Beyond Decoding: Meta-Generation Algorithms for Large Language Models

Presenters: Matthew Finlayson, Hailey Schoelkopf, Sean Welleck

December 11, 2024

I. Primitive Generators

I. Primitive Generators

Generating one token at a time

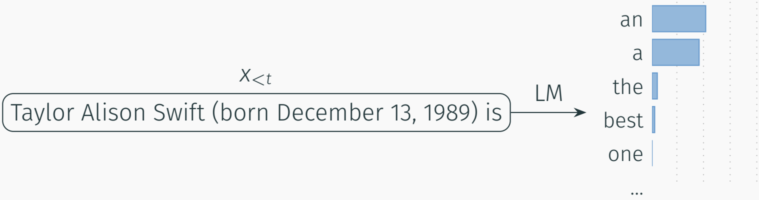
This tutorial

Beyond Decoding: Meta-Generation Algorithms for LLMs

- Primitive Generators
- Meta-generators
- Efficient meta-generation

Token-level generation

Auto-regressive language modeling uses a causal language model, which defines a conditional distribution over tokens $p_\theta[x_t \mid x_{<t}]$.



Token-level generation

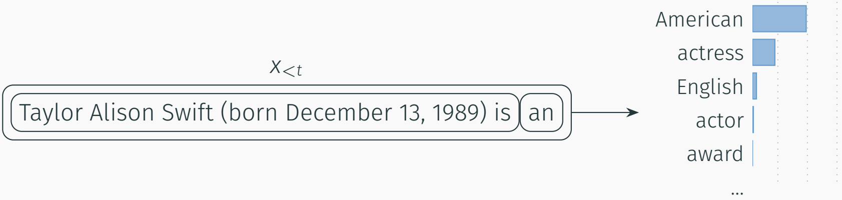
Auto-regressive language modeling uses a causal language model, which defines a conditional distribution over tokens $p_\theta[x_t \mid x_{<t}]$.

$x_{<t}$ x_t

Taylor Alison Swift (born December 13, 1989) is | an

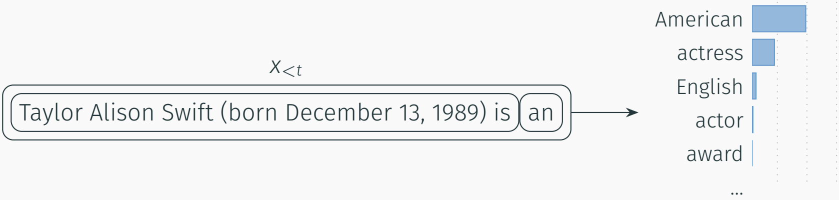
Token-level generation

Auto-regressive language modeling uses a causal language model, which defines a conditional distribution over tokens $p_\theta[x_t \mid x_{<t}]$.



Token-level generation

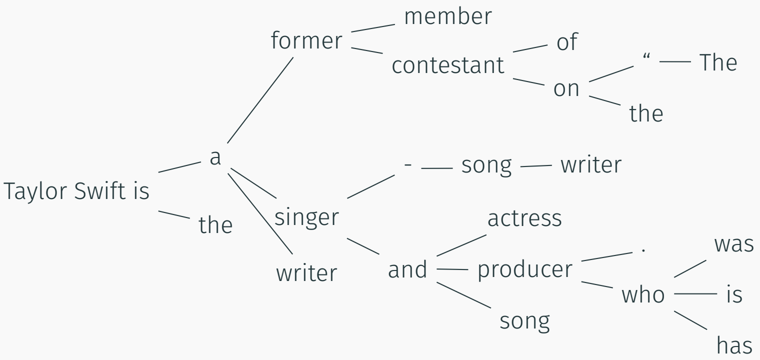
Auto-regressive language modeling uses a causal language model, which defines a conditional distribution over tokens $p_\theta[x_t \mid x_{<t}]$.



Token-level decoding algorithms are primarily concerned with *how to choose the next token.*

Decoding is search

Each time-step during decoding requires a choice.



But a search for what? What is our *objective*? How do we make *local* choices that achieve the objective?

Token-level generation (outline)

Objectives for decoding

- Optimization
- Sampling
- Constrained generation, structured outputs

I. Primitive Generators

Decoding as optimization

Maximum A Posteriori (MAP)

MAP decoding seeks to find the *most likely sequence*

$$\arg\max_{x} p_{\theta}[x]$$

- Greedy decoding
- Beam search

Greedy decoding

- Choose the *most-likely* token at each step.

$$x_t = \arg\max_{x} p_\theta[x \mid x_{<t}]$$

Greedy decoding

- Choose the *most-likely* token at each step.

$$x_t = \arg\max_x p_\theta[x \mid x_{<t}]$$

- Does not guarantee the most-likely sequence.

	Prefix	Continuation			Prob.
Greedy	Taylor Swift is a	former	contestant	on	
Token prob.		0.023	0.022	0.80	0.0004

Greedy decoding

- Choose the *most-likely* token at each step.

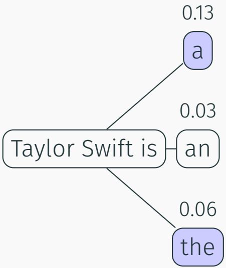
$$x_t = \arg\max_x p_\theta[x \mid x_{<t}]$$

- Does not guarantee the most-likely sequence.

	Prefix	Continuation			Prob.
Greedy	Taylor Swift is a	former	contestant	on	
Token prob.		0.023	0.022	0.80	0.0004
Non-greedy	Taylor Swift is a	singer	,	song	
Token prob.		0.012	0.26	0.21	**0.0007**

Beam Search

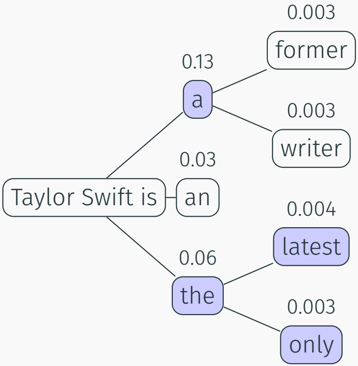
Beam-search is a width-limited breadth-first search (BFS).



GPT2, beam size 2

Beam Search

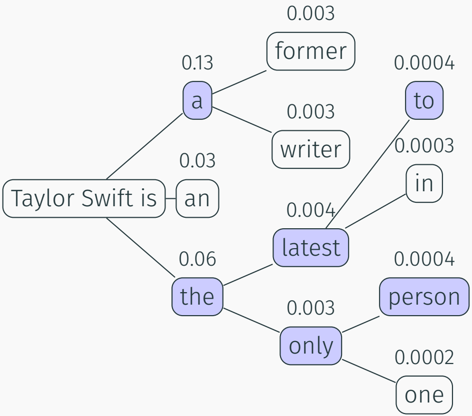
Beam-search is a width-limited breadth-first search (BFS).



GPT2, beam size 2

Beam Search

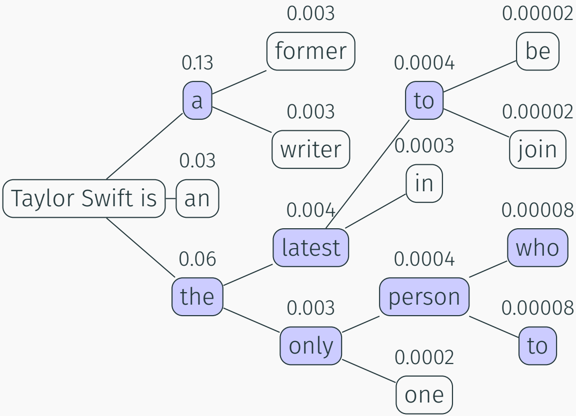
Beam-search is a width-limited breadth-first search (BFS).



GPT2, beam size 2

Beam Search

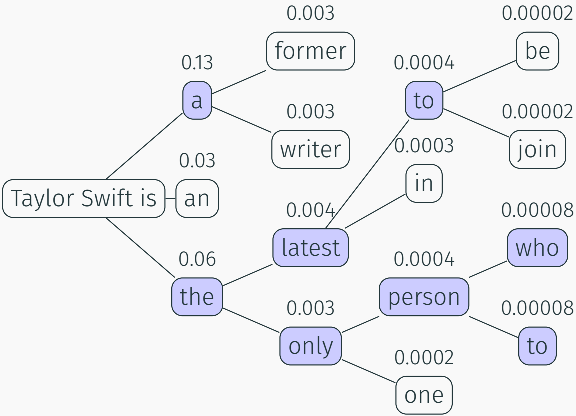
Beam-search is a width-limited breadth-first search (BFS).



GPT2, beam size 2

Beam Search

Beam-search is a width-limited breadth-first search (BFS).

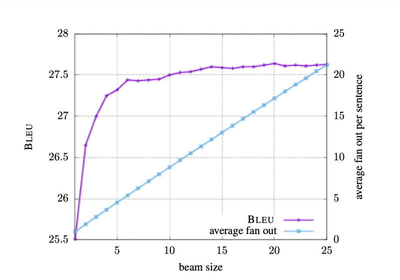


GPT2, beam size 2

Note: Beam search with beam size 1 is greedy decoding.

Benefits of MAP

MAP decoding works well for closed-ended tasks like translation, question answering.



[Freitag and Al-Onaizan, 2017]

Model	Dataset	Metric	Greedy	BS
Llama2-7B	HumanEval	Pass@1	12.80	15.24
	MBPP		17.80	19.40
	GSM8K	Acc	13.87	17.21
	XSUM	R-L	27.21	21.88
	CNN/DM		23.43	20.69
	De⇒En	B-4	28.80	30.14
	En⇒De		22.63	23.99
	Zh⇒En		19.44	20.11
	En⇒Zh		15.15	14.50
	CQA	Acc	62.90	64.37
	SQA		60.76	62.25

[Shi et al., 2024]

Pitfalls of MAP

Probability maximization causes decoding problems.

- Repetition traps
- Short sequences [Stahlberg and Byrne, 2019]
- Atypicality [Meister et al., 2022]

Pitfalls of MAP

Probability maximization causes decoding problems.

- **Repetition traps**
- Short sequences [Stahlberg and Byrne, 2019]
- Atypicality [Meister et al., 2022]

GPT2, Beam size 32.

> *Taylor Alison Swift (born December 13, 1989) is an American singer-songwriter, singer-songwriter, songwriter, and songwriter. She is best known for her work as a singer-songwriter, songwriter-songwriter, songwriter-songwriter, songwriter-songwriter...*

Remedies:

- repetition penalty
- unlikelihood training [Welleck et al., 2020]

Pitfalls of MAP

Probability maximization causes decoding problems.

- Repetition traps
- **Short sequences** [Stahlberg and Byrne, 2019]
- Atypicality [Meister et al., 2022]

$\mathbf{Pr}[\text{Taylor Swift is } \texttt{<eos>}] > \mathbf{Pr}[\text{Taylor Swift is an American singer-...}]$

Remedy: length normalization

Pitfalls of MAP

Probability maximization causes decoding problems.

- Repetition traps
- Short sequences [Stahlberg and Byrne, 2019]
- **Atypicality** [Meister et al., 2022]

- Biased coin Pr[H] = 0.6, Pr[T] = 0.4.
- Most likely outcome from 100 flips is all heads
  H H H H H H H H H H ...
- But this outcome is *atypical*.
- Similarly, the *most likely generation* may *also* be atypical.

Remedy to all of the above: *sampling*

Pitfalls of MAP

Probability maximization causes decoding problems.

- Repetition traps
- Short sequences [Stahlberg and Byrne, 2019]
- Atypicality [Meister et al., 2022]

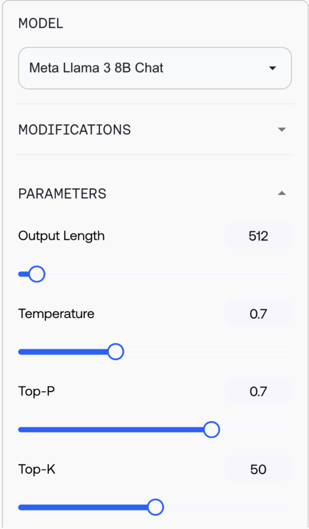
Takeaway: Approximate MAP (e.g., narrow beam search) works better than exact MAP [Meister et al., 2020].

I. Primitive Generators

Sampling

Objective: Sampling

Modern LLM APIs like Together.AI offer settings for *sampling*.



Together.ai playground.

Ancestral sampling

- $y_1 \sim p_\theta(\cdot \mid x)$
- $y_2 \sim p_\theta(\cdot \mid x, y_1)$
- $y_3 \sim p_\theta(\cdot \mid x, y_2, y_3)$
- ...

Ancestral sampling

- $y_1 \sim p_\theta(\cdot \mid x)$
- $y_2 \sim p_\theta(\cdot \mid x, y_1)$
- $y_3 \sim p_\theta(\cdot \mid x, y_2, y_3)$
- ...

Ancestral sampling is equivalent to sequence sampling.

$$p_\theta(\mathbf{y}) = p_\theta(y_1) p_\theta(y_2 \mid y_1) p_\theta(y_3 \mid y_1 y_2) \ldots p_\theta(y_T \mid \mathbf{y}_{<T})$$

What is wrong with ancestral sampling?

- Greedy decoding causes repetition traps

Greedy
(repetition trap)

Taylor Swift is a former contestant on the reality show … "I think it's a very sad day for the show," he said. "It's a very sad day for the show. It's a very sad day for the show. It's a very sad …

What is wrong with ancestral sampling?

- Greedy decoding causes repetition traps
- But ancestral sampling causes incoherence. Why?
- Low-probability tokens are *too likely*
- I.e., the distribution has a *heavy tail*.

Greedy (repetition trap)	Ancestral (incoherent)
Taylor Swift is a former contestant on the reality show … "I think it's a very sad day for the show," he said. "It's a very sad day for the show. It's a very sad day for the show. It's a very sad …	Taylor Swift is a huge fan of her latest album 'Famous.' The singer got her first reaction when she uploaded to Twitter a video of her dancing and singing at a reception for a Grammy-nominated female songstress, Beyoncé.

What is wrong with ancestral sampling?

- Greedy decoding causes repetition traps
- But ancestral sampling causes incoherence. Why?
- Low-probability tokens are *too likely*
- I.e., the distribution has a *heavy tail*.
- Solution: chop off the tail!

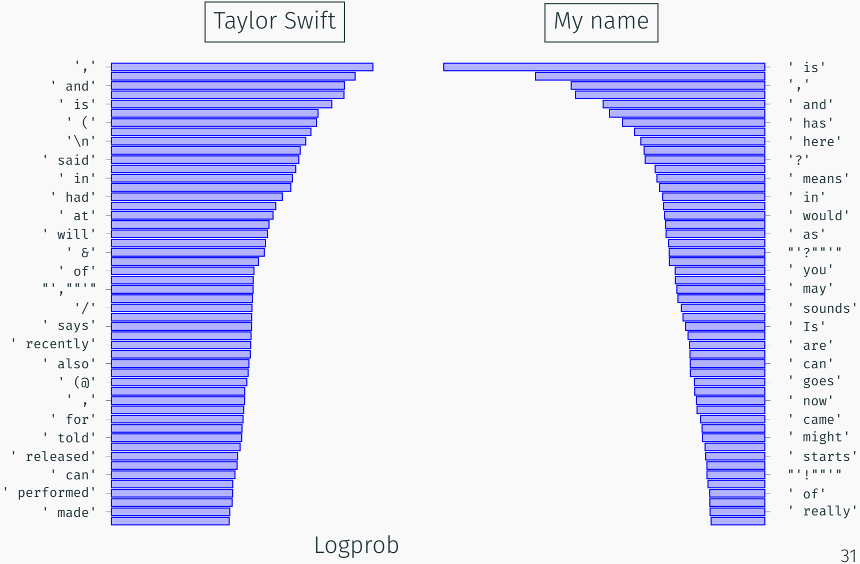
Greedy (repetition trap)	Ancestral (incoherent)	**Top-k (acceptable)**
Taylor Swift is a former contestant on the reality show … "I think it's a very sad day for the show," he said. "It's a very sad day for the show. It's a very sad day for the show. It's a very sad …	Taylor Swift is a huge fan of her latest album 'Famous.' The singer got her first reaction when she uploaded to Twitter a video of her dancing and singing at a reception for a Grammy-nominated female songstress, Beyoncé.	Taylor Swift is a writer for IGN and a member of IGN's Television Critics Association. You can follow her on Twitter at @_MsSwift, IGN at MsSwiftIGN, Facebook at MrsSwift, or subscribe to her video channels.

Truncation sampling

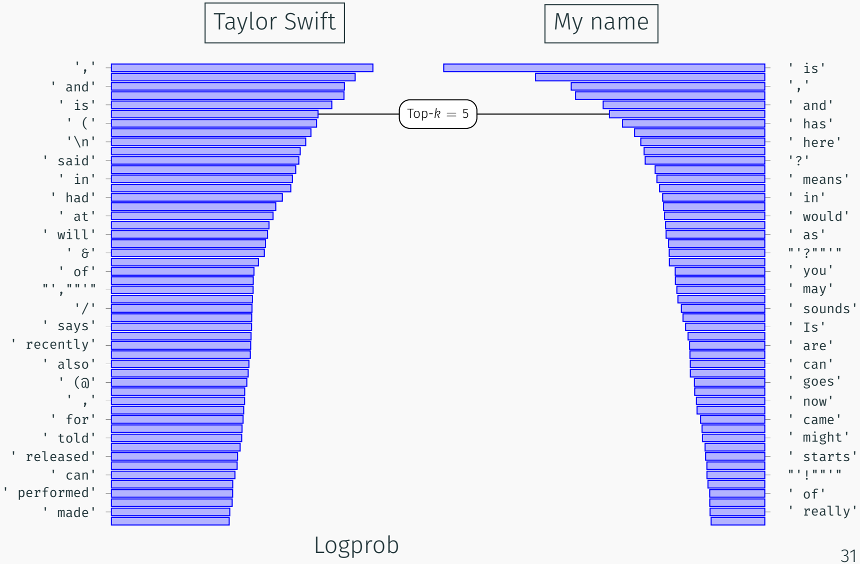
Truncation sampling interpolates greedy and ancestral sampling by choosing a minimum probability threshold at each time step.

Method	Threshold strategy
Top-k	Sample from k-most-probable
Top-p	Cumulative probability at most p
ϵ	Probability at least ϵ
η	Min prob. proportional to entropy
Min-p	Prob. at least p_{min} scaled by max token prob.

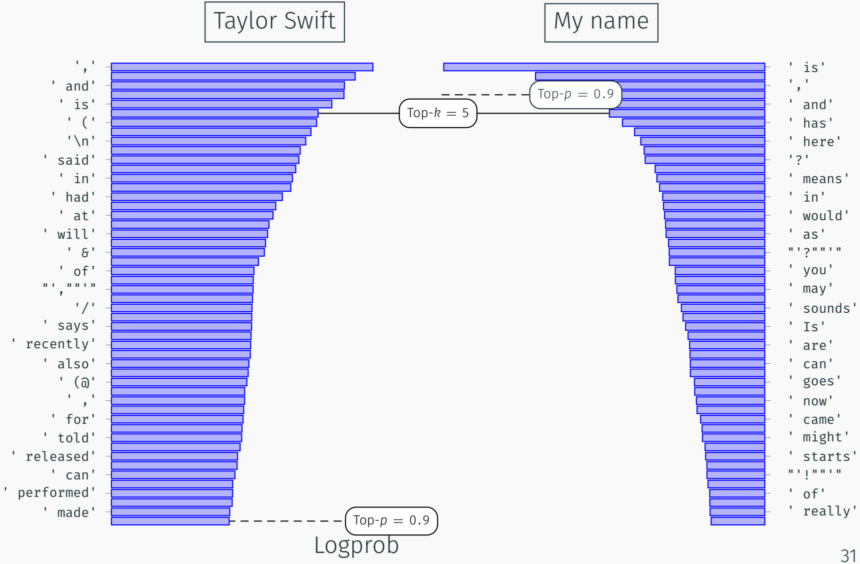
Truncation sampling



Truncation sampling



Truncation sampling



Temperature Sampling

Instead of truncating the tail, make the distribution more “peaked”.

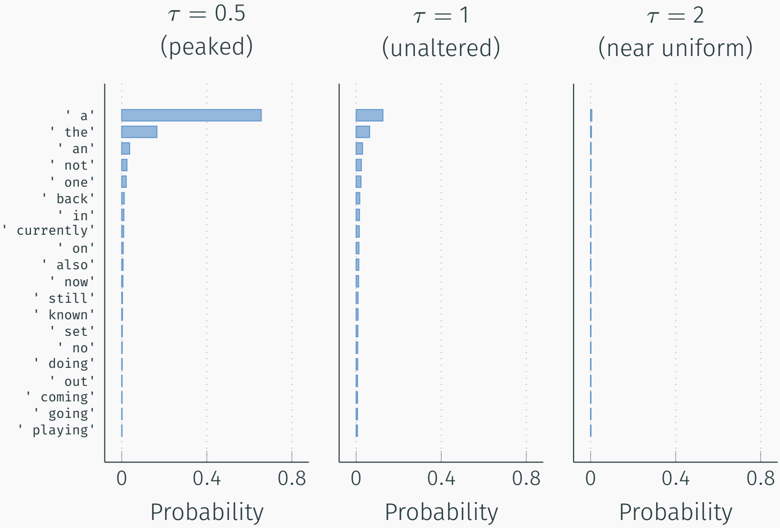
$$\text{softmax}(\boldsymbol{x}, \tau) = \frac{\exp(\boldsymbol{x}/\tau)}{\sum_i \exp(x_i/\tau)}$$

Temperature	Parameter	Pro	Con
High	$\tau \geq 1$	Diverse	Incoherent
Low	$\tau < 1$	Coherent	Repetitive

Temperature Sampling

Taylor Swift is...

$$\mathrm{softmax}(\boldsymbol{x}/\tau)$$



Sampling implementations

```python
probs = model(sequence)

# Greedy
indices, weights = probs.argmax(keepdim=True), None

# Ancestral
indices, weights = vocab_size, probs

# Top-k
topk = probs.topk(k)
indices, weights = topk.indices, topk.values

# Top-p
argsort = probs.argsort(descending=True)
top_p = (argsort.values.cumsum() < p).sum() + 1
indices, weights = argsort.indices[:top_p], argsort.values[:top_p]

# Epsilon
indices, weights = vocab_size, probs * (probs > epsilon)

# Temperature
indices, weights = vocab_size, (logits / temp).softmax(-1)

# Sample
next_token = random.choices(indices, weights=weights, k=1)
```

Batteries-included inference frameworks

```python
# vLLM
from vllm import LLM, SamplingParams
llm = LLM(model="facebook/opt-125m")
prompts = ["Hello, my name is"]
sampling_params = SamplingParams(temperature=0.8, top_p=0.95)
outputs = llm.generate(prompts, sampling_params)

# Huggingface
from transformers import AutoModelForCausalLM, AutoTokenizer
model = AutoModelForCausalLM.from_pretrained("gpt2")
tokenizer = AutoTokenizer.from_pretrained("gpt2")
text = "Hello, my name is"
tokens = tokenizer(text, return_tensors="pt")
output = model(**tokens).generate(
    temperature=0.8, top_p=0.95, do_sample=True
)
```

Why are next-token distributions heavy-tailed?

- Under-training

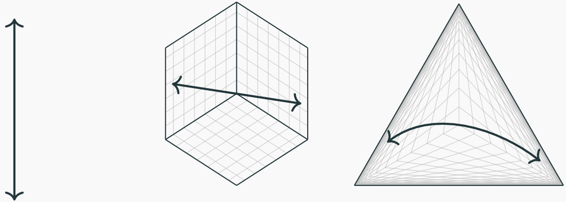
Why are next-token distributions heavy-tailed?

- Under-training
- Mode-seeking: cross-entropy loss punishes probability *underestimation* more than overestimation.

Why are next-token distributions heavy-tailed?

- Under-training
- Mode-seeking: cross-entropy loss punishes probability *underestimation* more than overestimation.
- By *design* low-rank constraints on the LLM outputs [Finlayson et al., 2024].

$$\mathbb{R}^{\text{embed}} \xrightarrow{\text{unembed}} \mathbb{R}^{\text{vocab}} \xrightarrow{\text{softmax}} \Delta_{\text{vocab}}$$



Sampling adapters

A sampling adapter takes a token distribution $p_\theta(\cdot \mid x)$ and re-adjusts the probabilities.

- Truncation and temperature are adapters.

Sampling adapters

A sampling adapter takes a token distribution $p_\theta(\cdot \mid x)$ and re-adjusts the probabilities.

- Truncation and temperature are adapters.
- Contrastive decoding [Li et al., 2023a, Liu et al., 2021]

$$p(\cdot \mid x) \propto \frac{p_{\text{expert}}(\cdot \mid x)}{p_{\text{antiexpert}}(\cdot \mid x)}$$

Sampling adapters

A sampling adapter takes a token distribution $p_\theta(\cdot \mid x)$ and re-adjusts the probabilities.

- Truncation and temperature are adapters.
- Contrastive decoding [Li et al., 2023a, Liu et al., 2021]

$$p(\cdot \mid x) \propto \frac{p_{\text{expert}}(\cdot \mid x)}{p_{\text{antiexpert}}(\cdot \mid x)}$$

- Many others

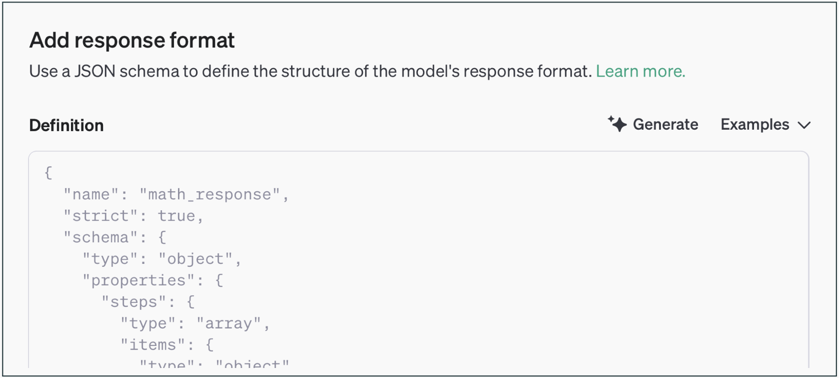
Method	Purpose	Adapter
Ancestral sampling	$y \sim p_\theta$	–
Temperature sampling [Ackley et al., 1985]	$y \sim q(p_\theta)$	Rescale
Greedy decoding	$y \leftarrow \max p_\theta$	Argmax (temperature$\rightarrow$ 0)
Top-k sampling [Fan et al., 2018]	$y \sim q(p_\theta)$	Truncation (top-k)
Nucleus sampling [Holtzman et al., 2020]	$y \sim q(p_\theta)$	Truncation (cumulative prob.)
Typical sampling [Meister et al., 2023]	$y \sim q(p_\theta)$	Truncation (entropy)
Epsilon sampling [Hewitt et al., 2022]	$y \sim q(p_\theta)$	Truncation (probability)
η sampling [Hewitt et al., 2022]	$y \sim q(p_\theta)$	Truncation (prob. and entropy)
Mirostat decoding [Basu et al., 2021]	Target perplexity	Truncation (adaptive top-k)
Basis-aware sampling [Finlayson et al., 2024]	$y \sim q(p_\theta)$	Truncation (linear program)
Contrastive decoding [Li et al., 2023a]	$y \sim q(p_\theta)$	$\log p_{\theta'} - \log p_\theta$ and truncation
DExperts [Liu et al., 2021]	$y \sim q_*(\cdot \mid x, c)$	$\propto p_\theta \cdot (p_{\theta+}/p_{\theta-})^\alpha$
Inference-time adapters [Lu et al., 2023]	$y \sim q_* \propto r(y)$	$\propto (p_\theta \cdot p_{\theta'})^\alpha$
Proxy tuning [Liu et al., 2024]	$y \sim q_*(\cdot \mid x, c)$	$\propto p_\theta \cdot (p_{\theta+}/p_{\theta-})^\alpha$

I. Primitive Generators

Constrained decoding

Embedding LLMs in larger systems requires that they can *communicate* with the larger system, e.g., with JSON.

Can we force LLMs to generate **structured outputs?**



From OpenAI Playground.

Worked example: decoding valid JSON

Language models can stuggle with controlled and structured generation. Prompt:

Key	Type
`name`	string
`birth year`	int

Format the following information using the JSON schema: "Taylor Swift was born December 13, 1989."

Worked example: decoding valid JSON

Language models can stuggle with controlled and structured generation. Prompt:

Key	Type
`name`	string
`birth year`	int

Format the following information using the JSON schema: "Taylor Swift was born December 13, 1989."

LLM:

`{"name": "Taylor Swift", "birth": "1998-12-13T01:00:00Z", "age…`

The LLM output does not match the JSON schema.

Format the following information using the JSON schema:

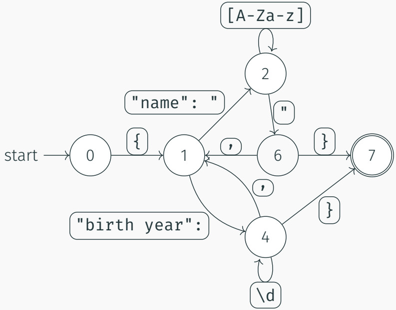
Taylor Swift was born December 13, 1989.

Key	Type
`name`	string
`birth year`	int

Format the following information using the JSON schema:

Taylor Swift was born December 13, 1989.

Key	Type
`name`	string
`birth year`	int

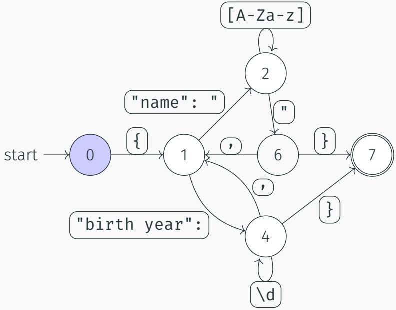


1. Compile the schema into a state machine.

Format the following information using the JSON schema:

Taylor Swift was born December 13, 1989.

Key	Type
`name`	string
`birth year`	int



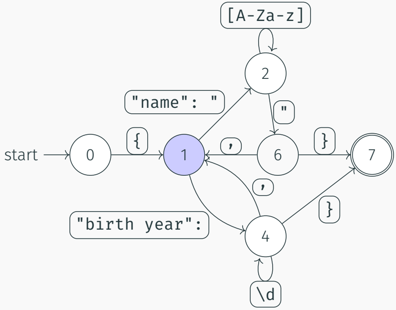
1. Compile the schema into a state machine.
2. Filter the next-token distribution for valid tokens.

Token	Prob.
`\n`	~~0.36~~
`"`	~~0.16~~
`{`	0.026
`https`	~~0.025~~
...	...

Format the following information using the JSON schema:

Taylor Swift was born December 13, 1989.

Key	Type
`name`	string
`birth year`	int



1. Compile the schema into a state machine.
2. Filter the next-token distribution for valid tokens.

GPT2:

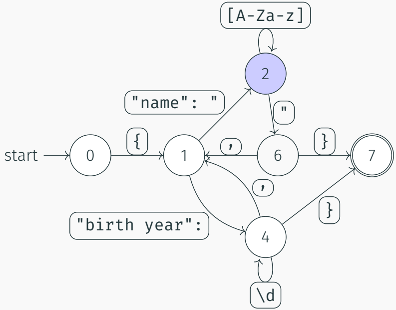
```
{
```

Token	Prob.
`name`	~~0.31~~
`date`	~~0.069~~
`"`	0.039
`id`	~~0.033~~
...	...

Format the following information using the JSON schema:

Taylor Swift was born December 13, 1989.

Key	Type
`name`	string
`birth year`	int



1. Compile the schema into a state machine.
2. Filter the next-token distribution for valid tokens.

GPT2:

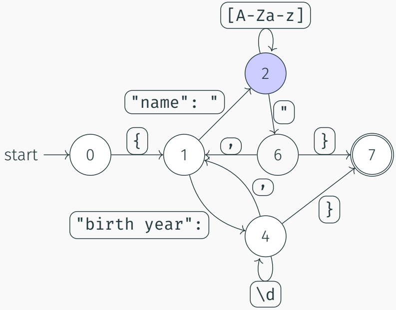
```
{"name": "
```

Token	Prob.
`Taylor`	0.85
`T`	0.034
`S`	0.024
`The`	0.022
...	...

Format the following information using the JSON schema:

Taylor Swift was born December 13, 1989.

Key	Type
`name`	string
`birth year`	int



1. Compile the schema into a state machine.
2. Filter the next-token distribution for valid tokens.

GPT2:

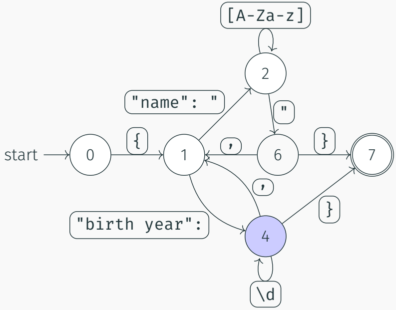
```
{"name": "Taylor Swift
```

Token	Prob.
`",`	0.85
`,"`	~~0.034~~
`"`	0.024
`,`	~~0.022~~
...	...

Format the following information using the JSON schema:

Taylor Swift was born December 13, 1989.

Key	Type
`name`	string
`birth year`	int



1. Compile the schema into a state machine.
2. Filter the next-token distribution for valid tokens.

GPT2:

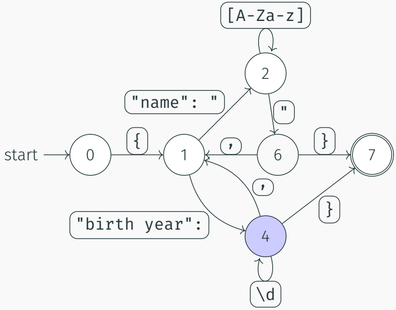
```
{"name": "Taylor Swift", "birth
year":
```

Token	Prob.
`"`	~~0.46~~
`int`	~~0.041~~
`'`	~~0.026~~
`1989`	0.020
...	...

Format the following information using the JSON schema:

Taylor Swift was born December 13, 1989.

Key	Type
`name`	string
`birth year`	int



1. Compile the schema into a state machine.
2. Filter the next-token distribution for valid tokens.

GPT2:

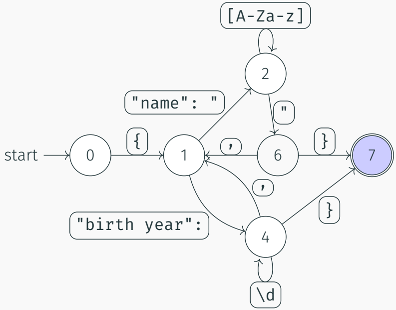
```
{"name": "Taylor Swift", "birth
year": 1989
```

Token	Prob.
,	0.39
}	0.34
},	~~0.11~~
}	0.082
...	...

Format the following information using the JSON schema:

Taylor Swift was born December 13, 1989.

Key	Type
`name`	string
`birth year`	int



1. Compile the schema into a state machine.
2. Filter the next-token distribution for valid tokens.

GPT2:

```
 {"name": "Taylor Swift", "birth
year": 1989}
```

Side effects of templated/constrained decoding

- Generation speedup
- Reduced performance

Token healing

- Templated generation can force unnatural token boundaries

`The␣url␣is␣http:`

Token healing

- Templated generation can force unnatural token boundaries

`The␣url␣is␣http:␣//`

- The model has rarely seen the tokenization `␣http` `:` `//` during training compared to `␣http` `://`.

Token healing

- Templated generation can force unnatural token boundaries

`The␣url␣is␣http:␣//`

- The model has rarely seen the tokenization `␣http:␣//` during training compared to `␣http://`.
- Token healing rewinds the tokenizer and enforces the untokenized text as a prefix to the next token.

`The␣url␣is␣http:`

Candidates
~~`s://`~~
`://`

Token healing

- Templated generation can force unnatural token boundaries

  `The` `␣url` `␣is` `␣http` `:` `␣//`

- The model has rarely seen the tokenization `␣http` `:` `//` during training compared to `␣http` `://`.
- Token healing rewinds the tokenizer and enforces the untokenized text as a prefix to the next token.

  `The` `␣url` `␣is` `␣http` `://`

Candidates
~~s://~~
://

- Alternative fix: tokenizer regularization during training [Kudo, 2018].

Summary

- Two views of decoding: optimization, sampling
- The diversity-coherence trade-off
- Constrained decoding enforces structure on LLM outputs

These are the building blocks of modern LLM generation methods.

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*.

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.