Generating and Evaluating Text

Aaron Mueller
CS599 B1: Advanced Natural Language Processing
Sep. 17, 2026

Reminders

- Homework 1 is due in **1 week**!
  - After today's lecture, you will have all the content you need to complete it

- Our first quiz will be the following Tuesday (5 days after the HW1 due date)
  - I will release a practice quiz at least one week beforehand

Resource Usage

Resource Usage

FLOPs and FLOPS

- How can we estimate how much compute/how long a certain experiment will take?
- **FLOPs:** floating-point operations
  - How computationally expensive is a matrix multiplication?
- **FLOPS:** floating-point operations *per second*
  - How quickly can an H100 run a certain number of computations?

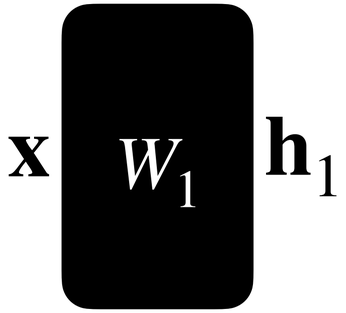
H100 has a peak performance of 1979 teraFLOPS: 1979e12

(This is only true with sparsity; take 50% of this with dense tensors.)

Resource Usage

FLOPs Examples

- How many FLOPs for a forward pass in a linear layer?
  - Batch size = 1024
  - Input dim = 256
  - Hidden dim = 64



$$2 * 1024 * 256 * 64$$

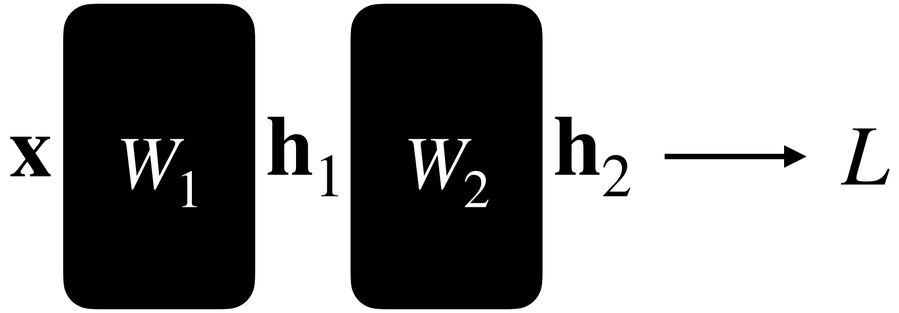
Where does the 2 come from?

Recall that in matrix multiplication, all floats are multiplied *and* added at some point

Resource Usage

FLOPs Examples

- How many FLOPs for a forward *and backward* pass in a linear model?
  - Batch size = 1024
  - Input dim = 256
  - Hidden dim = 64
  - Output dim = 64



Forward pass required ≈2 operations per float. How about the backward pass?

Compute gradient for W_2: $\mathbf{h}_1^\top \left(\frac{\partial L}{\partial \mathbf{h}_2} \right)$ -> 2 * 1024 * 64 * 64

2× more expensive than the forward pass!

Backward pass via $\mathbf{h}_1$: $\left(\frac{\partial L}{\partial \mathbf{h}_2} \right) W_2^\top$ -> 2 * 1024 * 64 * 64

Total forward + backward FLOPs:

*≈ 6 * data points * parameters*

Resource Usage

Model FLOPs Utilization (MFU)

- The maximal FLOPs a GPU can process at once is not usually the same as what it does in practice
- The MFU is the observed throughput (tokens/sequences/FLOPs per second) divided by the theoretical peak throughput of the hardware
- Example: with torch.float16, an A100 has a theoretical peak of 312 teraFLOPS
    - Using the $6N$ FLOPs estimate from before, when training a 125M parameter model, we have a throughput of 200,000 tokens per second:

$$\text{MFU} = \frac{6ND}{P} = \frac{6 \cdot 125000000 \cdot 200000}{312 \times 10^{12}}$$

$$= 0.48$$

Resource Usage

Memory

What all do we need to store in memory?

- Copies of the optimizer states (2 for Adam): $M_{\text{optim}} = 8N_{\text{param}}$
- Forward pass (assuming torch.float32, s.t. 4 bytes per parameter):
  - Copies of the weights of the model: $M_{\text{model}} = 4N_{\text{param}} + 4N_{\text{buf}}$
  - Copies of the data and targets/labels: $M_{\text{data}} = 2 \times B \times T \times 8$
    - B = batch size, $T =$ data point, 8 assumes *64-bit precision* in the data
  - Backward pass (assuming torch.float32):
    - $M_{\text{gradients}} = 4N_{\text{param}}$

Resource Usage

Exercises

- How long would it take to train Llama 3 (a 70B-parameter model) on 15T tokens given 1024 H100 GPUs?

```
FLOPs = 6 * 70B * 15T
h100_FLOPS = 1979T / 2
MFU = 0.5
FLOPs_per_day = h100_FLOPS * 1024 * (60 * 60 * 24) * MFU
days = FLOPs / FLOPs_per_day
```

This is about 3.5M H100 hours.

```
days ≈ 143.93
```

Same ballpark as the actual reported time of 6.4M H100 hours!

Resource Usage

Exercises

- What is the largest model trainable on 8 H100s with AdamW using torch.float32?

```
h100_memory = 80B
bytes_per_parameter = 4 + 4 + (4 + 4)
num_parameters = (h100_memory * 8) / bytes_per_parameter
```

```
num_parameters ≈ 40B
```

Generation

From Training to Inference

- So far, we've talked almost exclusively about how to *train* language models.

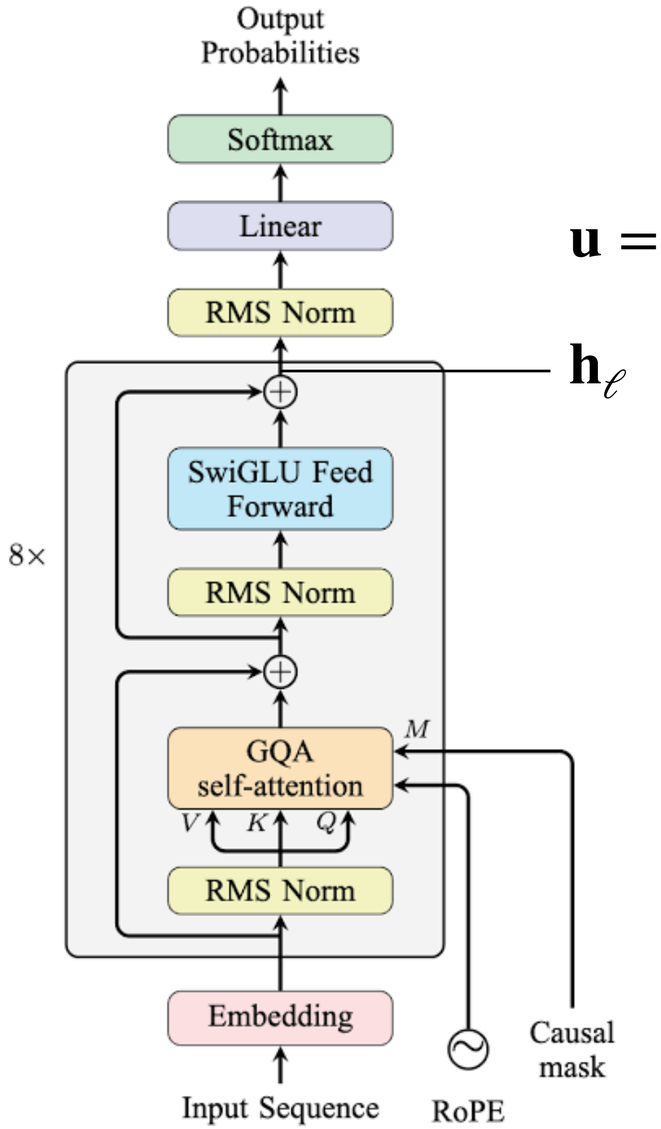
- Now, we'll talk about **inference**, which (in this context) refers to using a trained LM to generate new sequences

Decoding from an LM

- LMs learn a distribution $p(x_i \mid x_1, \ldots, x_{i-1})$

- How can we use this to generate text?
  - **Greedy decoding:** pick whichever token maximizes $p(x_i \mid x_1, \ldots, x_{i-1})$
  - **Beam search:** keep track of top-k most probable sequences
  - **Sampling:** pick a token at random according to the probability distribution
    - There are many variants of this!

Decoding from an LM



$$\mathbf{u} = W_U \text{RMSNorm}(\mathbf{h}_\ell)$$

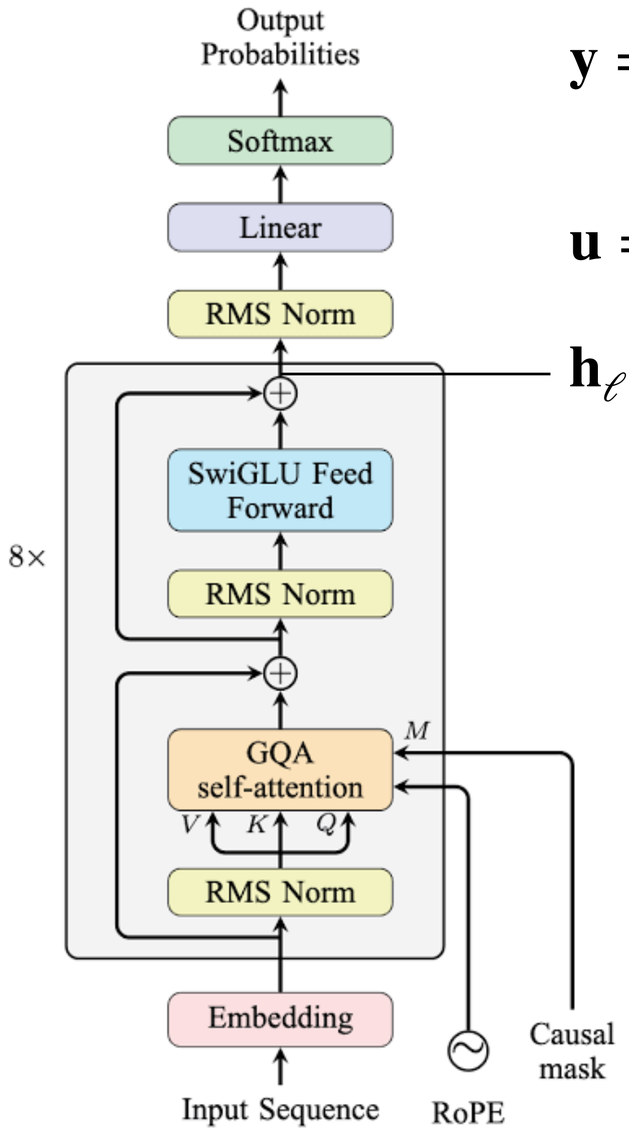
$\mathbf{h}_\ell$

We'll refer to the output probability distribution as $\mathbf{y}$.

Take the final layer representation $\mathbf{h}_\ell$, pass it through a final RMSNorm, and multiply it with the unembedding matrix W_U to get the logits $\mathbf{u}$.

$$W_U \in \mathbb{R}^{vocab\_size \times d\_model}$$

Decoding from an LM



$\mathbf{y} = \text{softmax}(\mathbf{u})$

$\mathbf{u} = W_U \text{RMSNorm}(\mathbf{h}_\ell)$

$\mathbf{h}_\ell$

To turn the logits into probabilities, pass them through a **softmax**:

$$\text{softmax}(\mathbf{a})_i = \frac{\exp(a_i)}{\sum_j \exp(a_j)}$$

The softmax normalizes all values in a vector such that they sum to 1.

softmax([5, 2, 1]) = [0.936, 0.047, 0.017]

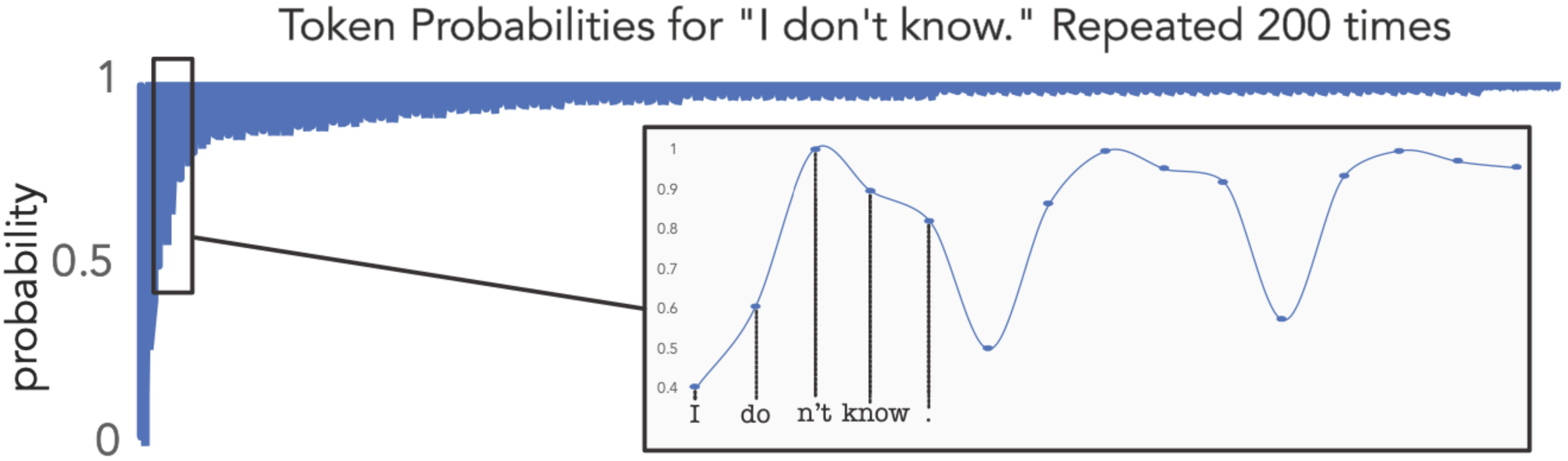
Greedy Decoding

- **Greedy decoding:** pick whichever token maximizes $p(x_i \mid x_1, \ldots, x_{i-1})$ at each step i
- Keep going until we generate some end-of-sequence token like `<|endoftext|>`

- What might be some issues with this?

Issues with Greedy Decoding

[Holtzman et al., 2020]

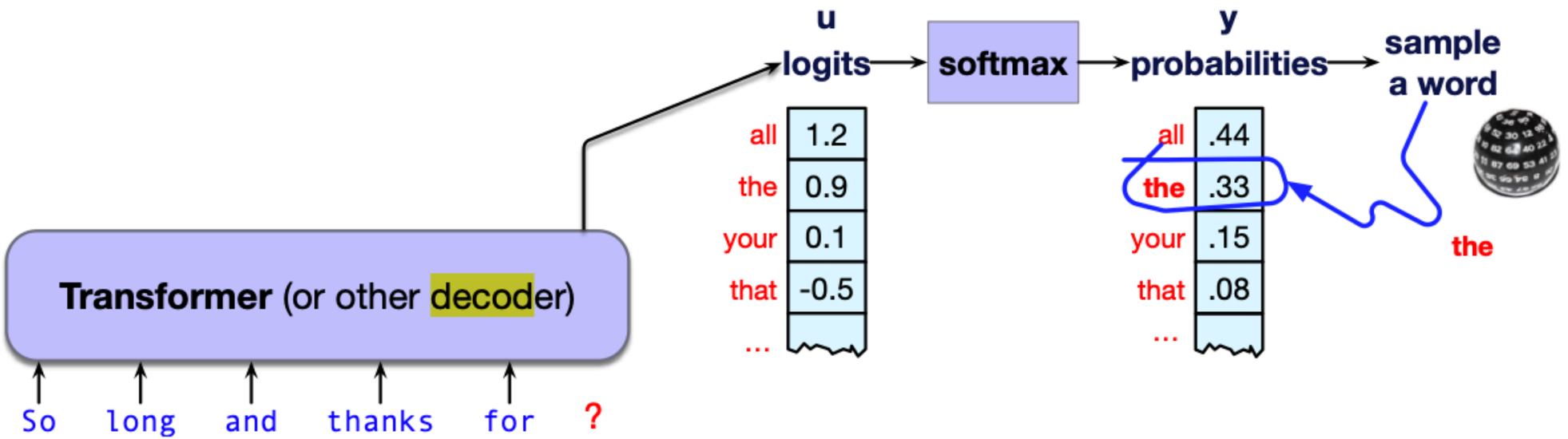


It is very easy to get **degenerate outputs** when using greedy decoding. The most likely tokens *right now* are not necessarily those that correspond to fluent language *overall*.

> The number of stranded whales has increased by more than 50 per cent in the past year, with the number of stranded whales on the West Australian coast increasing by more than 50 per cent in the past year. The number of whales stranded on the West Australian coast has increased by more than 50 per cent in the past year, with the number of stranded whales on the West Australian coast increasing by more than 50 per cent in the past year.

(Pure) Sampling

The most common methods involve **sampling** from the LM's output distribution.



Just pick a token from the vocab at random, according to the probabilities output by the LM.

This will generate mostly sensible outputs, but there are *a lot* of weird tokens in the tail of the probability distribution.

Issues with (Pure) Sampling

[Holtzman et al., 2020]

Context: In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

Beam Search, *b*=32:
"The study, published in the Proceedings of the National Academy of Sciences of the United States of America (PNAS), was conducted by researchers from the Universidad Nacional Autónoma de México (UNAM) and the Universidad Nacional Autónoma de México (UNAM/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de ..."

Pure Sampling:
They were cattle called Bolivian Cavalleros; they live in a remote desert uninterrupted by town, and they speak huge, beautiful, paradisiacal Bolivian linguistic thing. They say, 'Lunch, marge.' They don't tell what the lunch is," director Professor Chuperas Omwell told Sky News. "They've only been talking to scientists, like we're being interviewed by TV reporters. We don't even stick around to be interviewed by TV reporters. Maybe that's how they figured out that they're cosplaying as the Bolivian Cavalleros."

Even in a well-trained model, greedy decoding (and its improved cousin beam search) lead to degenerate outputs. Pure sampling leads to ungrammatical gibberish.

Issues with (Pure) Sampling

- Sampling is *too* random

Pure Sampling:
They were cattle called Bolivian Cavalleros; they live in a remote desert uninterrupted by town, and they speak huge, beautiful, paradisiacal Bolivian linguistic thing. They say, 'Lunch, marge.' They don't tell what the lunch is," director Professor Chuperas Omwell told Sky News. "They've only been talking to scientists, like we're being interviewed by TV reporters. We don't even stick around to be interviewed by TV reporters. Maybe that's how they figured out that they're cosplaying as the Bolivian Cavalleros."

$p(y \mid \ldots \text{and in the evenings, they enjoyed playing})$

0.1	cards
0.05	basketball
0.05	Civilization
0.01	pretend
...	
0.00005	JsonObj

Top of distribution looks fine.

≈90% chance of getting something good.

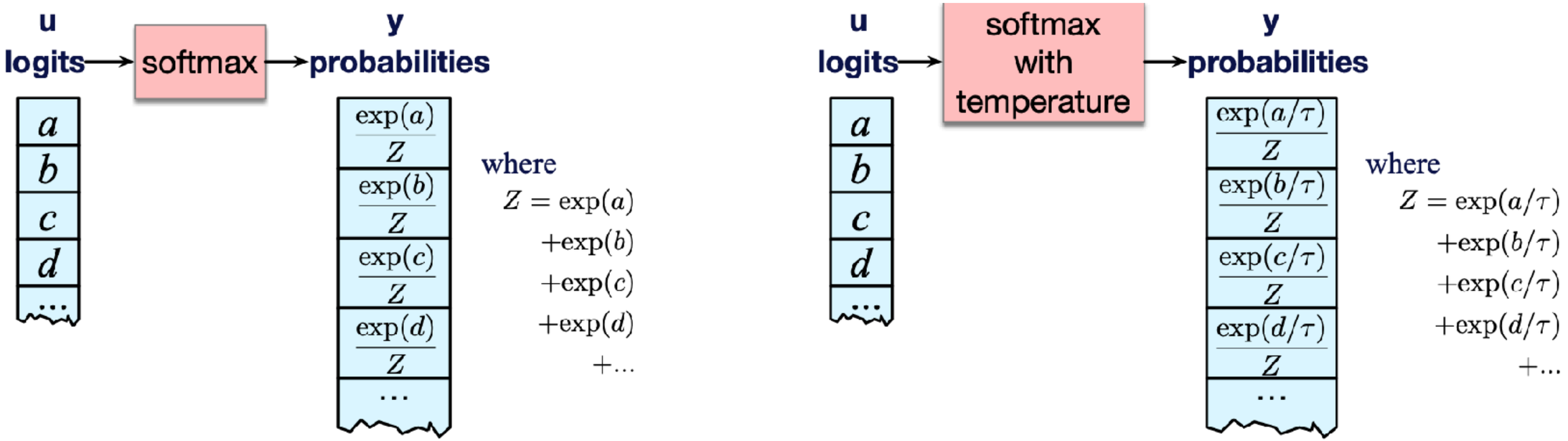
Long tail of bad completions with ≈10% of the probability.

Temperature Sampling

Idea: reshape the probability distribution to increase the probability of high-probability tokens, and decrease the probability of low-probability tokens.

We use **temperature** τ, a hyperparameter, to do this.

$$\mathbf{y} = \text{softmax}(\mathbf{u}/\tau)$$



Temperature Sampling

- Low-temperature sampling ($\tau < 1$) makes more probable tokens even more probable, decreasing the "randomness" of the outputs.
  - As $\tau \to 0$, we get the same thing as greedy decoding!
- High-temperature sampling ($\tau > 1$) increases the probability assigned to lower-probability tokens, increasing the "randomness" of the outputs.

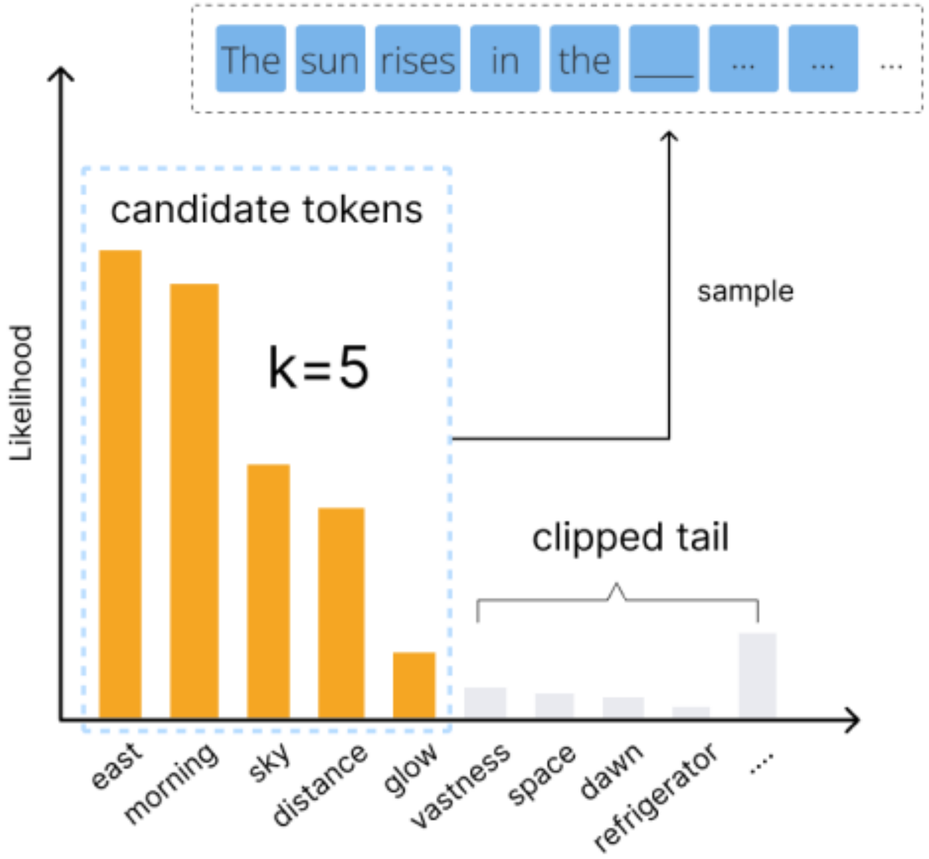
	logits	τ=0.1	τ=0.5	τ=1	τ=10	τ=100
all	1.2	.95	.59	.44	.27	.25
the	0.9	.05	.32	.33	.26	.25
your	0.1	0	.07	.15	.24	.25
that	-0.5	0	.02	.08	.23	.25

Top-k Sampling

Top-k sampling: We can truncate the distribution by sampling from *only* the top-k most probable tokens.

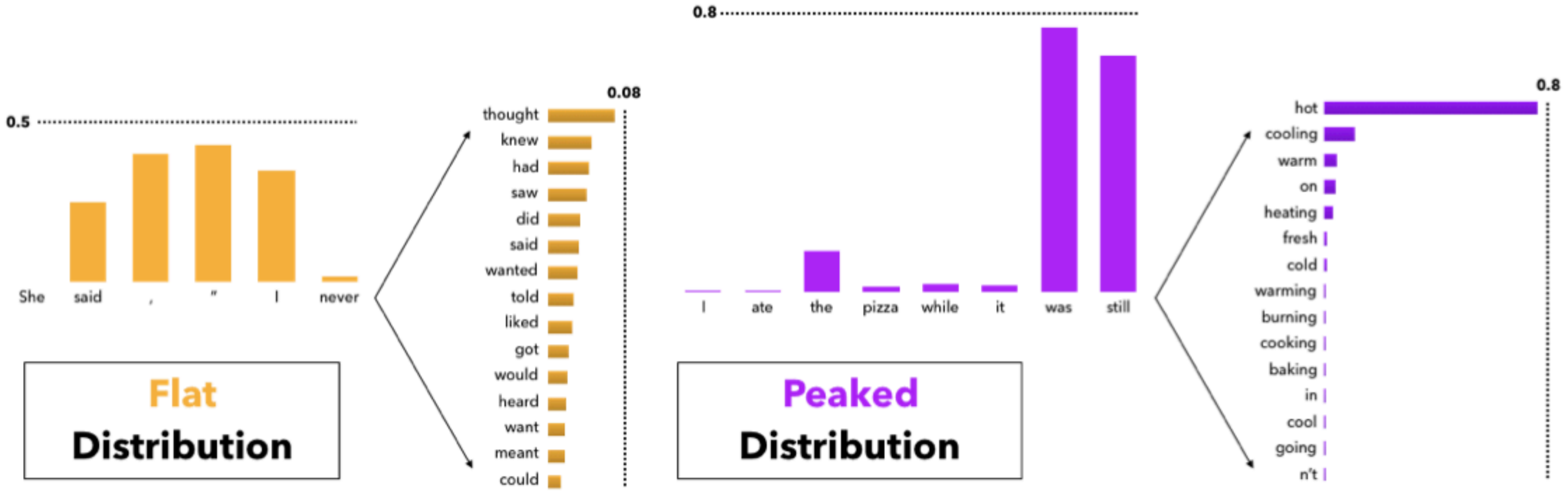
Surprisingly good method, but how do you pick k?

Is the same k always going to be best?



Nucleus Sampling

Nucleus (top-p) sampling: Or, we can sample from only the top tokens whose cumulative probability exceeds some threshold *p*.



(It's generally agreed that nucleus sampling is best.)

Nucleus Sampling

$p(y \mid \ldots$and in the evenings, they enjoyed playing$)$

0.1	cards
0.05	basketball
0.05	Civilization
0.01	pretend

Sample from top of distribution only.

Cut off distribution after *p*% of probability mass.

- Keep the most probable options accounting for *p* of the probability mass (the "nucleus"), and sample only among these.
- To implement: sort tokens by probability, truncate list as soon as you exceed *p*, then renormalize and sample from that new distribution

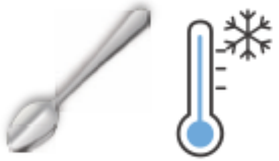
Examples

WebText

An unprecedented number of mostly young whales have become stranded on the West Australian coast since 2008.

Pure Sampling

The Australian Food Safety Authority has warned Australia's beaches may be revitalised this year because healthy seabirds and seals have been on the move. More than 50,000 seabirds, sea mammals and seahorses have been swept into the sea by the Holden CS118 and Adelaide Airport CS300 from 2013. A major white-bat and umidauda migration across Australia is under way in Australia for the first time, with numbers reaching an estimated 50,000.



Sampling, t=0.9

Last week's intense storms and a series of powerful cyclones have been officially blamed for the deaths of at least nine large fin whales near Whitsundays - the largest loss of any species globally. The fin whales: packed in the belly of one killer whale thrashing madly in fear as another tries to bring it to safety. When the colossal animal breached the waters of Whitsundays, he'd been seen tagged for a decade.

Examples

WebText

An unprecedented number of mostly young whales have become stranded on the West Australian coast since 2008.

Top-*k*, *k*=40, *t*=0.7

The whale's fate was confirmed late last week when the animal was found by fishermen off the coast of Bundaberg. Experts believe the whale was struck by a fishing vessel off the coast of Bundaberg, and died after being sucked into the ocean. The whale's fate was confirmed late last week when the animal was found by fishermen off the coast of Bundaberg.

Nucleus, *p*=0.95

There has been an unprecedented number of calves caught in the nets of whaling stations that operate in WA. Pilot whales continue to migrate to feeding grounds to feed their calves. They are now vulnerable due to the decline of wild populations; they are restricted to one breeding site each year. Image copyright Yoon Bo Kim But, with sharp decline in wild populations the size of the Petrels are shrinking and dwindling population means there will only be room for a few new fowl.

Practical Details

- In practice, we often combine temperature with nucleus sampling!
  - *Order matters*: Reshape distribution with temperature first
  - Then, truncate to get only the p most probable tokens

Inference Efficiency

- Parallelizing training is easy: we know what should come next
  - When you **prefill** text, you can also process the input in parallel
- During generation, we *must* process sequentially. Fully using compute is much harder

The quick brown fox jumped _____

The quick brown fox jumped over _____

The quick brown fox jumped over the _____

Evaluation

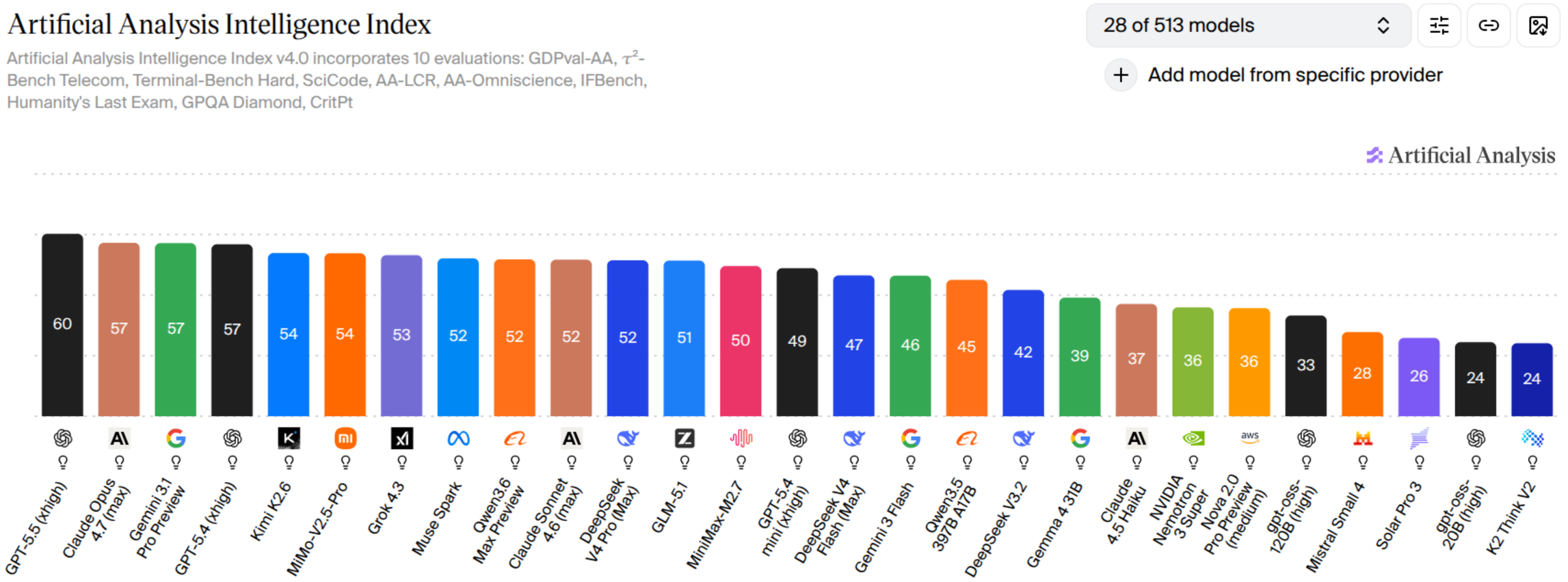
Evaluating Text

- So now we've used our LM to generate some text in response to some prompt(s). How can we tell how good it is?

- Seems easy at first: compute accuracy

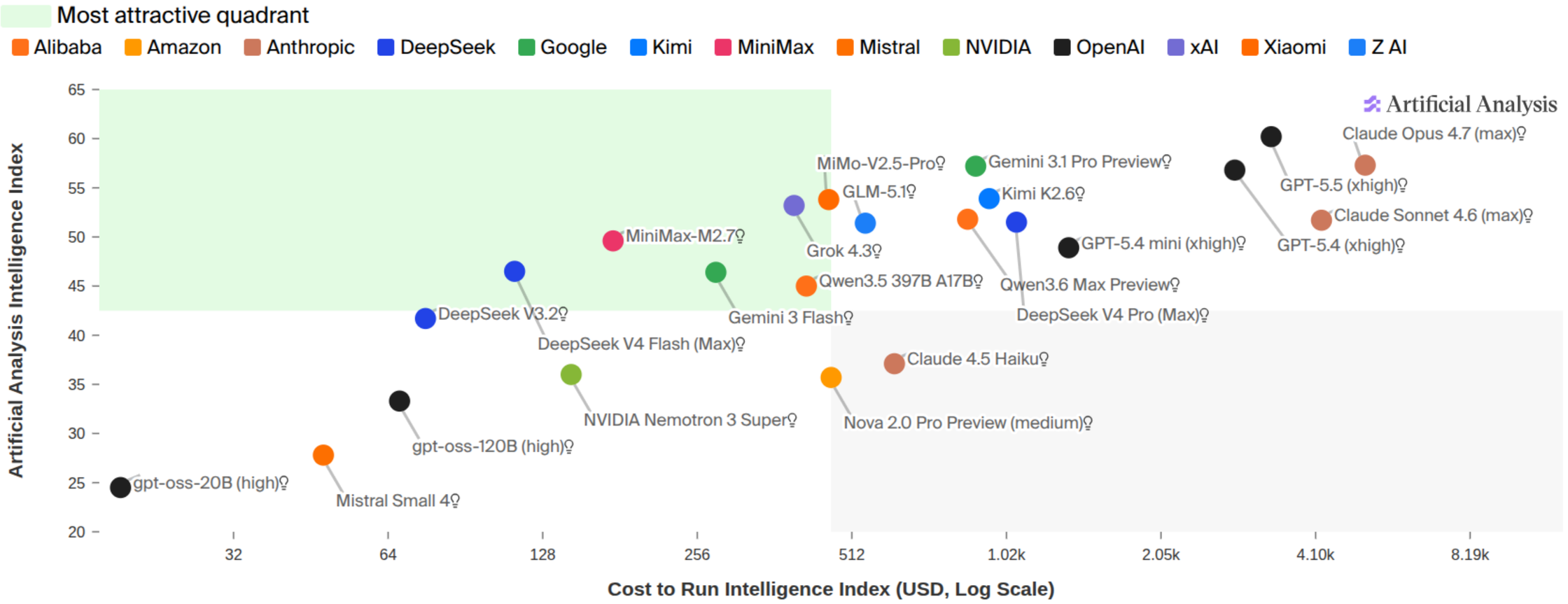
- Actually, evaluation is a deep and active area of research that is advancing quickly

The Benchmark View



Maybe we consider a model good if it performs well on pre-constructed benchmarks

The Benchmark + Efficiency View



Maybe we say a model is good if it performs well on benchmarks *and* is efficient to run

The Relative Ranking View

Text

4 days ago

Rank	Model	Score
1	claude-opus-4-7-thinking	1503
2	claude-opus-4-6-thinking	1502
3	claude-opus-4-6	1497
4	gemini-3.1-pro-preview	1493
5	claude-opus-4-7	1491
6	muse-spark	1491
7	gpt-5.5-high	1488
8	gemini-3-pro	1486
9	grok-4.20-beta1	1480
10	grok-4.20-beta-0309-reasoning	1477

Maybe we pit models against each other and say that a model is good if it generates responses that people prefer more often than others.

The Usage View

Compare the most popular models on OpenRouter

1.	Hy3 preview (free) by tencent	3.66T tokens ↑298%	
2.	Kimi K2.6 by moonshotai	1.8T tokens ↓11%	
3.	Claude Sonnet 4.6 by anthropic	1.34T tokens ↓0%	
4.	Gemini 3 Flash Preview by google	974B tokens ↓5%	
5.	Claude Opus 4.7 by anthropic	919B tokens ↓21%	
6.	DeepSeek V4 Flash by deepseek	819B tokens ↑158%	
7.	DeepSeek V3.2 by deepseek	815B tokens ↓33%	
8.	MiniMax M2.7 by minimax	738B tokens ↓4%	
9.	Grok 4.1 Fast by x-ai	706B tokens ↓6%	
10.	Step 3.5 Flash by stepfun	663B tokens ↓19%	
11.	Gemini 2.5 Flash Lite by google	624B tokens ↓3%	
12.	Gemini 2.5 Flash by google	602B tokens ↓0%	
13.	DeepSeek V4 Pro by deepseek	577B tokens ↑616%	
14.	Nemotron 3 Super (free) by nvidia	546B tokens ↓17%	
15.	Ling-2.6-1T (free) by inclusionai	467B tokens ↑10%	
16.	Claude Opus 4.6 by anthropic	418B tokens ↓36%	
17.	gpt-oss-120b by openai	391B tokens ↑1%	
18.	GLM 5.1 by z-ai	362B tokens ↓8%	
19.	Gemini 3.1 Pro Preview by google	318B tokens ↓4%	
20.	Gemini 3.1 Flash Lite Previ... by google	312B tokens ↑12%	

Maybe we consider a model good if people choose to pay for it and use it more often

Evaluation Metrics

Perplexity

- We haven't even started discussing how to measure goodness in a benchmarking setting yet

- Let's start with the original language model evaluation metric: **perplexity**

- A language model is a probability distribution over tokens $p(x_i \mid x_{<i})$. Using this, we can define perplexity as follows:

$$\text{ppl} = \exp\left(-\frac{1}{N} \sum_{i=1}^{N} \log p(x_i \mid x_{<i}) \right)$$

Cross-entropy (average surprisal)

Perplexity

$$\text{ppl} = \exp\left(-\frac{1}{N} \sum_{i=1}^{N} \log p(x_i \,|\, x_{<i}) \right)$$

- Perplexity is inversely correlated to the probability of the dataset according to your model
    - Intuition: captures how “surprised” or “perplexed” your model is upon seeing some sequence of text. A good model should minimize this on text that looks coherent

- We implicitly train LMs to minimize this on a training set
    - Ideally, this model should also get low perplexity on held-out test data

Perplexity Benchmarks

- Standard datasets:
  - Penn Treebank (Wall Street Journal)
  - WikiText-103
  - One Billion Word Benchmark

From **Radford et al. (2019):**

Language Models are Unsupervised Multitask Learners

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	1BW (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3	**21.8**
117M	**35.13**	45.99	**87.65**	**83.4**	**29.41**	65.85	1.16	1.17	37.50	75.20
345M	**15.60**	55.48	**92.35**	**87.1**	**22.76**	47.33	1.01	**1.06**	26.37	55.72
762M	**10.87**	**60.12**	**93.45**	**88.0**	**19.93**	**40.31**	**0.97**	**1.02**	22.05	44.575
1542M	**8.63**	**63.24**	**93.30**	**89.05**	**18.34**	**35.76**	**0.93**	**0.98**	**17.48**	42.16

Is perplexity all you need?

- Let's call the true distribution of language t, and the LM's learned distribution p
  - The best possible perplexity is achieved only when $p = t$
  - By continually decreasing perplexity, some hope we might eventually "solve language modeling" or "reach AGI"

- ...But post-training can *increase* perplexity on the pre-training data
  - And yet, post-trained models are more useful

Perplexity in Disguise

- Consider a fill-in-the-blank task like LAMBADA:

Context: "Why?" "I would have thought you'd find him rather dry," she said. "I don't know about that," said Gabriel. "He was a great craftsman," said Heather. "That he was," said Flannery.
Target sentence: "And Polish, to boot," said _____.
Target word: Gabriel

Context: Preston had been the last person to wear those chains, and I knew what I'd see and feel if they were slipped onto my skin-the Reaper's unending hatred of me. I'd felt enough of that emotion already in the amphitheater. I didn't want to feel anymore. "Don't put those on me," I whispered. "Please."
Target sentence: Sergei looked at me, surprised by my low, raspy please, but he put down the _____.
Target word: chains

- This **cloze task** is really just measuring the model's ability to predict the same word as in the true distribution—i.e., to minimize perplexity

Perplexity in Disguise

ACTIVITYNET

A woman is outside with a bucket and a dog. The dog is running around trying to avoid a bath. She…

A. rinses the bucket off with soap and blow dry the dog's head.
B. uses a hose to keep it from getting soapy.
C. gets the dog wet, then it runs away again.
D. gets into a bath tub with the dog.

\+ Adversarial Filtering

---

wikiHow to do anything

How to determine who has right of way.

Come to a complete halt at a stop sign or red light. At a stop sign, come to a complete halt for about 2 seconds or until vehicles that arrived before you clear the intersection. If you're stopped at a red light, proceed when the light has turned green. …

A. Stop for no more than two seconds, or until the light turns yellow. A red light in front of you indicates that you should stop.
B. After you come to a complete stop, turn off your turn signal. Allow vehicles to move in different directions before moving onto the sidewalk.
C. Stay out of the oncoming traffic. People coming in from behind may elect to stay left or right.
D. If the intersection has a white stripe in your lane, stop before this line. Wait until all traffic has cleared before crossing the intersection.

\+ Adversarial Filtering

You could imagine an easier task where you give the model some preset list of options, and see how often it chooses the correct (most probable) one

(This task is called HellaSwag **[Zellers et al., 2019]**)

Exam Benchmarks

- As with humans, exams are a useful way to test language models
  - We have control over difficulty and subject matter
  - Can design s.t. there is always an unambiguous correct answer that is *easy to grade*
- A common task: **Massive Multitask Language Understanding (MMLU) [Hendrycks et al., 2020]**

Few Shot Prompt and Predicted Answer

The following are multiple choice questions about high school mathematics.

How many numbers are in the list 25, 26, ..., 100?
(A) 75 (B) 76 (C) 22 (D) 23
Answer: B

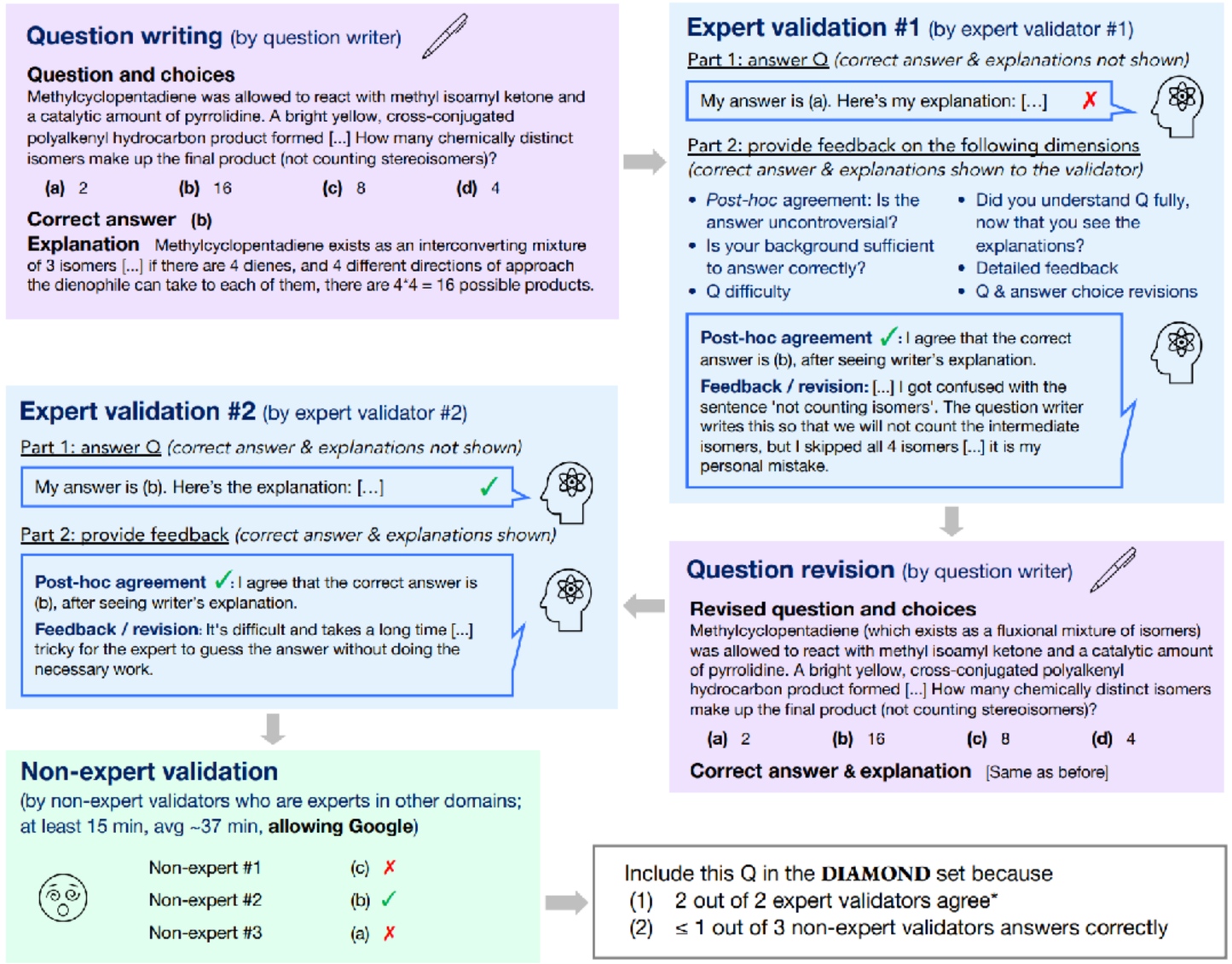
Compute $i + i^2 + i^3 + \cdots + i^{258} + i^{259}$.
(A) -1 (B) 1 (C) i (D) -i
Answer: A

If 4 daps = 7 yaps, and 5 yaps = 3 baps,
how many daps equal 42 baps?
(A) 28 (B) 21 (C) 40 (D) 30
Answer: C

Exam Benchmarks

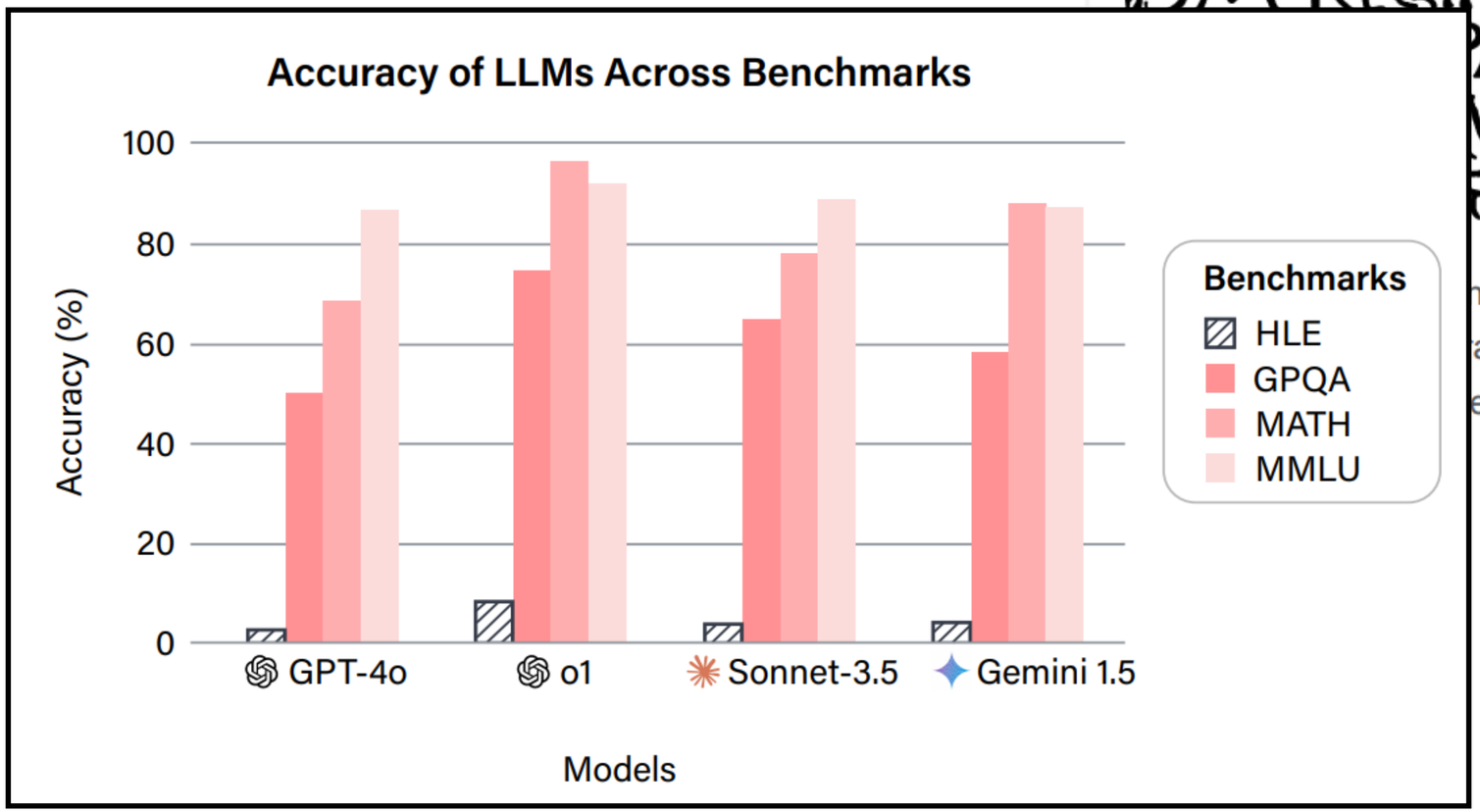
Graduate-level Google-proof QA (GPQA) **[Rein et al., 2023]**

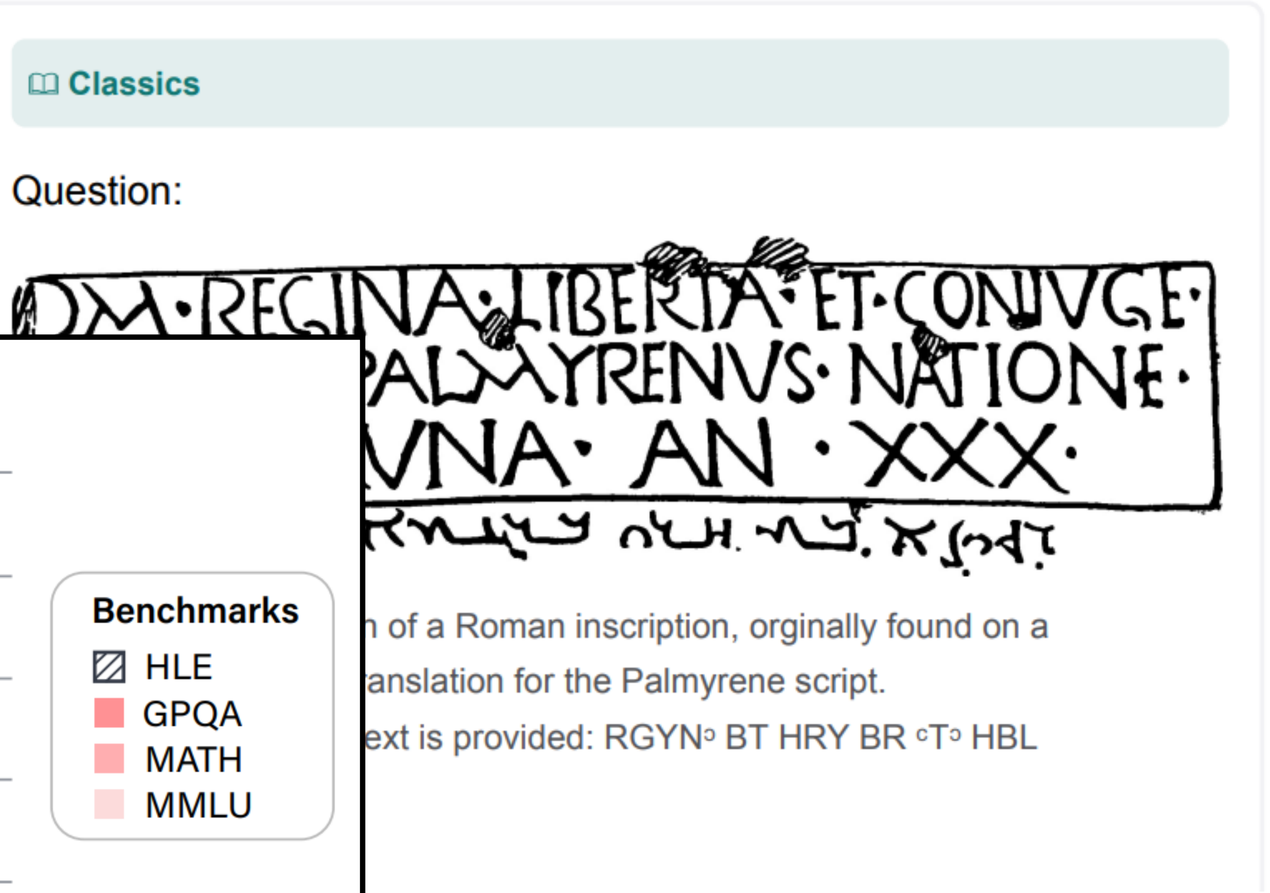
- PhD-level experts get 65% accuracy
- Non-experts achieve 34% accuracy *with access to Google*
- GPT-4 achieves 39%
  - The latest Claude, ChatGPT, and Gemini models all get >94%!



Exam Benchmarks

Humanity's Last Exam (HLE)





Chat Benchmarks

Most people don’t ask their models multiple-choice questions.

Can you give me a good travel itinerary for a four-year trip to Boston?

How do we evaluate how good this is compared to anything else it could have said?

I’d think of it as **four seasons of becoming a Bostonian:**

Year 1 — “I live in Boston now”

Goal: learn the physical city.

- Explore every major Boston neighborhood on foot: Back Bay, Beacon Hill, South End, Fenway, Charlestown, North End, etc.
- Learn the MBTA until you can navigate it without thinking.

...

Year 3 — “New England”

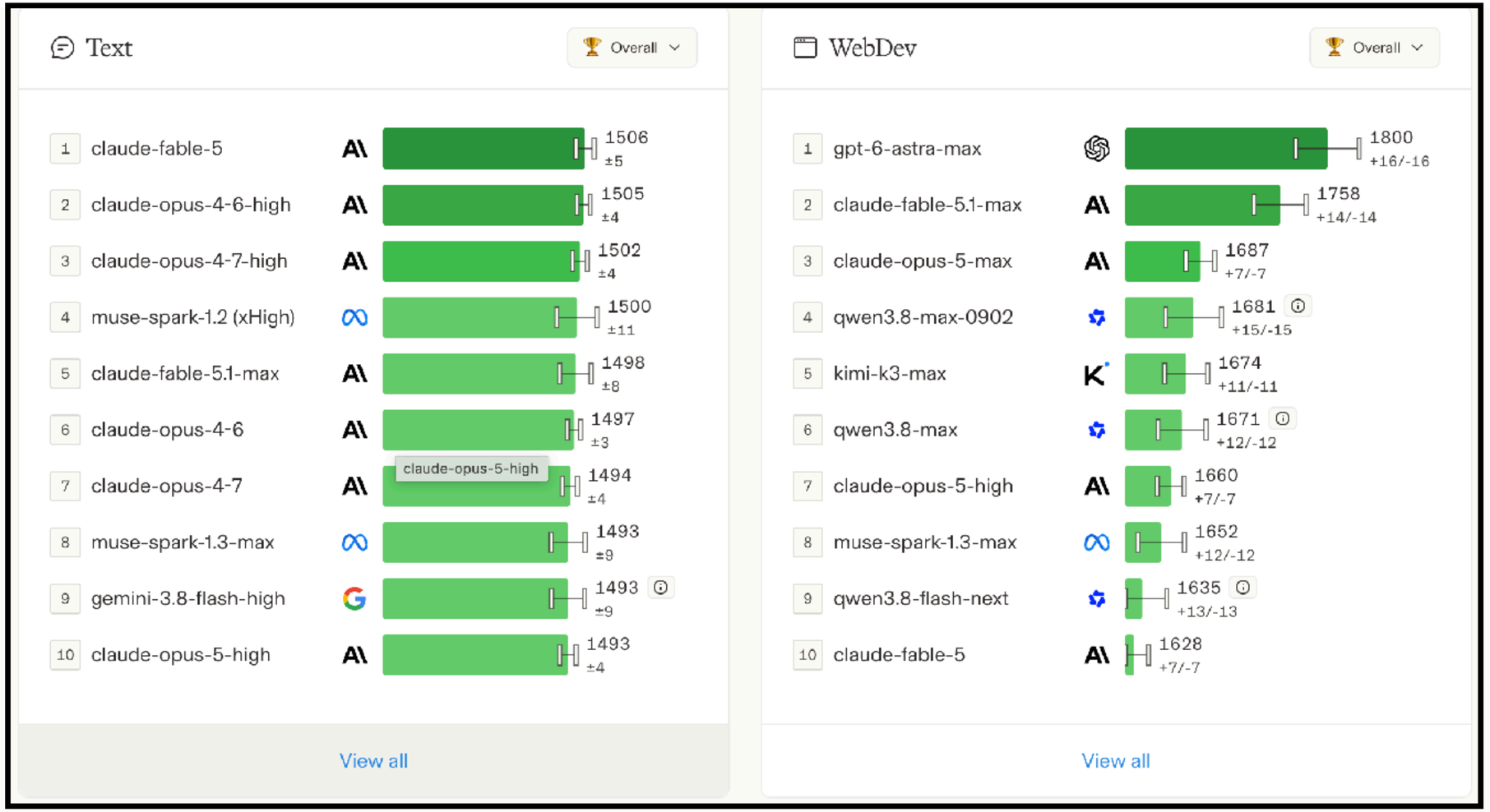
Goal: realize Boston is the center of a much larger region

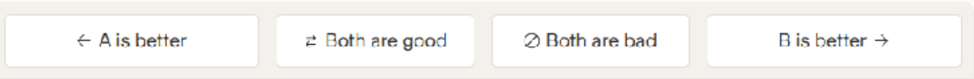
Chat Benchmarks

Arena involves having someone on the Internet type a prompt.

They get a response from two different LMs, and pick which one they prefer.

We can use these preferences to rank models using Bradley-Terry ratings (similar to Elo scores used for chess players).





Evaluation Validity

- What makes a good evaluation?
- **Ecological validity:** how well does the benchmark capture real-world use cases?
  - Exams are far from real-world use
  - Arena is made of real prompts, but distribution is not controlled
- **Train-test generalization:** can models perform well on unseen data?
  - Back in the day, we knew exactly what was in our data
  - These days, we train on the whole internet; companies don’t say what their models were trained on exactly
- **Dataset quality:** Are there enough test cases? Are they well constructed?

How to Devise Your Own Evaluation

- First, *why* are you performing an evaluation?
  - There is no one true/best evaluation: it's all about your use case
  - Researchers often want to measure the raw capabilities of a model
  - Companies want to make purchase decisions about which models to base their systems on
  - We want to understand the risks and benefits of a given system
  - Developers want feedback to improve the model

- Think about *what* you're evaluating
  - Are you evaluating a *method*? An agentic *system*?

- Can you devise single correct answers? Or will you need qualitative judgments (e.g., from a human or LLM?)

How to Devise Your Own Evaluation

- Dataset collection and filtering
    - Internet text? Human-written text?

- Annotation
    - Language modeling data: little to no annotation
    - Classification/Exam: must label every example
    - Relative ranking: need human preferences

- Scoring design
    - What metrics are most appropriate?
    - How will you score? String matching/regex? Have a human or LLM judge quality?