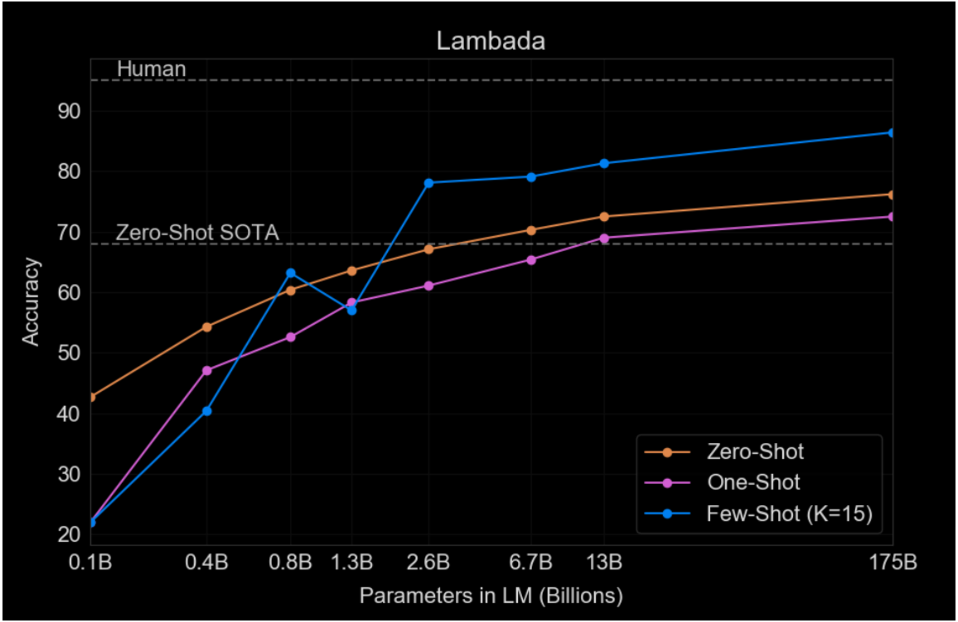
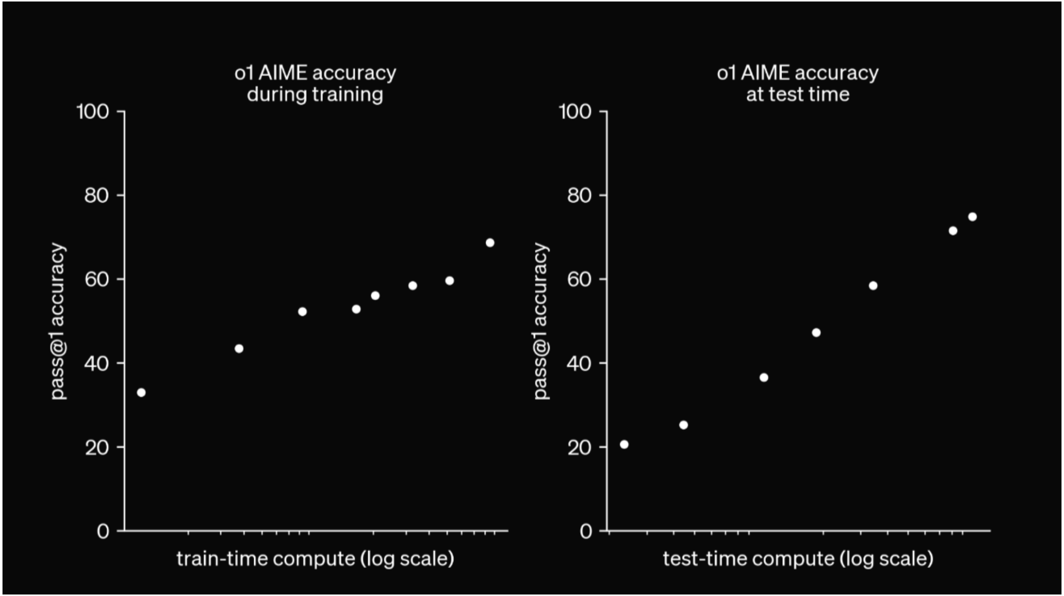
Speculations on Test-Time Scaling

Sasha Rush Daniel Ritter

Cornell

[?]


Lambada
Human
Zero-Shot SOTA
Accuracy
90
80
70
60
50
40
30
20
0.1B
0.4B
0.8B
1.3B
2.6B
6.7B
13B
175B
Parameters in LM (Billions)
Zero-Shot
One-Shot
Few-Shot (K=15)


o1 AIME accuracy
during training
100
80
60
40
20
0
pass@1 accuracy
train-time compute (log scale)
o1 AIME accuracy
at test time
100
80
60
40
20
0
pass@1 accuracy
test-time compute (log scale)

[?]

AIME

[?]

For any finite set X, let $|X|$ denote the number of elements in X. Define

$$S_n = \sum |A \cap B|,$$

where the sum is taken over all ordered pairs (A, B) such that A and B are subsets of $\{1, 2, 3, \cdots, n\}$ with $|A| = |B|$. For example, $S_2 = 4$ because the sum is taken over the pairs of subsets

$$(A, B) \in \{(\emptyset, \emptyset), (\{1\}, \{1\}), (\{1\}, \{2\}), (\{2\}, \{1\}), (\{2\}, \{2\}), (\{1, 2\}, \{1, 2\})\}$$

giving $S_2 = 0 + 1 + 0 + 0 + 1 + 2 = 4$. Let $\frac{S_{2022}}{S_{2021}} = \frac{p}{q}$, where p and q are relatively prime positive integers. Find the remainder when $p + q$ is divided by 1000.

The Bitter Lesson

[?]

The bitter lesson is based on the historical observations that 1) AI researchers have often tried to build knowledge into their agents, 2) this always helps in the short term, and is personally satisfying to the researcher, but 3) in the long run it plateaus and even inhibits further progress, and 4) breakthrough progress eventually arrives by an opposing approach based on scaling computation by **search and learning**.

Importance of Search

https://x.com/polynoamial/status/1840822629625688469

> The most important [lesson] is that I and other researchers simply didn't know how much of a difference scaling up search would make. If I had seen those scaling results at the start of my PhD, I would have shifted to researching search algorithms for poker much sooner and we probably would have gotten superhuman poker bots much sooner.

Sources

- Survey of the public literature
- Synthesis of discussions with expert
- Rumors from social media

Thanks to Lewis Tunstall, Edward Beeching, Aviral Kumar, Charlie Snell, Michael Hassid, Yoav Artzi, Risab Agarwal, Kanishk Gandhi, Wenting Zhao, Yuntian Deng, Nathan Lambert, Noah Goodman

Outline

Introduction

The Clues

Technical Background

The Suspects

What do we do now?

[?]

o1 Description

Our large-scale **reinforcement learning algorithm** teaches the model how to think productively using its **chain of thought** in a highly **data-efficient** training process.

Implication

- RL; Signal from verifiable problems
- CoT; Test-time occurs in token stream
- Data Efficient; Bounded set of problems

Current Assumptions

- Single final language model
- Not following from expert examples
- Behavior is “learned”.

Review: Chain of Thought

[?, ?]

- In the process of answering a question, the model generates intermediate steps.
- These steps provide an scratchpad for hard technical problems.
- A limited number of examples provided.

o1 Chain of Thought

[?]

o1 learns to hone its chain of thought and refine the strategies it uses. It learns to recognize and **correct its mistakes**. It learns to **break down tricky steps** into simpler ones. It learns to try a **different approach** when the current one isn't working.

o1 CoT: Outlining

[?]

Implementation Outline:

1. Capture input string as argument.
2. Remove any spaces (if any).
3. Parse the input string to extract numbers into arrays.

- Since the input is in the format ’[1,2],[3,4],[5,6]’, we can:
- Remove outer brackets if necessary.
- Split the string by ’],’ to get each row.
- For each row, remove ’[’ and ’]’, then split by ’,’ to get elements.

4. Build a 2D array in bash (arrays containing arrays).

o1 CoT: Planning

[?]

First, the cipher seems connected to the plaintext.
Given the time constraints, perhaps the easiest way is to try to see patterns.
Option 1: Try to find mapping from letters to letters.
Do any letters match?
First, let's write down the ciphertext and plaintext letters on top of each other.

o1 CoT: Backtracking

[?]

Similarly, m(x)⍰(−x 2) = (−x2n + 2 + m2n−2x2n + lower terms)m(x) * (−x 2) = (−x 2n + 2 + m 2n−2 x 2n + lower terms). Wait, actually, this may not help us directly without specific terms.An alternative is to consider the known polynomials. So m(x) = k ...

o1 CoT: Self-Evaluation

Let's analyze each option.
Option A: "because appetite regulation is a field of staggering complexity."
Is that a good explanation? Hmm.
Option B: "because researchers seldom ask the right questions."
Does this make sense with the main clause?

Summary

- CoT provides test-time scaling
- CoT looks like search / planning in a classical sense
- RL needed to induce this behavior

Outline

Introduction

The Clues

Technical Background

The Suspects

What do we do now?

Technical Background

[?]

- Formalize sampling of latent reasoning
- Techniques from combinatorial sampling
- No learning yet.

Question: 4 baskets. 3 have 9 apples, 15 oranges, 14 bananas each. 4th has 2 less of each. Total fruits?
Let's solve step-by-step:
Fruits in one of first 3 baskets: 9 + 15 + 14 = 38
Total in first 3 baskets: 38 * 3 = 114
4th basket: (9-2) + (15-2) + (14-2) = 32
Total fruits: 114 + 32 = 146
Answer: 146 fruits

Stepwise CoT Sampling

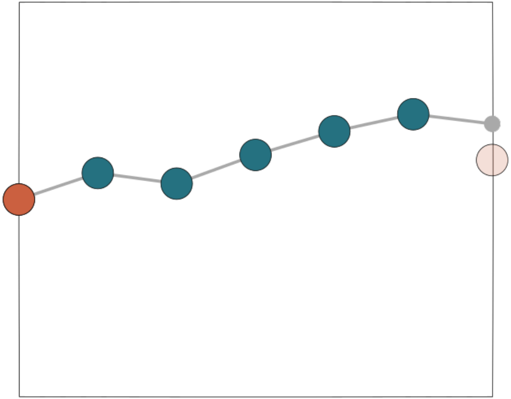
- x; problem specification
- $z_{1:T} \in \mathcal{S}^T$; chain of thought (CoT) steps
- $y \in \mathcal{Y}$; final answer

$$p(y|x) = \mathbb{E}_z p(y|x, z)$$

Warm-up: Ancestral Sampling

[?]

$$z_{1:T} \sim p(\cdot | x)$$
$$y \sim p(\cdot | x, z_{1:T})$$



T is the amount of test-time compute

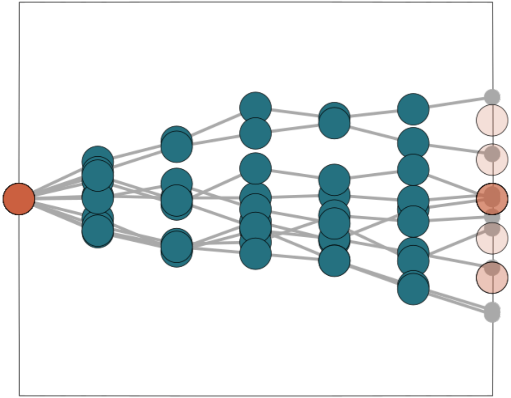
Warm-up: Monte-Carlo (Self-Consistency)

[?]

For N samples,

$$z_{1:T} \sim p(\cdot|x)$$
$$y^n \sim p(\cdot|x, z_{1:T})$$

Pick majority choice y^n



[Uesato et al., 2022]

Assumption: Verifier

$$\mathsf{Ver}_x : \mathcal{Y} \to \{0, 1\}$$

Common datasets:

- Regular expression for math [?]
- Unit test for code [?]
- Test questions for science [?]

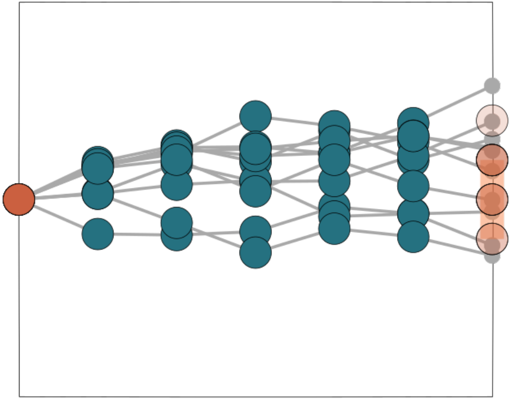
Rejection Sampling / Best-of-N

[?, Lightman et al., 2023]

For $n = 1$ to N:

$$z^n \sim p(z|x)$$
$$y^n \sim p(y|x, z^n)$$

Verified set $\{y^n : \text{Ver}(y^n)\}$



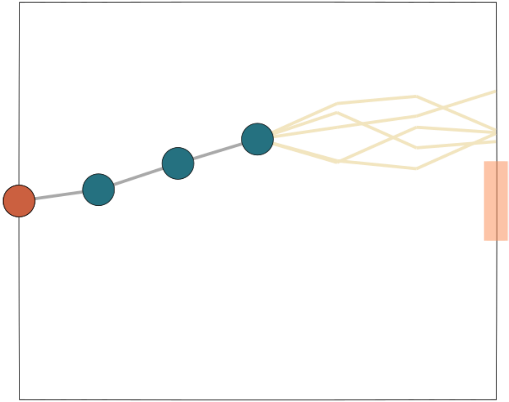
Monte-Carlo Roll-Outs

[Wang et al., 2023]

Given partial CoT $z_{1:t}$, expected value,

$$\mathbb{E}_{y \sim p(\cdot|x)} \mathrm{Ver}(y)$$

Monte Carlo for this expectation.



Goal: Learning with Latent CoTs

Maximum likelihood;

$$\max_{\theta} \sum \log p(y|x;\theta) =$$
$$\sum \log \mathbb{E}_z p(y|x, z; \theta)$$

Classic combinatorial
expectation

Outline

Introduction

The Clues

Technical Background

The Suspects

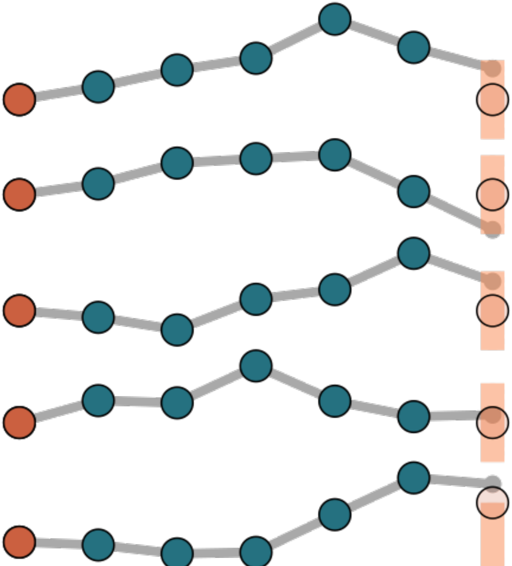
What do we do now?

The Suspects

- Guess + Check
- Guided Search
- AlphaZero
- Learn to Search

Suspect 1: Guess + Check

- 1) Sample N CoTs
- 2) Check if successful
- 3) Train on good ones



Formalization: Rejection Sampling EM

[?, ?]

$$\max_{\theta} \sum \log E_{z \sim p(z|x;\theta)} p(y|x,z)$$

- E-Step: For $n = 1$ to N:

$$z^n \sim p(\cdot|x)$$
$$y^n \sim p(\cdot|x, z^n)$$

  Keep verified set $\mathcal{Z} = \{z^n : \text{Ver}(y^n)\}$

- M-Step: Fit $\theta' \leftarrow \arg\max_{\theta} \sum_{z \in \mathcal{Z}} \log p(z|x;\theta)$

Variants

- Best-of-N Training [?]
- STaR [Zelikman et al., 2022]
- ReST [Gulcehre et al., 2023]
- ReST-EM [Singh et al., 2023]
- Filtered Rejection Sampling [?]

Reinforcement Learning?

- Batched $\rightarrow$ Compute trajectories first, then train
- Online $\rightarrow$ Update after each example
- Specific algorithm choice (PPO, etc)

Empirical Results

[Singh et al., 2023]

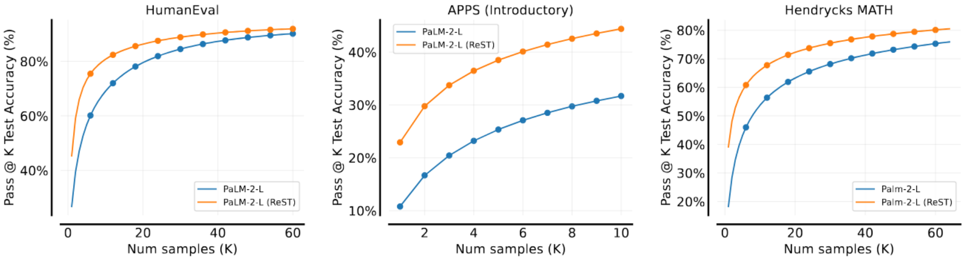


Figure 5 | **Pass@K results** for PaLM-2-L pretrained model as well as model fine-tuned with ReST^{EM}. For a fixed number of samples K, fine-tuning with ReST^{EM} substantially improves Pass@K performance. We set temperature to 1.0 and use nucleus sampling with $p = 0.95$.

Is this o1?

- ✓ Extremely simple and scalable
- ✓ Positive results in past work

Is this o1?

- ✓ Extremely simple and scalable
- ✓ Positive results in past work

- × No evidence this learns to correct, plan
- × Computationally inefficient search

Towards More Structure

- Rejection sampling may be really inefficient.
- Particularly on hard problems, may get no signal

Suspect 2: Guided Search

- During CoT sampling, use a heuristic to improve trajectories

- Check if final versions are successful

- Train on good ones

GS: Beam Search with Guide

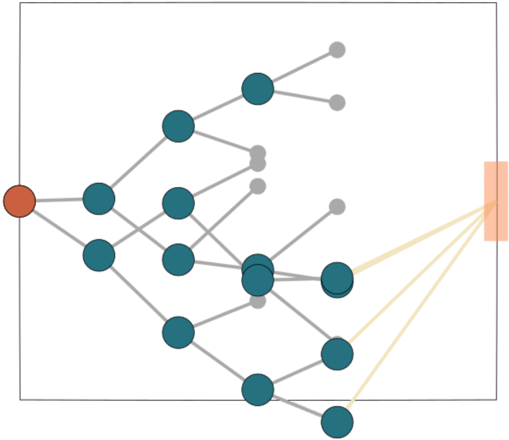
$r : \mathcal{S}^t \to \mathbb{R}$; Guide function

For each step t

1. Sample next step,

$$z_t \sim p(\cdot | x, z^i_{1:t-1})$$

2. Keep the top N samples, ordered by $r(z_t)$



What to use as Guide?

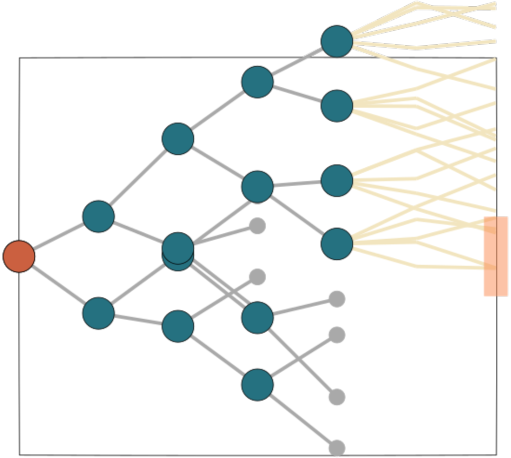
- Monte Carlo Roll-outs
- Learned Value Function
- Interleaved Value Function

Beam Search with Roll-Outs

For a z_t, sample answers

$$y^n \sim p(\cdot | x, z_{1:t-1})$$

$$r_{MC}(z_t) = \frac{1}{N} \sum_{i=1}^{n} \text{Ver}(y^i)$$

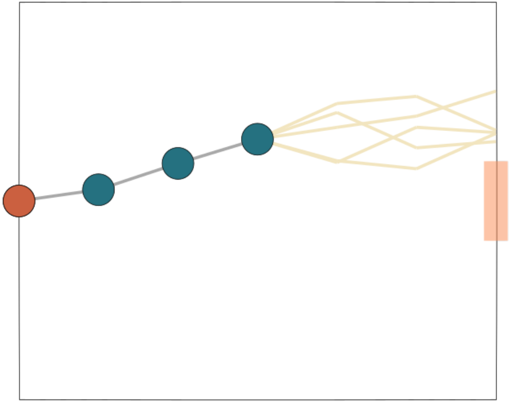


Amortized Roll-Outs

[Wang et al., 2023]

- Rollouts are costly, so instead learn a model $r_\psi(z_t)$ to approximate rollouts

- Use r_{MC} to determine labels to train r_ψ



Interleaved

[Wang et al., 2023]

-
-

What about test time?

- Learned rewards can improve test-time without verifier.
-

Terminology

[Uesato et al., 2022, Setlur et al., 2024, Wang et al., 2023, Lightman et al., 2023, Snell et al., 2024]

- Value
- PRM
- PAV
- Math Shepard.
- snell.

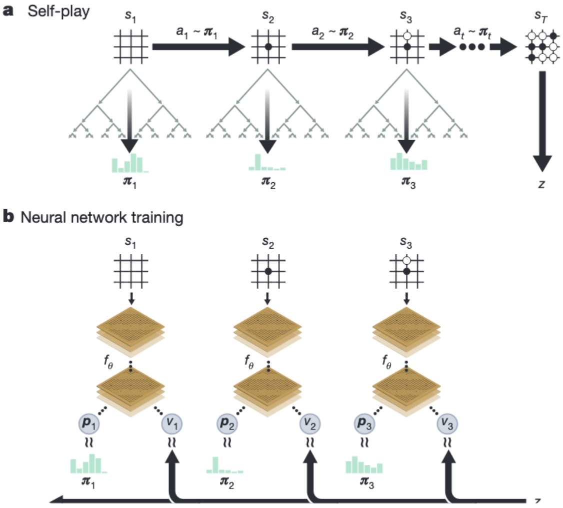
Why might this be right?

- OpenAI is exploring
- Makes RS more efficient.
- Learned rewards are effective
- Assumption: o1 is a single test-time model (although could train or distill-in)
- Not clear if it learns planning.

More Structure

- Improving search seems critical.

Reminder: AlphaZero



Suspect 3: AlphaZero

- Self-play using guided-search with exploration
- Label final outcomes of self-play games
- Train guide and generator

Formalized: Expert Iteration

[Anthony et al., 2017]

- Iterative algorithm combining learned model + expert search with a verifier.

- Generate samples using $p(y, z|x)$, reward model $r(z_t)$, and search algorithm (e.g. beam search)

- Label samples using $Ver(y)$

- Train $p(y, z|x)$, $r(z_t)$ on the labeled samples, and repeat

MCTS exploration

UCB for Language

- Selection: Walk down tree to leaf z_{t-1}
- Expand: Sample K next steps z_t^i, pick one at random
- Rollouts: Sample $z_{t+1} \ldots z_T$
- Backprop: Update nodes counts $z_{1:t}$ based on results

Compared with Search

- System builds in exploration
- Scales to more train-time search

- Costly to maintain open states
- More complex algorithmically

Empirical Results

[?]

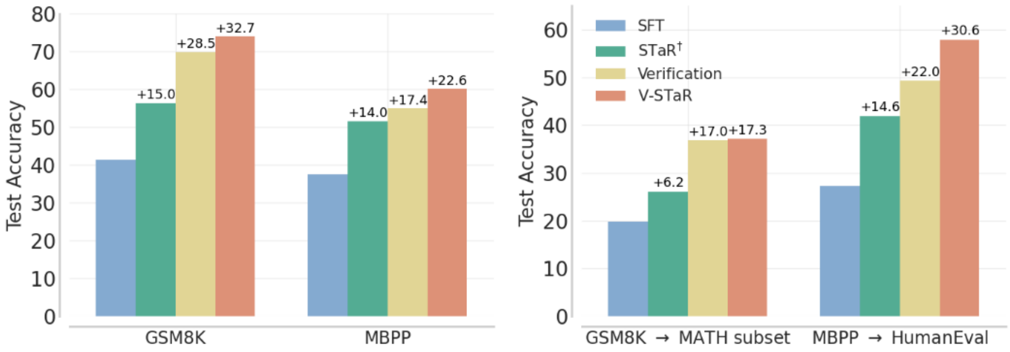


Figure 8: Test accuracy of 13B V-STaR compared to baselines. We report Best-of-64 for verification-based methods and Pass@1 for others. **(Left)** Test accuracy for training tasks. **(Right)** Transfer evaluation of GSM8K and MBPP trained models on MATH subset and HumanEval respectively.

Why might this be right?

- Major demonstrated RL result
-
-
-

More Structure

- Can we force the model to search?

Suspect 4: Learning to Correct

- Sample N Successful CoTs
- Edit to inject incorrect expansions before correct ones.
- Train on correcting trajectories

Self-Correction

[Gandhi et al., 2024]

- Argument: Training on x, z_1^*, y is too easy.
- Train instead on x, z', z_1^*, y
- Model should learn to self-correct

Score

- Positive rewards

Challenge: Collapse

- Model may learn to just ignore negative
-
-

[Gandhi et al., 2024]

Generalized: Stream of Search

- Find $z^*_{1:T}$ as optimal length CoT
- Find $z'_{1:T'}$ with $T' > T$ through backtracking tree search
- Train model on $z'_{1:T'}$

[Gandhi et al., 2024]

From Tree to Stream

Empirical Results

Score results

Why might this be right?

Less Structure?

- Maybe this is all too much...
- Could this be done without a verifier?

Outline

Introduction

The Clues

Technical Background

The Suspects

What do we do now?

Replication

Does it need to be the same?

Reference I

[Anthony et al., 2017] Anthony, T., Tian, Z., and Barber, D. (2017).
Thinking fast and slow with deep learning and tree search.
arXiv [cs.AI].

[Brown et al., 2024] Brown, B., Juravsky, J., Ehrlich, R., Clark, R., Le, Q. V., Ré, C., and Mirhoseini, A. (2024).
Large language monkeys: Scaling inference compute with repeated sampling.
arXiv [cs.LG].

Reference II

[Gandhi et al., 2024] Gandhi, K., Lee, D., Grand, G., Liu, M., Cheng, W., Sharma, A., and Goodman, N. D. (2024). Stream of search (SoS): Learning to search in language. *arXiv [cs.LG]*.

[Gulcehre et al., 2023] Gulcehre, C., Paine, T. L., Srinivasan, S., Konyushkova, K., Weerts, L., Sharma, A., Siddhant, A., Ahern, A., Wang, M., Gu, C., Macherey, W., Doucet, A., Firat, O., and de Freitas, N. (2023). Reinforced self-training (ReST) for language modeling. *arXiv [cs.CL]*.

Reference III

[Lightman et al., 2023] Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. (2023).
Let's verify step by step.
arXiv [cs.LG].

[Setlur et al., 2024] Setlur, A., Nagpal, C., Fisch, A., Geng, X., Eisenstein, J., Agarwal, R., Agarwal, A., Berant, J., and Kumar, A. (2024).
Rewarding progress: Scaling automated process verifiers for LLM reasoning.
arXiv [cs.LG].

Reference IV

[Singh et al., 2023] Singh, A., Co-Reyes, J. D., Agarwal, R., Anand, A., Patil, P., Garcia, X., Liu, P. J., Harrison, J., Lee, J., Xu, K., Parisi, A., Kumar, A., Alemi, A., Rizkowsky, A., Nova, A., Adlam, B., Bohnet, B., Elsayed, G., Sedghi, H., Mordatch, I., Simpson, I., Gur, I., Snoek, J., Pennington, J., Hron, J., Kenealy, K., Swersky, K., Mahajan, K., Culp, L., Xiao, L., Bileschi, M. L., Constant, N., Novak, R., Liu, R., Warkentin, T., Qian, Y., Bansal, Y., Dyer, E., Neyshabur, B., Sohl-Dickstein, J., and Fiedel, N. (2023).
Beyond human data: Scaling self-training for problem-solving with language models.

Reference V

arXiv [cs.LG].

[Snell et al., 2024] Snell, C., Lee, J., Xu, K., and Kumar, A. (2024).
Scaling LLM test-time compute optimally can be more effective than scaling model parameters.
arXiv [cs.LG].

[Uesato et al., 2022] Uesato, J., Kushman, N., Kumar, R., Song, F., Siegel, N., Wang, L., Creswell, A., Irving, G., and Higgins, I. (2022).
Solving math word problems with process- and outcome-based feedback.

Reference VI

arXiv [cs.LG].

[Wang et al., 2023] Wang, P., Li, L., Shao, Z., Xu, R. X., Dai, D., Li, Y., Chen, D., Wu, Y., and Sui, Z. (2023). Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. *arXiv [cs.AI]*.

[Zelikman et al., 2022] Zelikman, E., Wu, Y., Mu, J., and Goodman, N. D. (2022). STaR: Bootstrapping reasoning with reasoning. *arXiv [cs.LG]*.