Language modeling

CS 5624: Natural Language Processing

Spring 2025

https://tuvllms.github.io/nlp-spring-2025

Tu Vu

VIRGINIA TECH.

Office hours

- **Instructor:** Tu Vu
  - **Office hours:** Thursday 3:00 - 4:00 PM, D&DS 374

- **Teaching Assistant:** Rishab Balasubramanian
  - **Office hours:** Monday 1:00 - 2:00 PM, D&DS 260E

Office hours (both in-person and via Zoom) will start next Monday, January 27th. Zoom links will be posted on Piazza.

Final project

- The class size has exceeded 50 students and is still growing
- Groups of ~~2-3~~ **4-5**; all groups should be formed by January 31st
- A Google form for submitting group information will be available next week
- Search for teammates on Piazza https://piazza.com/class/m63qacreewc2fs/post/5

  or reach out to us at cs5624instructors@gmail.com

Final project (cont’d)

“If I were given one hour to save the planet, I would spend ***59 minutes*** *defining the problem and one minute resolving it.”*

— Albert Einstein?

Homework

- Homework 0 will be released tomorrow (due February 7th)

Reminder

- Conditional probability $$P(B|A) = \frac{P(A,B)}{P(A)}$$

  Rewriting

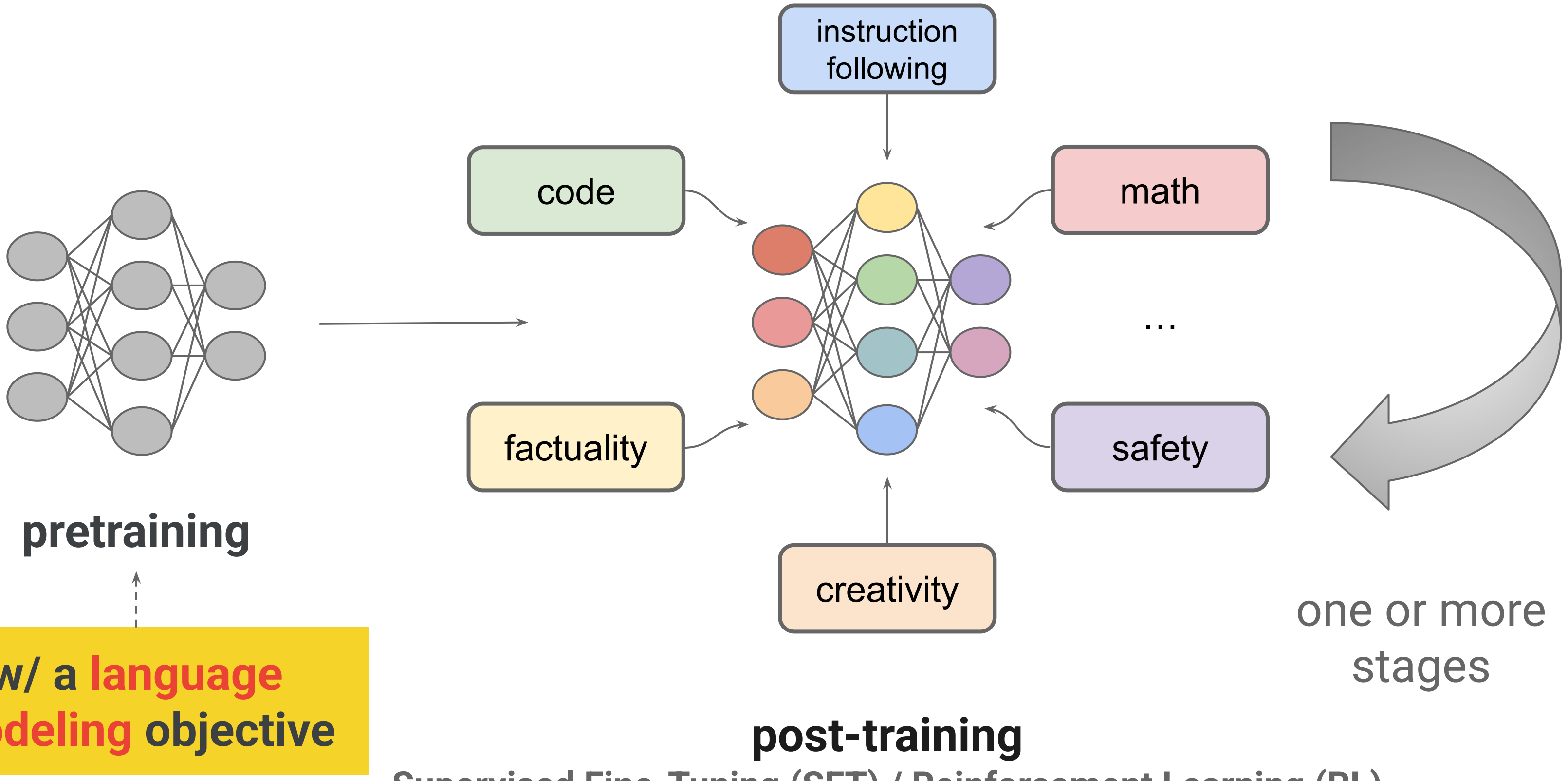
$$P(A,B) = P(A) \times P(B|A)$$

- Chain rule

$$\begin{aligned} P(X_1, X_2, \ldots, X_n) &= P(X_1, X_2, \ldots, X_{n-1}) \times P(X_n|X_1, X_2, \ldots, X_{n-1}) \\ &= P(X_1, X_2, \ldots, X_{n-2}) \times P(X_{n-1}|X_1, X_2, \ldots, X_{n-2}) \times P(X_n|X_1, X_2, \ldots, X_{n-1}) \\ &= P(X_1) \times P(X_2|X_1) \times \ldots \times P(X_n|X_1, X_2, \ldots, X_{n-1}) \end{aligned}$$

$$P(w_1, w_2, \ldots, w_n) = P(w_1) \times P(w_2|w_1) \times \ldots \times P(w_n|w_1, w_2, \ldots, w_{n-1})$$

The development of modern LLMs



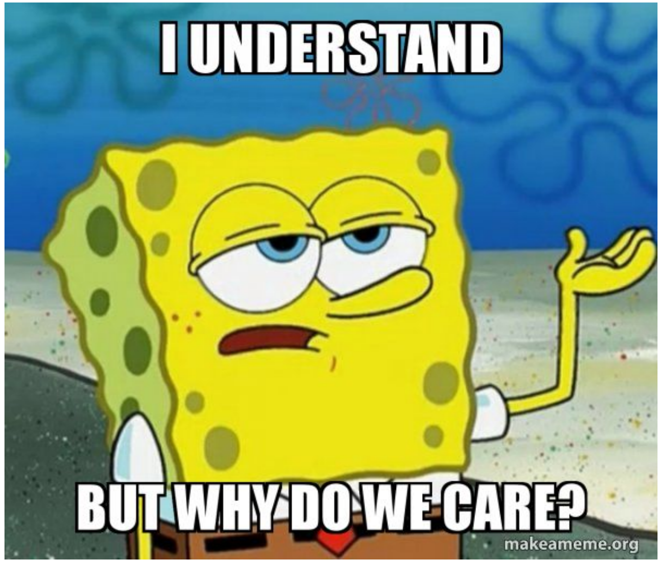
w/ a language modeling objective

Supervised Fine-Tuning (SFT) / Reinforcement Learning (RL)

Language modeling

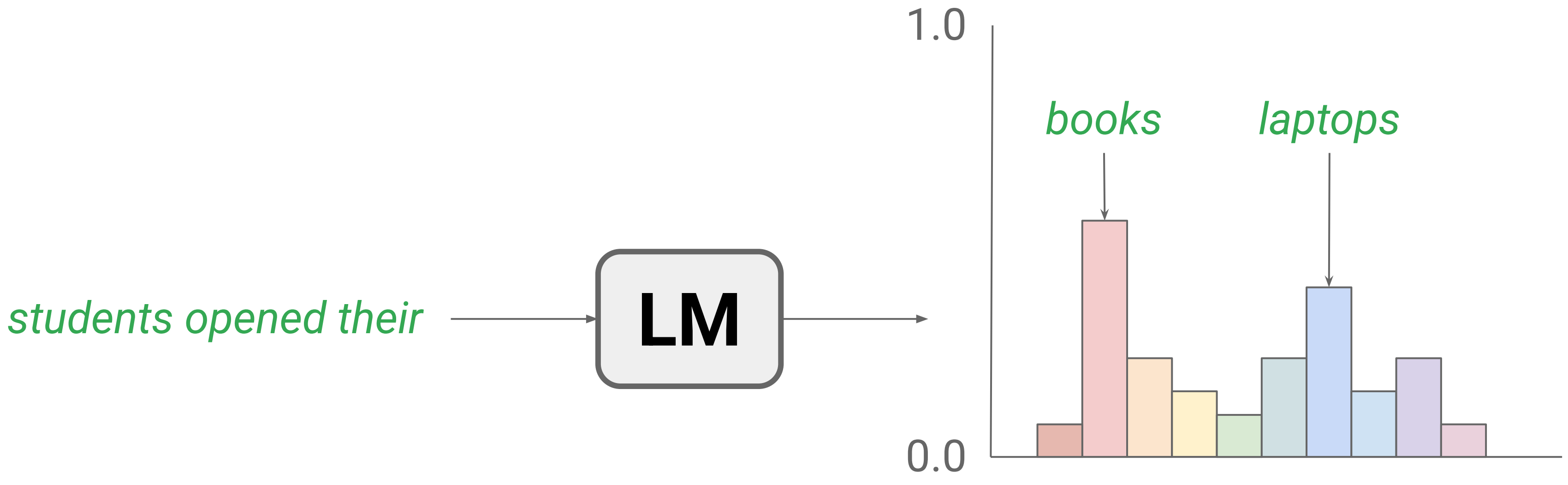
- **Predicting the next/missing word**

  Example: “The cat is on the ___.” → Predicted: “mat”.



What is a language model?

- A machine learning model that assigns a ***probability*** to each possible next word, or a ***probability distribution*** over possible next words



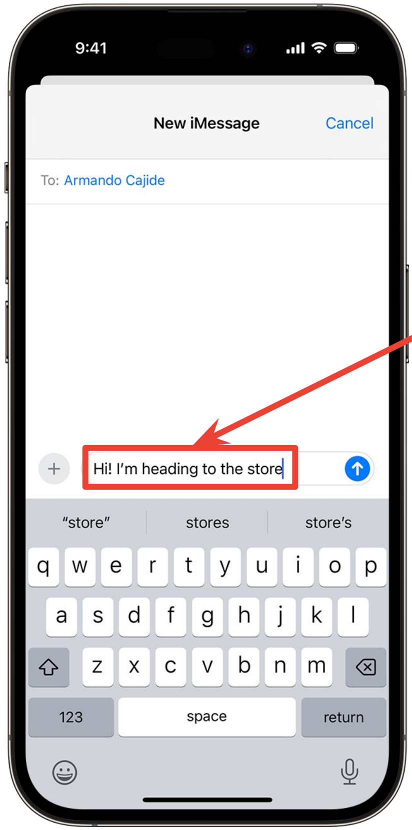
What is a language model? (cont'd)

- A language model can also assign a probability to an entire sentence

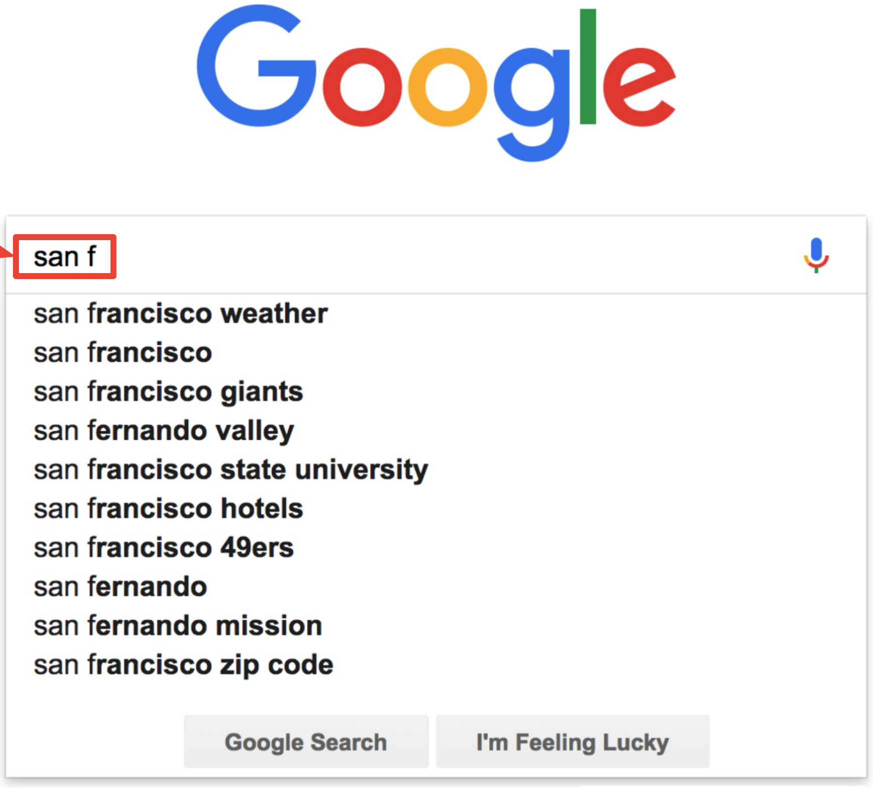
$$P(w_1, w_2, \ldots, w_n) = P(w_1) \times P(w_2|w_1) \times \ldots \times P(w_n|w_1, w_2, \ldots, w_{n-1})$$

P(*"The cat is on the mat"*) = ***P***(*"The"*) *x* ***P***(*"cat" | "The"*) *x* ***P***(*"is" | "The cat"*) *x* ***P***(*"on" | "The cat is"*) *x* ***P***(*"the" | "The cat is on"*) *x* ***P***(*"mat" | "The cat is on the"*)

You use language models everyday!


New iMessage
Cancel
To: Armando Cajide
Hi! I'm heading to the store

prefix


san f
san francisco weather
san francisco
san francisco giants
san fernando valley
san francisco state university
san francisco hotels
san francisco 49ers
san fernando
san fernando mission
san francisco zip code

source: Apple Support

source: Google Blog

Two categories of language models

- Statistical language models
    - N-gram / Count-based language models

- Neural language models (e.g., ChatGPT, Gemini)

N-grams

- An n-gram is a sequence of n words
- Unigram (n=1)
  - “The”, “water”, “of”, “Walden”, “Pond”
- Bigram (n=2)
  - “The water”, “water of”, “of Walden”, “Pond”
- Trigram (n=3)
  - “The water of”, “water of Walden”, “of Walden Pond”
- 4-gram
- ...

N-grams (cont'd)

- Notation
  - **word type**: a unique word in our vocabulary
  - **token**: an individual occurrence of a word type

Example: "I am Sam. Sam am I. I do not like green eggs and ham."

→ one word type of "I", three tokens of "I"

N-grams (cont'd)

- How to compute the probabilities?

$$P(w_1, w_2, \ldots, w_n) = P(w_1) \times P(w_2|w_1) \times \ldots \times \boxed{P(w_n|w_1, w_2, \ldots, w_{n-1})}$$

P(*"blue" | "The water of Walden Pond is so beautifully"*)

What is the problem with this approach?

The Markov assumption

- n-gram model: Approximate the prefix by just the last ***n-1*** words
- bigram (n=2) model

$$P(\text{“blue”} \mid \text{“The water of Walden Pond is so beautifully”}) \\ = P(\text{“blue”} \mid \text{beautifully”})$$

- trigram (n=3) model

$$P(\text{“blue”} \mid \text{“The water of Walden Pond is so beautifully”}) \\ = P(\text{“blue”} \mid \text{so beautifully”})$$

The Markov assumption (cont'd)

- unigram model

$$\begin{aligned} P(w_1, w_2, \dots, w_n) &= P(w_1) \times P(w_2|w_1) \times \dots \times P(w_n|w_1, w_2, \dots, w_{n-1}) \\ &\approx P(w_1) \times P(w_2) \times \cdots \times P(w_n) \\ &= \prod_{k=1}^{n} P(w_k) \end{aligned}$$

- bigram model

$$\begin{aligned} P(w_1, w_2, \dots, w_n) &\approx P(w_1) \times P(w_2|w_1) \times \cdots \times P(w_n|w_{n-1}) \\ &= \prod_{k=1}^{n} P(w_k|w_{k-1}) \end{aligned}$$

Maximum likelihood estimation (MLE)

$$P(w_n|w_{n-1}) = \frac{Count(w_{n-1}w_n)}{\sum_w Count(w_{n-1}w)} = \frac{Count(w_{n-1}w_n)}{Count(w_{n-1})}$$

relative frequency

```
<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>
```

Here are the calculations for some of the bigram probabilities

$P(\texttt{I}|\texttt{<s>}) = \frac{2}{3} = 0.67$ $P(\texttt{Sam}|\texttt{<s>}) = \frac{1}{3} = 0.33$ $P(\texttt{am}|\texttt{I}) = \frac{2}{3} = 0.67$

$P(\texttt{</s>}|\texttt{Sam}) = \frac{1}{2} = 0.5$ $P(\texttt{Sam}|\texttt{am}) = \frac{1}{2} = 0.5$ $P(\texttt{do}|\texttt{I}) = \frac{1}{3} = 0.33$

Example

- From a restaurant corpus

  “can you tell me about any good cantonese restaurants close by”

  “tell me about chez panisse”

  “i’m looking for a good place to eat breakfast”

  “when is caffe venezia open during the day”

Example (cont'd)

unigram counts

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

target

prefix

want followed i 827 times

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Example (cont'd)

827/2533

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

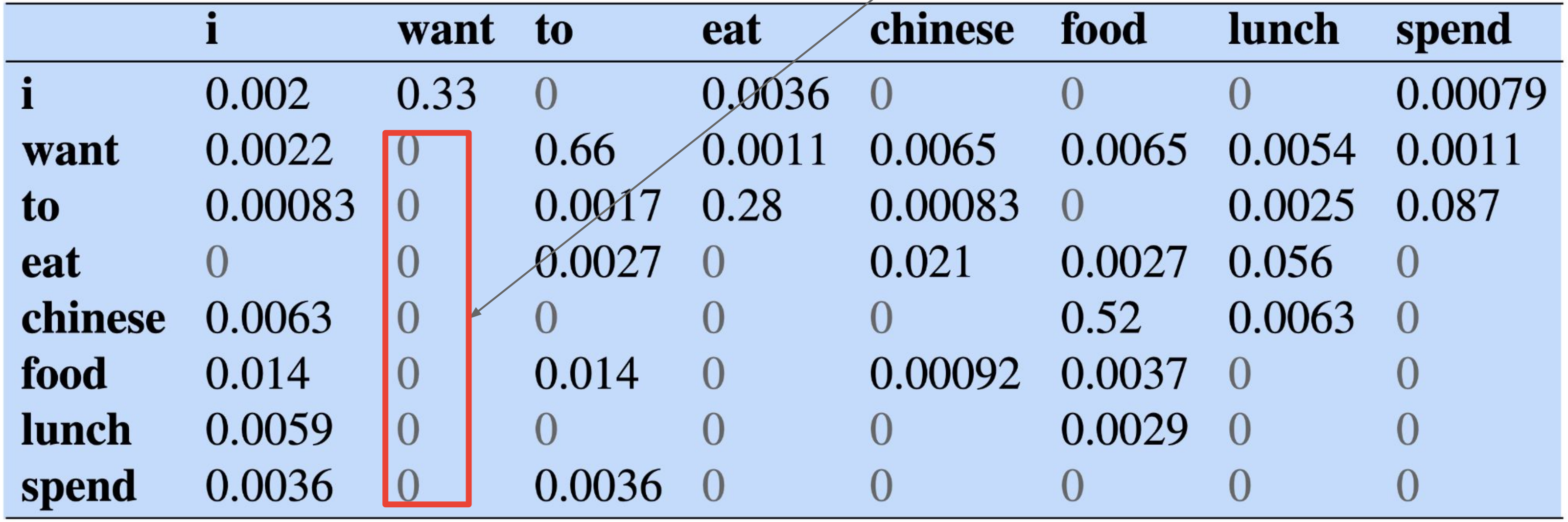
Here are a few other useful probabilities:

$$P(\texttt{i}|\texttt{<s>}) = 0.25$$
$$P(\texttt{food}|\texttt{english}) = 0.5$$
$$P(\texttt{english}|\texttt{want}) = 0.0011$$
$$P(\texttt{</s>}|\texttt{food}) = 0.68$$

$$\begin{aligned} P(\texttt{<s> i want english food </s>}) &= P(\texttt{i}|\texttt{<s>})P(\texttt{want}|\texttt{i})P(\texttt{english}|\texttt{want}) \\ &\qquad P(\texttt{food}|\texttt{english})P(\texttt{</s>}|\texttt{food}) \\ &= 0.25 \times 0.33 \times 0.0011 \times 0.5 \times 0.68 \\ &= 0.000031 \end{aligned}$$

Example (cont'd)

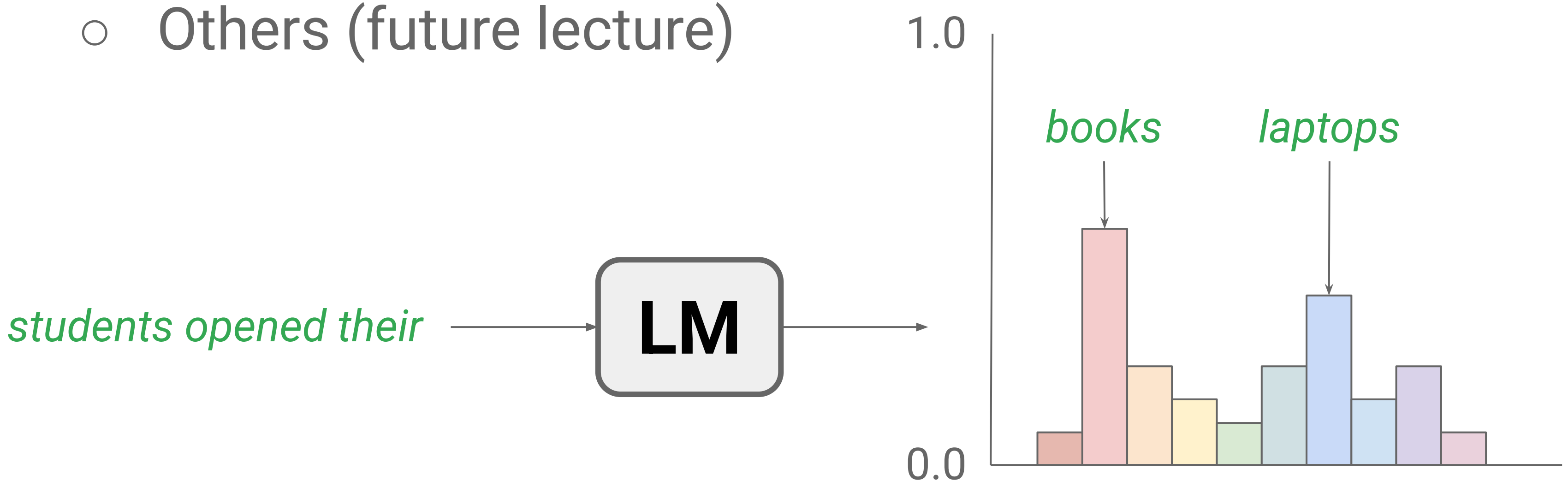
sparsity issue



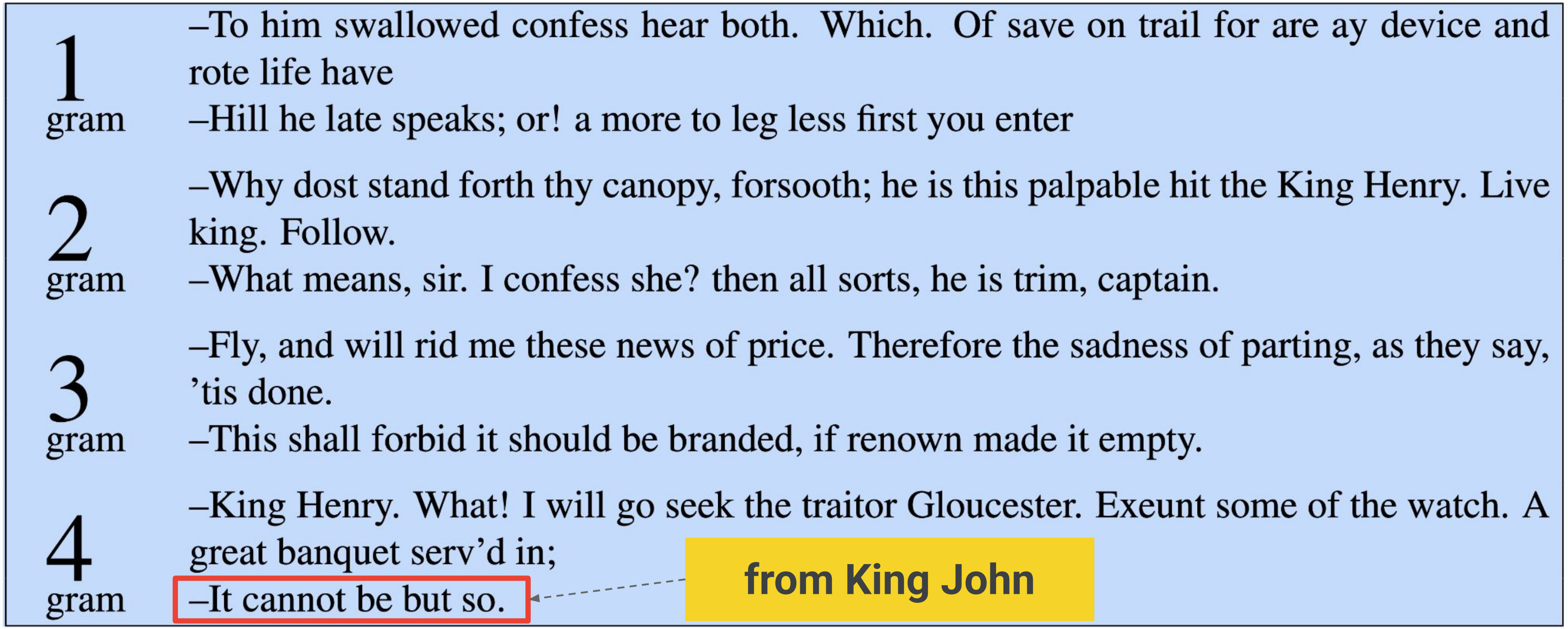
	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

How to sample sentences from a language model?

- Decoding strategies
  - Greedy decoding
  - Sampling
  - Others (future lecture)



Sample generations



1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have –Hill he late speaks; or! a more to leg less first you enter
2 gram	–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. –What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done. –This shall forbid it should be branded, if renown made it empty.
4 gram	–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in; –It cannot be but so.

from King John

Figure 3.4 Eight sentences randomly generated from four n-grams computed from Shakespeare's works. All characters were mapped to lower-case and punctuation marks were treated as words. Output is hand-corrected for capitalization to improve readability.

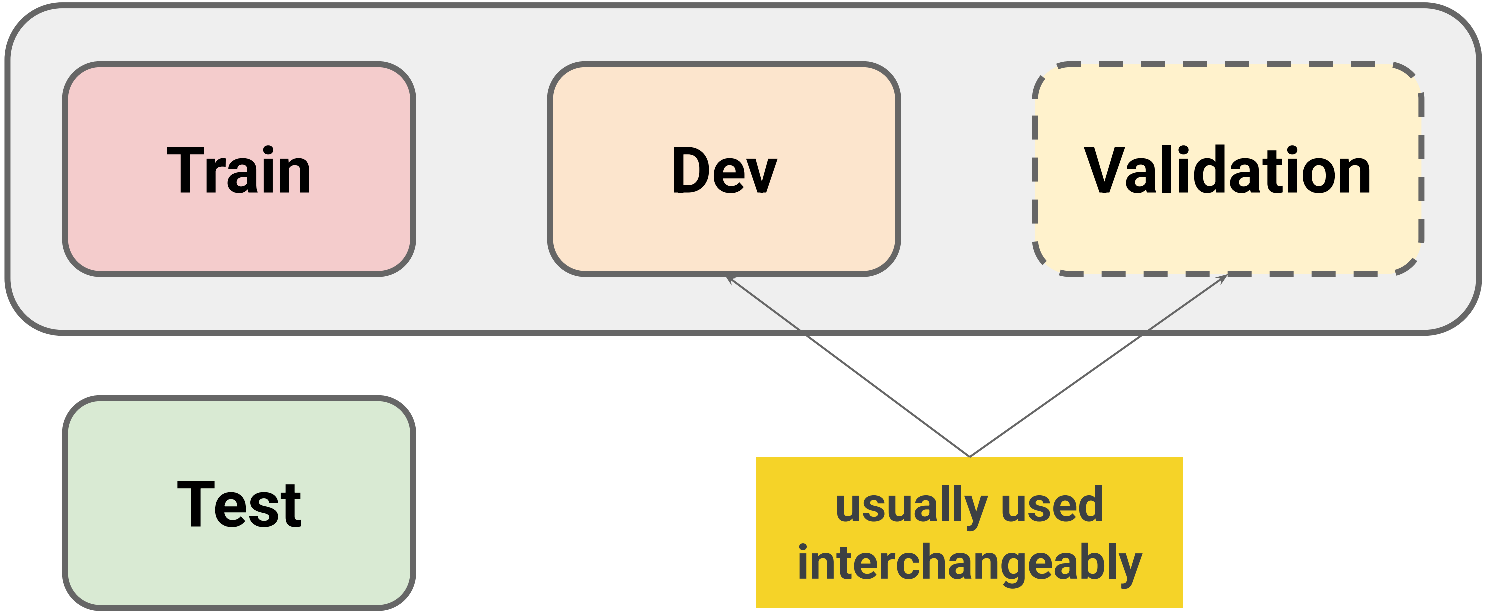
Should we increase the value of n?

- As n increases, the number of possible nnn-grams grows exponentially (many n-grams have insufficient or no data)
- Storing and processing large n-grams requires more memory and computational power
- Beyond a certain point, increasing n may not yield significant performance improvements, especially if the dataset does not contain sufficient examples of longer n-grams

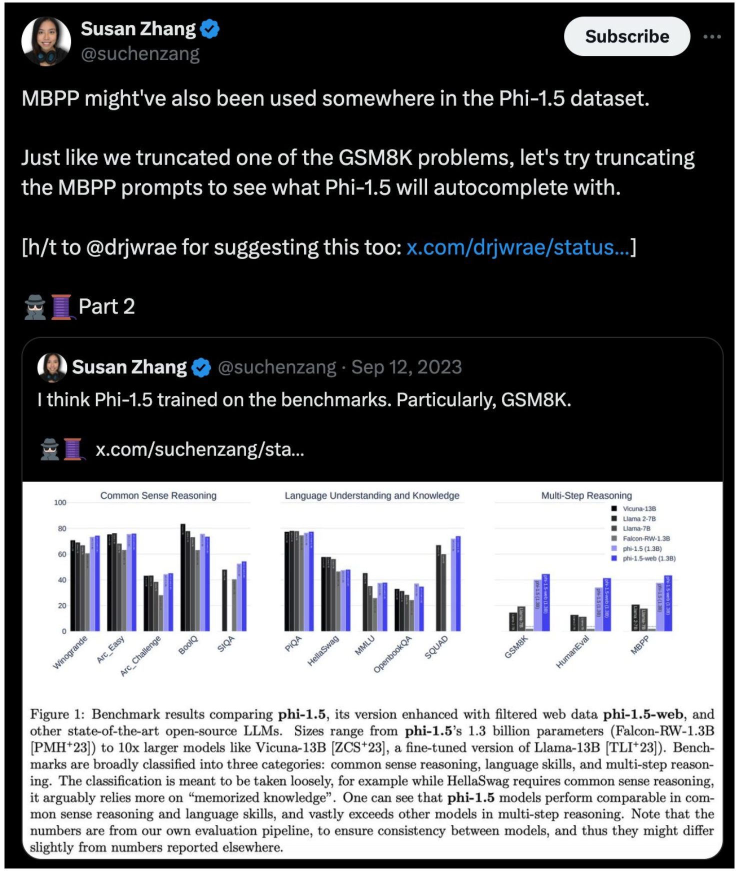
Shakespeare as corpus

- T=884,647 tokens, V=29,066
- Shakespeare produced 300,000 bigram types out of V^2= 844,000,000 possible bigrams.
- **99.96%** of the possible bigrams have zero entries in the bigram table (were never seen)!

Evaluating language models



Never train on the test set!



Susan Zhang @suchenzang

Subscribe

MBPP might've also been used somewhere in the Phi-1.5 dataset.

Just like we truncated one of the GSM8K problems, let's try truncating the MBPP prompts to see what Phi-1.5 will autocomplete with.

[h/t to @drjwrae for suggesting this too: x.com/drjwrae/status...]

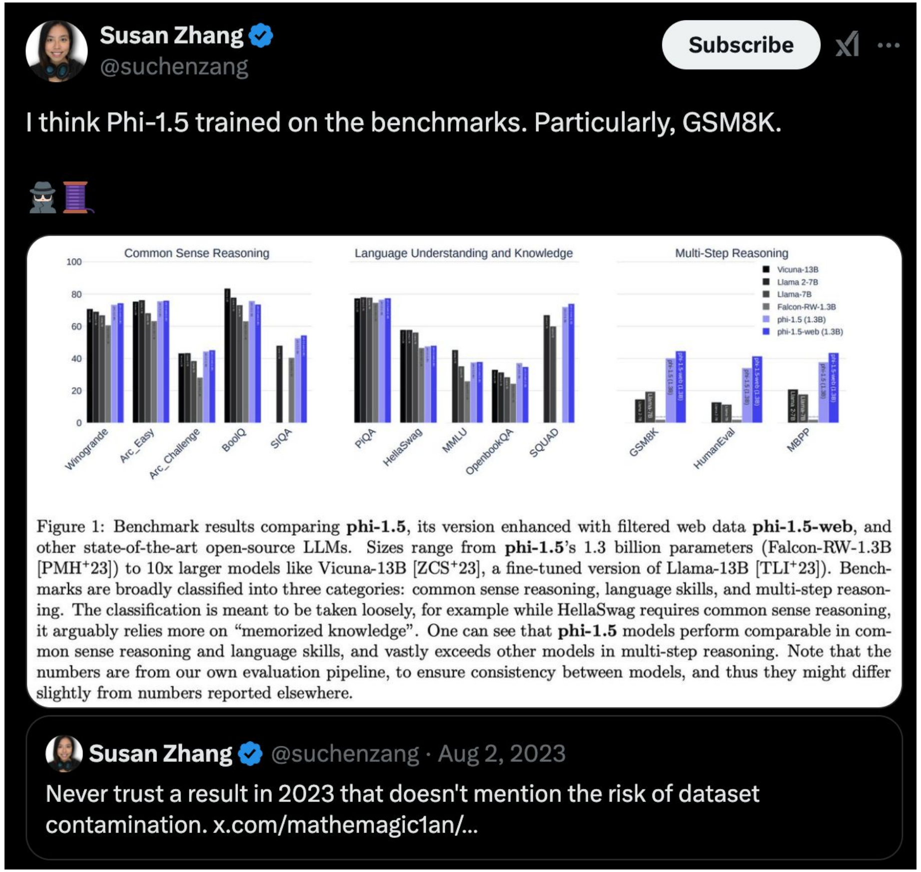
🕵️🧵 Part 2

Susan Zhang @suchenzang · Sep 12, 2023

I think Phi-1.5 trained on the benchmarks. Particularly, GSM8K.

🕵️🧵 x.com/suchenzang/sta...

Figure 1: Benchmark results comparing **phi-1.5**, its version enhanced with filtered web data **phi-1.5-web**, and other state-of-the-art open-source LLMs. Sizes range from **phi-1.5**'s 1.3 billion parameters (Falcon-RW-1.3B [PMH+23]) to 10x larger models like Vicuna-13B [ZCS+23], a fine-tuned version of Llama-13B [TLI+23]). Benchmarks are broadly classified into three categories: common sense reasoning, language skills, and multi-step reasoning. The classification is meant to be taken loosely, for example while HellaSwag requires common sense reasoning, it arguably relies more on "memorized knowledge". One can see that **phi-1.5** models perform comparable in common sense reasoning and language skills, and vastly exceeds other models in multi-step reasoning. Note that the numbers are from our own evaluation pipeline, to ensure consistency between models, and thus they might differ slightly from numbers reported elsewhere.



Susan Zhang @suchenzang

Subscribe

I think Phi-1.5 trained on the benchmarks. Particularly, GSM8K.

🕵️🧵

Figure 1: Benchmark results comparing **phi-1.5**, its version enhanced with filtered web data **phi-1.5-web**, and other state-of-the-art open-source LLMs. Sizes range from **phi-1.5**'s 1.3 billion parameters (Falcon-RW-1.3B [PMH+23]) to 10x larger models like Vicuna-13B [ZCS+23], a fine-tuned version of Llama-13B [TLI+23]). Benchmarks are broadly classified into three categories: common sense reasoning, language skills, and multi-step reasoning. The classification is meant to be taken loosely, for example while HellaSwag requires common sense reasoning, it arguably relies more on "memorized knowledge". One can see that **phi-1.5** models perform comparable in common sense reasoning and language skills, and vastly exceeds other models in multi-step reasoning. Note that the numbers are from our own evaluation pipeline, to ensure consistency between models, and thus they might differ slightly from numbers reported elsewhere.

Susan Zhang @suchenzang · Aug 2, 2023

Never trust a result in 2023 that doesn't mention the risk of dataset contamination. x.com/mathemagic1an/...

Perplexity

$$\begin{aligned}\text{perplexity}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}\end{aligned}$$

We normalize by the number of words N by taking the Nth root

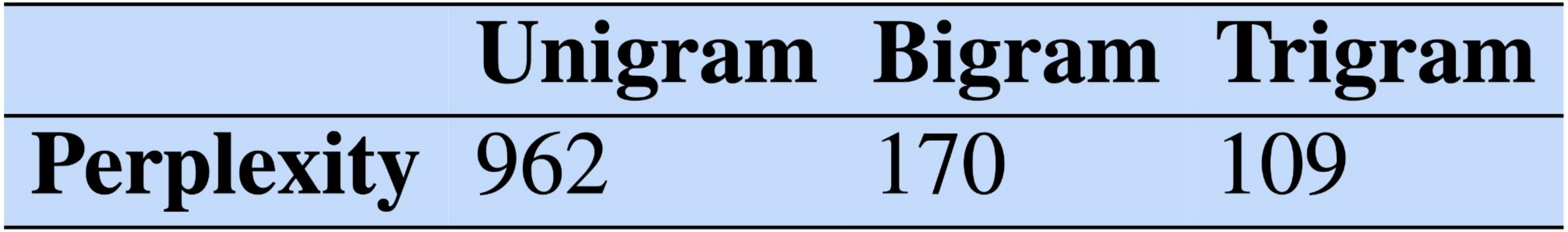
Or we can use the chain rule to expand the probability of W:

$$\text{perplexity}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 \ldots w_{i-1})}}$$

Perplexity as Weighted Average Branching Factor

- Suppose a sentence consists of random digits.
  What is the perplexity of this sentence for a model that assigns a probability of 1/10 to each digit?

Lower perplexity = Better language model



	Unigram	Bigram	Trigram
Perplexity	962	170	109

In practice, we use log probs

$$log \prod p(w_i|w_{i-1}) = \sum logp(w_i|w_{i-1})$$

logs to avoid numerical underflow

sentence: I love love love love love the movie

$$p(\text{i}) \cdot p(\text{love})^5 \cdot p(\text{the}) \cdot p(\text{movie}) = 5.95374181\text{e-}7$$

$$\log p(\text{i}) + 5 \log p(\text{love}) + \log p(\text{the}) + \log p(\text{movie})$$

$$= -14.3340757538$$

In practice, we use log probs (cont'd)

$$perplexity(W) = exp(-\frac{1}{N}\sum_{i}^{N} logp(w_i|w_{<i}))$$

perplexity is the exponentiated token-level negative log-likelihood

Infini-gram: Scaling Unbounded n-gram Language Models to a Trillion Tokens

https://arxiv.org/pdf/2401.17377

Thank you!