Beyond Decoding: Meta-Generation Algorithms for Large Language Models

Presenters: Matthew Finlayson, Hailey Schoelkopf, Sean Welleck

December 11, 2024

Meta-generators

Goal (system designer)

Design a system G that generates acceptable sequences:

$$\arg\max_G \mathbb{E}_{y \sim G(\cdot)} A(y) \tag{1}$$

Example acceptability: correctness, human preferences

Goal (system designer)

Design a system G that generates acceptable sequences:

$$\arg\max_G \mathbb{E}_{y\sim G(\cdot)} A(y) \tag{1}$$

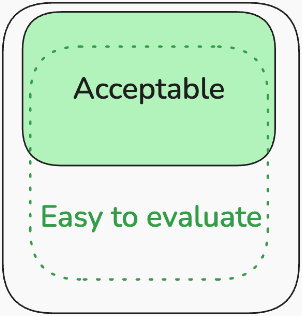
Example acceptability: correctness, human preferences

We know how to sample *probable* outputs, $y \sim p_\theta(y|x)$

- What if these outputs are not *acceptable*?

Meta-generation | Key ideas

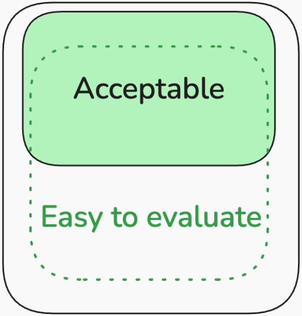
1. Take advantage of external information during generation



- Example: Learn an evaluator $v(y) \approx A(y)$ and use it in generation

Meta-generation | Key ideas

1. Take advantage of external information during generation

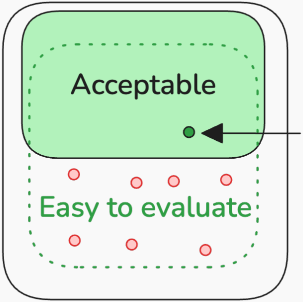


- Example: Learn an evaluator $v(y) \approx A(y)$ and use it in generation

Terminology: Evaluator ≈ critic ≈ verifier ≈ value ≈ reward model ≈ scoring model

Meta-generation | Key ideas

1. Take advantage of external information during generation
2. Call the generator more than once to search for good sequences



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

Meta-generation | Key ideas

What if we had an oracle verifier, $v(y)$?

Repeat:

- $z \sim p_\theta(z|x)$
- $y \sim p_\theta(y|x, z)$
- Stop if $v(y)$ says answer is correct

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

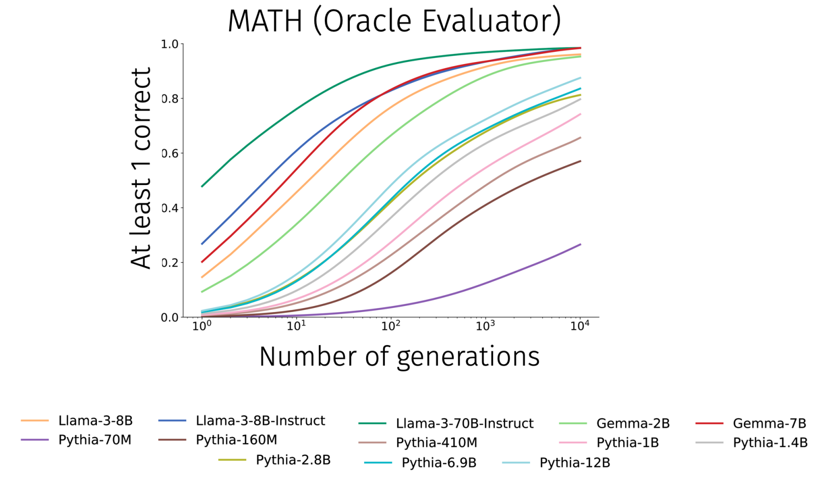
z:

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

y:

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

Meta-generation | Key ideas[1]



[1] Adapted from [Brown et al., 2024]. See also [Li et al., 2022, Cobbe et al., 2021, Jiang et al., 2023]

Meta-generation

We formalize these kinds of strategies as *meta-generators* [2]

$$y \sim G(y|x; \underbrace{g_1, g_2, \ldots, g_G}_{\text{generators}}, \underbrace{\phi}_{\text{Other parameters}})$$

Key design choices:

- G: strategy for calling generators
- $g_1, g_2, \ldots, g_G$: choice of generators
- ϕ: other models, number of tokens to generate, ...

[2] [Welleck et al., 2024] *From Decoding to Meta-Generation: Inference-time Algorithms for LLMs.* S. Welleck, A. Bertsch*, M. Finlayson*, H. Schoelkopf*, A. Xie, G. Neubig, I. Kulikov, Z. Harchaoui.

Meta-generation

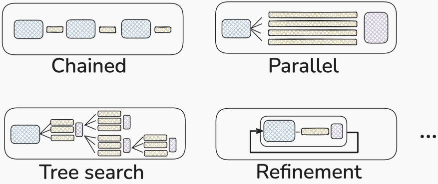
Token-level generators from part 1 are a special case of calling:

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

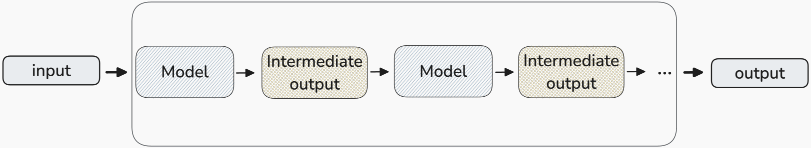
Design choices:

- g: sampling adapters, beam search, ...
- ϕ: temperature, beam width, ...

- Strategies
  - **Chain**
  - Parallel
  - Tree search
  - Refinement/Self-Correction
- Scaling meta-generators



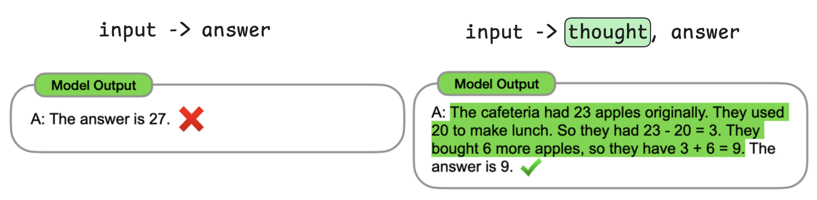
Meta-generators | chain



Compose generators:

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

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

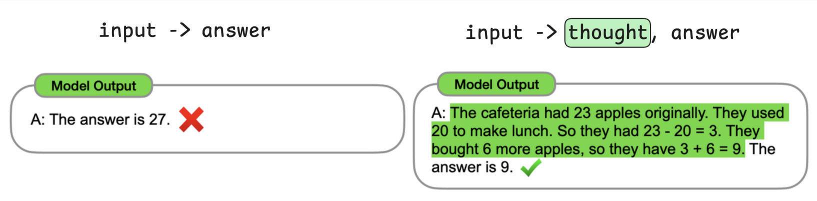


A simple decomposition:

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

Meta-generators | chain

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



Increases expressivity[3]

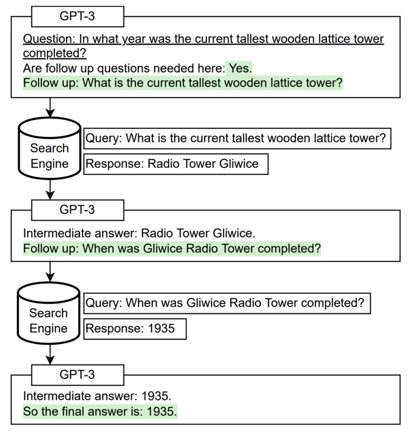
- Variable output length, analogous to a writeable tape

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

Meta-generators | chain

Extend to multiple steps:

- Each step:
    - Generate query
    - Call API
- Then generate an answer



Self-Ask [Press et al., 2023]

Meta-generators | chain[4]

View as programs:

- Outer function ≈ meta-generator
- Inner function ≈ generator

```
def search(x: Example) -> Example:
  x.hop1 = generate(hop_template)(x).pred
  x.psg1 = retrieve(x.hop1, k=1)[0]
  x.hop2 = generate(hop_template)(x).pred
  x.psg2 = retrieve(x.hop2, k=1)[0]
  return x

def predict(x: Example) -> Example:
  x.context = [x.psg1, x.psg2]
  x.pred = generate(qa_template)(x).pred
  return x
```

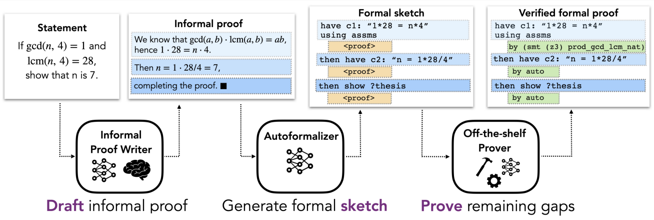
Demonstrate-Search-Predict (DSP)
[Khattab et al., 2022]

[4] [Khattab et al., 2022, Dohan et al., 2022, Schlag et al., 2023, Zheng et al., 2024]

Meta-generators | chain

Many other examples!

- Rewrite input before generating
  (*System-2 Attention* [Weston and Sukhbaatar, 2023])
- Sketch proof, fill gaps, check proof
  (*Draft-Sketch-Prove* [Jiang et al., 2023])
- ...



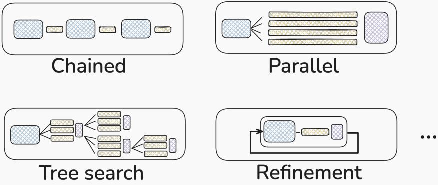
Meta-generators | chain

Chained meta-generation

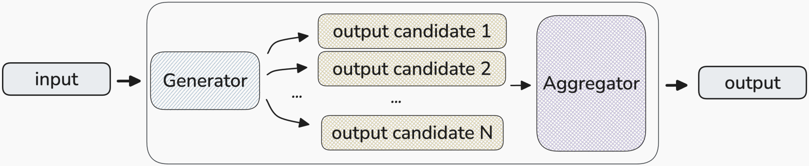
- **Key idea:** decompose generation and incorporate tools/models
- Chaining alone does not explore the output space

Meta-generators | outline

- Strategies
  - Chain
  - Parallel
  - Tree search
  - Refinement



Meta-generators | parallel



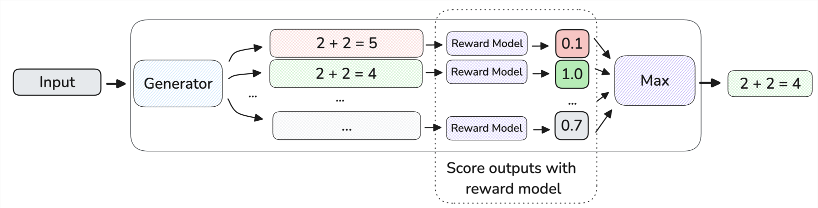
- Generate candidates:

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

- Aggregate:

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

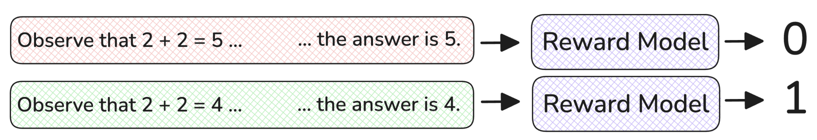
Meta-generators | parallel | Best-of-N/Rejection Sampling[5]



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

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

Reward model $v(y) \rightarrow [0, 1]$:

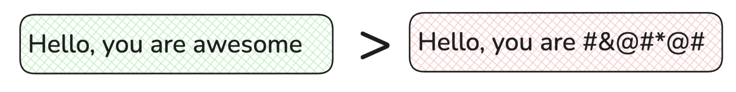


Train reward model with correct and incorrect examples.[6]

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

Meta-generators | parallel | Best-of-N/Rejection Sampling

Reward model $v(y) \to [0, 1]$:



Train reward model with preference data.[6]

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

Why Best-of-N?

- Approximates maximum acceptability:

$$
\begin{aligned}
\text{Best-of-}N &= \operatorname*{arg\,max}_{y \in \{y^{(1)},\ldots,y^{(N)}\}} v(y) \\
&\approx \operatorname*{arg\,max}_{y} v(y) && (2) \\
&\approx \operatorname*{arg\,max}_{y} A(y) && (3)
\end{aligned}
$$

Why Best-of-N?

- Approximates maximum acceptability:

$$
\begin{aligned}
\text{Best-of-}N &= \underset{y\in\{y^{(1)},\ldots,y^{(N)}\}}{\arg\max}\; v(y) \\
&\approx \arg\max_y v(y) && (2) \\
&\approx \arg\max_y A(y) && (3)
\end{aligned}
$$

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

Meta-generators | parallel | Best-of-N/Rejection Sampling

Why Best-of-N?

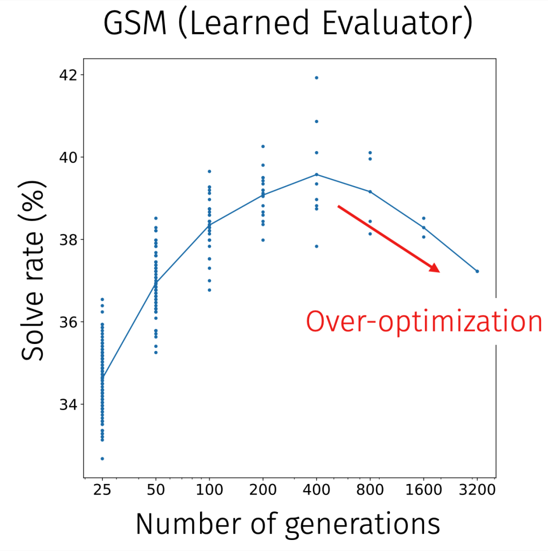
- Approximates maximum acceptability:

$$
\begin{aligned}
\text{Best-of-}N &= \arg\max_{y\in\{y^{(1)},\dots,y^{(N)}\}} v(y) \\
&\approx \arg\max_{y} v(y) \qquad (2)\\
&\approx \arg\max_{y} A(y) \qquad (3)
\end{aligned}
$$

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

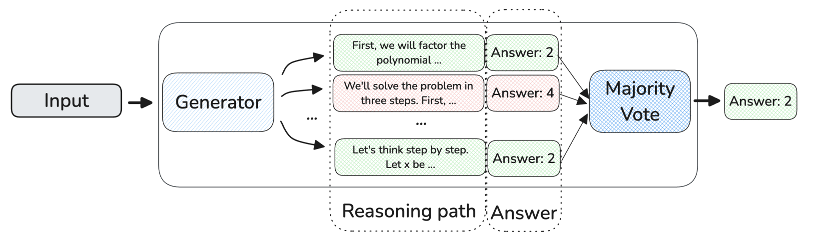
(3) Suffers from imperfect reward model, aka "over-optimization"

Meta-generators | parallel | Best-of-N / Rejection Sampling[7]



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

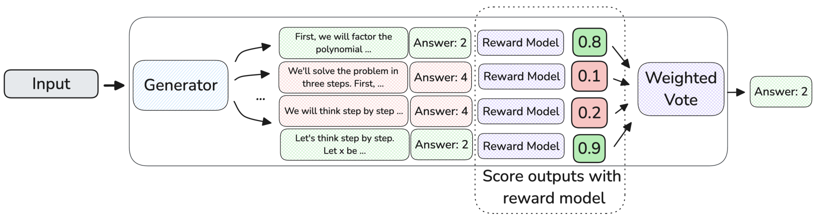
Voting aggregation:[8]



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

8[Wang et al., 2023]

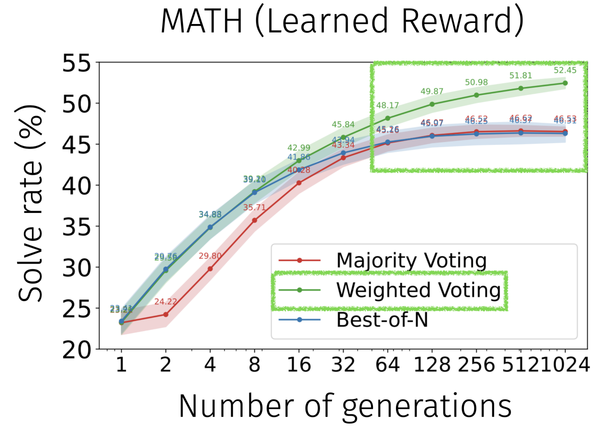
Weighted Voting:



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

[9] [Li et al., 2023b]

Can outperform Best-of-N, e.g.:[10]



[10][Sun et al., 2024] *Easy-to-Hard Generalization: Scalable Alignment Beyond Human Supervision.* Z. Sun, L. Yu, Y. Shen, W. Liu, Y. Yang, S. Welleck, C. Gan. NeurIPS 2024.

Meta-generators | parallel | why (weighted) voting?

As the number of candidates $N \to \infty$, voting accuracy converges to...[11]

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

Notation:

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

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

Meta-generators | parallel | why (weighted) voting?

As the number of candidates $N \to \infty$, voting accuracy converges to...[11]

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

Takeaway 1: Will accuracy keep improving with more samples?

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

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

As the number of candidates $N \to \infty$, voting accuracy converges to...[11]

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

Takeaway 2: When is weighted voting better than voting?

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

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

Meta-generators | parallel | why (weighted) voting?

As the number of candidates $N \to \infty$, voting accuracy converges to...[11]

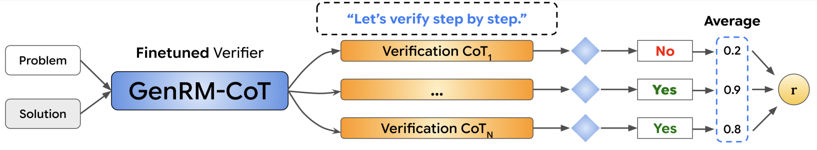
$$\frac{1}{M}\sum_{i=1}^{M}\mathbb{I}\left[a_i^* = \arg\max_a \underbrace{\sum_z v(x,z,a)g(z,a|x)}_{\text{"Marginalize out paths } z\text{"}}\right]$$

Takeaway 3: How do we improve performance further?

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

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

Improve the reward model:



Parallel generation *in the reward model too*[12]

Active area of research!

[12] [Zhang et al., 2024]

Meta-generators | parallel

Parallel meta-generators

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

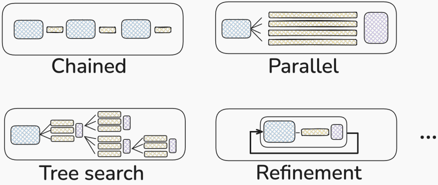
Parallel meta-generators

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

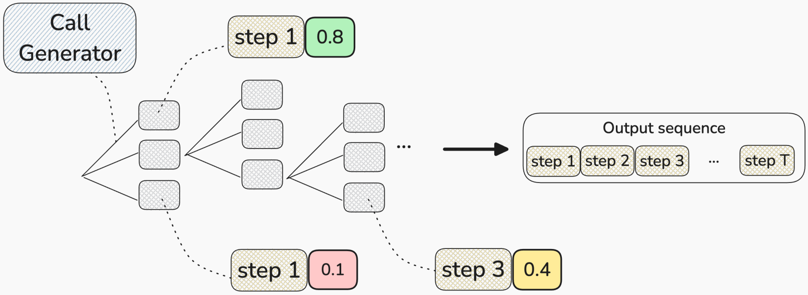
Insight: only uses the verifier at the end (on full sequences)

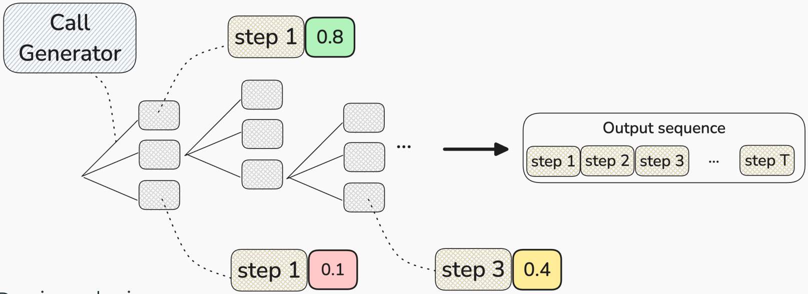
- *Next:* Can we better leverage *intermediate* evaluation?

- Strategies
    - Chain
    - Parallel
    - Tree search
    - Refinement



Meta-generators | tree search | basic idea

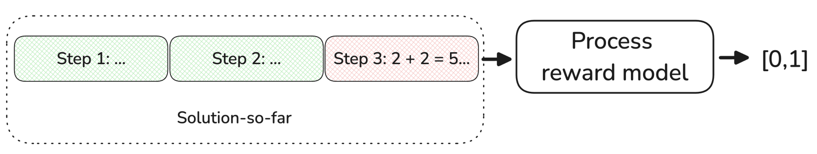




Design choices:

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

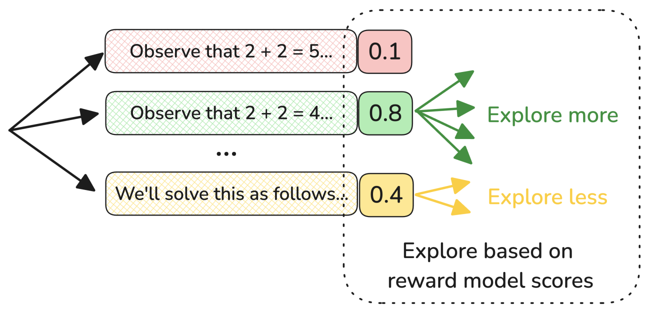
1. Scores: “process reward model (PRM)”[13]



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

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

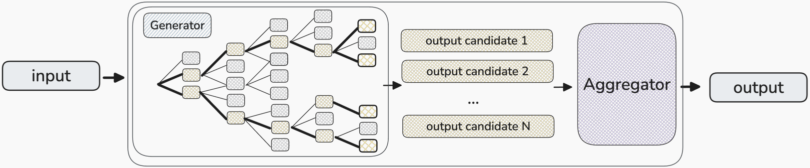
2. *Reward Balanced Search (Rebase)*[14]



$$\text{explore}_i = \text{Round}\left(\text{Budget}\frac{\exp\left(v(s_i)/\tau\right)}{\sum_j \exp\left(v(s_j)/\tau\right)}\right), \tag{4}$$

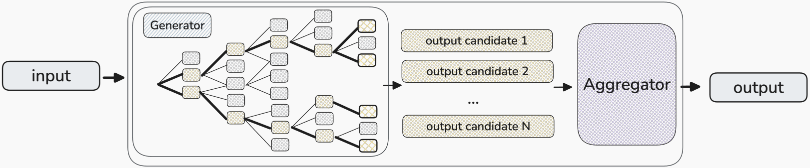
[14] [Wu et al., 2024b] *Inference Scaling Laws: An Empirical Analysis of Compute-Optimal Inference.*

Meta-generators | tree search | aggregation



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

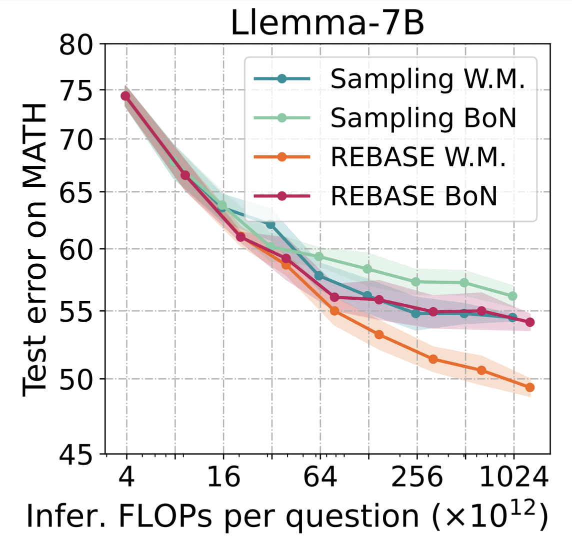
Meta-generators | tree search | aggregation



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

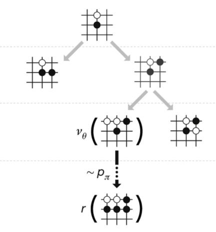
- **Key idea:** Leverages scores on *intermediate* states
  - Backtracking
  - Exploration

Meta-generators | tree search | example[15]

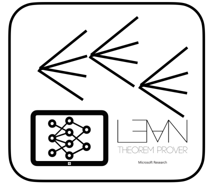


[15] [Wu et al., 2024b] *Inference Scaling Laws: An Empirical Analysis of Compute-Optimal Inference.*

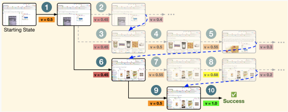
Meta-generators | tree search | example



Go [Silver et al., 2016]



Proofs [Polu and Sutskever, 2020]



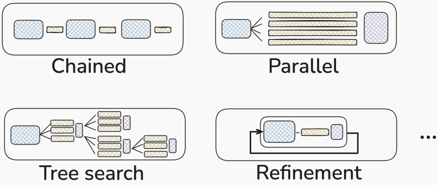
Agents [Koh et al., 2024]

Meta-generators | tree search

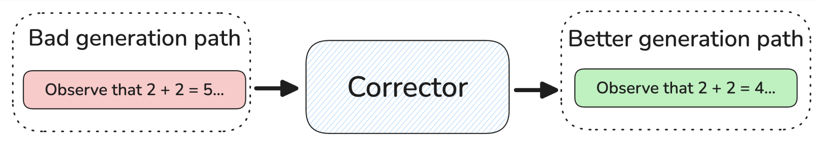
Tree-search meta-generators

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

- Strategies
  - Chain
  - Parallel
  - Tree search
  - Refinement/self-correction
- Scaling meta-generators

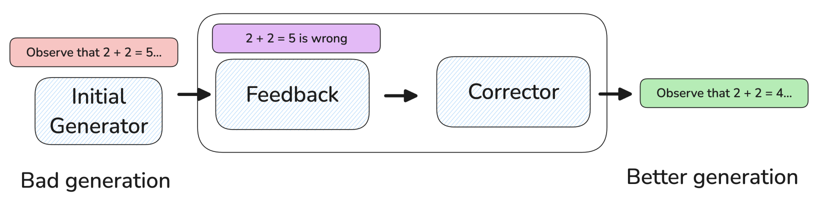


Meta-generators | refinement / self-correction



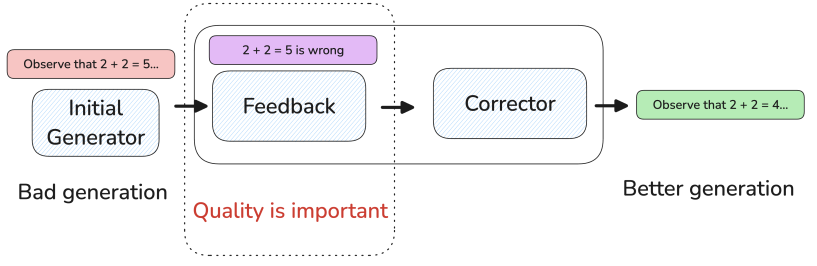
Improve a generation

Meta-generators | refinement / self-correction



Improve a generation using feedback

Meta-generators | refinement / self-correction

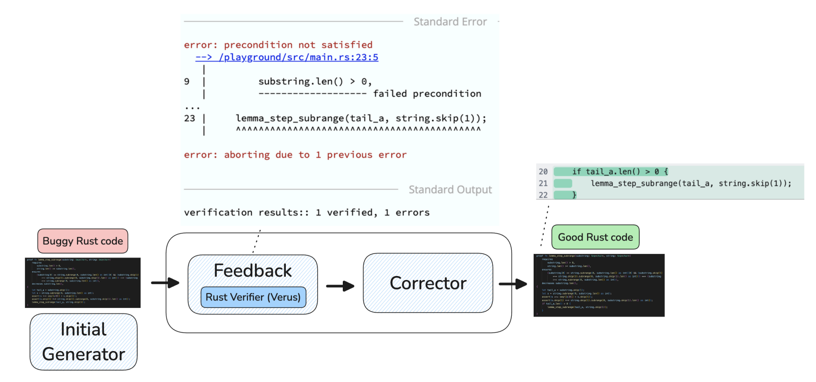


Improve a generation using feedback

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

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

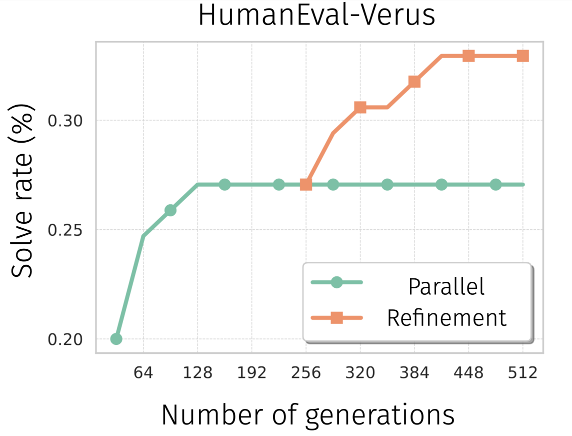
1. **Extrinsic**: external feedback



Feedback: external program verifier[16]

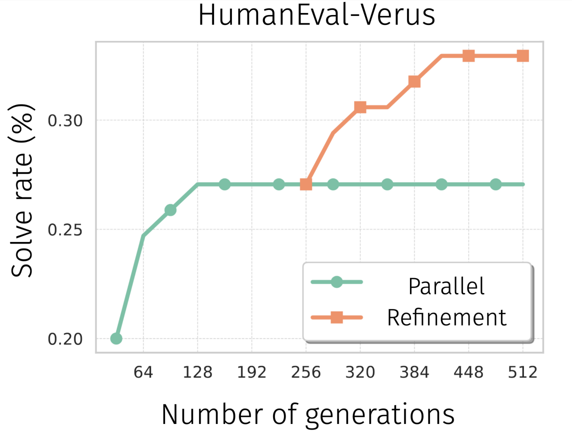
[16] [Aggarwal et al., 2024], *AlphaVerus*. P. Aggarwal, B. Parno, S. Welleck.

1. **Extrinsic**: external feedback



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

1. **Extrinsic**: external feedback



Tutorial code demo: github.com/cmu-l3/neurips2024-inference-tutorial-code

Meta-generators | refinement | extrinsic

1. **Extrinsic**: external feedback

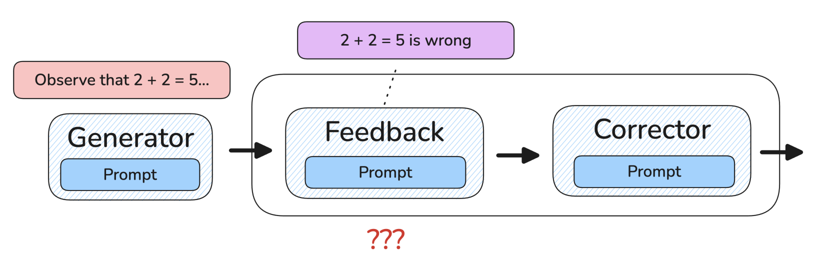
Several success cases:

- Verifiers [Aggarwal et al., 2024]
- Code interpreters [Chen et al., 2024b]
- Retrievers [Asai et al., 2024]
- Tools + agent environment[16]
- ...

Intuition: adds new information, can detect and localize errors

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

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



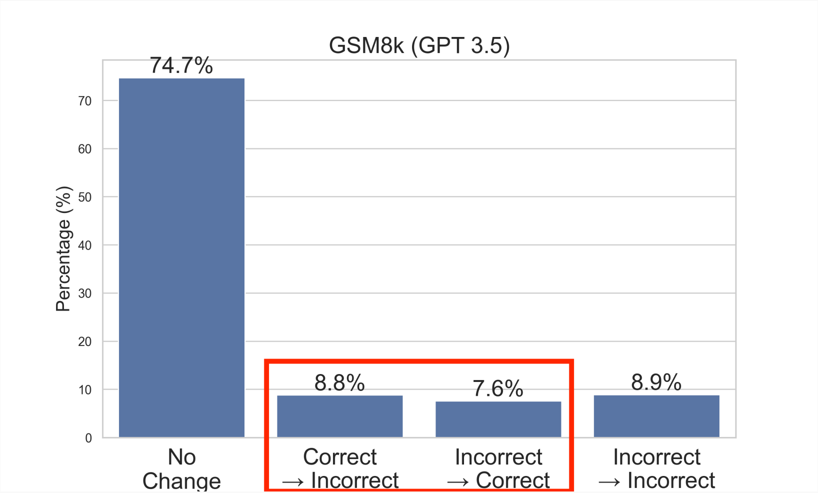
Re-prompt a single LLM, e.g. [Madaan et al., 2023]

Mixed results:

- Easy to evaluate tasks: positive [Wang et al., 2024b]
  - E.g., missing info [Asai et al., 2024]
- Mathematical reasoning: mixed[17]

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

Meta-generators | refinement | intrinsic

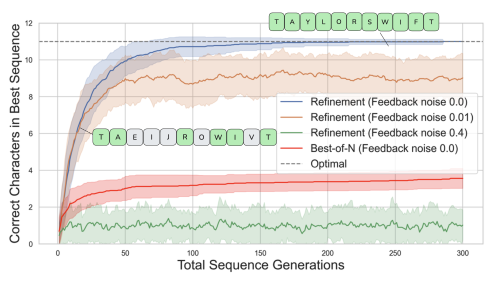


Takeaway: feedback is too noisy From [Huang et al., 2024]

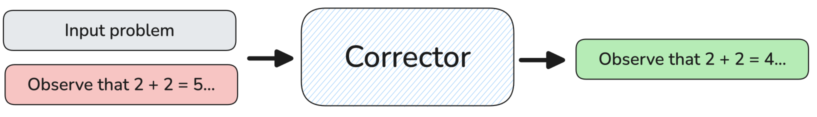
Meta-generators | refinement

Generate "TAYLORSWIFT"

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



3. **Intrinsic**: trained corrector



Directly *learn* to correct[17]

[17] [Welleck et al., 2023], *Generating Sequences by Learning to [Self-]Correct.*

Meta-generators | refinement | Case 3: fine-tuning

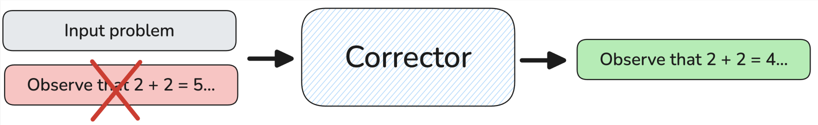
General pattern:[18]

- Collect (bad, better) pairs by generating and evaluating reward
- Update corrector $p_\theta(better|bad)$ using the collected data
- Repeat

[18] E.g., Self-corrective learning [Welleck et al., 2023], SCoRe [Kumar et al., 2024].

General pattern:[18]

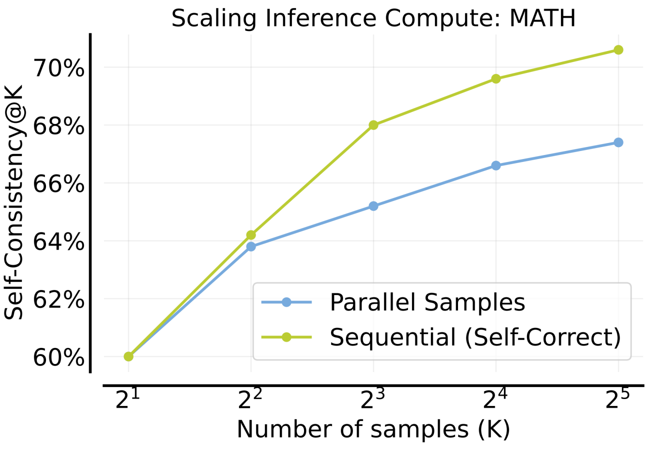
- Collect (bad, better) pairs by generating and evaluating reward
- Update corrector $p_\theta(better|bad)$ using the collected data
- Repeat



Prone to *behavior collapse*

- [Kumar et al., 2024]: overcome with regularization + RL

[18] E.g., Self-corrective learning [Welleck et al., 2023], SCoRe [Kumar et al., 2024].



From *SCoRe* [Kumar et al., 2024]

Meta-generators | refinement / self-correction

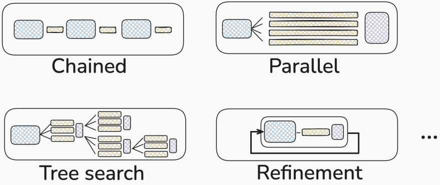
Refinement / self-correction

- Extrinsic
  - Positive results for environments that detect or localize errors
- Intrinsic, prompted
  - Mixed results, depends on difficulty of verification
- Intrinsic, trained
  - Possible improvements, requires specific training strategies

Meta-generators | outline

This talk:

- Strategies
  - Chain
  - Parallel
  - Tree search
  - Refinement
- Scaling meta-generators

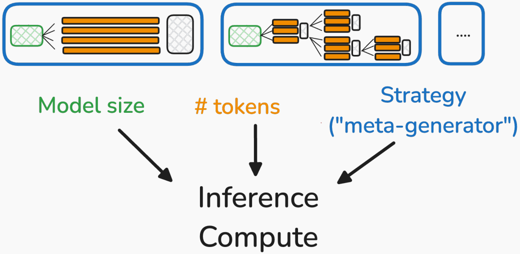


Meta-generation | how do we allocate test-time compute?

Choose strategies based on **task performance** and **compute cost**

Cost is a function of:

- Model size
- Number of generated tokens



Meta-generation | compute-optimal inference[19]

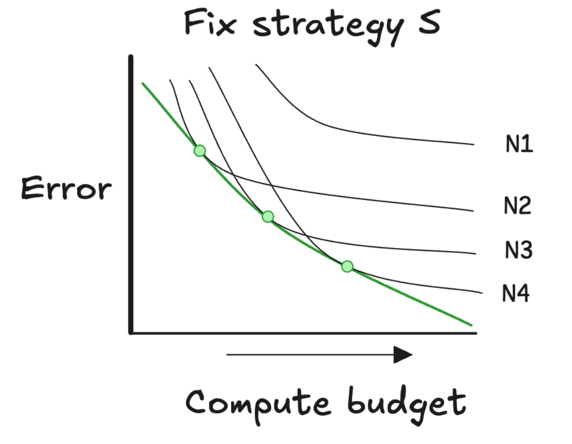
For a compute budget C:

$$\mathrm{argmin}_{N,T,S \text{ s.t. } cost(N,T,S)=C}\,\mathrm{error}(N,T,S)$$

- N: number of model parameters
- T: number of generated tokens
- S: inference strategy
- $cost(N,T,S)$: in floating-point operations

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

Meta-generation | compute-optimal inference[20]



Choose configurations on the *compute-optimal frontier* (green)

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

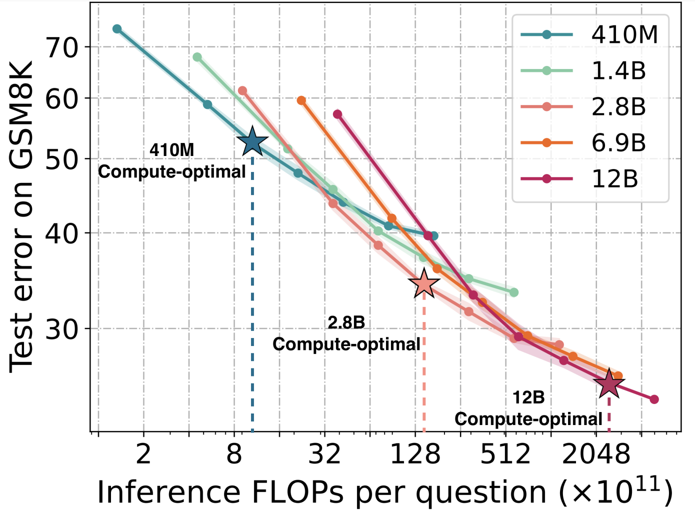
Meta-generation | how do we choose a meta-generator?

Question 1: is it better to use:

- A **small** model and more generations
- A **large** model and fewer generations

Experiment: Fix strategy, vary model size N and number of tokens T

Meta-generation | how do we choose a meta-generator?



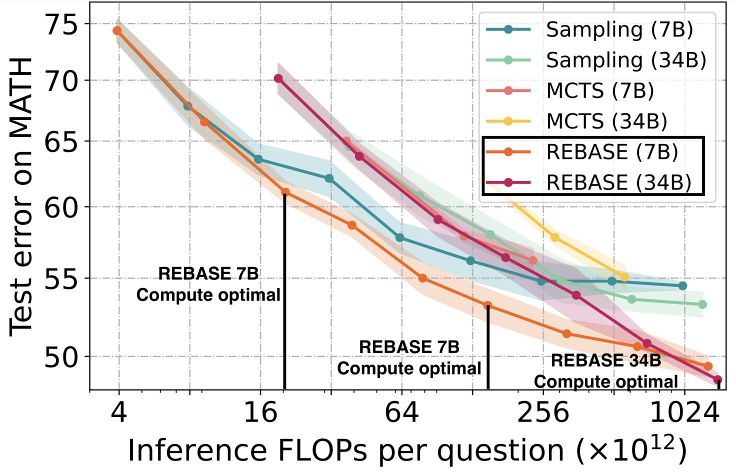
Smaller models can be compute optimal [Wu et al., 2024b].

Meta-generation | how do we choose a meta-generator?

Question 2: what is the compute-optimal meta-generation strategy?

Experiment: vary strategy (and model size and number of tokens)

Meta-generation | how do we choose a meta-generator?



Tree search (REBASE) can be compute-optimal [Wu et al., 2024b].

Meta-generation | recap

- Performance improves with increased compute…
  - … but it varies by the choice of model size and meta-generator
- The optimal model size and strategy varies with the compute budget
  - Sometimes smaller models are better!
  - Goal: design strategies that are universally optimal

Meta-generation | recap

- Meta-generators: strategies for calling generators and incorporating external information
- Several patterns: chain, parallel, tree search, refinement
- They can be combined and mixed together
- Choose and design methods based on task performance *and* cost

Meta-generation | speeding up

Next: The preceding meta-generators

- Generate many tokens
- In diverse ways (e.g., tree search)

How do we do this quickly and efficiently?

Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. (1985).
A learning algorithm for boltzmann machines.
Cognitive Science, 9(1):147–169.

Adams, G., Ladhak, F., Schoelkopf, H., and Biswas, R. (2024).
Cold compress: A toolkit for benchmarking kv cache compression approaches.

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`.

Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebrón, F., and Sanghai, S. (2023).
Gqa: Training generalized multi-query transformer models from multi-head checkpoints.

Ankner, Z., Paul, M., Cui, B., Chang, J. D., and Ammanabrolu, P. (2024).
Critique-out-loud reward models.

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.

Basu, S., Ramachandran, G. S., Keskar, N. S., and Varshney, L. R. (2021).
Mirostat: a neural text decoding algorithm that directly controls perplexity.
In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Bertsch, A., Xie, A., Neubig, G., and Gormley, M. (2023).
It's MBR all the way down: Modern generation techniques through the lens of minimum Bayes risk.
In Elazar, Y., Ettinger, A., Kassner, N., Ruder, S., and A. Smith, N., editors, *Proceedings of the Big Picture Workshop*, pages 108–122, Singapore. Association for Computational Linguistics.

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, J., Tiwari, V., Sadhukhan, R., Chen, Z., Shi, J., Yen, I. E.-H., and Chen, B. (2024a).
Magicdec: Breaking the latency-throughput tradeoff for long context generation with speculative decoding.

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

References v

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Castro-Ros, A., Pellat, M., Robinson, K., Valter, D., Narang, S., Mishra, G., Yu, A., Zhao, V., Huang, Y., Dai, A., Yu, H., Petrov, S., Chi, E. H., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q. V., and Wei, J. (2022).
Scaling instruction-finetuned language models.
`https://arxiv.org/abs/2210.11416`.

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`.

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and R'e, C. (2022).
Flashattention: Fast and memory-efficient exact attention with io-awareness.
ArXiv preprint, abs/2205.14135.

Dohan, D., Xu, W., Lewkowycz, A., Austin, J., Bieber, D., Lopes, R. G., Wu, Y., Michalewski, H., Saurous, R. A., Sohl-dickstein, J., Murphy, K., and Sutton, C. (2022).
Language model cascades.
`https://arxiv.org/abs/2207.10342`.

Fan, A., Lewis, M., and Dauphin, Y. (2018).
Hierarchical neural story generation.
In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898. Association for Computational Linguistics.

Fedus, W., Zoph, B., and Shazeer, N. (2022).
Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity.

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*.

Finlayson, M., Hewitt, J., Koller, A., Swayamdipta, S., and Sabharwal, A. (2024).
Closing the curious case of neural text degeneration.
In *The Twelfth International Conference on Learning Representations*.

Freitag, M. and Al-Onaizan, Y. (2017).
Beam search strategies for neural machine translation.
In *Proceedings of the First Workshop on Neural Machine Translation*, pages 56–60. Association for Computational Linguistics.

He, H. (2022).
Making deep learning go brrrr from first principles.

Hewitt, J., Manning, C., and Liang, P. (2022).
Truncation sampling as language model desmoothing.
In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3414–3427. Association for Computational Linguistics.

Hobbhahn, M., Heim, L., and Aydos, G. (2023).
Trends in machine learning hardware.
Accessed: 2024-11-26.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. (2020).
The curious case of neural text degeneration.
In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.
OpenReview.net.

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*.

Jiang, A. Q., Welleck, S., Zhou, J. P., Lacroix, T., Liu, J., Li, W., Jamnik, M., Lample, G., and Wu, Y. (2023).
Draft, sketch, and prove: Guiding formal theorem provers with informal proofs.
In *The Eleventh International Conference on Learning Representations*.

Juravsky, J., Brown, B., Ehrlich, R., Fu, D. Y., Ré, C., and Mirhoseini, A. (2024).
Hydragen: High-throughput llm inference with shared prefixes.

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`.

Khattab, O., Santhanam, K., Li, X. L., Hall, D. L. W., Liang, P., Potts, C., and Zaharia, M. A. (2022).
Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp.
ArXiv, abs/2212.14024.

Kim, S., Suk, J., Longpre, S., Lin, B. Y., Shin, J., Welleck, S., Neubig, G., Lee, M., Lee, K., and Seo, M. (2024).
Prometheus 2: An open source language model specialized in evaluating other language models.
`https://arxiv.org/abs/2405.01535`.

Koh, J. Y., McAleer, S., Fried, D., and Salakhutdinov, R. (2024).
Tree search for language model agents.
arXiv preprint arXiv:2407.01476.

Kudo, T. (2018).
Subword regularization: Improving neural network translation models with multiple subword candidates.
In Gurevych, I. and Miyao, Y., editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Kumar, A., Zhuang, V., Agarwal, R., Su, Y., Co-Reyes, J. D., Singh, A., Baumli, K., Iqbal, S., Bishop, C., Roelofs, R., Zhang, L. M., McKinney, K., Shrivastava, D., Paduraru, C., Tucker, G., Precup, D., Behbahani, F., and Faust, A. (2024).
Training language models to self-correct via reinforcement learning.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. (2023).
Efficient memory management for large language model serving with pagedattention.

Li, X. L., Holtzman, A., Fried, D., Liang, P., Eisner, J., Hashimoto, T., Zettlemoyer, L., and Lewis, M. (2023a).
Contrastive decoding: Open-ended text generation as optimization.
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 12286–12312. Association for Computational Linguistics.

- Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., Eccles, T., Keeling, J., Gimeno, F., Lago, A. D., Hubert, T., Choy, P., de Masson d'Autume, C., Babuschkin, I., Chen, X., Huang, P.-S., Welbl, J., Gowal, S., Cherepanov, A., Molloy, J., Mankowitz, D. J., Robson, E. S., Kohli, P., de Freitas, N., Kavukcuoglu, K., and Vinyals, O. (2022).
  Competition-level code generation with alphacode.
  Science, 378(6624):1092–1097.
- Li, Y., Lin, Z., Zhang, S., Fu, Q., Chen, B., Lou, J.-G., and Chen, W. (2023b).
  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*.

Liu, A., Han, X., Wang, Y., Tsvetkov, Y., Choi, Y., and Smith, N. A. (2024).
Tuning language models by proxy.

Liu, A., Sap, M., Lu, X., Swayamdipta, S., Bhagavatula, C., Smith, N. A., and Choi, Y. (2021).
DExperts: Decoding-time controlled text generation with experts and anti-experts.
In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6691–6706. Association for Computational Linguistics.

Lu, X., Brahman, F., West, P., Jung, J., Chandu, K., Ravichander, A., Ammanabrolu, P., Jiang, L., Ramnath, S., Dziri, N., Fisher, J., Lin, B., Hallinan, S., Qin, L., Ren, X., Welleck, S., and Choi, Y. (2023).
Inference-time policy adapters (IPA): Tailoring extreme-scale LMs without fine-tuning.
In Bouamor, H., Pino, J., and Bali, K., editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6863–6883. Association for Computational Linguistics.

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*.

Meister, C., Cotterell, R., and Vieira, T. (2020).
If beam search is the answer, what was the question?
In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2173–2185. Association for Computational Linguistics.

Meister, C., Pimentel, T., Wiher, G., and Cotterell, R. (2022).
Locally typical sampling.
Transactions of the Association for Computational Linguistics, 11:102–121.

Meister, C., Pimentel, T., Wiher, G., and Cotterell, R. (2023).
Locally typical sampling.
Transactions of the Association for Computational Linguistics, 11:102–121.

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.

Nebius (2024).
Leveraging training and search for better software engineering agents.
https://nebius.com/blog/posts/training-and-search-for-software-engineering-agents.

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.

- Press, O., Zhang, M., Min, S., Schmidt, L., Smith, N., and Lewis, M. (2023).
  Measuring and narrowing the compositionality gap in language models.
  In Bouamor, H., Pino, J., and Bali, K., editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711. Association for Computational Linguistics.
- Schlag, I., Sukhbaatar, S., Celikyilmaz, A., tau Yih, W., Weston, J., Schmidhuber, J., and Li, X. (2023).
  Large language model programs.
  `https://arxiv.org/abs/2305.05364`.
- Shazeer, N. (2019).
  Fast transformer decoding: One write-head is all you need.

Shi, C., Yang, H., Cai, D., Zhang, Z., Wang, Y., Yang, Y., and Lam, W. (2024).
A thorough examination of decoding methods in the era of LLMs.
In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N., editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8601–8629, Miami, Florida, USA. Association for Computational Linguistics.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016).

Mastering the game of go with deep neural networks and tree search.
Nature, 529:484–503.

Stahlberg, F. and Byrne, B. (2019).
On nmt search errors and model errors: Cat got your tongue?
ArXiv, abs/1908.10090.

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.

Welleck, S., Kulikov, I., Roller, S., Dinan, E., Cho, K., and Weston, J. (2020).
Neural text generation with unlikelihood training.
In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

- Welleck, S., Lu, X., West, P., Brahman, F., Shen, T., Khashabi, D., and Choi, Y. (2023).
  Generating sequences by learning to self-correct.
  In *The Eleventh International Conference on Learning Representations*.
- Weston, J. and Sukhbaatar, S. (2023).
  System 2 attention (is something you might need too).
- Wu, I., Fernandes, P., Bertsch, A., Kim, S., Pakazad, S., and Neubig, G. (2024a).
  Better instruction-following through minimum bayes risk.
  `https://arxiv.org/abs/2410.02902`.

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

Xia, H., Yang, Z., Dong, Q., Wang, P., Li, Y., Ge, T., Liu, T., Li, W., and Sui, Z. (2024).
Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding.

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.

Zheng, L., Yin, L., Xie, Z., Sun, C., Huang, J., Yu, C. H., Cao, S., Kozyrakis, C., Stoica, I., Gonzalez, J. E., Barrett, C., and Sheng, Y. (2024).
Sglang: Efficient execution of structured language model programs.