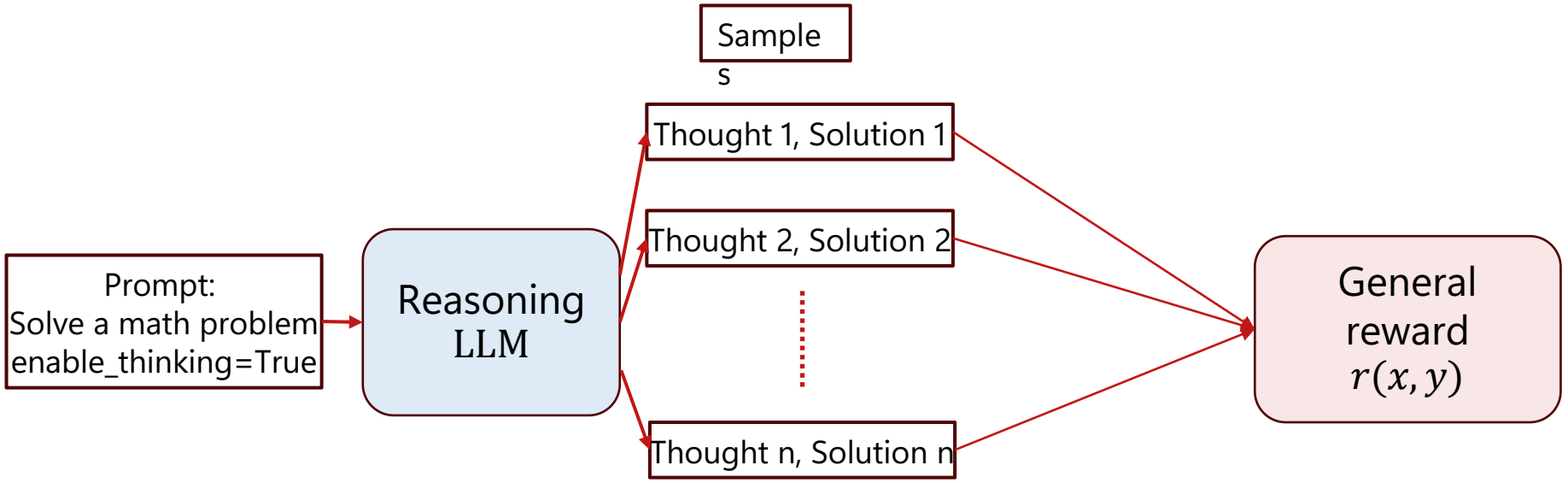
Using synthetic data from reasoning models



- Keep the (prompt, thought, solution) pairs with high reward
- Perform SFT on these triplets

Inducing reasoning patterns in weaker models

Using synthetic data from reasoning models



- Keep the (prompt, thought, solution) pairs with high reward
- Perform SFT on these triplets

Distillation vs RL for weak models

- Always prefer distillation from a powerful reasoning model.
- RL on weak models may never be able to find trajectories that have already been discovered by more powerful models

Model	AIME 2024		MATH-500	GPQA Diamond	LiveCodeBench
	pass@1	cons@64	pass@1	pass@1	pass@1
QwQ-32B-Preview	50.0	60.0	90.6	54.5	41.9
DeepSeek-R1-Zero-Qwen-32B	47.0	60.0	91.6	55.0	40.2
DeepSeek-R1-Distill-Qwen-32B	**72.6**	**83.3**	**94.3**	**62.1**	**57.2**

References & Further Reading

- **Chain-of-Thought Prompting Elicits Reasoning in LLMs** (Wei et al., 2022) — Prompt-only emergence of multi-step solutions.
- **STaR: Bootstrapping Reasoning With Rationales** (Zelikman et al., 2022) — Self-generate explanations → filter → SFT.
- **Thinking LLMs: Thought Preference Optimization (TPO)** (2025) — Optimize preferences over thoughts, not just finals.
- **DeepSeek-R1: Incentivizing Reasoning via RL** (2025) — GRPO + verifiers; emergence of planning/backtracking.
- **Cognitive Behaviors that Enable Self-Improving Reasoners** (2025) — behavior presence predicts RL self-improvement