Can we use classical probabilistic inference methods to scale small LMs to o1 level?

the promise of inference scaling

Isha Puri, MIT CSAIL

Isha Puri, Shivchander Sudalairaj, GX Xu, Kai Xu, Akash Srivastava. Rollout Roulette, …. arxiv.org/abs/2502.01618

Talk

1. Intro
2. Current inference scaling methods
3. Our method - particle-based inference scaling
4. Results
5. Why should you care?

Why does inference scaling matter?

why inference scaling?

Unlocking hidden capabilities of LLMs, improving quality & reliability — *without retraining*

bridging the gap to larger, more powerful models

why inference scaling?

We all know that closed-source, frontier models like GPT-4o and Claude 3.5 Sonnet are fantastic at a variety of tasks.

why inference scaling? (Part 1)

We all know that closed-source, frontier models like GPT-4o and Claude 3.5 Sonnet are fantastic at a variety of tasks.

<u>Privacy Concerns:</u>

- hidden away behind an API wall, requiring users to send data to external entities
- problem for entities such as healthcare, finance, & enterprises with sensitive data

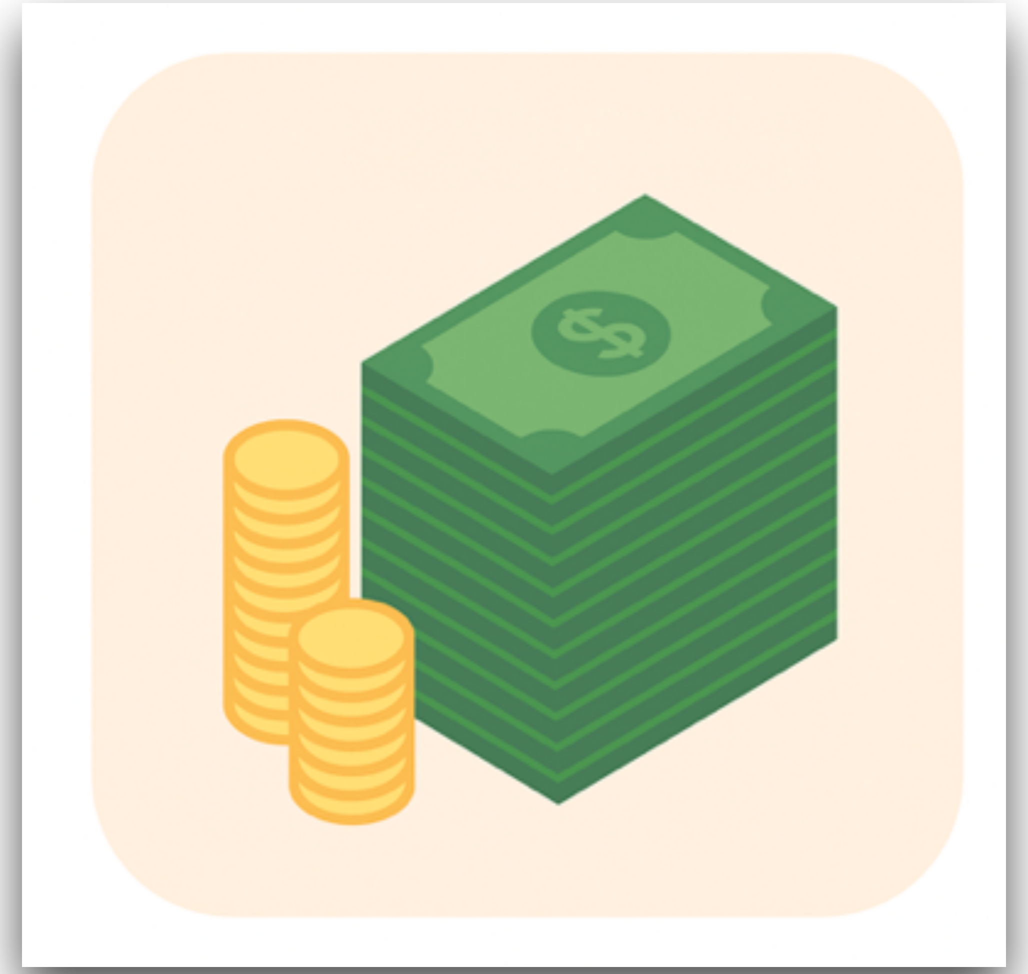
why inference scaling? (Part I)

We all know that closed-source, frontier models like GPT-4o and Claude 3.5 Sonnet are fantastic at a variety of tasks.

Cost Concerns:

- they are exceedingly expensive to run: both:

energy wise
monetarily

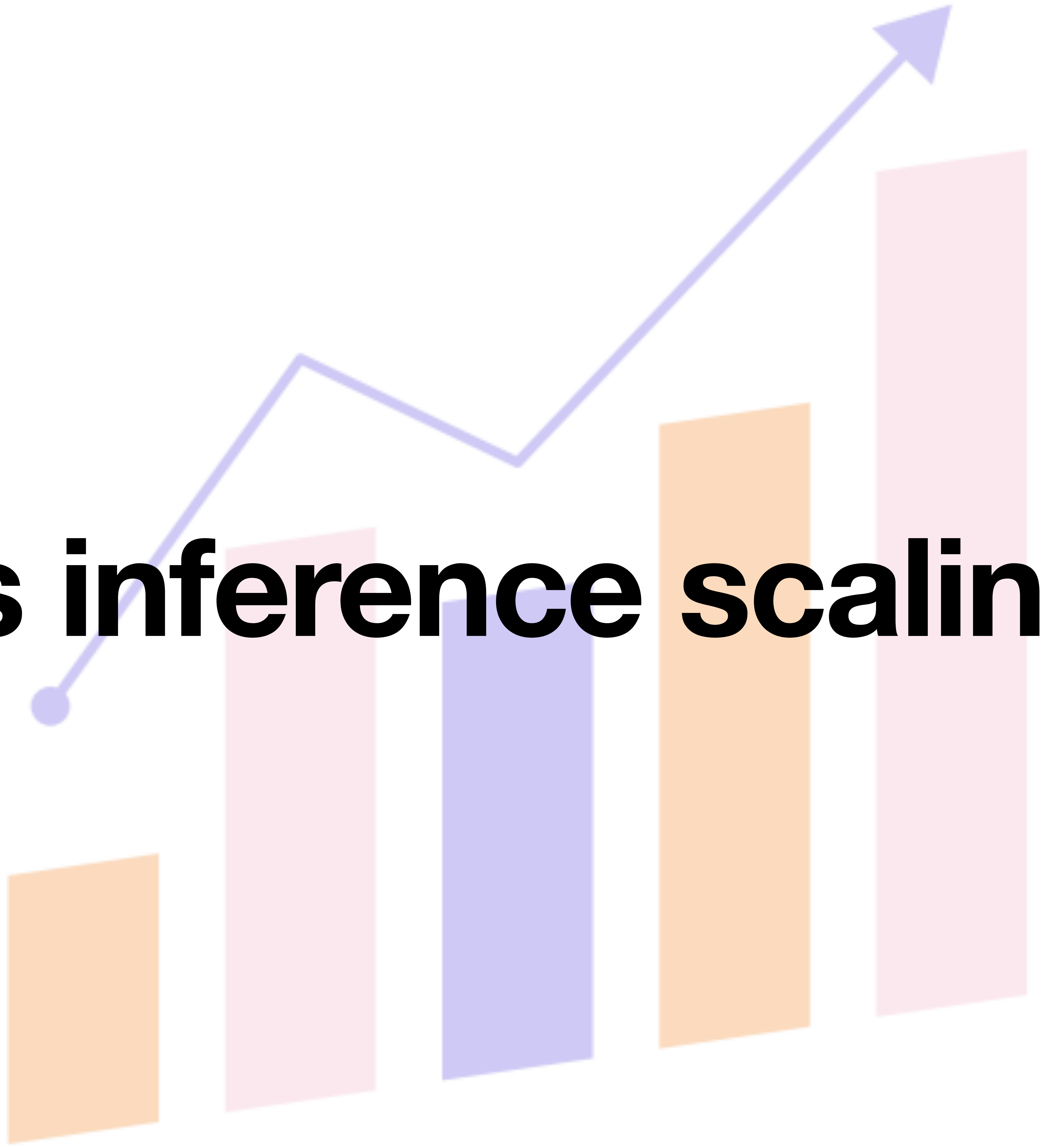


	Model	Provider	Input $/1M	Output $/1M
closed source models	claude-3-opus	Anthropic	$15	$75
	gpt-4o	OpenAI	$5	$15
	gemini-1.5-pro	Google	$3.5	$10.5
open source models	llama-3.1-8b-instruct	Deepinfra	$0.09	$0.09
	llama-3.1-70b-instruct	Deepinfra	$0.52	$0.75
	mixtral-8x7b	Mistral	$0.7	$0.7

Prices from Feb 2025.

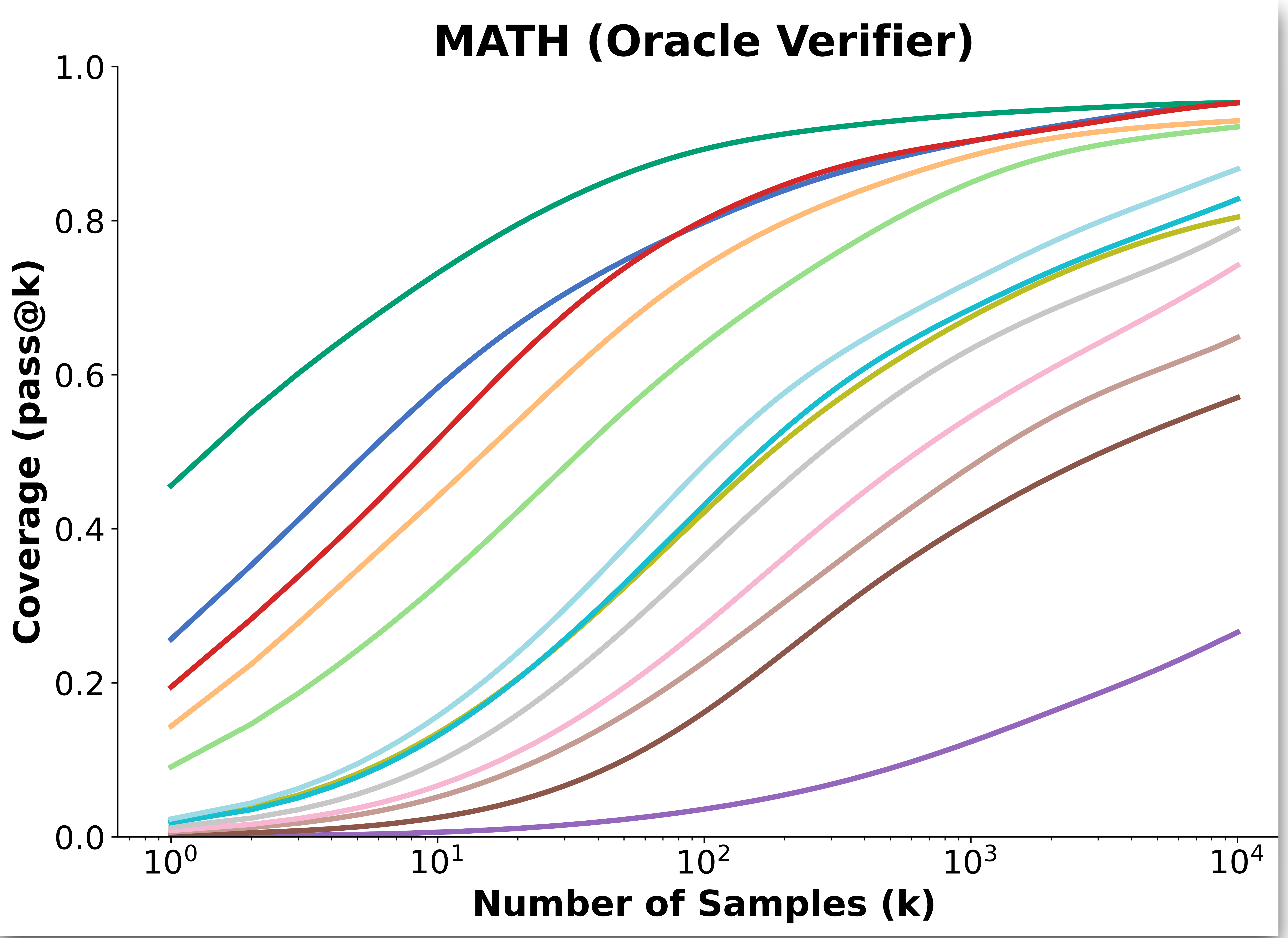
Source: llmpricecheck.com/

What is inference scaling?



What is inference scaling?

Now, studies have shown that smaller, much cheaper, open models - even those as small as 1B models - when queried several times, will often eventually correctly answer challenging reasoning questions.



(from Large Language Monkeys: Scaling Inference Compute with Repeated Sampling, Brown et al., 2024) so… if we know that small language models have it in them to answer difficult questions,

we just have to find a way to squeeze it out of them!

Our problem then becomes:

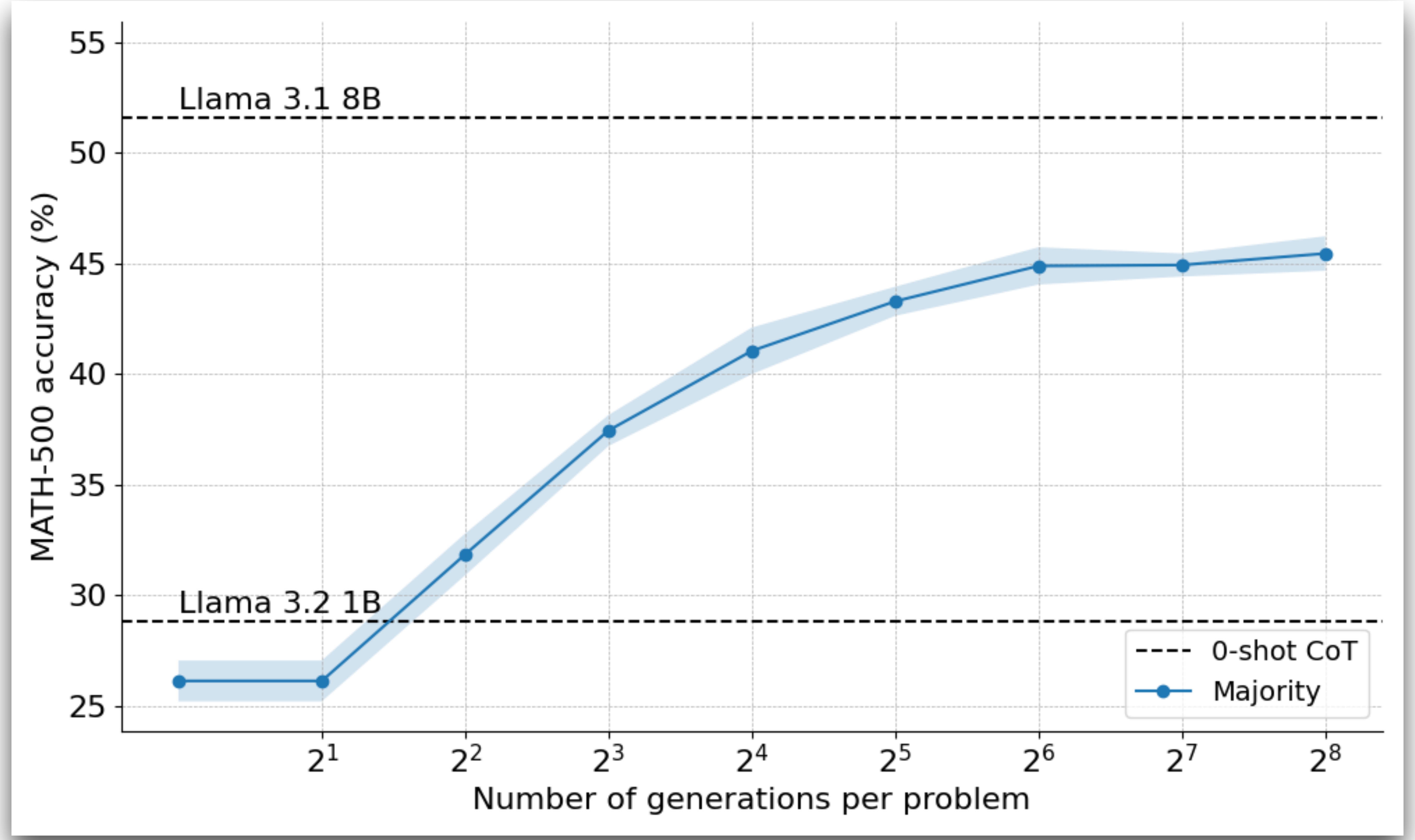
how can we intelligently navigate the search space of these smaller models to find the best answer they can provide?

Therein lies the problem and promise of inference-time scaling.

Background in some current inference scaling methods

Simple methods - majority voting

In majority voting, I ask the LM the question N times and pick the most common answer

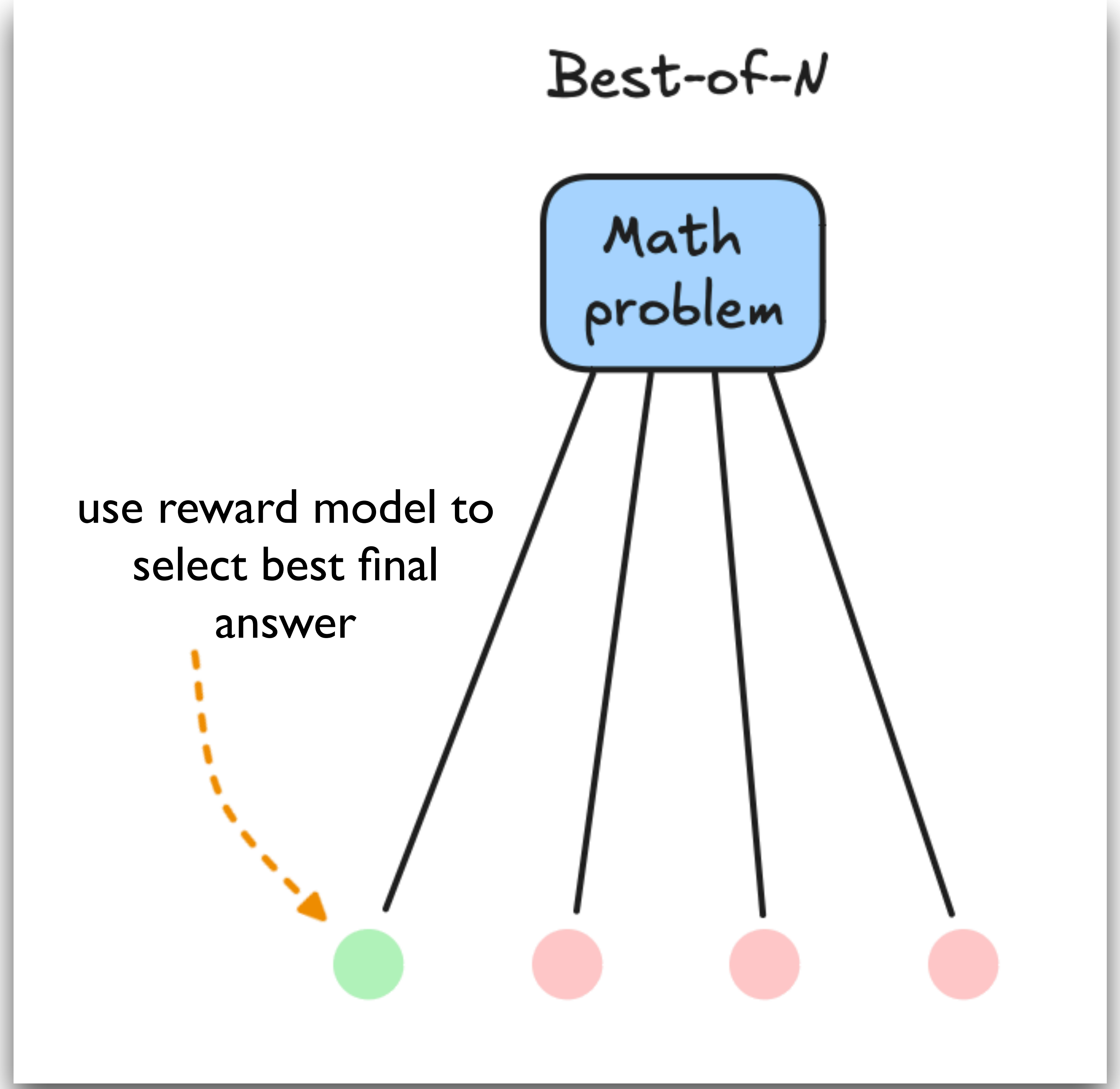


from HuggingFace, scaling test time compute.

Simple methods - Best of N

In best of N sampling, I ask a language model a question N times.

- gather up all N independent answers
- ask the reward model to score all of them
- choose the answer with the highest reward score as my final answer.

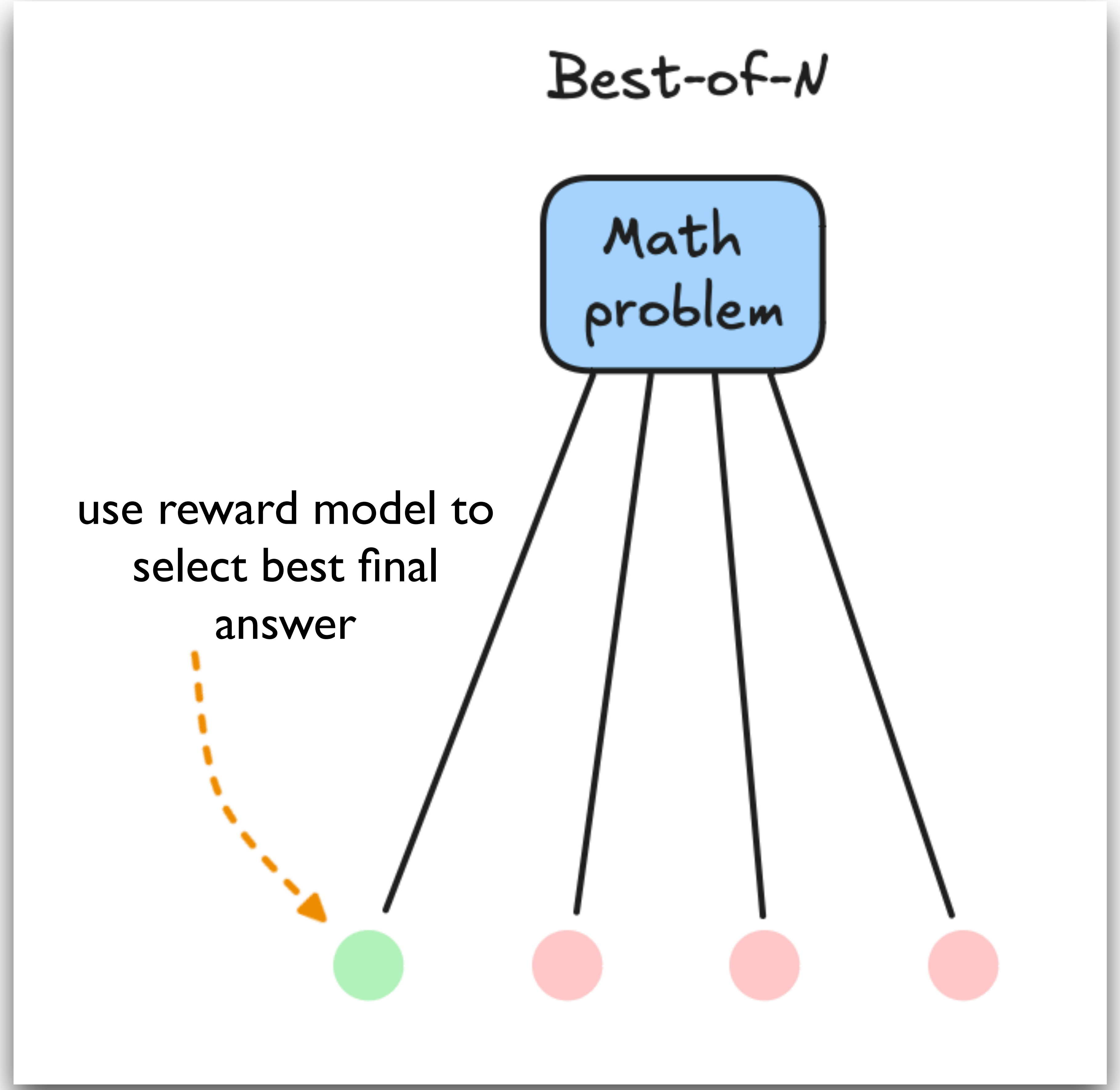


from HuggingFace, scaling test time compute.

Simple methods - Weighted Best of N

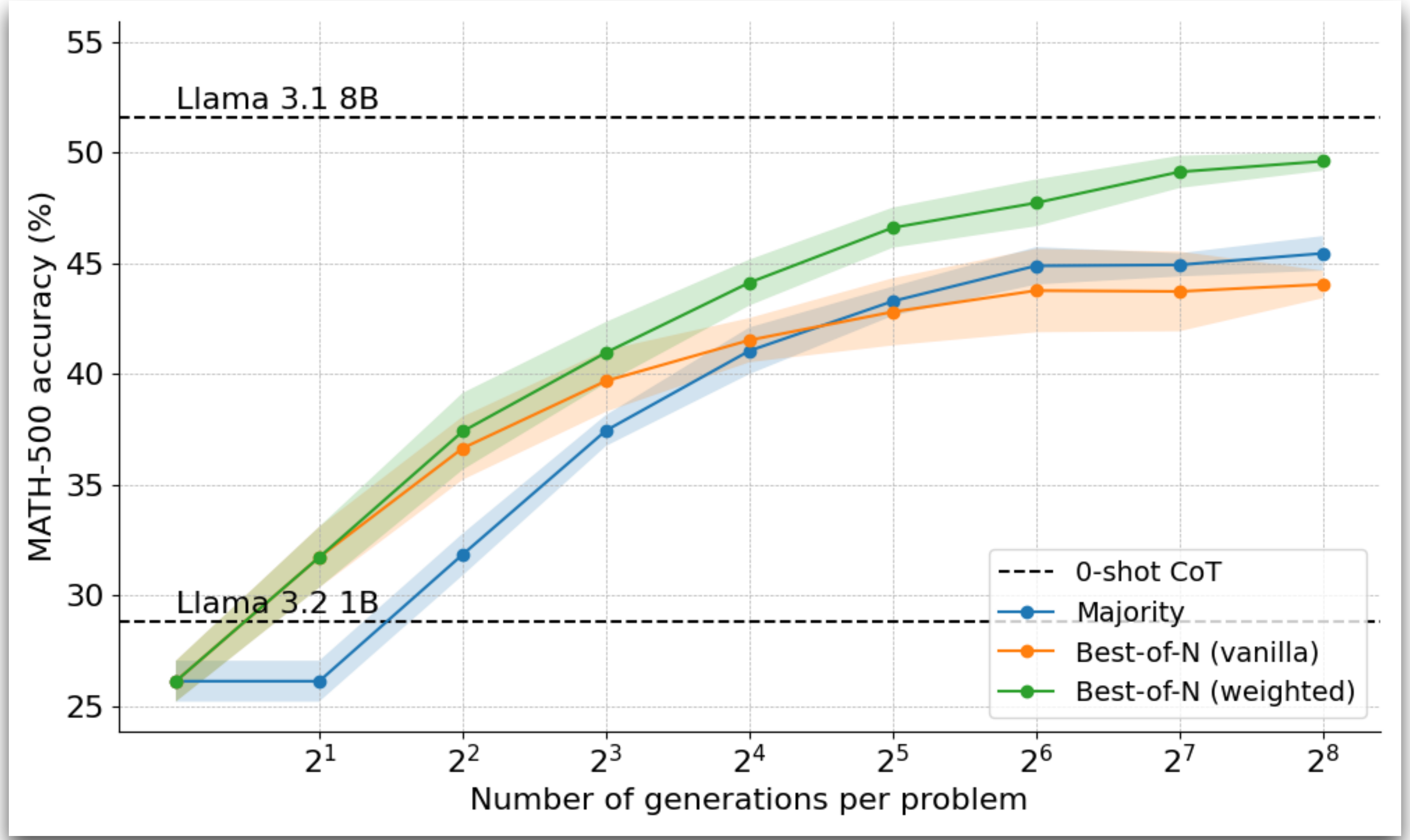
In Weighted Best-of-N sampling, I also consider how many times an answer is generated.

- ask a language model a question N times
- gather up all N independent answers
- ask the reward model to score all of them
- multiply how many times each answer shows up with the reward score



from HuggingFace, scaling test time compute.

Simple methods - Weighted Best of N

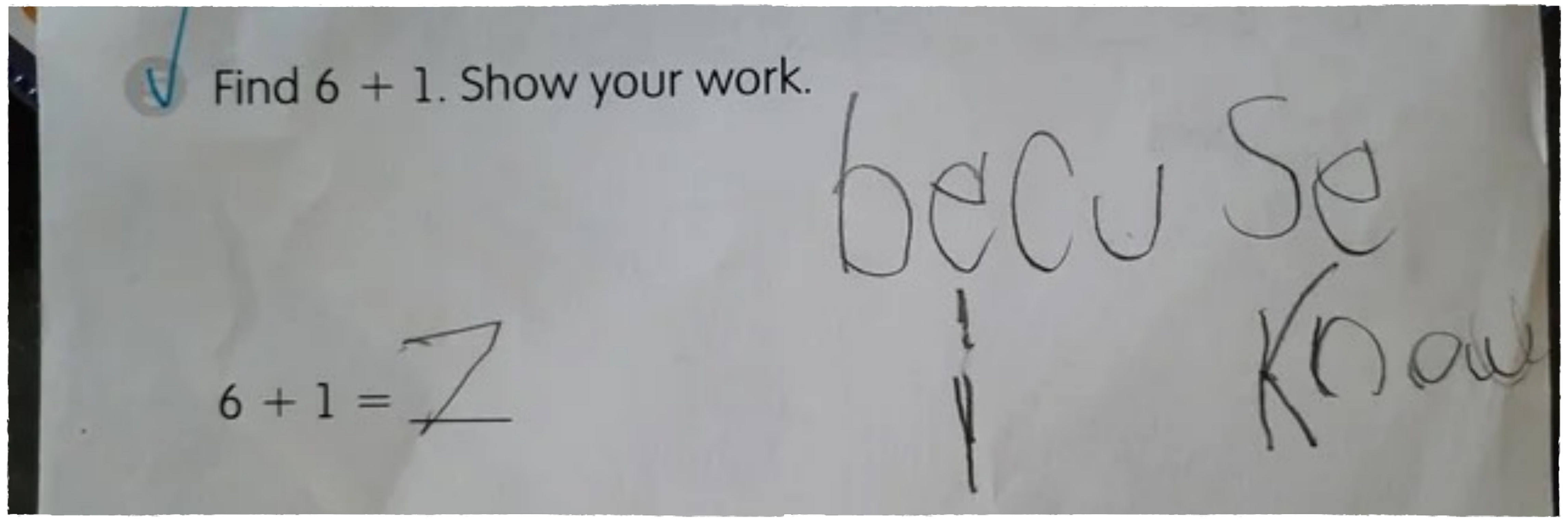


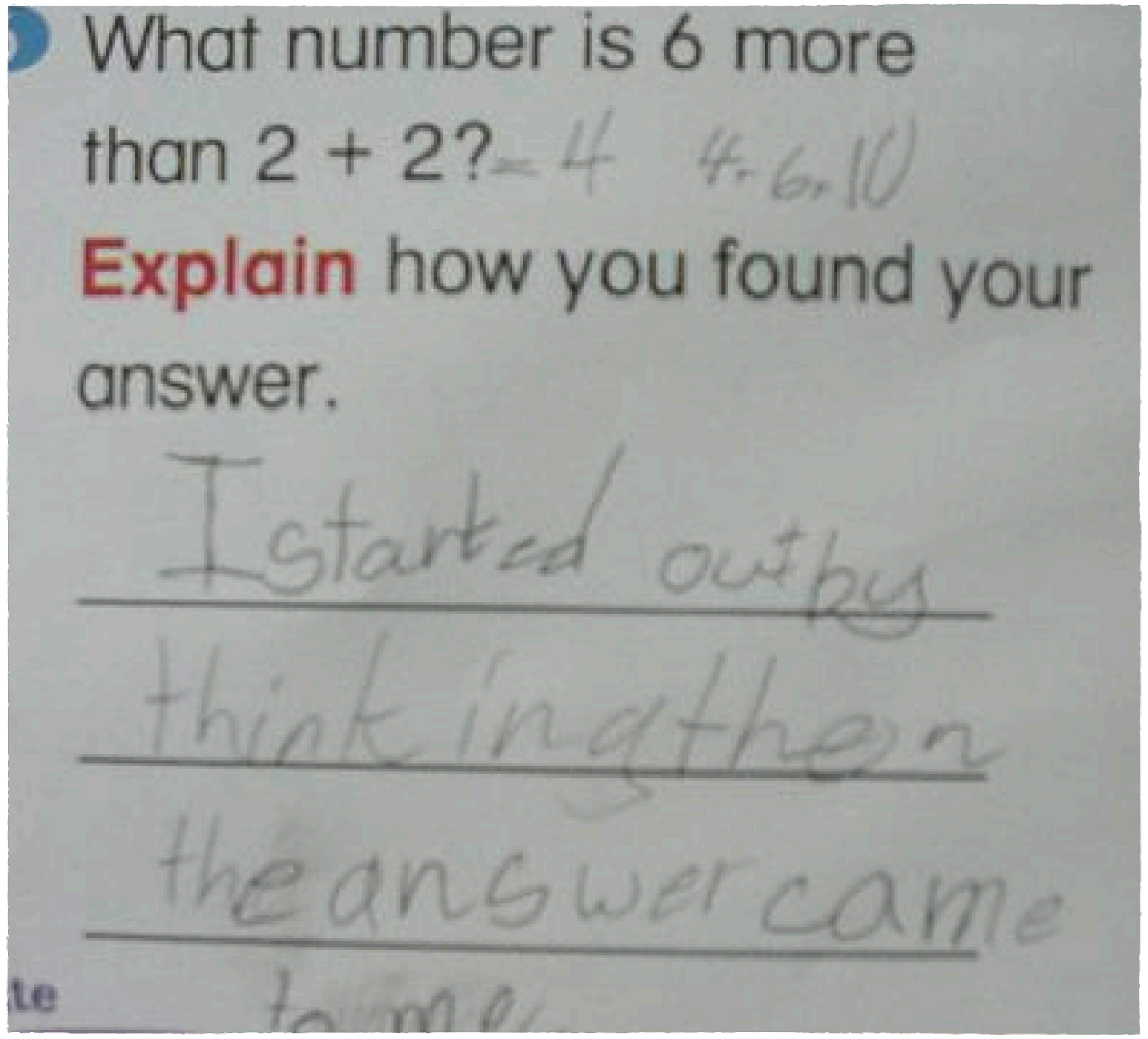
from HuggingFace, scaling test time compute.

Process Reward Models (PRM)

the process of showing one's work is just as important as the final answer!

so we want something to **judge a LM's reasoning trajectory**





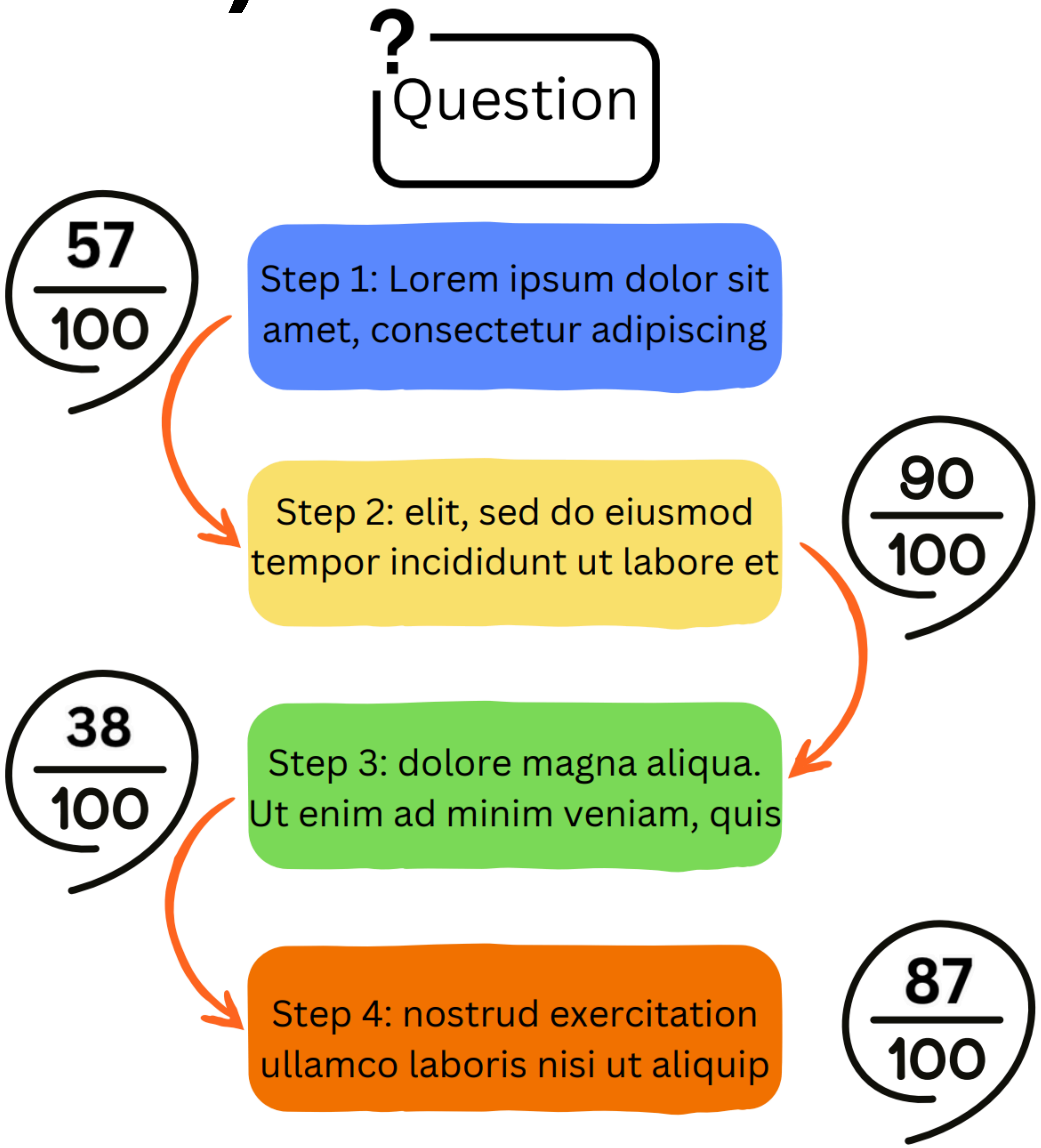
Process Reward Models (PRM)

a process reward model is a LM
that is specially trained to take in

(1) a question

(2) a *partial* answer

and return: a score!

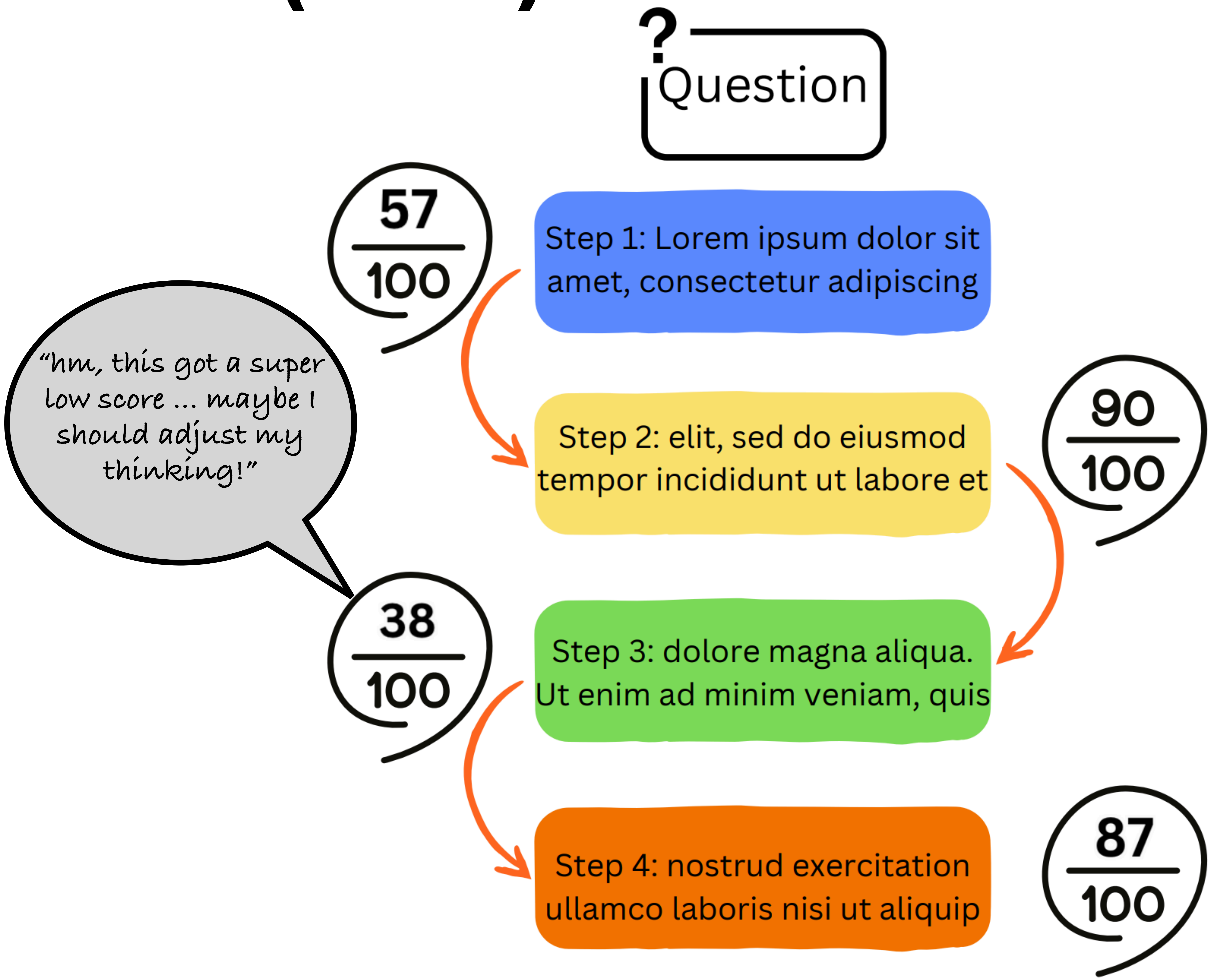


Process Reward Models (PRM)

now that we have a PRM,

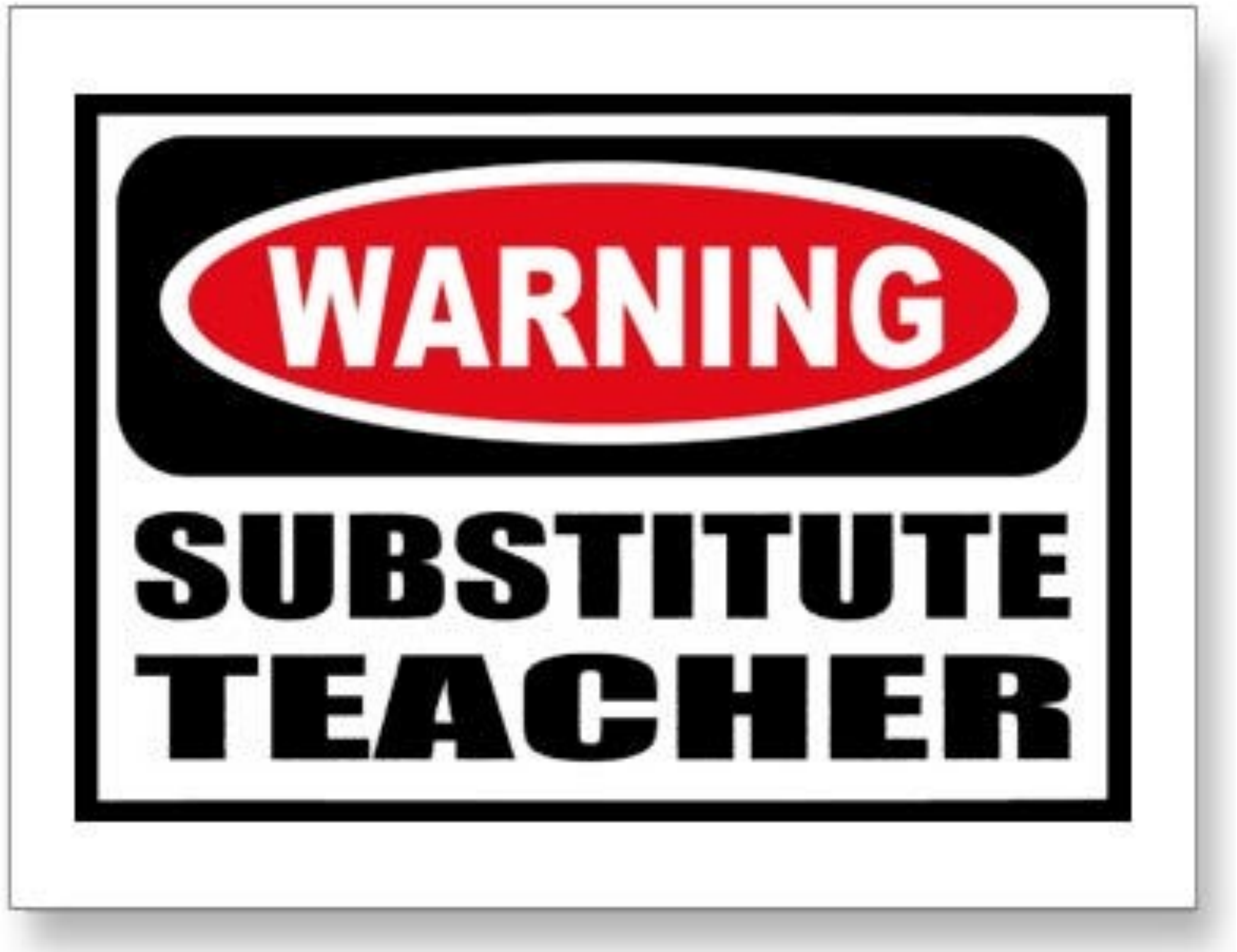
we can use it to "guide" the search to the best possible answer by scoring our partial answers *as we generate them* and adjust according to the scores we see!

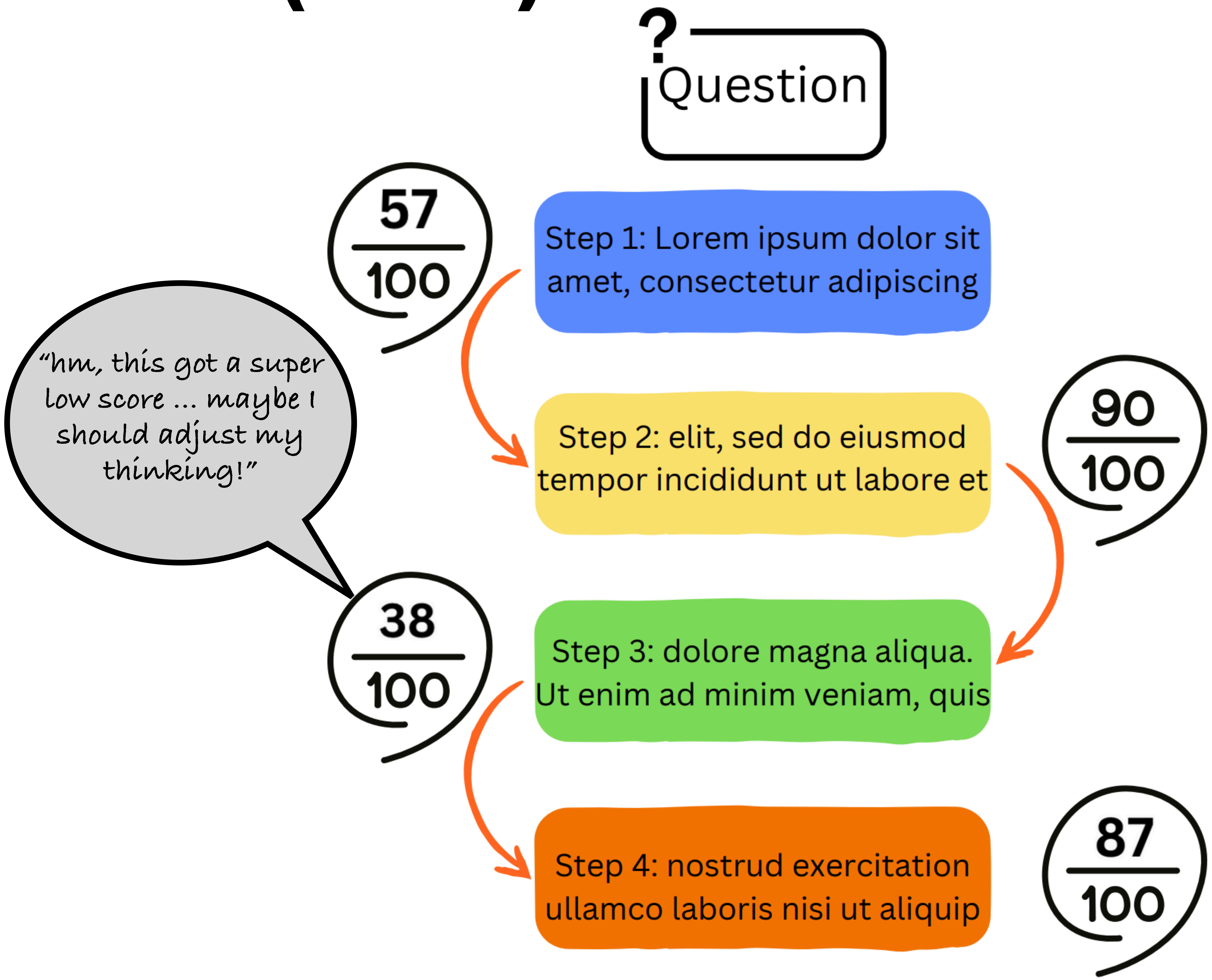
it's like getting live feedback as we reason.



Process Reward Models (PRM)

but… let's remember! the PRM is just a model - it's not an exact scoring mechanism!



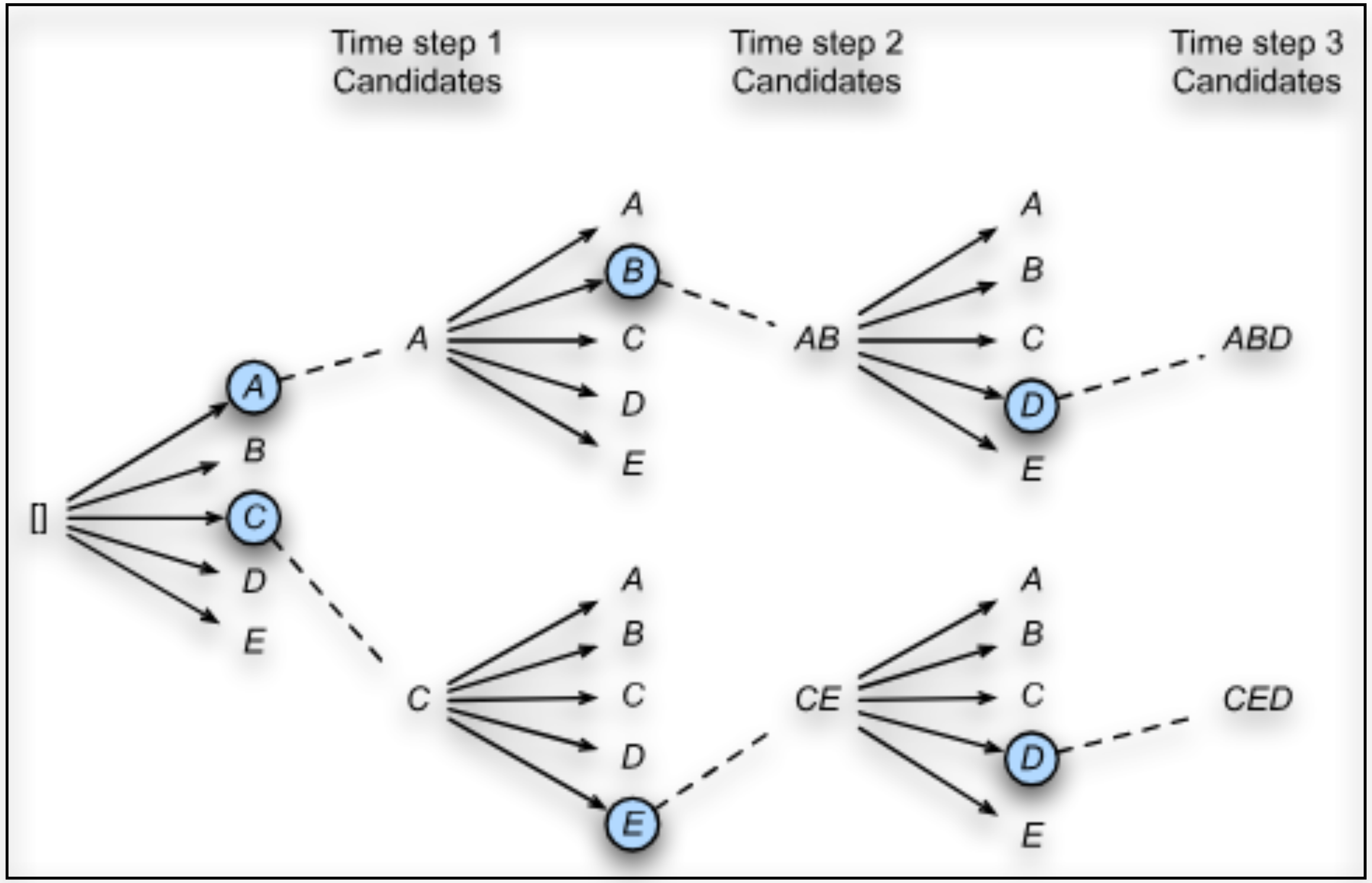


Beam Search

this is where previous methods, like beam search, falter a bit!

many current inference scaling methods automatically select only the top N ranked partial responses at every step.

<u>they rely completely on the PRM to determine what is right or not.</u>

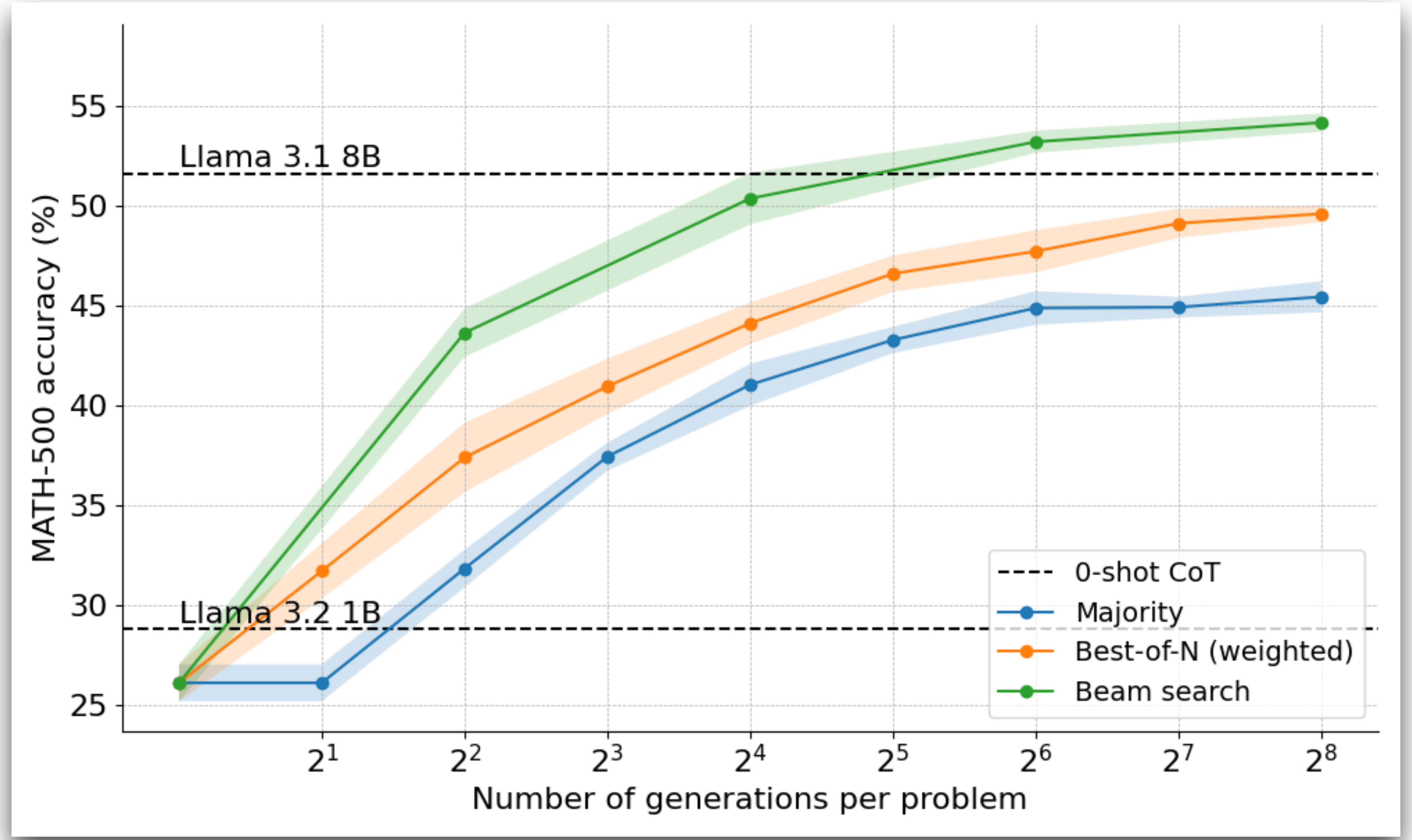


Beam Search

this is where previous methods, like beam search, falter a bit!

many current inference scaling methods automatically select only the top N ranked partial responses at every step.

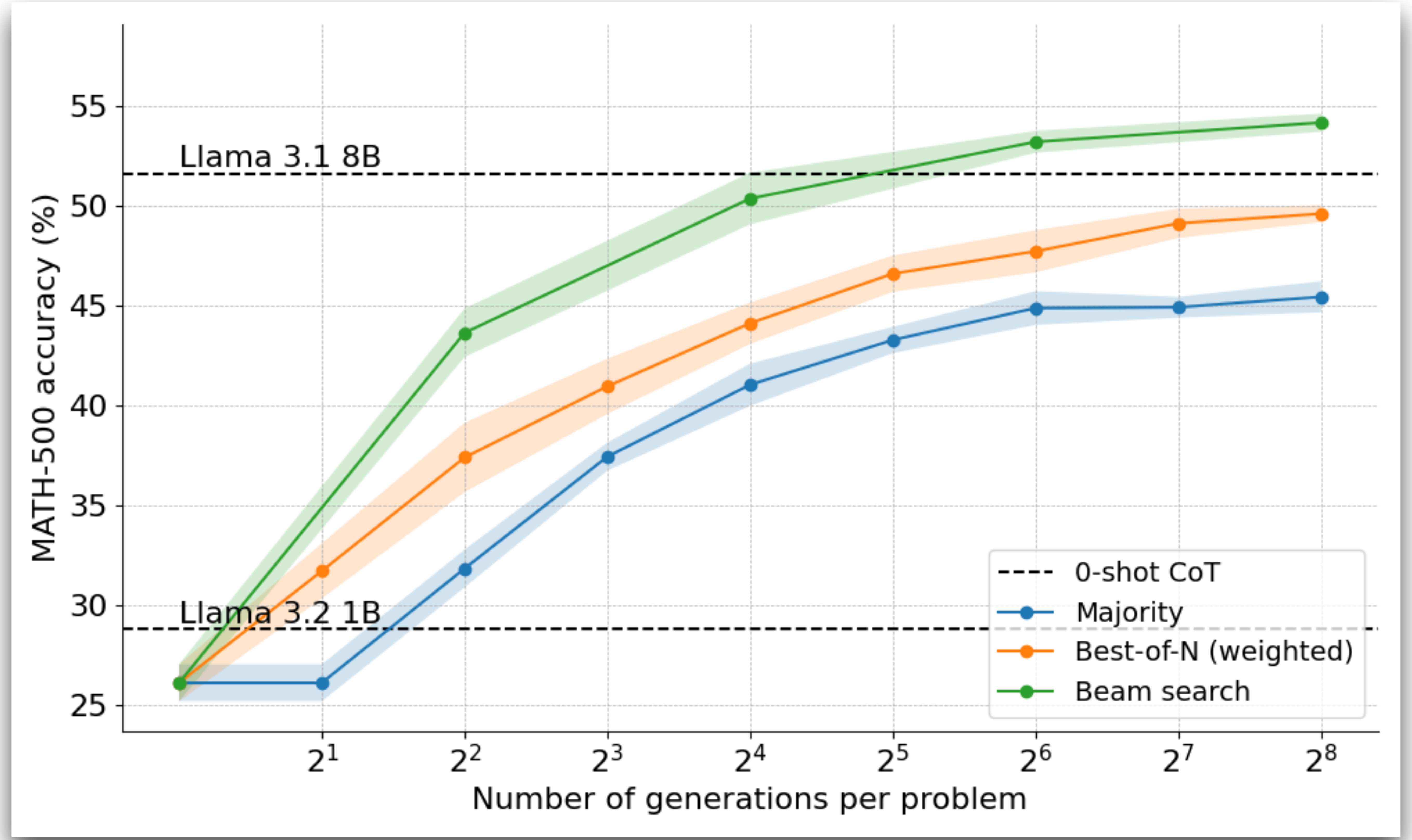
<u>they rely completely on the PRM to determine what is right or not.</u>



from HuggingFace, scaling test time compute.

Beam Search

Many search methods rely completely on the PRM to determine what is right or not.

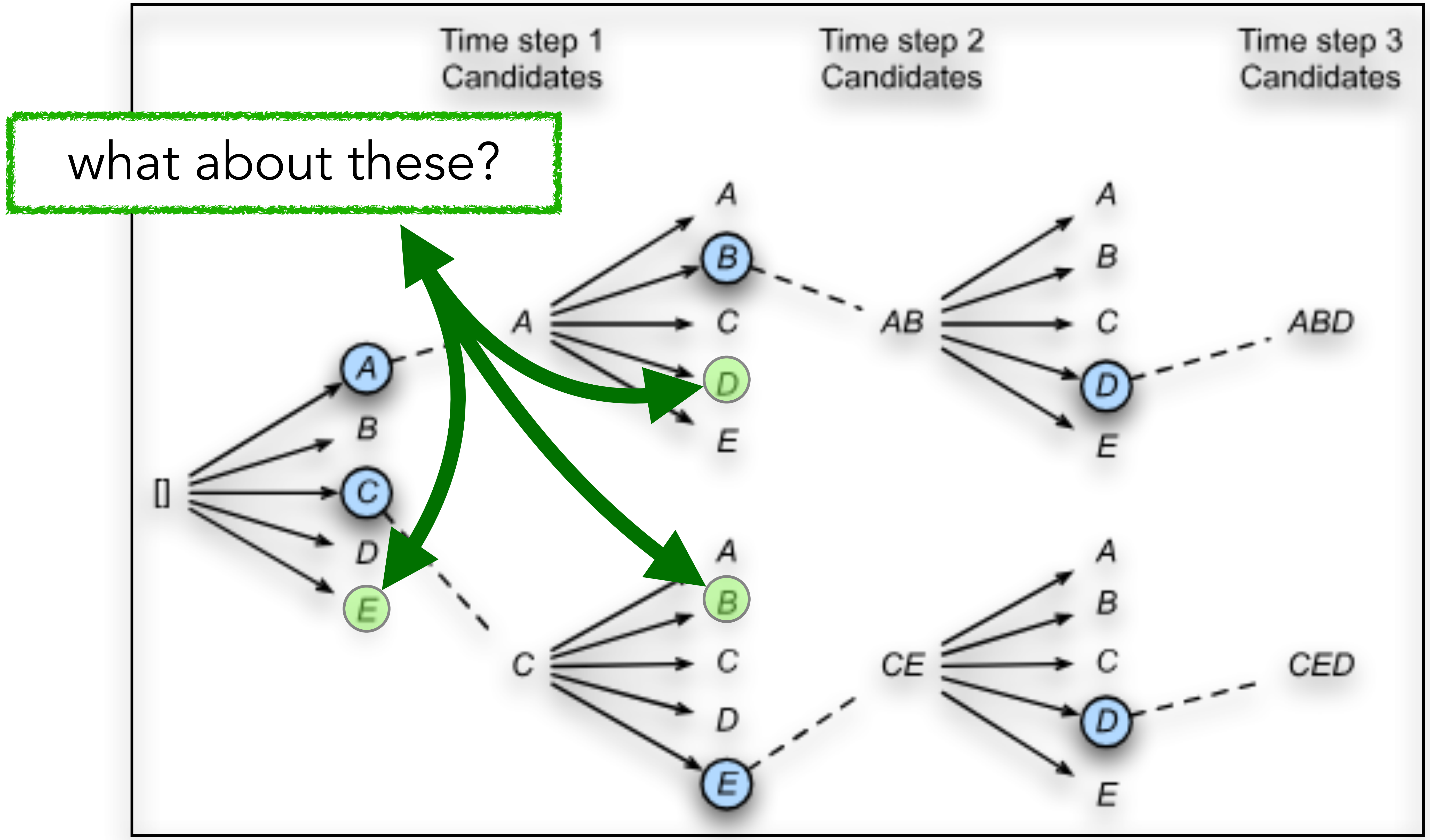


from HuggingFace, scaling test time compute.

Previous Methods

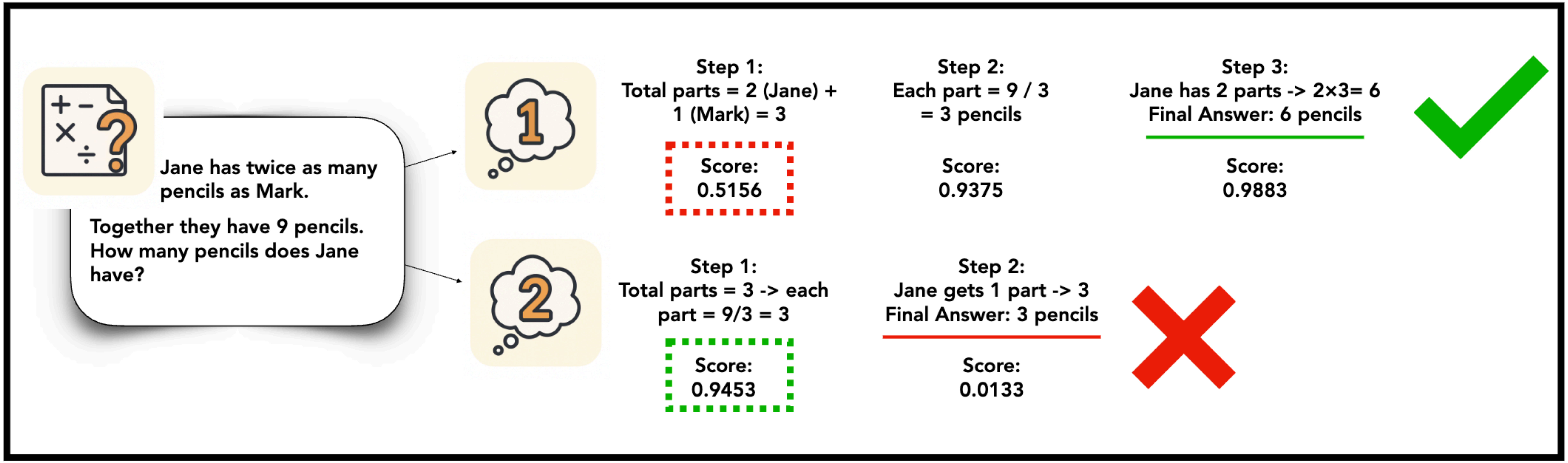
following our PRM **blindly** to determine which partial answers we want to continue expanding during our reasoning process can lead to **reward hacking**

where the final output is optimized to score well according to the reward model but fails to be useful and/or correct



from HuggingFace, scaling test time compute.

Early Pruning



A Probabilistic Inference Approach to Inference-Time Scaling of LLMs using Particle-Based Monte Carlo Methods

Isha Puri, Shivchander Sudalairaj, GX Xu, Kai Xu, Akash Srivastava

Particle Filtering
for
Inference Scaling

Isha Puri, Shivchander Sudalairaj, GX Xu, Kai Xu, Akash Srivastava

Formalism

Algorithm 1 Particle Filtering for Inference-Time Scaling

Input: the number of particles N, a reward model $\hat{r}$, a LLM p_M and the prompt c

Initialize N particles $\{x_1^{(i)} \sim p_M(\cdot \mid c)\}_{i=1}^N$

$t \leftarrow 1$

while not all particles stop **do**

  Update rewards $\mathbf{w} = [\hat{r}(x_{1:t}^{(1)}), \ldots, \hat{r}(x_{1:t}^{(N)})]$

  Compute softmax distribution $\theta = \mathrm{softmax}(\mathbf{w})$

  Sample indices $\{j_t^{(i)}\}_{i=1}^N \sim \mathbb{P}_t(j = i) = \theta_i$

  Update the set of particles as $\{x_{1:t}^{(j_t^{(i)})}\}_{i=1}^N$

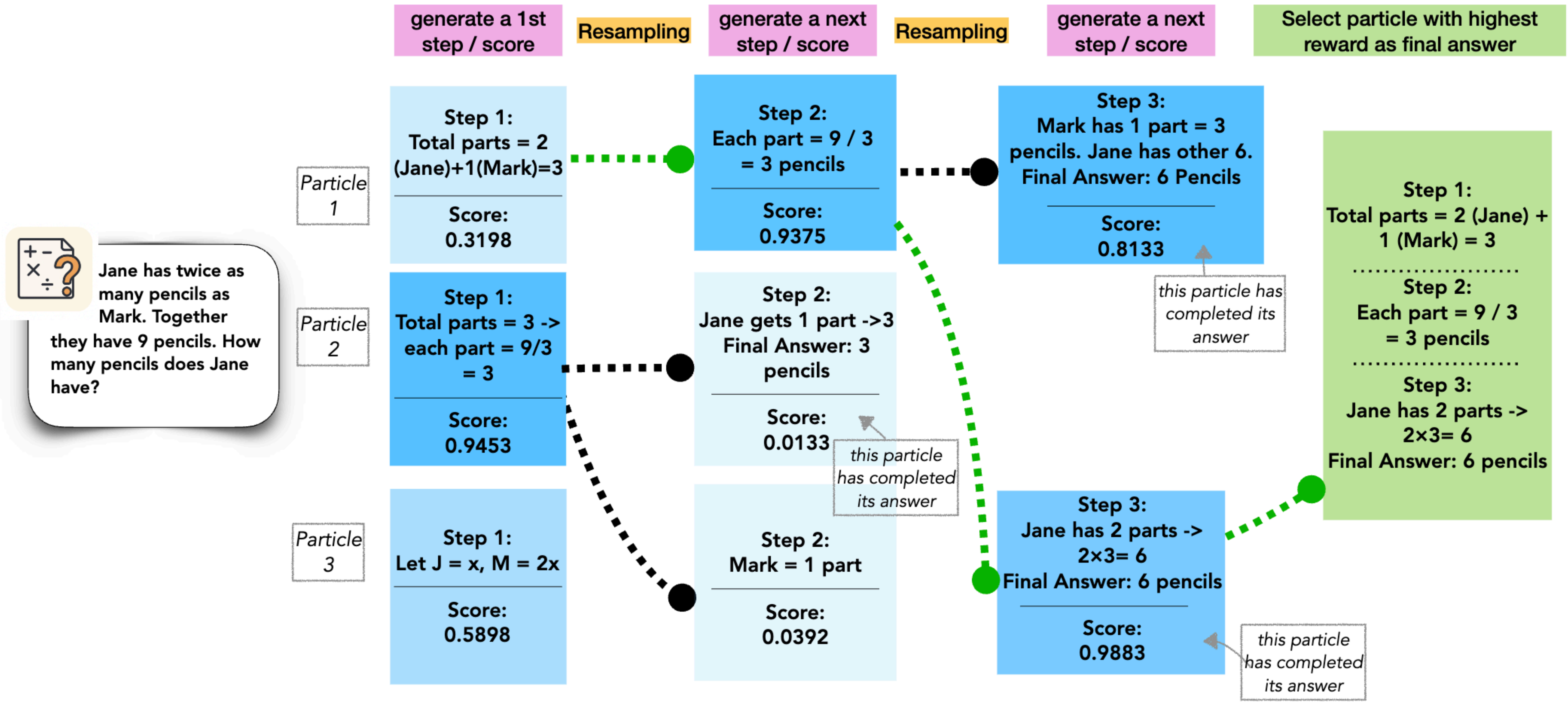
  Transition $\{x_{t+1}^{(i)} \sim p_M(\cdot \mid c, x_{1:t}^{(i)})\}_{i=1}^N$

  $t \leftarrow t + 1$

end while

Return: the set of particles in the end

PARTICLE FILTERING FOR INFERENCE SCALING



<u>Problem:</u>

A Senate committee has 8 Republicans and 6 Democrats. In how many ways can we form a subcommittee of 5 members that has at least one member from each party?

initialize N particles

Problem:

A Senate committee has 8 Republicans and 6 Democrats. In how many ways can we form a subcommittee of 5 members that has at least one member from each party?

p_1

p_2

p_3

p_4

Problem:

A Senate committee has 8 Republicans and 6 Democrats. In how many ways can we form a subcommittee of 5 members that has at least one member from each party?

p_1

p_2

p_3

p_4

- **each particle "takes a step", where a "step" is the text the LLM generates until it hits the "\n\n" Delimiter.**
- **Each particle will have a slightly different "first step because we set a high model temperature of 0.8. Temperature basically controls how "creative"/"random" a model's generations are.**
- **You can think of each particle like a new person involved in collaboration! Everyone will have slightly different ideas and will bring something different to the table.**

Problem:

A Senate committee has 8 Republicans and 6 Democrats. In how many ways can we form a subcommittee of 5 members that has at least one member from each party?

p_1

Step 1: Use casework: Count the number of ways to form subcommittees with at least one Democrat by considering (1D,4R), (2D,3R), (3D,2R), (4D,1R), and (5D,0R), then sum these cases

p_2

Step 1: Use permutations instead of combinations to count the ways to choose a subcommittee of 5 from 14.

p_3

Step 1: Assume each senator is equally likely to be chosen and compute the probability of selecting at least one Democrat by considering individual selections.

p_4

Step 1: Calculate the total number of ways to choose 5 members from the entire committee without any restrictions\nWe can use combinations to calculate this. The total number of members is 14 (8 Republicans + 6 Democrats), so the total number of ways to choose 5 members is given by C(14,5) = 14! / (5! * (14-5)!) = 14! / (5! * 9!) = 2002.

Problem:

A Senate committee has 8 Republicans and 6 Democrats. In how many ways can we form a subcommittee of 5 members that has at least one member from each party?

p_1

Step 1: Use casework: Count the number of ways to form subcommittees with at least one Democrat by considering (1D,4R), (2D,3R), (3D,2R), (4D,1R), and (5D,0R), then sum these cases

0.8419

PRM = an off-the-shelf "process reward model"

PRM

assigns scores to each question and partial answer

p_2

Step 1: Use permutations instead of combinations to count the ways to choose a subcommittee of 5 from 14.

0.3125

p_3

Step 1: Assume each senator is equally likely to be chosen and compute the probabil of selecting at least one Democrat by considering individual selections.

0.2724

p_4

Step 1: Calculate the total number of ways to choose 5 members from the entire committee without any restrictions\nWe can use combinations to calculate this. The total number of members is 14 (8 Republicans + 6 Democrats), so the total number of ways to choose 5 members is given by C(14,5) = 14! / (5! * (14-5)!) = 14! / (5! * 9!) = 2002.

0.9483

Problem:

A Senate committee has 8 Republicans and 6 Democrats. In how many ways can we form a subcommittee of 5 members that has at least one member from each party?

p_1 ## Step 1: Use casework: Count the number of ways to form subcommittees with at least one Democrat by considering (1D,4R), (2D,3R), (3D,2R), (4D,1R), and (5D,0R), then sum these cases

0.306

p_2 ## Step 1: Use permutations instead of combinations to count the ways to subcommittee of 5 from 14.

0.180

softmax the reward scores to get probabilities. each probability represents the chance that that particle will be evolved in the next step of this process!

p_3 ## Step 1: Assume each senator is equally likely to be chosen and compute of selecting at least one Democrat by considering individual selections.

0.173

p_4 ## Step 1: Calculate the total number of ways to choose 5 members from the entire committee without any restrictions\nWe can use combinations to calculate this. The total number of members is 14 (8 Republicans + 6 Democrats), so the total number of ways to choose 5 members is given by C(14,5) = 14! / (5! * (14-5)!) = 14! / (5! * 9!) = 2002.

0.340

Problem:

A Senate committee has 8 Republicans and 6 Democrats. In how many ways can we form a subcommittee of 5 members that has at least one member from each party?

p_1

p_2

p_3

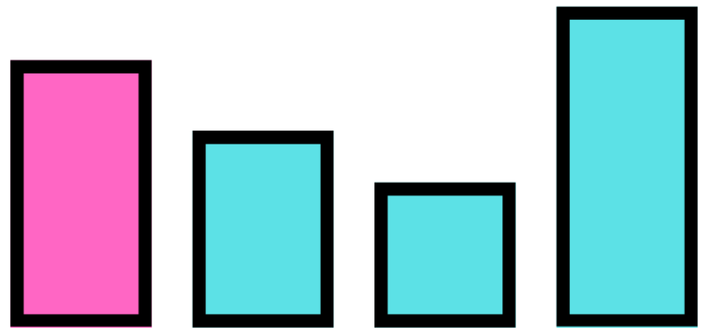
p_4

“resampling”

Problem:

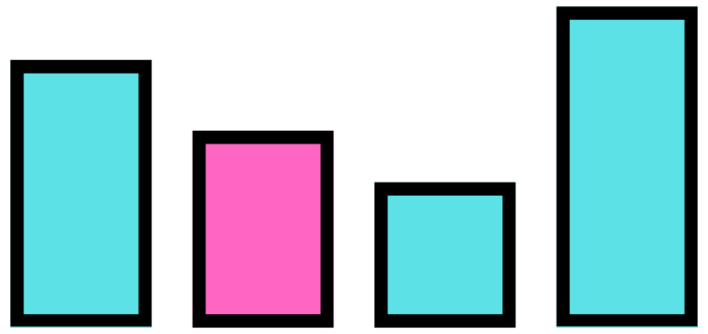
A Senate committee has 8 Republicans and 6 Democrats. In how many ways can we form a subcommittee of 5 members that has at least one member from each party?

p_1 | 0.306



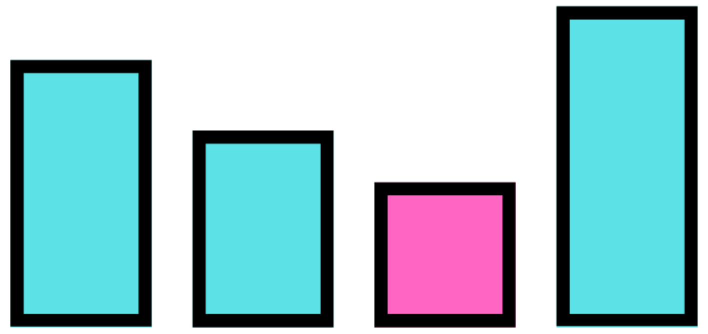
each particle in step 2 will be continued from this particle p_1 with a 30.6% probability

p_2 | 0.180



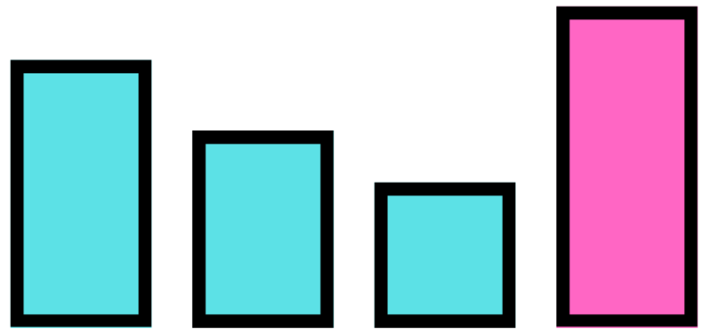
each particle in step 2 will be continued from this particle p_2 with a 18.0% probability

p_3 | 0.173



each particle in step 2 will be continued from this particle p_3 with a 17.3% probability

p_4 | 0.340



each particle in step 2 will be continued from this particle p_4 with a 34.0% probability

p_1

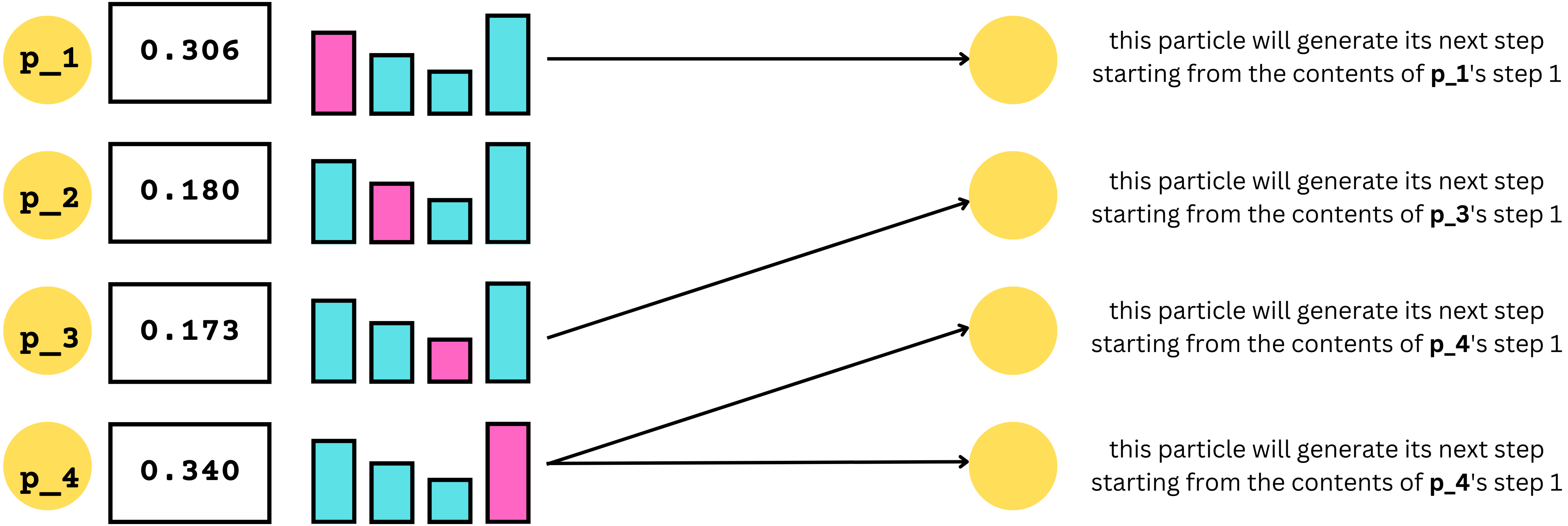
p_2

p_3

p_4

Problem:

A Senate committee has 8 Republicans and 6 Democrats. In how many ways can we form a subcommittee of 5 members that has at least one member from each party?



every particle independently, randomly selects from the previous set of particles with the weights specified.

each particle generates the next step

p_1

Step 1: Use casework: Count the number of ways to form subcommittees with at least one Democrat by considering (1D,4R), (2D,3R), (3D,2R), (4D,1R), and (5D,0R), then sum these cases. ## Step 2: Compute the number of ways for each valid case Using combinations, calculate: C(6,1) * C(8,4) + C(6,2) * C(8,3) + C(6,3) * C(8,2) + C(6,4) * C(8,1) + C(6,5) * C(8,0) = 1940.

p_2

Step 1: Assume each senator is equally likely to be chosen and compute the probability of selecting at least one Democrat by considering individual selections. ## Step 2: Convert the probability approach into counting Instead of using probability, we should count the valid subcommittees directly. Re-evaluate by considering cases where at least one Democrat is included, using combinations to count valid selections.

p_3

Step 1: Calculate the total number of ways to choose 5 members from the entire committee without any restrictions\nWe can use combinations to calculate this. The total number of members is 14 (8 Republicans + 6 Democrats), so the total number of ways to choose 5 members is given by C(14,5) = 14! / (5! * (14-5)!) = 14! / (5! * 9!) = 2002. ## Step 2: Calculate the number of subcommittees that have only Democrats Using the same concept as before, the number of ways to choose 5 members from the 6 Democrats is: C(6,5) = 6! / (5! * (6-5)!) = 6! / (5! * 1!) = 6.

p_4

Step 1: Calculate the total number of ways to choose 5 members from the entire committee without any restrictions\nWe can use combinations to calculate this. The total number of members is 14 (8 Republicans + 6 Democrats), so the total number of ways to choose 5 members is given by C(14,5) = 14! / (5! * (14-5)!) = 14! / (5! * 9!) = 2002. '## Step 2: Calculate the number of subcommittees that have only Republicans\nUsing the same concept as before, the number of ways to choose 5 members from the 8 Republicans is C(8,5) = 8! / (5! * (8-5)!) = 8! / (5! * 3!) = 56.

now, we do this entire thing again!

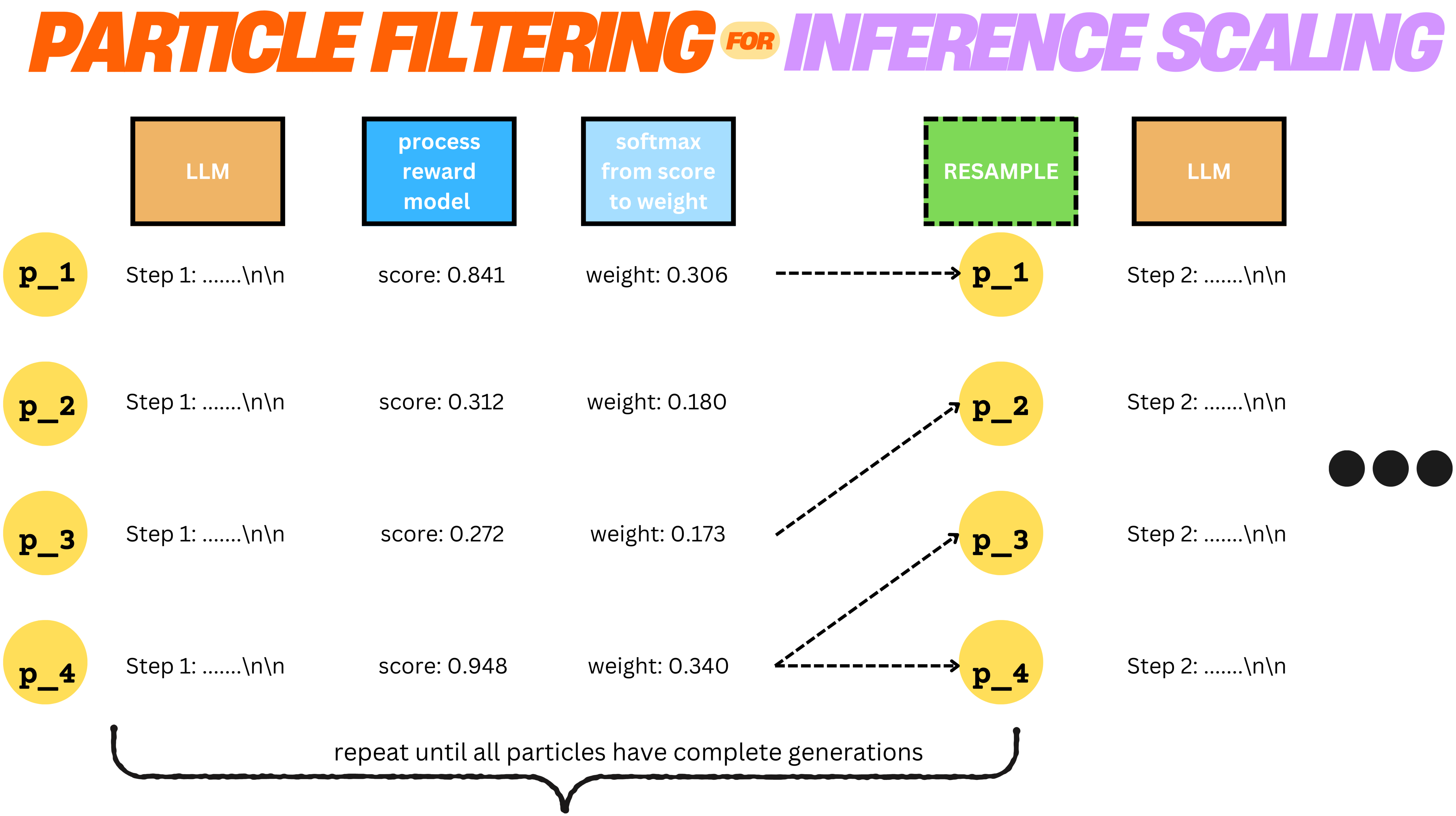
1 . each particle generates its “next step”

2 . we use the PRM (Process Reward Model) to calculate a score using the question and the entire answer generated by that particle so far

3 . we convert that score to a weight with softmax

4 . we then resample the particles according to those weights

we continue doing this until every single particle has generated an “end of sequence” token, and thus, finished its answer!


PARTICLE FILTERING FOR INFERENCE SCALING
LLM
process reward model
softmax from score to weight
RESAMPLE
LLM
p_1
Step 1: ....\n\n
score: 0.841
weight: 0.306
p_1
Step 2: ....\n\n
p_2
Step 1: ....\n\n
score: 0.312
weight: 0.180
p_2
Step 2: ....\n\n
p_3
Step 1: ....\n\n
score: 0.272
weight: 0.173
p_3
Step 2: ....\n\n
p_4
Step 1: ....\n\n
score: 0.948
weight: 0.340
p_4
Step 2: ....\n\n
repeat until all particles have complete generations

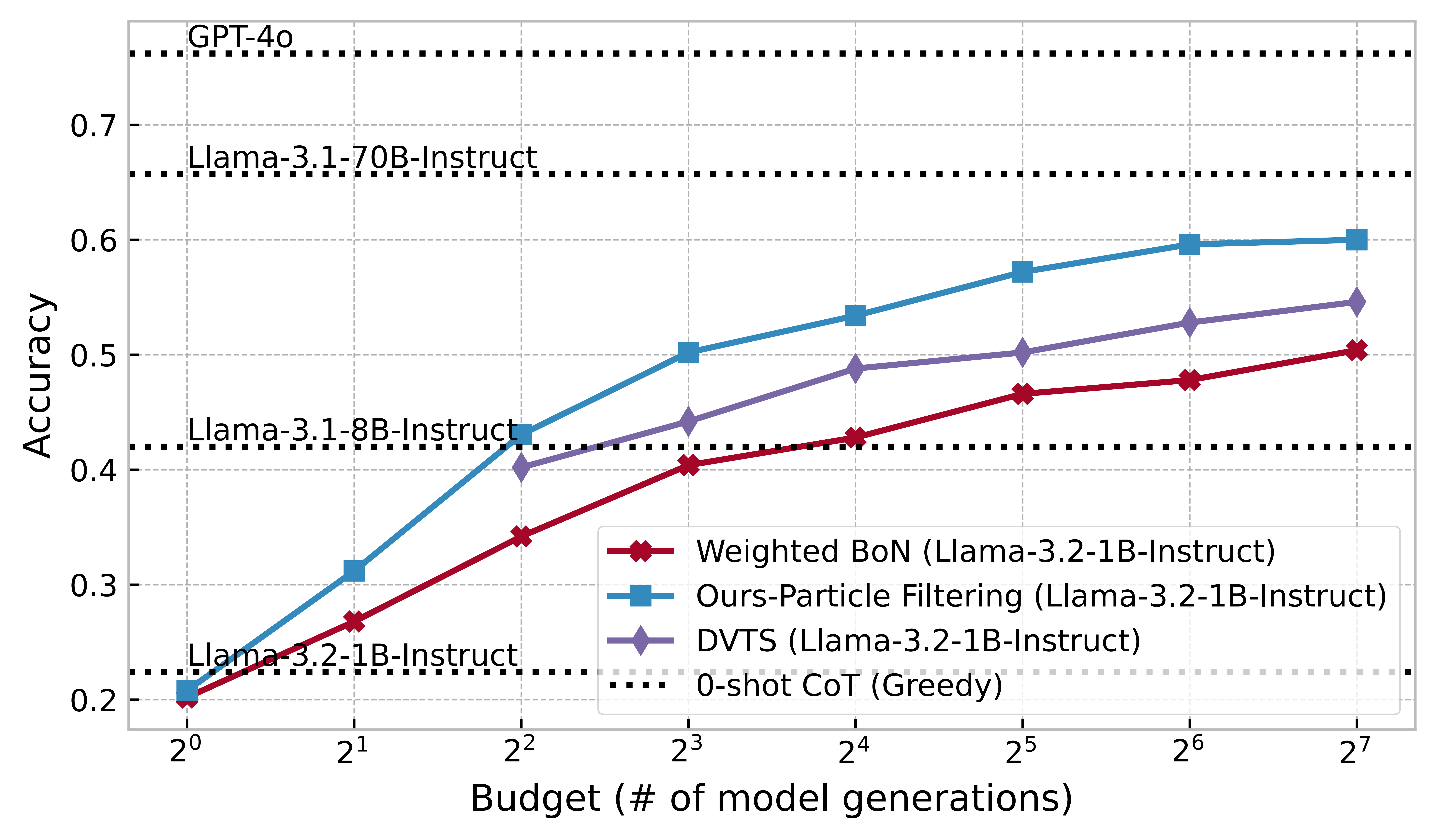
Results

Results

- **4-16x better scaling rate compared to deterministic search methods on challenging mathematical reasoning tasks**
- Qwen2.5-Math-1.5B-Instruct surpasses GPT-4o accuracy with only 4 rollouts
- Qwen2.5-Math-7B-Instruct achieves o1 level accuracy with only 32 rollouts

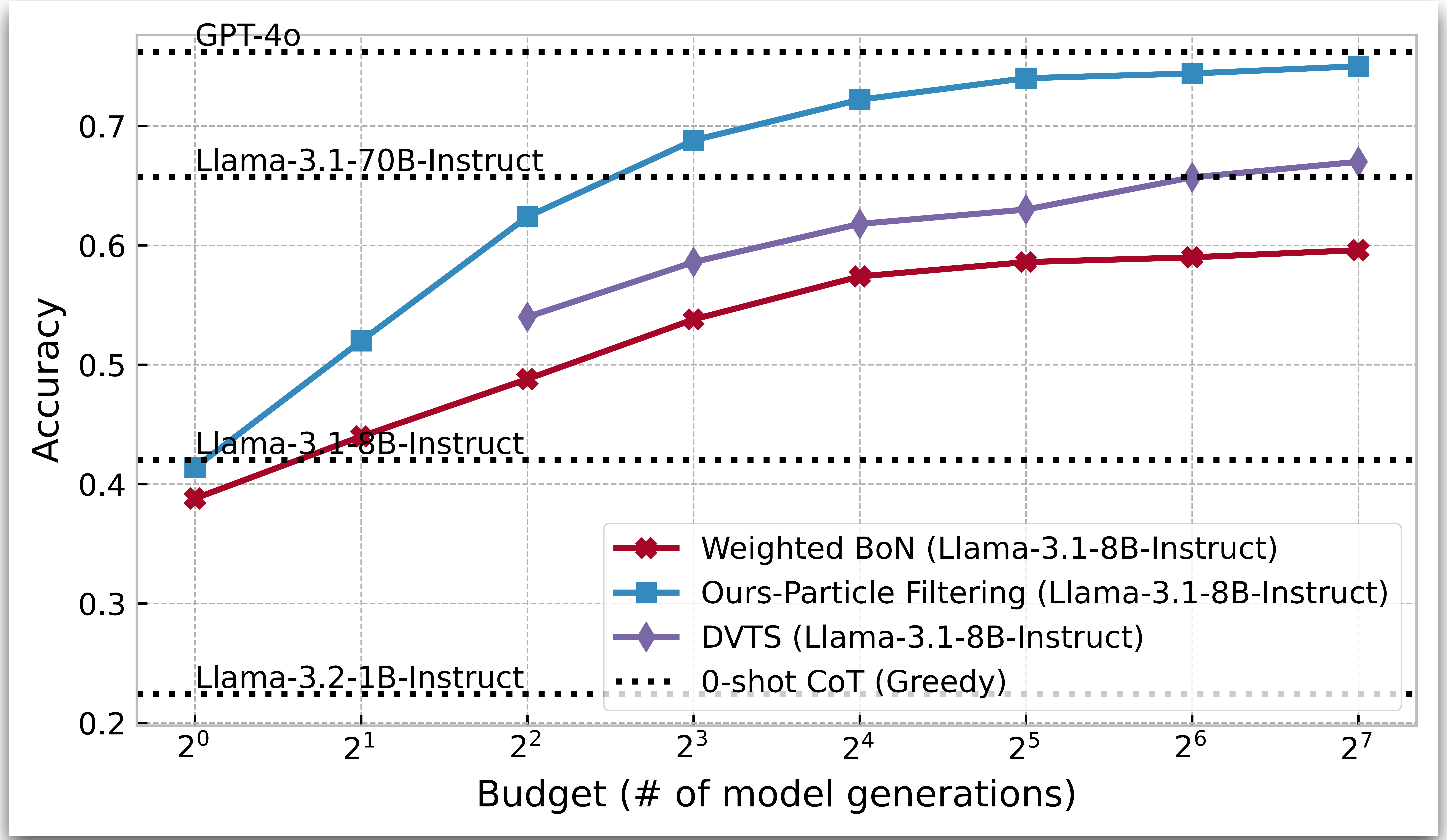
Model	Method	MATH500	AIME 2024
Closed-Source LLMs			
GPT-4o	-	76.2	13.3
o1-preview	-	***87.0***	40.0
Claude3.5-Sonnet	-	78.3	16.0
Open-Source LLMs			
Llama-3.1-70B-Instruct	-	65.7	16.6
Qwen2.5-Math-72B-Instruct	-	82.0	30.0
Open-Source SLMs			
Llama-3.2-1B-Instruct	Pass@1	26.8	0.0
	Ours - PF	59.6	10.0
Llama-3.1-8B-Instruct	Pass@1	49.9	6.6
	Ours - PF	74.4	16.6
phi-4	Pass@1	79.8	16.6
	Ours - PF	83.6	26.6
Mistral-Small-24B-Instruct-2501	Pass@1	69.2	10.0
	Ours - PF	83.4	23.3
Qwen2.5-32B-Instruct	Pass@1	82.8	16.6
	Ours - PF*	***89.9***	***43.3***
Open-Source Math SLMs			
Qwen2.5-Math-1.5B-Instruct	Pass@1	70.0	10.0
	Ours - PF	85.4	23.3
Qwen2.5-Math-7B-Instruct	Pass@1	79.6	16.6
	Ours - PF	87.0	23.3

Results



GPT-4o
Llama-3.1-70B-Instruct
Llama-3.1-8B-Instruct
Llama-3.2-1B-Instruct
Accuracy
0.7
0.6
0.5
0.4
0.3
0.2
2^0
2^1
2^2
2^3
2^4
2^5
2^6
2^7
Budget (# of model generations)
Weighted BoN (Llama-3.2-1B-Instruct)
Ours-Particle Filtering (Llama-3.2-1B-Instruct)
DVTS (Llama-3.2-1B-Instruct)
0-shot CoT (Greedy)

Results



GPT-4o
Llama-3.1-70B-Instruct
Llama-3.1-8B-Instruct
Llama-3.2-1B-Instruct
Accuracy
0.7
0.6
0.5
0.4
0.3
0.2
2^0
2^1
2^2
2^3
2^4
2^5
2^6
2^7
Budget (# of model generations)
Weighted BoN (Llama-3.1-8B-Instruct)
Ours-Particle Filtering (Llama-3.1-8B-Instruct)
DVTS (Llama-3.1-8B-Instruct)
0-shot CoT (Greedy)

Results

Method	FinanceBench	NumGLUE Task 2 (Chemistry)
Greedy	62.67	71.69
BoN	68.00	80.92
Self Consistency	68.67	79.32
Beam Search	67.33	80.47
Particle Filtering (Ours)	70.33	84.22

Why does it matter?

Results

Model	Method	MATH500	AIME 2024
Closed-Source LLMs			
GPT-4o	-	76.2	[illegible]
o1-preview	-	*87.0*	[illegible]
Claude3.5-Sonnet	-	78.3	[illegible]
Open-Source LLMs			
Llama-3.1-70B-Instruct	-	[illegible]	16.6
Qwen2.5-Math-72B-Instruct	-		30.0
Open-Source SLMs			
Llama-3.2-1B-Instruct	Pass@1	26.8	0.0
	Ours - PF	59.6	10.0
Llama-3.1-8B-Instruct	Pass@1	49.9	6.6
	Ours - PF	74.4	16.6
phi-4	Pass@1	79.8	16.6
	Ours - PF	83.6	26.6
Mistral-Small-24B-Instruct-2501	Pass@1	69.2	10.0
	Ours - PF	83.4	23.3
Qwen2.5-32B-Instruct	Pass@1	82.8	16.6
	Ours - PF*	*89.9*	*43.3*
Open-Source Math SLMs			
Qwen2.5-Math-1.5B-Instruct	Pass@1	70.0	10.0
	Ours - PF	85.4	23.3
Qwen2.5-Math-7B-Instruct	Pass@1	79.6	16.6
	Ours - PF	87.0	23.3

- We do all of this - scaling small models to such large numbers - **without training anything at all!**
- The method is able to efficiently guide a small, open source off-the-shelf model to "discover its potential" and make massive improvements, just by intelligently navigating the search space

Why Does Inference Time Scaling Matter?

Unlocking hidden capabilities of LLMs, improving quality & reliability — *without retraining*

&

bridging the gap to larger, more powerful models

Provides insights that leads to o1-r1-style-reasoning models:

- observing that CoT improves performance → training on CoT examples directly
- observing that diverse reasoning paths help → training models to explore diverse paths internally.

State of The Art:

Even today, many domains will rely upon inference time scaling, because just querying even the best models is not enough. This is often the case in settings when high accuracy and verifiability matter, where you may want to sample and rank outputs for max confidence.

Why Does Inference Time Scaling Matter?

Inference-time scaling is the most open, cheapest, and often the only way to extract better reasoning, reliability, and robustness from language models — especially when you can't retrain them.

thank you!

Rollout Roulette: A Probabilistic Inference Approach to Inference-Time Scaling of LLMs using Particle-Based Monte Carlo Methods

Isha Puri[1] Shiv Sudalairaj[2] GX Xu[2] Kai Xu[2] Akash Srivastava[2]

[1]MIT CSAIL [2]RedHat AI Innovation

thank you!

please check out our website

probabilistic-inference-scaling.github.io/

for more information!

Self consistency comparison