Tutorial: Test-Time Scaling for Mathematical Reasoning

Sean Welleck

May 28, 2025

Carnegie Mellon University

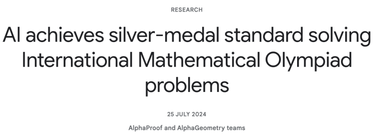
Today's talk

Algorithms for generating outputs with a language model

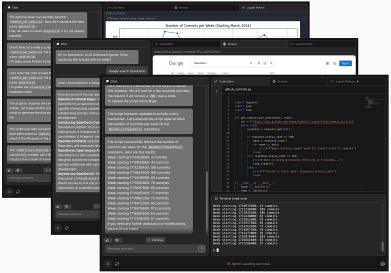
Algorithms for generating outputs with a language model

Why? *Use **test-time compute** to improve performance*

Language models



Solving olympiad problems

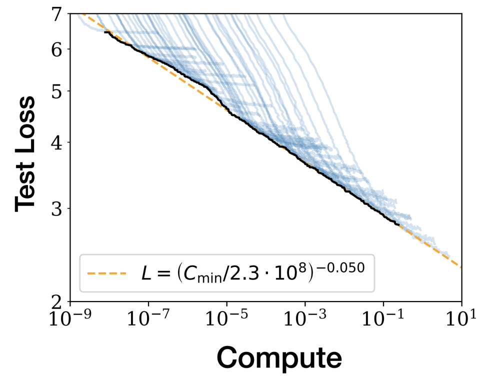


Writing code

Tasks framed as generating sequences: many other applications

Approach 1: scale pretraining compute

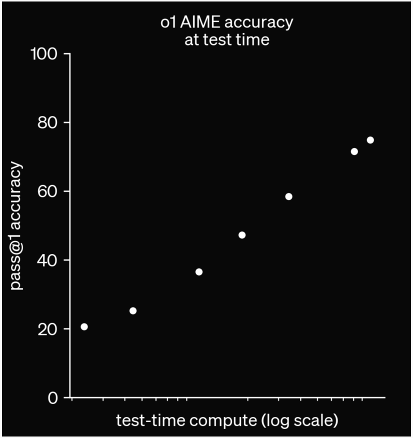
[2020-] **Scale training-time compute:** larger model, larger dataset



Scaling Laws for Neural Language Models [Kaplan et al., 2020]

Approach 2: scale *test-time* compute

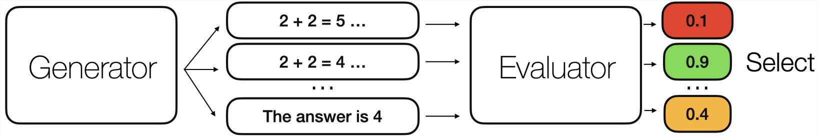
[Now] **Test-time scaling:** increase compute at generation time



Test-time compute vs. accuracy ([OpenAI, 2024])

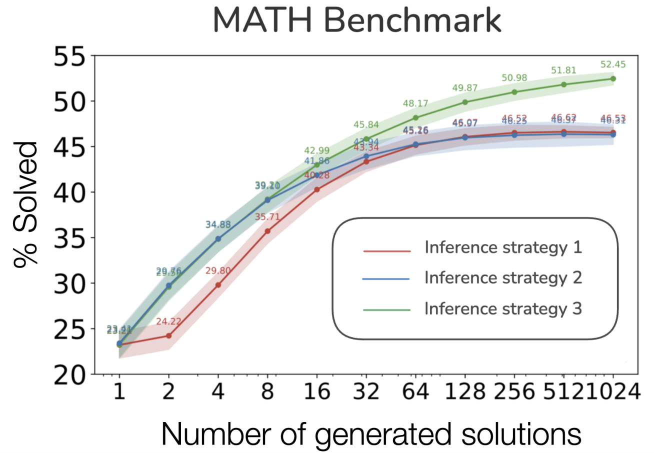
Test-time scaling strategies

1. Generate multiple times



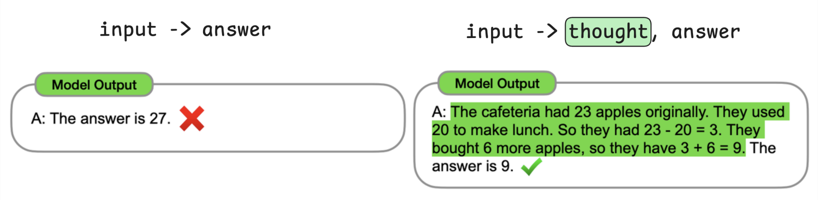
Test-time scaling strategies

1. Generate multiple times



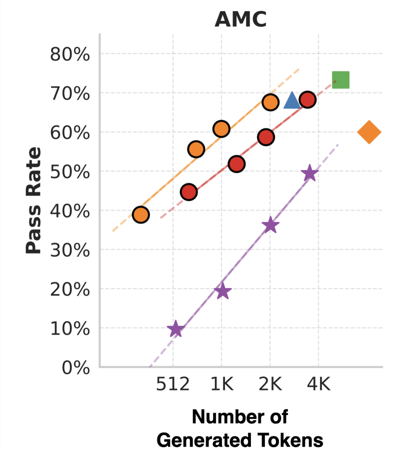
Test-time scaling strategies

1. Generate multiple times
2. Generate longer outputs

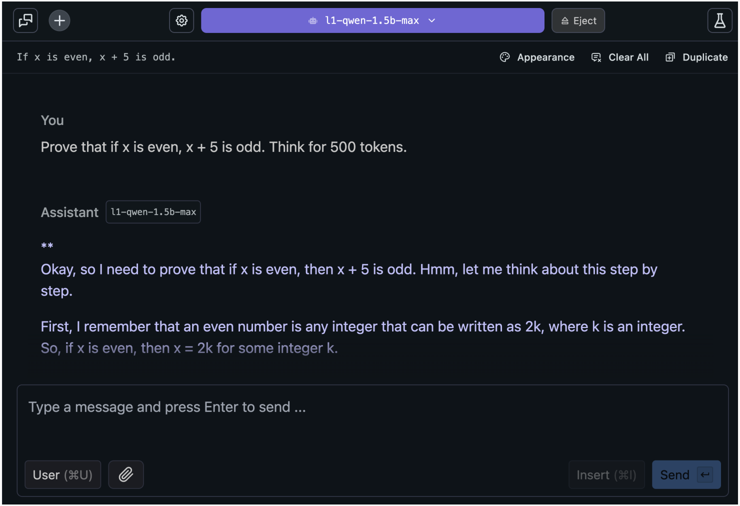


Test-time scaling strategies

1. Generate multiple times
2. Generate longer outputs



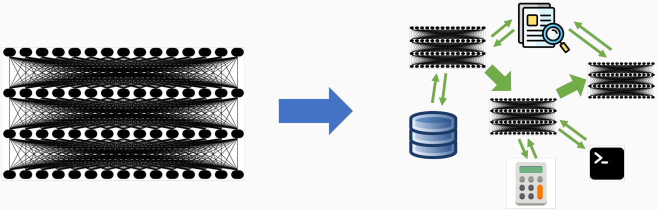
Test-time scaling strategies



Demo: L1 reasoning model [Aggarwal and Welleck, 2025] on a laptop

Test-time scaling strategies

1. Generate multiple times
2. Generate longer outputs
3. Incorporate other models/tools



[Zaharia et al., 2024]

Verifiers, code interpreters, search engines, ...

Today's tutorial

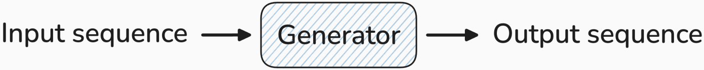
1. **Part 1:** Generate multiple times
    - *Meta-generation*: chain, parallel, refinement, tree search
2. **Part 2:** Generate longer outputs
    - *Long chain-of-thought*

Today's tutorial

1. Part 1: Generate multiple times
    - *Meta-generation*: chain, parallel, refinement, tree search
2. Part 2: Generate longer outputs
    - *Long chain-of-thought*

Recap: generation and decoding algorithms

Generator: Generates a sequence with a language model.

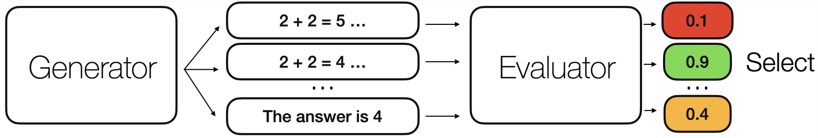


- Example: calling an LLM API
- Decoding algorithms
  - Temperature sampling
  - ...

$$y \sim g(p_\theta, x; \phi)$$

Unified view: Meta-Generation

Meta-generator: Test-time strategies for calling a generator multiple times and incorporating external information[1]

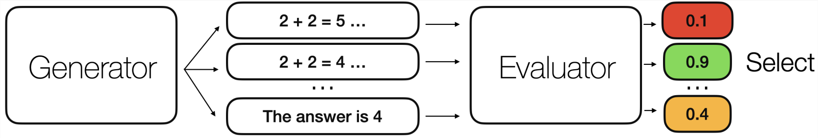


- Example: call API multiple times, select the best sequence with a separate model

[1] *From Decoding to Meta-Generation: Inference-time Algorithms for LLMs* [Welleck et al., 2024]

Unified view: Meta-Generation

Meta-generator: Test-time strategies for calling a generator multiple times and incorporating external information[1]



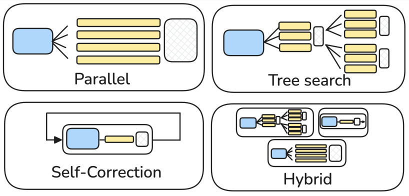
- Example: call API multiple times, select the best sequence with a separate model

$$y \sim G(x, g; \Phi)$$

[1] *From Decoding to Meta-Generation: Inference-time Algorithms for LLMs* [Welleck et al., 2024]

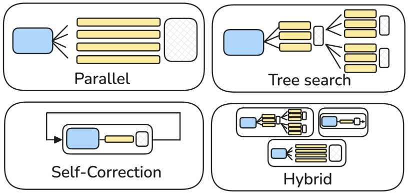
Meta-generators | outline

- Strategies
  - Parallel
  - Tree search
  - Refinement/self-correction
  - Chain (*Part II*)

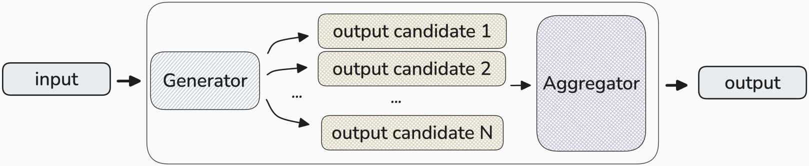


Meta-generators | outline

- Strategies
    - Parallel
    - Tree search
    - Refinement/self-correction
    - Chain (Part II)



Meta-generators | parallel



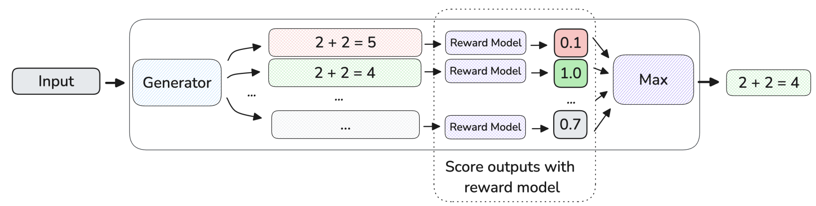
- Generate candidates:

$$\{y^{(1)}, \ldots, y^{(N)}\} \sim G(\cdot|x)$$

- Aggregate:

$$y = h(y^{(1)}, \ldots, y^{(N)})$$

Parallel | Best-of-N[2]



$$\underset{\{y^{(1)},\ldots,y^{(N)}\}}{\arg\max} \underbrace{v(y)}_{\text{reward model}}$$

[2] [Stiennon et al., 2020, Nakano et al., 2022]

x:

Input:
Let $f(r) = \sum_{j=2}^{2008} \frac{1}{j^r} = \frac{1}{2^r} + \frac{1}{3^r} + \cdots + \frac{1}{2008^r}$. Find $\sum_{k=2}^{\infty} f(k)$.

Llemma 34B solution:
We have

z:

$$\sum_{k=2}^{\infty} f(k) = \sum_{k=2}^{\infty} \sum_{j=2}^{2008} \frac{1}{j^k} = \sum_{j=2}^{2008} \sum_{k=2}^{\infty} \frac{1}{j^k} = \sum_{j=2}^{2008} \frac{1}{j^2} \sum_{k=0}^{\infty} \frac{1}{j^k} = \sum_{j=2}^{2008} \frac{1}{j^2} \frac{1}{1 - \frac{1}{j}}$$
$$= \sum_{j=2}^{2008} \frac{1}{j(j-1)} = \sum_{j=2}^{2008} \left(\frac{1}{j-1} - \frac{1}{j} \right)$$
$$= \left(\frac{1}{1} - \frac{1}{2} \right) + \left(\frac{1}{2} - \frac{1}{3} \right) + \cdots + \left(\frac{1}{2007} - \frac{1}{2008} \right)$$
$$= 1 - \frac{1}{2008}$$
$$= \boxed{\frac{2007}{2008}}.$$

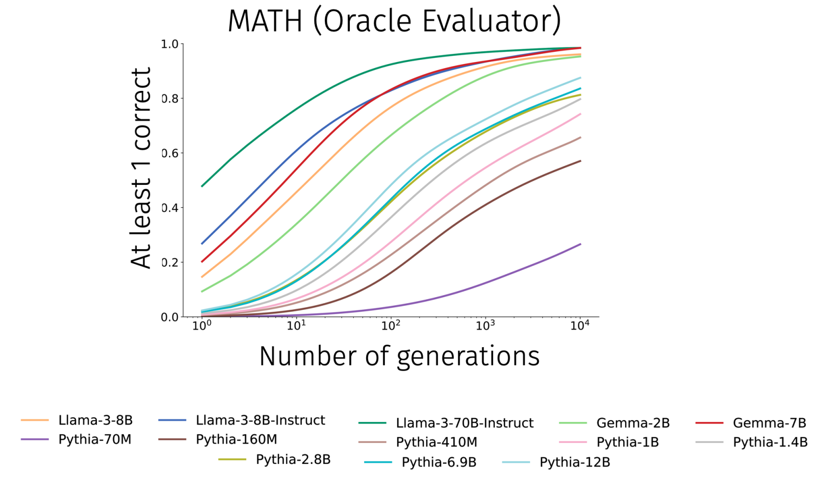
y:

Final Answer: The final answer is $\frac{2007}{2008}$.

Example: solve a math problem

Best-of-N

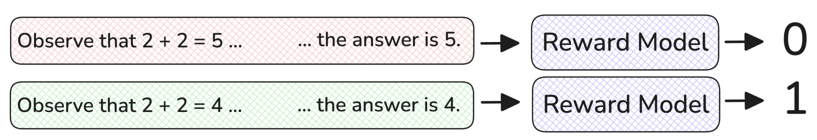
What if we had a perfect reward model $v^*(y)$?



[Brown et al., 2024]

Best-of-N | Reward Model

"Outcome reward model" $v(y) \to [0, 1] \approx R(y)$:

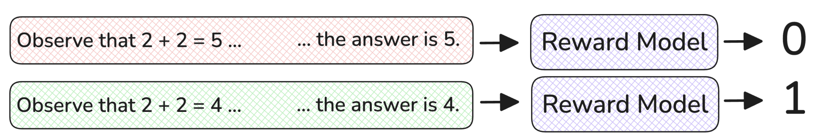


Train reward model with correct and incorrect examples.[3]

[3] E.g., [Cobbe et al., 2021]

Best-of-N | Reward Model

"Outcome reward model" $v(y) \rightarrow [0, 1] \approx R(y)$:



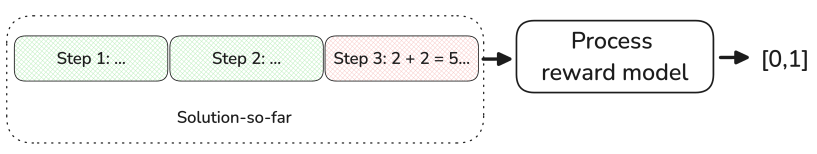
Train reward model with correct and incorrect examples.[3]

Terminology: Reward model $\approx$ evaluator $\approx$ critic $\approx$ verifier $\approx$ value $\approx$ scoring model

[3] E.g., [Cobbe et al., 2021]

Best-of-N | Reward Model

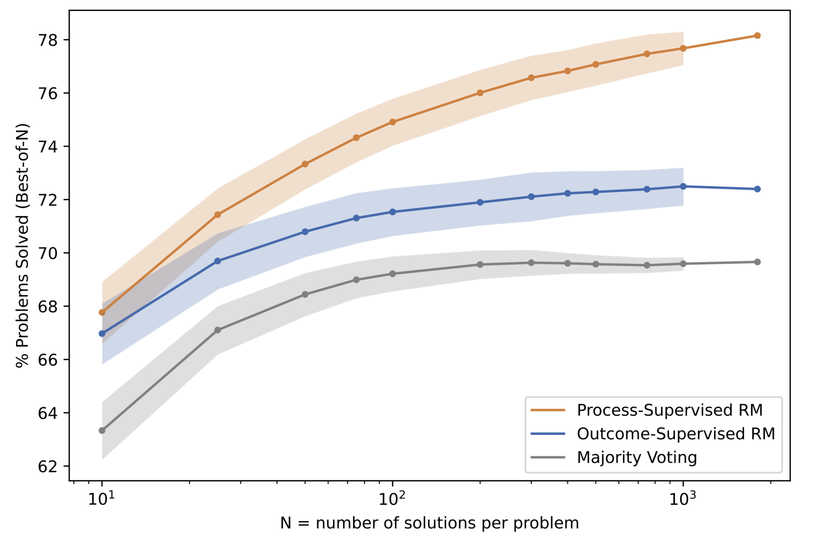
"Process reward model (PRM)"[4]



$$v(x, s_1, s_2, \ldots, s_t) \rightarrow [0, 1]$$

[4] [Uesato et al., 2022, Lightman et al., 2024, Wang et al., 2024a]

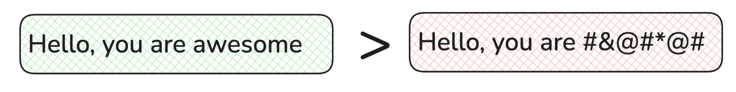
Best-of-N | Reward Model



Lightman et al 2023

Best-of-N | Reward Model

Note: Preference reward model



Train reward model with preference data.[5]

[5] E.g., [Stiennon et al., 2020]

Best-of-N

Why Best-of-N?

- Approximates maximum (true) reward:

$$
\begin{aligned}
\text{Best-of-}N &= \operatorname*{arg\,max}_{y \in \{y^{(1)}, \ldots, y^{(N)}\}} v(y) \\
&\approx \operatorname*{arg\,max}_{y} v(y) \qquad (1) \\
&\approx \operatorname*{arg\,max}_{y} R(y) \qquad (2)
\end{aligned}
$$

Best-of-N

Why Best-of-N?

- Approximates maximum (true) reward:

$$
\begin{aligned}
\text{Best-of-}N &= \operatorname*{arg\,max}_{y \in \{y^{(1)},\ldots,y^{(N)}\}} v(y) \\
&\approx \operatorname*{arg\,max}_{y} v(y) \qquad (1) \\
&\approx \operatorname*{arg\,max}_{y} R(y) \qquad (2)
\end{aligned}
$$

(1) gets better as number of generations N increases!

Best-of-N

Why Best-of-N?

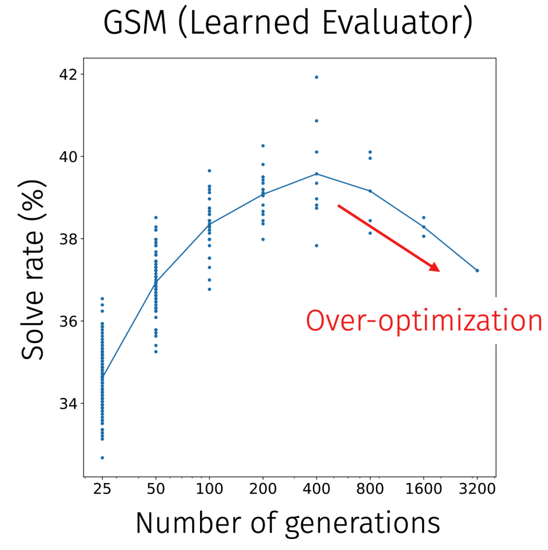
- Approximates maximum (true) reward:

$$\begin{aligned}\text{Best-of-}N &= \arg\max_{y\in\{y^{(1)},\ldots,y^{(N)}\}} v(y) \\ &\approx \arg\max_{y} v(y) \qquad (1) \\ &\approx \arg\max_{y} R(y) \qquad (2)\end{aligned}$$

(1) gets better as number of generations N increases!

(2) can suffer from imperfect reward model, aka “over-optimization”

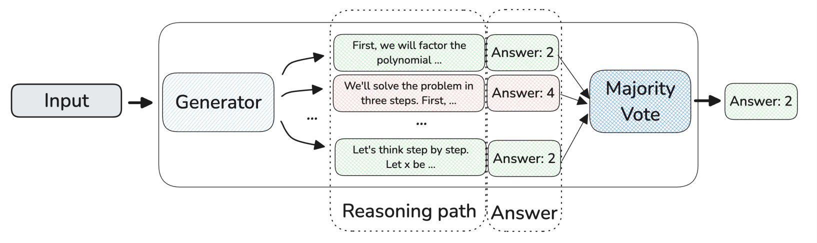
Parallel | Best-of-N[6]



[6] Plot adapted from *Training Verifiers to Solve Math Word Problems* [Cobbe et al., 2021]

Voting

Voting aggregation:[7]

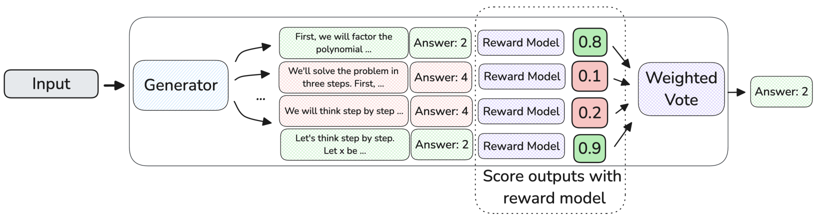


$$\arg\max_a \sum_{i=1}^{N} \mathbb{1}\{y^{(i)} = a\},$$

[7]Also called *self-consistency* [Wang et al., 2023]

Weighted voting[8]

Weighted Voting:

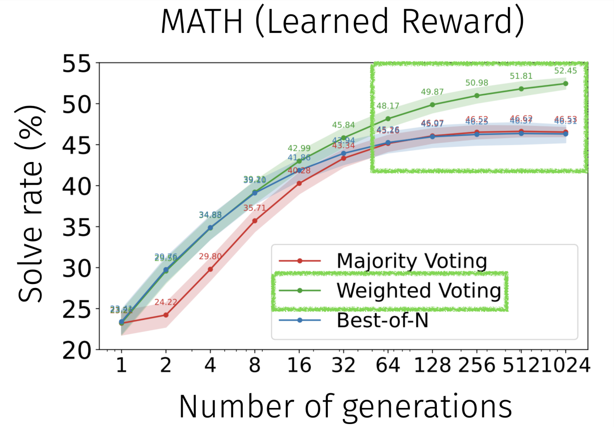


$$\arg\max_a \sum_{i=1}^{N} \underbrace{v(y^{(i)})}_{\text{reward model}} \cdot \mathbb{1}\{y^{(i)} = a\},$$

[8] [Li et al., 2023]

Voting

Can outperform Best-of-N, e.g.:[9]



[9] [Sun et al., 2024] *Easy-to-Hard Generalization: Scalable Alignment Beyond Human Supervision.*

Why (weighted) voting?

As the number of candidates $N \to \infty$, voting accuracy converges to...[10]

$$\frac{1}{M}\sum_{i=1}^{M}\mathbb{I}\left[a_i^* = \arg\max_a \underbrace{\sum_z v(x,z,a)g(z,a|x)}_{\text{“Marginalize out paths } z\text{”}}\right]$$

Notation:

- (x, z, a): (input, solution, answer)
- M: number of test examples

[10] Theorem 2, [Wu et al., 2024] *Inference Scaling Laws.* Y. Wu, Z. Sun, S. Li, S. Welleck, Y. Yang.

Why (weighted) voting?

As the number of candidates $N \to \infty$, voting accuracy converges to...[10]

$$\frac{1}{M}\sum_{i=1}^{M}\mathbb{I}\left[a_i^* = \arg\max_a \underbrace{\sum_z v(x,z,a)g(z,a|x)}_{\text{“Marginalize out paths } z\text{”}}\right]$$

Takeaway 1: Will accuracy keep improving with more samples?

- **No**, it eventually converges to the accuracy shown above

[10] Theorem 2, [Wu et al., 2024] *Inference Scaling Laws.* Y. Wu, Z. Sun, S. Li, S. Welleck, Y. Yang.

Why (weighted) voting?

As the number of candidates $N \to \infty$, voting accuracy converges to…[10]

$$\frac{1}{M}\sum_{i=1}^{M}\mathbb{I}\left[a_i^* = \arg\max_a \underbrace{\sum_z v(x,z,a)g(z,a|x)}_{\text{“Marginalize out paths } z\text{”}}\right]$$

Takeaway 2: When is weighted voting better than voting?

- When $v \cdot g$ assigns more total mass to correct answers than g

[10] Theorem 2, [Wu et al., 2024] *Inference Scaling Laws.* Y. Wu, Z. Sun, S. Li, S. Welleck, Y. Yang.

Why (weighted) voting?

As the number of candidates $N \to \infty$, voting accuracy converges to...[10]

$$\frac{1}{M}\sum_{i=1}^{M}\mathbb{I}\left[a_i^* = \arg\max_a \underbrace{\sum_z v(x,z,a)g(z,a|x)}_{\text{“Marginalize out paths } z\text{”}}\right]$$

Takeaway 3: How do we improve performance further?

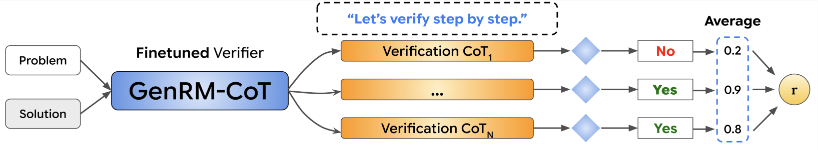
- Improve the reward model v
- Improve the generator g (better model and/or better algorithm)

---

[10] Theorem 2, [Wu et al., 2024] *Inference Scaling Laws.* Y. Wu, Z. Sun, S. Li, S. Welleck, Y. Yang.

Parallel

Improve the reward model:



Parallel generation *in the reward model too*[11]

Active area of research! (More later in this tutorial!!)

[11] [Zhang et al., 2024]

Meta-generation strategies | parallel

Parallel

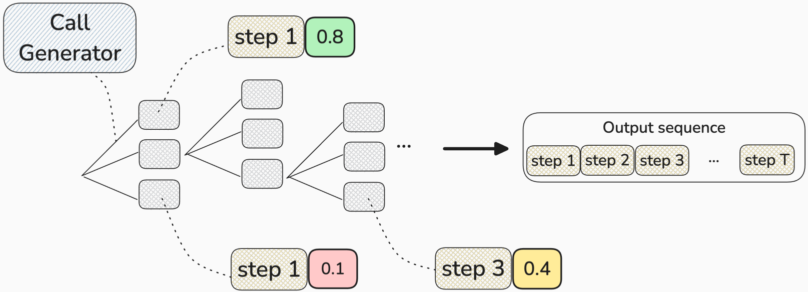
- Explores output space by generating full sequences
- Large performance gains in practice
- Bounded by the quality of the evaluator and generator

- *Next:* Can we better leverage *intermediate* evaluation?

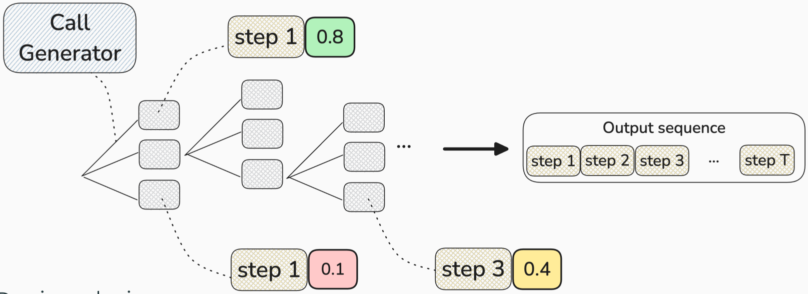
Strategies

- Strategies
  - Parallel
  - <u>Tree search</u>
  - Refinement

Tree search | basic idea



Tree search | basic idea

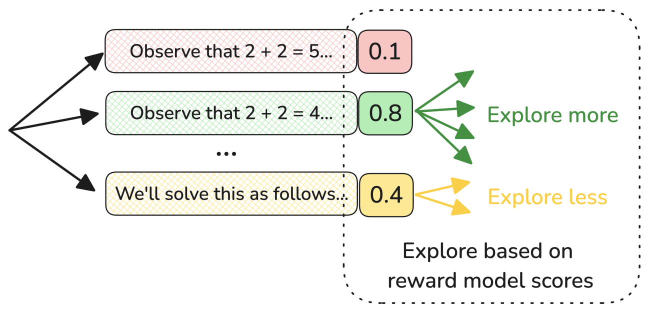


Design choices:

- States s
- Transitions $s \to s'$
- Scores $v(s)$
- Strategy (breadth-first, depth-first, ...)

Tree search | example (Rebase)

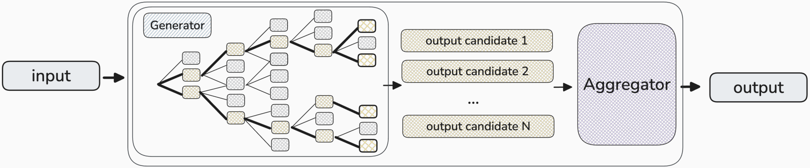
2. *Reward Balanced Search (Rebase)*[12]



$$\text{explore}_i = \text{Round}\left(\text{Budget}\frac{\exp\left(v(s_i)/\tau\right)}{\sum_j \exp\left(v(s_j)/\tau\right)}\right), \tag{3}$$

[12] [Wu et al., 2024] *Inference Scaling Laws: An Empirical Analysis of Compute-Optimal Inference.*

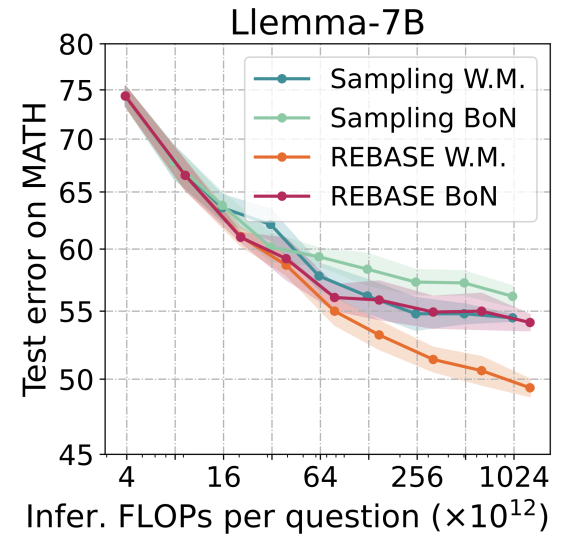
Tree search | example



Run tree search to get candidates for aggregation (e.g., voting).

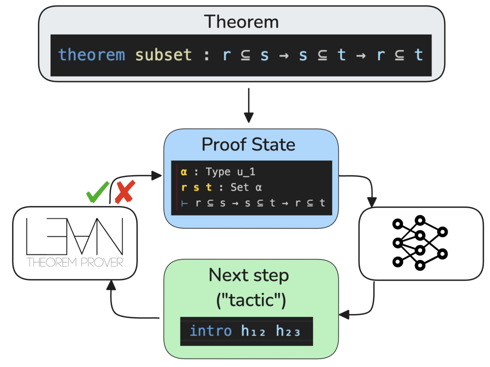
- **Key idea:** Leverages scores on *intermediate* states
  - Backtracking
  - Exploration

Tree search | examples[13]



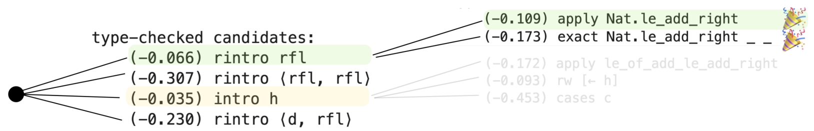
[13] [Wu et al., 2024] *Inference Scaling Laws: An Empirical Analysis of Compute-Optimal Inference.*

Tree search | examples



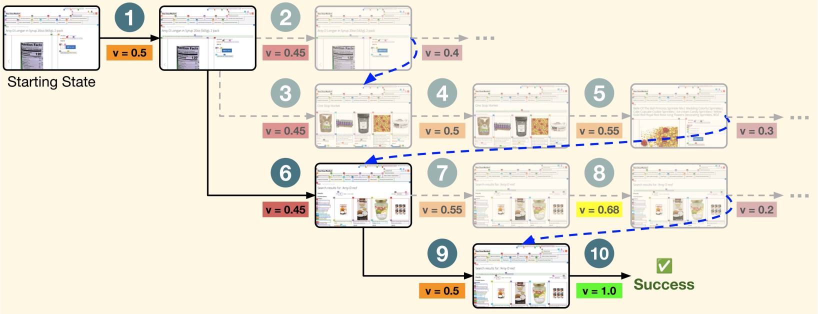
Formal theorem proving [Polu and Sutskever, 2020]

Tree search | examples



Best-first search in formal theorem proving

Tree search | examples



Best-first search in web agents [Koh et al., 2024]

Meta-generation strategies | Tree search

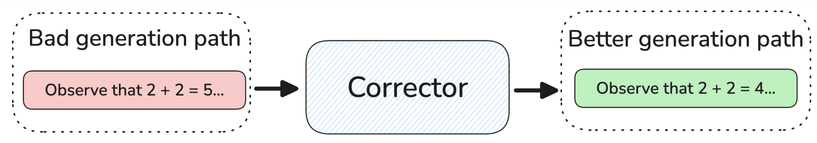
Tree-search

- Can backtrack and explore using intermediate scores
- Requires a suitable environment and value function
    - Decomposition into states
    - Good reward signal

Strategies

- Strategies
  - Parallel
  - Tree search
  - Refinement

Refinement / self-correction

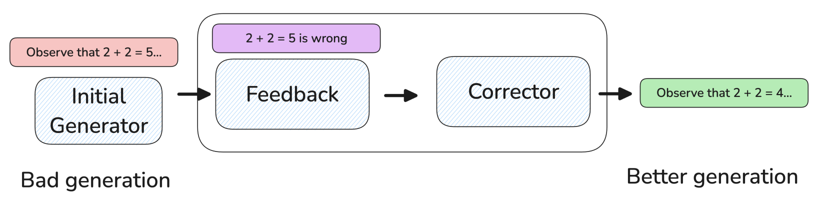


Improve a generation

Repeat:

- $y^{(i+1)} \sim g(x, y^{(i)})$

Refinement / self-correction

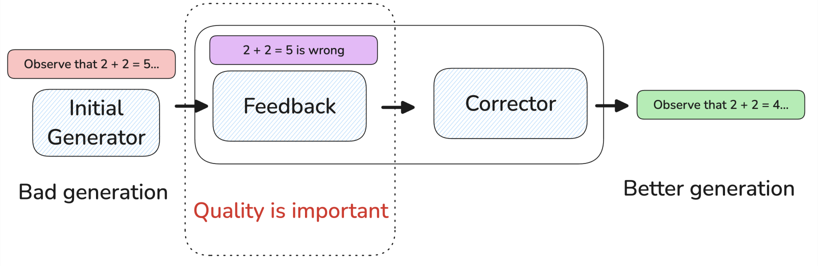


Improve a generation using feedback

Repeat:

- $y^{(i+1)} \sim g(x, y^{(i)}, F(y^{(i)}))$

Refinement / self-correction



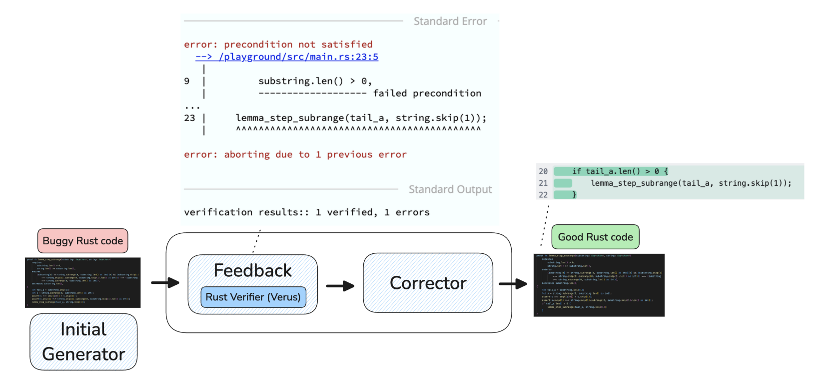
Improve a generation using feedback

Meta-generators | refinement

In practice, the quality and source of feedback is crucial:

- **Extrinsic**: external information at inference time
- **Intrinsic**: no external information at inference time

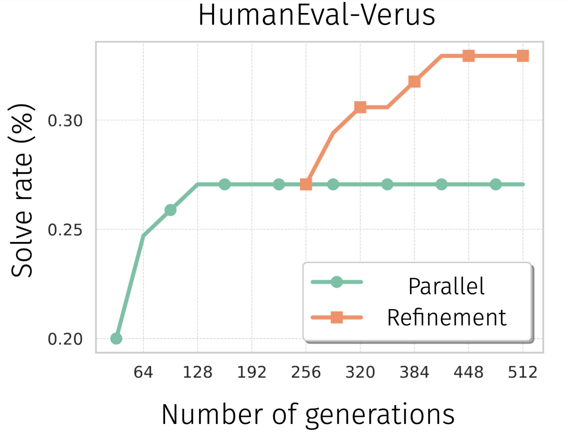
1. **Extrinsic**: external feedback



Feedback: external program verifier[14]

[14] [Aggarwal et al., 2024], *AlphaVerus*. P. Aggarwal, B. Parno, S. Welleck.

1. **Extrinsic**: external feedback



AlphaVerus. P. Aggarwal, B. Parno, S. Welleck.

Meta-generators | refinement | extrinsic

1. **Extrinsic**: external feedback

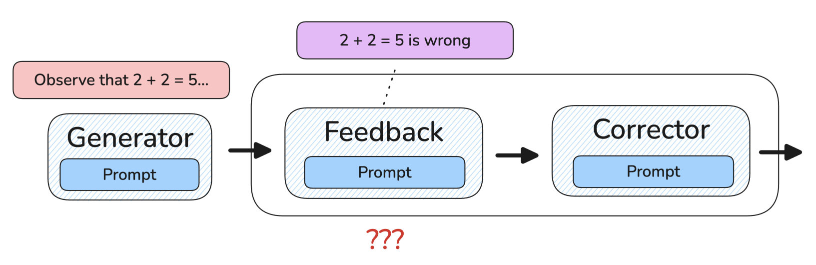
Several success cases:

- Verifiers [Aggarwal et al., 2024]
- Code interpreters [Chen et al., 2024]
- Retrievers [Asai et al., 2024]
- Tools + agent environment[14]
- ...

Intuition: adds new information, can detect and localize errors

[14] https://x.com/gneubig/status/1866172948991615177

2. **Intrinsic**: Re-prompt the same model:



Re-prompt a single LLM, e.g. [Madaan et al., 2023]

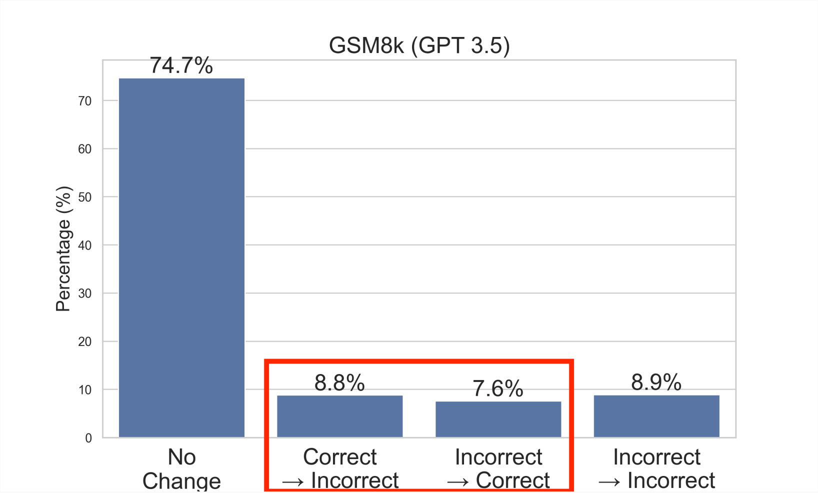
Meta-generators | refinement | intrinsic

Mixed results:

- Easy to evaluate tasks: positive [Wang et al., 2024b]
  - E.g., missing info [Asai et al., 2024]
- Mathematical reasoning: mixed[15]

[15] E.g., [Huang et al., 2024] *Large Language Models Cannot Self-Correct Reasoning Yet*

Meta-generators | refinement | intrinsic

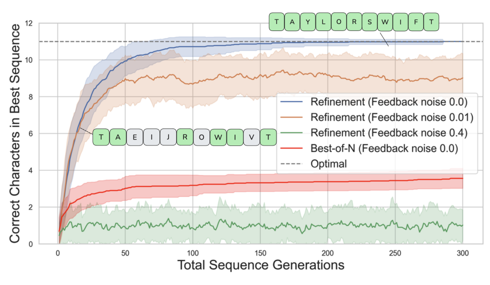


Takeaway: feedback is too noisy From [Huang et al., 2024]

Meta-generators | refinement

Generate “**TAYLORSWIFT**”

- Generator:
  - p(character)
- Feedback:
  - Incorrect characters
- Corrector:
  - Regenerate incorrect



Meta-generators | refinement / self-correction

Refinement / self-correction

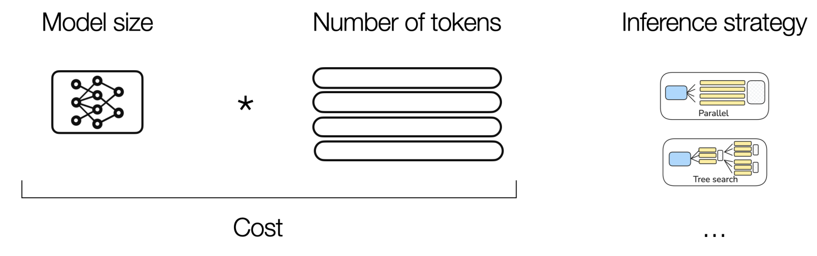
- Extrinsic
  - Positive results for environments that detect or localize errors
- Intrinsic
  - Mixed results, depends on difficulty of verification

Meta-generators | outline

- Strategies
    - Parallel
    - Tree search
    - Refinement
- *Inference scaling laws*

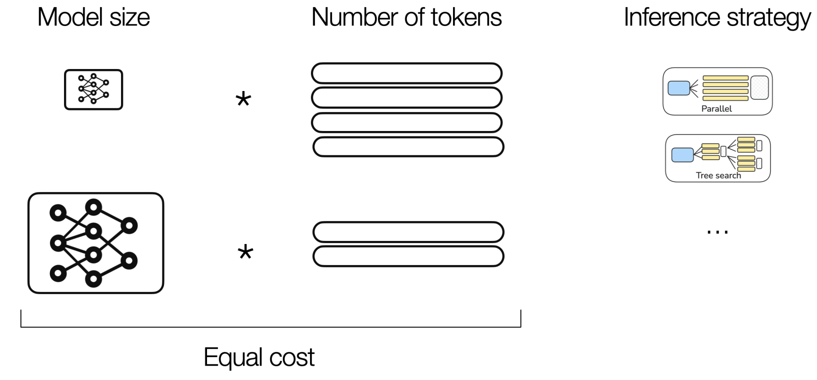
Inference scaling laws [Wu et al., 2024]

Compute is a function of model size and number of generated tokens

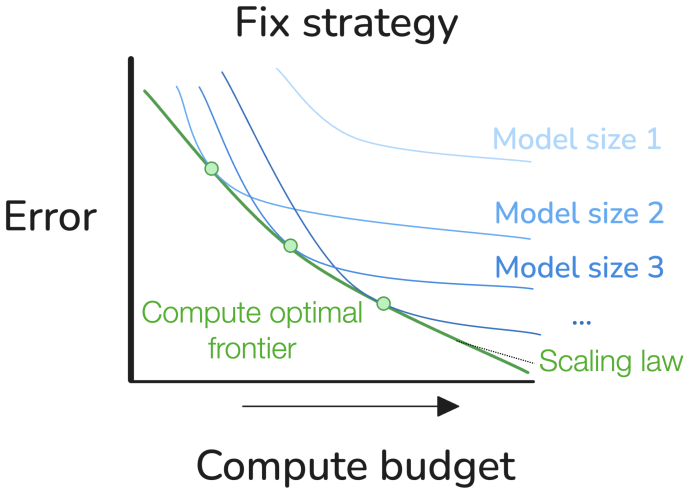


Inference scaling laws [Wu et al., 2024]

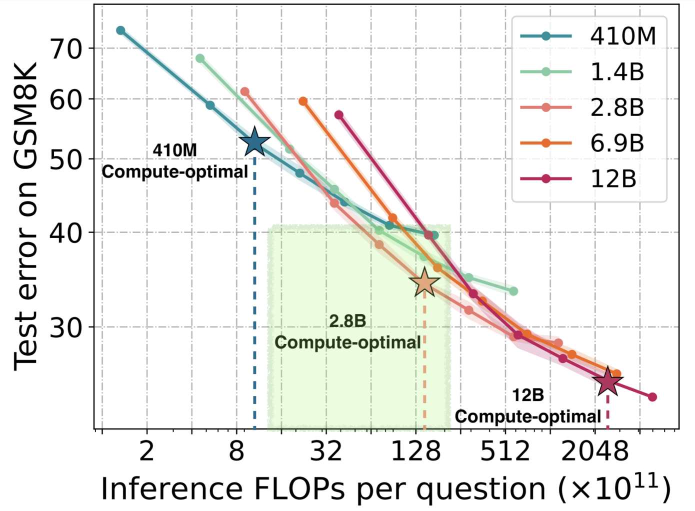
We can choose to increase model size or number of tokens



Inference scaling laws [Wu et al., 2024]



Inference scaling laws [Wu et al., 2024]



Using a smaller model and generating more is often best [Wu et al., 2024].

Inference scaling laws

Designing better strategies

- Example: design a better tree search [Wu et al., 2024]
- Example: select inference strategy based on problem difficulty [Snell et al., 2024]
- ...

Inference scaling laws | recap

- When allocated optimally, performance improves with compute
- Best model size and strategy varies with the budget
    - Sometimes smaller models are better!

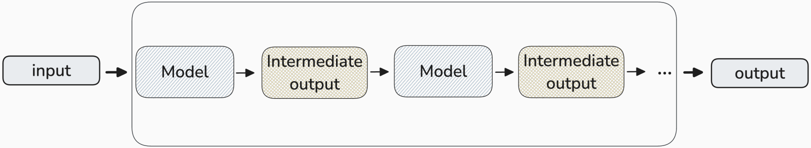
Meta-generation | recap

- Strategies for generating multiple sequences
- Parallel, tree search, refinement
- Choose methods based on task performance and cost

Today's lecture

1. Part 1: Generating multiple sequences
2. Part 2: Generating a single long sequence
    - *Long chain-of-thought*

Meta-generators | chain

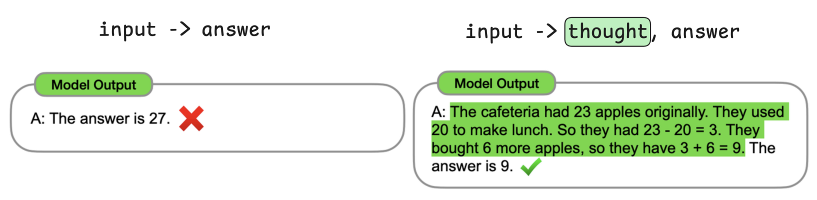


Compose generators:

$$
\begin{aligned}
y_1 &\sim g_1(x) \\
y_2 &\sim g_2(x, y_1) \\
y_3 &\sim g_3(x, y_2) \\
&\vdots
\end{aligned}
$$

Meta-generators | chain

Simple example: *Chain-of-thought* [Wei et al., 2022]:

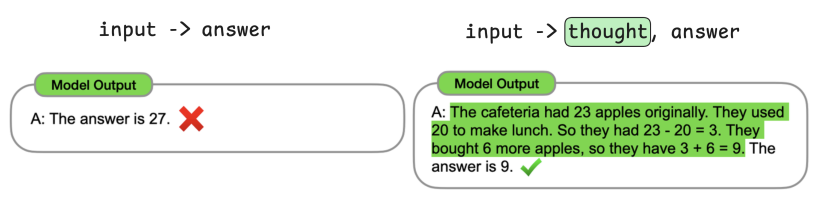


A simple decomposition:

- Generate a thought, $z \sim g(\cdot|x)$
- Generate an answer, $a \sim g(\cdot|x, z)$

Meta-generators | chain

Simple example: *Chain-of-thought* [Wei et al., 2022]:



Increases expressivity[15]

- Variable output length, analogous to a writeable tape
- Idea: train a model that searches on its own in the thought!

[15] E.g., [Feng et al., 2023, Merrill and Sabharwal, 2024, Nowak et al., 2024]

Training a model to search: Basic idea

- Train a model to generate a “thought” prior to a final output

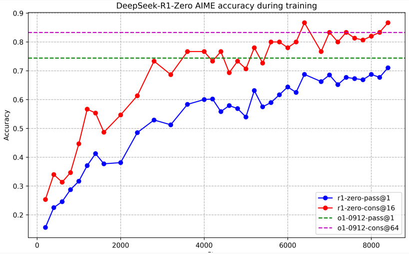
$$p_\theta(\underbrace{y}_{\text{“output”}}, \underbrace{z}_{\text{“thought”}} | x)$$

- At inference time, just sample a thought+output
- In principle, the model can learn to try alternatives, perform refinement, backtrack **within the thought**

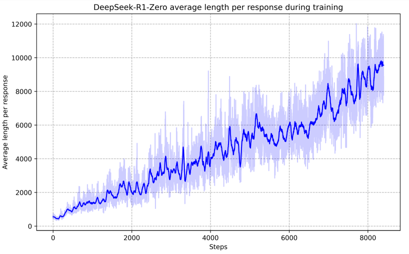
Training for long chain-of-thought [DeepSeek-AI et al., 2025]

- Approach 1: reinforcement learning
  - Policy: given a math problem x, generate a thought + answer
  - Reward: is the answer correct

Training for long chain-of-thought [DeepSeek-AI et al., 2025]



Accuracy improves during training



Response length increases to > 10,000

Training for long chain-of-thought [DeepSeek-AI et al., 2025]

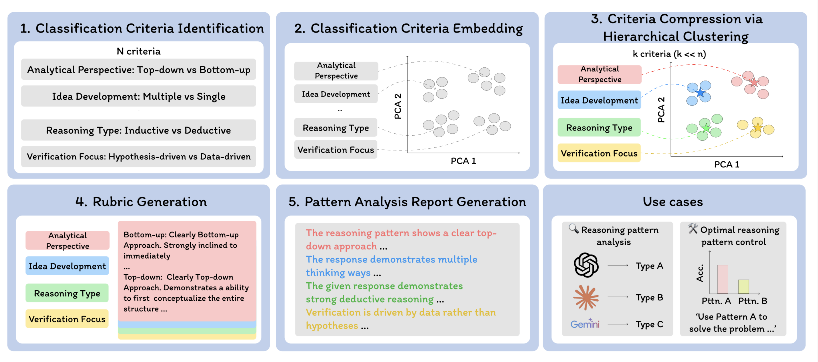
Question: If $a > 1$, then the sum of the real solutions of $\sqrt{a - \sqrt{a+x}} = x$ is equal to
Response: <think> To solve the equation $\sqrt{a - \sqrt{a+x}} = x$, let's start by squaring both $\cdots$ $\left(\sqrt{a - \sqrt{a+x}}\right)^2 = x^2 \implies a - \sqrt{a+x} = x^2.$ Rearrange to isolate the inner square root term: $(a - x^2)^2 = a + x \implies a^2 - 2ax^2 + (x^2)^2 = a + x \implies x^4 - 2ax^2 - x + (a^2 - a) = 0$ ... Wait, wait. Wait. That's an aha moment I can flag here. Let's reevaluate this step-by-step to identify if the correct sum can be $\cdots$ We started with the equation: $\sqrt{a - \sqrt{a+x}} = x$ First, let's square both sides: $a - \sqrt{a+x} = x^2 \implies \sqrt{a+x} = a - x^2$ Next, I could square both sides again, treating the equation: $\cdots$ ...

Example response

Long chain-of-thought patterns Credits: Weihua Du

1. Uncertainty
    - *Wait... / Hold on...*
    - *Wait–actually, does this formula apply here?*
2. Branching, backtracking, retrying
    - *Alternatively, generating functions could model this problem...*
    - *Revisiting...*
    - *Wait, I'm overthinking. Let's try again...*
3. Verification
    - *Let's check if we made an error. We should verify...*
    - *This is a contradiction, so we must have made a mistake.*
    - *Let's test this with...*
4. Key Points
    - *Key takeaway... / It's worth noting...*
5. Clarification
    - *In other words... / To clarify...*
6. Synthesis
    - *Ultimately... / Putting it all together...*

CoT Encyclopedia [Lee et al., 2025]

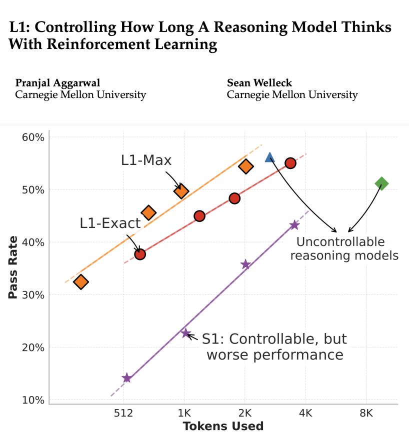


The CoT Encyclopedia: Analyzing, Predicting, and Controlling how a Reasoning Model will Think

Seongyun Lee[1,3*] Seungone Kim[2*] Minju Seo[1] Yongrae Jo[3]
Dongyoung Go[4,5] Hyeonbin Hwang[1] Jinho Park[1]
Xiang Yue[2] Sean Welleck[2] Graham Neubig[2] Moontae Lee[3] Minjoon Seo[1]
KAIST AI[1] Carnegie Mellon University[2] LG AI Research[3]
NAVER Search US[4] Cornell University[5]
{seongyun, minjoon}@kaist.ac.kr seungone@cmu.edu

Controlling the length: L1 [Aggarwal and Welleck, 2025]

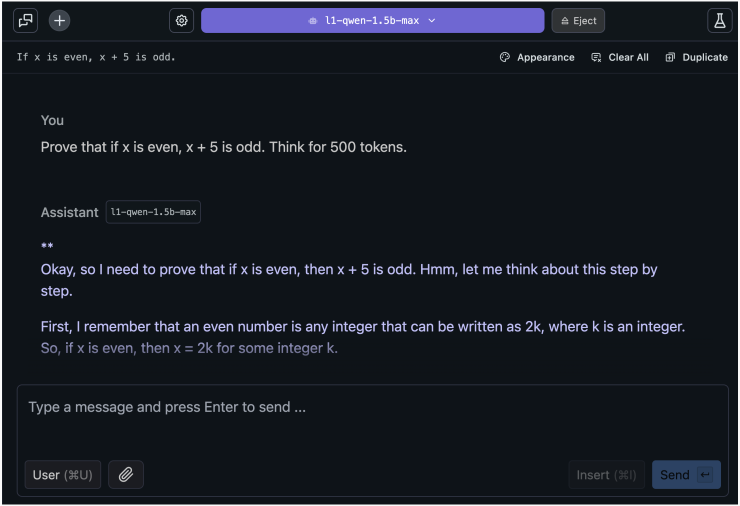
- Train model with reinforcement learning to adhere to length constraints
- E.g. “use up to 1000 tokens” provided in the prompt
- Reward: correctness and length constraint penalty


L1: Controlling How Long A Reasoning Model Thinks With Reinforcement Learning

Pranjal Aggarwal
Carnegie Mellon University

Sean Welleck
Carnegie Mellon University

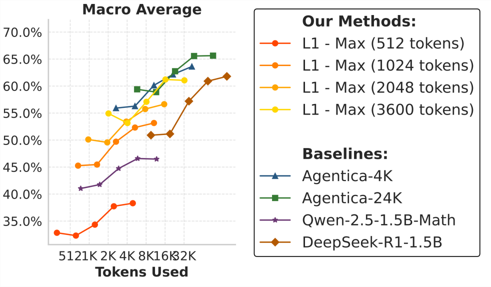
Controlling the length: L1 [Aggarwal and Welleck, 2025]



Demo: Using L1 on a laptop in LM studio

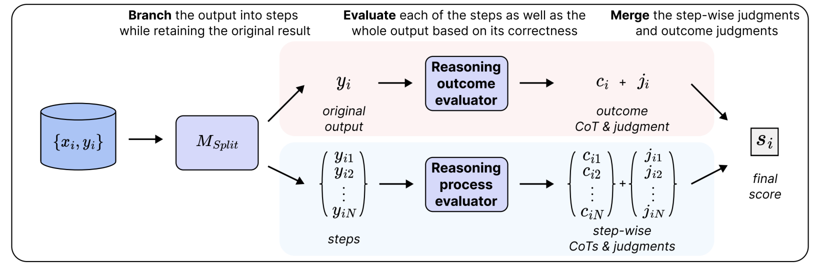
Long chain-of-thought | sequential vs. parallel

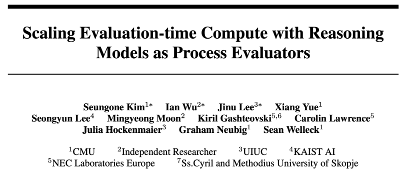
- **Sequential**: long chain-of-thought
- **Parallel**: majority voting (multiple long COTs)



[Aggarwal and Welleck, 2025]

Reasoning models as evaluators in Best-of-N [Kim et al., 2025]



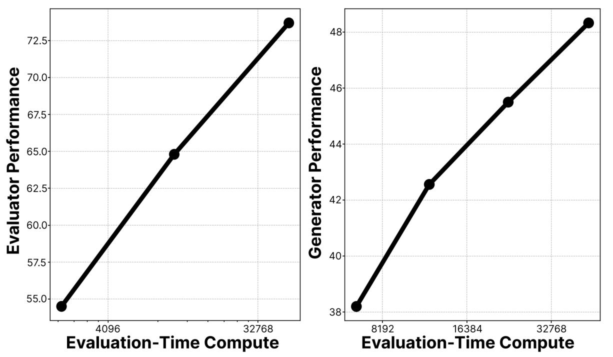


Scaling Evaluation-time Compute with Reasoning Models as Process Evaluators

Seungone Kim[1*] **Ian Wu**[2*] **Jinu Lee**[3*] **Xiang Yue**[1]
Seongyun Lee[4] **Mingyeong Moon**[2] **Kiril Gashteovski**[5,6] **Carolin Lawrence**[5]
Julia Hockenmaier[3] **Graham Neubig**[1] **Sean Welleck**[1]

[1]CMU [2]Independent Researcher [3]UIUC [4]KAIST AI
[5]NEC Laboratories Europe [7]Ss.Cyril and Methodius University of Skopje

Reasoning models as evaluators in Best-of-N [Kim et al., 2025]



Scaling Evaluation-time Compute with Reasoning Models as Process Evaluators

Seungone Kim[1*] Ian Wu[2*] Jinu Lee[3*] Xiang Yue[1]
Seongyun Lee[4] Mingyeong Moon[2] Kiril Gashteovski[5,6] Carolin Lawrence[5]
Julia Hockenmaier[3] Graham Neubig[1] Sean Welleck[1]

[1]CMU [2]Independent Researcher [3]UIUC [4]KAIST AI
[5]NEC Laboratories Europe [7]Ss.Cyril and Methodius University of Skopje

Part 2 recap

- Train a model to generate a long sequence, then use a simple inference algorithm
- Internally can perform backtracking, self-correction, etc.
- Emerging area of research!

Today's tutorial | recap

1. Test-time inference strategies take a trained model and improve performance by:
    - Generating tokens according to a strategy
    - Incorporate external information
        - Reward models
        - Environment feedback

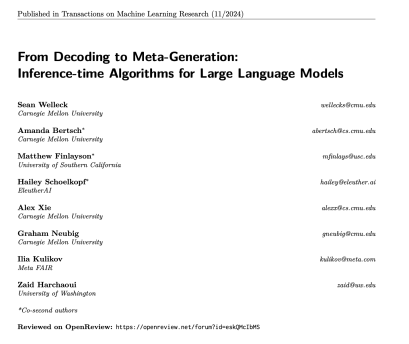
Very active and evolving research area!

Today’s tutorial | recap

1. Test-time inference strategies take a trained model and improve performance by:
    - Generating tokens according to a strategy
    - Incorporate external information
        - Reward models
        - Environment feedback
2. Two complementary meta-generation strategies
    - Call a generator to generate a thought prior to an answer
        - Long chain-of-thought
    - Call a generator multiple times in a structured way
        - parallel, tree search, refinement

Very active and evolving research area!

Additional resources



Published in Transactions on Machine Learning Research (11/2024)

From Decoding to Meta-Generation: Inference-time Algorithms for Large Language Models

Sean Welleck *wellecks@cmu.edu*
Carnegie Mellon University

Amanda Bertsch* *abertsch@cs.cmu.edu*
Carnegie Mellon University

Matthew Finlayson* *mfinlays@usc.edu*
University of Southern California

Hailey Schoelkopf* *hailey@eleuther.ai*
EleutherAI

Alex Xie *alexx@cs.cmu.edu*
Carnegie Mellon University

Graham Neubig *gneubig@cmu.edu*
Carnegie Mellon University

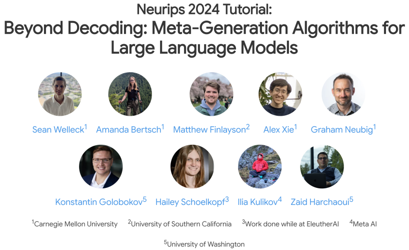
Ilia Kulikov *kulikov@meta.com*
Meta FAIR

Zaid Harchaoui *zaid@uw.edu*
University of Washington

**Co-second authors*

Reviewed on OpenReview: https://openreview.net/forum?id=eskQMcIbMS

TMLR Survey [Welleck et al., 2024]



NeurIPS 2024 LLM Inference Tutorial: https://cmu-l3.github.io/neurips2024-inference-tutorial/

Additional resources

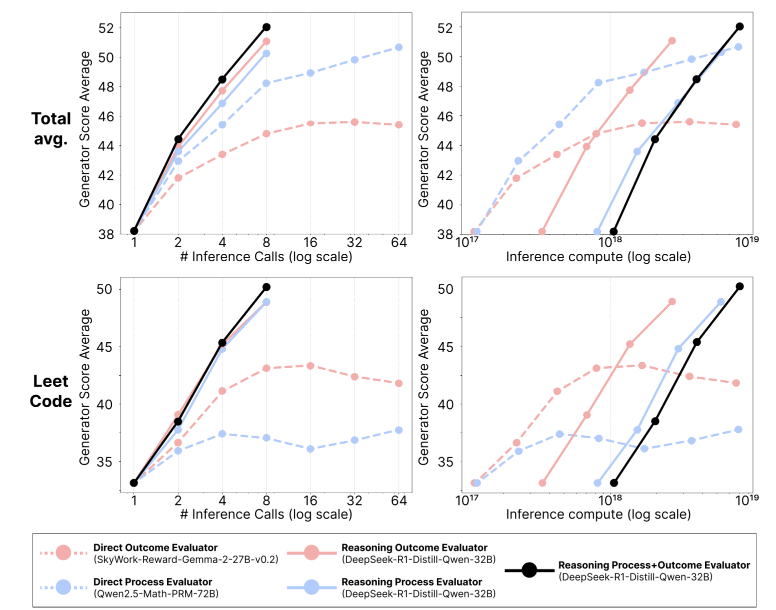
1st Workshop on Test-Time Scaling and Reasoning Models at COLM 2025
https://scalr-workshop.github.io/

Thank you!

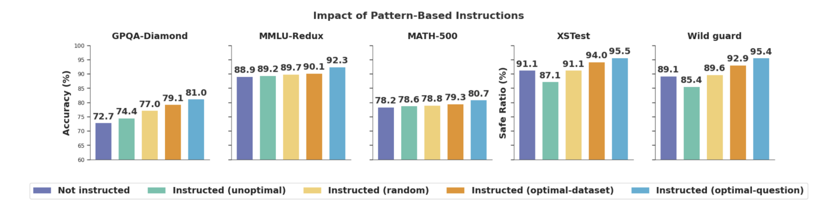
Sean Welleck
Carnegie Mellon University
Learning, Language, and Logic (L3) Lab
www.wellecks.com

Appendix

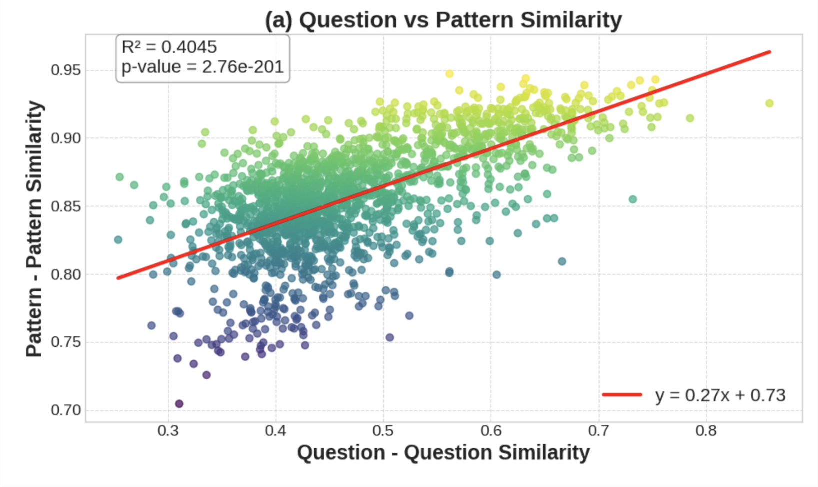
Additional results



Additional results



Additional results



References i

Aggarwal, P., Parno, B., and Welleck, S. (2024).
Alphaverus: Bootstrapping formally verified code generation through self-improving translation and treefinement.
https://arxiv.org/abs/2412.06176.

Aggarwal, P. and Welleck, S. (2025).
L1: Controlling how long a reasoning model thinks with reinforcement learning.
https://arxiv.org/abs/2503.04697.

Asai, A., He⋆, J., Shao⋆, R., Shi, W., Singh, A., Chang, J. C., Lo, K., Soldaini, L., Feldman, Tian, S., Mike, D., Wadden, D., Latzke, M., Minyang, Ji, P., Liu, S., Tong, H., Wu, B., Xiong, Y., Zettlemoyer, L., Weld, D., Neubig, G., Downey, D., Yih, W.-t., Koh, P. W., and Hajishirzi, H. (2024).
OpenScholar: Synthesizing scientific literature with retrieval-augmented language models.
Arxiv.

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.
https://arxiv.org/abs/2407.21787.

Chen, X., Lin, M., Schärli, N., and Zhou, D. (2024).
Teaching large language models to self-debug.
In *The Twelfth International Conference on Learning Representations*.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. (2021).
Training verifiers to solve math word problems.
`https://arxiv.org/abs/2110.14168`.

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Ding, H., Xin, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Wang, J., Chen, J., Yuan, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S. S., Zhou, S., Wu, S., Ye, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Zhao, W.,

Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W. L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X. Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang, X., Shan, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y. X., Xu, Y., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z., and Zhang, Z. (2025).
Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.

Feng, G., Zhang, B., Gu, Y., Ye, H., He, D., and Wang, L. (2023).
Towards revealing the mystery behind chain of thought: A theoretical perspective.
In *Thirty-seventh Conference on Neural Information Processing Systems.*

Huang, J., Chen, X., Mishra, S., Zheng, H. S., Yu, A. W., Song, X., and Zhou, D. (2024).
Large language models cannot self-correct reasoning yet.
In *The Twelfth International Conference on Learning Representations.*

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020).
Scaling laws for neural language models.
`https://arxiv.org/abs/2001.08361`.

- Kim, S., Wu, I., Lee, J., Yue, X., Lee, S., Moon, M., Gashteovski, K., Lawrence, C., Hockenmaier, J., Neubig, G., and Welleck, S. (2025). **Scaling evaluation-time compute with reasoning models as process evaluators.** https://arxiv.org/abs/2503.19877.
- Koh, J. Y., McAleer, S., Fried, D., and Salakhutdinov, R. (2024). **Tree search for language model agents.** *arXiv preprint arXiv:2407.01476.*
- Lee, S., Kim, S., Seo, M., Jo, Y., Go, D., Hwang, H., Park, J., Yue, X., Welleck, S., Neubig, G., Lee, M., and Seo, M. (2025). **The cot encyclopedia: Analyzing, predicting, and controlling how a reasoning model will think.** https://arxiv.org/abs/2505.10185.

Li, Y., Lin, Z., Zhang, S., Fu, Q., Chen, B., Lou, J.-G., and Chen, W. (2023).
Making language models better reasoners with step-aware verifier.
In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5315–5333, Toronto, Canada. Association for Computational Linguistics.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. (2024).
Let's verify step by step.
In *The Twelfth International Conference on Learning Representations*.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., Gupta, S., Majumder, B. P., Hermann, K., Welleck, S., Yazdanbakhsh, A., and Clark, P. (2023).
Self-refine: Iterative refinement with self-feedback.
In *Thirty-seventh Conference on Neural Information Processing Systems.*

Merrill, W. and Sabharwal, A. (2024).
The expressive power of transformers with chain of thought.
In *The Twelfth International Conference on Learning Representations.*

Nakano, R., Hilton, J., Balaji, S., Wu, J., Ouyang, L., Kim, C., Hesse, C., Jain, S., Kosaraju, V., Saunders, W., Jiang, X., Cobbe, K., Eloundou, T., Krueger, G., Button, K., Knight, M., Chess, B., and Schulman, J. (2022).
Webgpt: Browser-assisted question-answering with human feedback.
`https://arxiv.org/abs/2112.09332`.

Nowak, F., Svete, A., Butoi, A., and Cotterell, R. (2024).
On the representational capacity of neural language models with chain-of-thought reasoning.
In Ku, L.-W., Martins, A., and Srikumar, V., editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12510–12548, Bangkok, Thailand. Association for Computational Linguistics.

OpenAI (2024).
Learning to reason with llms.
https://openai.com/index/learning-to-reason-with-llms/.

Polu, S. and Sutskever, I. (2020).
Generative language modeling for automated theorem proving.

Snell, C., Lee, J., Xu, K., and Kumar, A. (2024).
Scaling llm test-time compute optimally can be more effective than scaling model parameters.
https://arxiv.org/abs/2408.03314.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. F. (2020).
Learning to summarize with human feedback.
In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 3008–3021. Curran Associates, Inc.

Sun, Z., Yu, L., Shen, Y., Liu, W., Yang, Y., Welleck, S., and Gan, C. (2024).
Easy-to-hard generalization: Scalable alignment beyond human supervision.
In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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.

Wang, P., Li, L., Shao, Z., Xu, R., Dai, D., Li, Y., Chen, D., Wu, Y., and Sui, Z. (2024a).
Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations.
In Ku, L.-W., Martins, A., and Srikumar, V., editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, Bangkok, Thailand. Association for Computational Linguistics.

Wang, X., Wei, J., Schuurmans, D., Le, Q. V., Chi, E. H., Narang, S., Chowdhery, A., and Zhou, D. (2023).
Self-consistency improves chain of thought reasoning in language models.
In *The Eleventh International Conference on Learning Representations*.

Wang, Y., Wu, Y., Wei, Z., Jegelka, S., and Wang, Y. (2024b).
A theoretical understanding of self-correction through in-context alignment.
`https://arxiv.org/abs/2405.18634`.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., brian ichter, Xia, F., Chi, E. H., Le, Q. V., and Zhou, D. (2022).
Chain of thought prompting elicits reasoning in large language models.
In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.

Welleck, S., Bertsch, A., Finlayson, M., Schoelkopf, H., Xie, A., Neubig, G., Kulikov, I., and Harchaoui, Z. (2024).
From decoding to meta-generation: Inference-time algorithms for large language models.
`https://arxiv.org/abs/2406.16838`.

Wu, Y., Sun, Z., Li, S., Welleck, S., and Yang, Y. (2024).
Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models.
https://arxiv.org/abs/2408.00724.

Zaharia, M., Khattab, O., Chen, L., Davis, J. Q., Miller, H., Potts, C., Zou, J., Carbin, M., Frankle, J., Rao, N., and Ghodsi, A. (2024).
The shift from models to compound ai systems.
https://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems/.

Zhang, L., Hosseini, A., Bansal, H., Kazemi, M., Kumar, A., and Agarwal, R. (2024).
Generative verifiers: Reward modeling as next-token prediction.