N-gram Language Models

EECS 183/283a: Natural Language Processing

N-Grams

- An *n*-gram is a sequence of *n* tokens
- Given a vocabulary $\mathcal{V}$, the set of all possible *n*-grams is $\mathcal{V}^n$

A language model assigns a

1-grams (unigrams): { a, assigns, language, model, probability, of, sequence, tokens } $\mathcal{V}$

2-grams (bigrams): { a a, a assigns, a language, ... , tokens of, tokens sequence, tokens tokens } $\mathcal{V}^2$

3-grams (trigrams): { a a a, a a assigns, a a language, ... , tokens tokens of, tokens tokens sequence, tokens tokens tokens} $\mathcal{V}^3$

...

Autoregressive Language Models

$$p(\overline{x}) = \prod_{i=1}^{|\overline{x}|} p(x_i \mid x_1, \ldots, x_{i-1})$$

$$= p(x_1)\, p(x_2 \mid x_1) \ldots p(x_n \mid x_1, \ldots, x_{n-1})$$

probability that the first word is x_1

probability that the second word is x_2, given that the first word is x_1

probability that the sentence ends after the sequence $\langle x_1, \ldots, x_{n-1} \rangle$

Core modeling challenge:

How do we compute these conditional probabilities?

Autoregressive N-Gram Language Models

$$p(\overline{x}) = \prod_{i=1}^{|\overline{x}|} p(x_i \mid x_1, \ldots, x_{i-1})$$

$$= p(x_1)p(x_2 \mid x_1) \ldots p(x_n \mid x_1, \ldots, x_{n-1})$$

$$p(X_i = x) = p(x \mid x_1, \ldots, x_{i-1})$$

Challenge: how do we parameterize this if there can be arbitrarily many previous tokens?

Let's make a Markov assumption:
The probability of word at index *i* only depends on the *n* - 1 words that came before it

Autoregressive N-Gram Language Models

$$p(\overline{x}) = \prod_{i=1}^{|\overline{x}|} p(x_i \mid x_1, \ldots, x_{i-1})$$

$$= p(x_1)p(x_2 \mid x_1) \ldots p(x_n \mid x_1, \ldots, x_{n-1})$$

$$p(X_i = x) = p(x \mid x_1, \ldots, x_{i-1})$$

$$\approx p(x \mid x_{i-n+1}, \ldots, x_{i-1})$$

Let's make a Markov assumption:

The probability of word at index *i* only depends on the *n* - 1 words that came before it

Autoregressive N-Gram Language Models

$$p(\overline{x}) = \prod_{i=1}^{|\overline{x}|} p(x_i \mid x_1, \ldots, x_{i-1})$$

$$= p(x_1)p(x_2 \mid x_1) \ldots p(x_n \mid x_1, \ldots, x_{n-1})$$

$$p(X_i = x) = p(x \mid x_1, \ldots, x_{i-1})$$

$$\approx p(x \mid x_{i-n+1}, \ldots, x_{i-1})$$

Preceding (n-1)-gram

$$\approx \frac{C(x_{i-n+1}, \ldots, x_{i-1}, x)}{C(x_{i-n+1}, \ldots, x_{i-1})}$$

count of n-gram

count of prev (n-1)-gram

N-Gram Language Models

- Now, all we need to model is *n*-gram and (*n*-1)-gram probabilities!

- We can do this by counting *n*-gram occurrences in the wild

- Simplest *n*-gram language model: *n* = 1 (unigrams, aka bag of words)

$$p(X_i = x) \approx p(x) \in \Delta^{\mathcal{V}}$$

count of word *x* across corpus

$$= \frac{C(x)}{\sum_{\overline{x} \in \mathcal{D}} |\overline{x}|}$$

total number of words in corpus

Unigram Language Model

$$p(\overline{x}) = \prod_{i=1}^{|\overline{x}|} p(x_i \mid x_1, \ldots, x_{i-1})$$

$$\approx \prod_{i-1}^{|\overline{x}|} p(x_i)$$

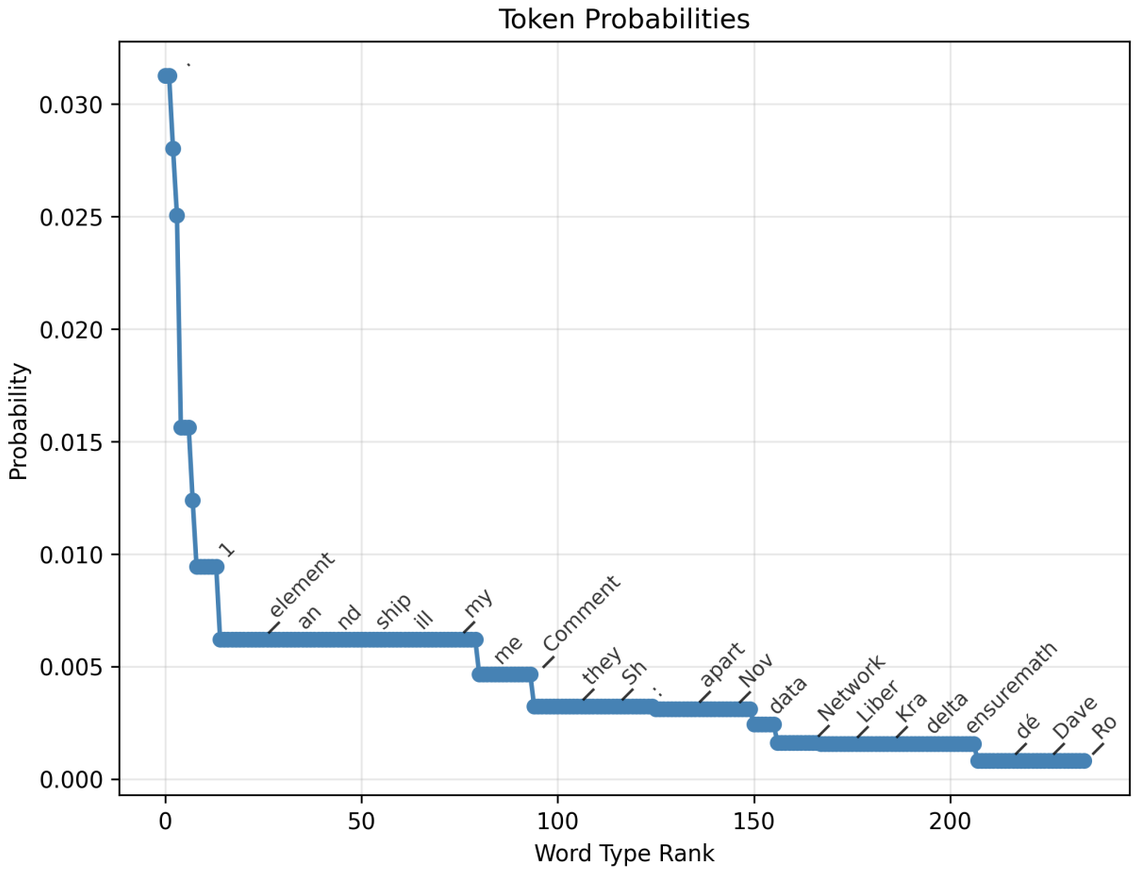
$$p(\text{yellow suitcase and red hat}) = p(\text{red suitcase and yellow hat})$$

Word order does not matter!

This is why it’s called “bag of words”

Unigram Language Model

$p(X)$ (unigram probabilities)



$$x_1 \sim p(X)$$
$$x_2 \sim p(X)$$

based
-it
displayed
-b
were
-an
-ly
-is
EOS

basedit displayedb wereanlyis

What are the parameters of this model?

How many parameters are there?

Infini-gram (Liu et al. 2024) stats, vibecoded with Claude

Bigram Language Model
(*n* = 2)

$$p(X_i = x) \approx \frac{C(x_{i-n+1}, \ldots, x_{i-1}, x)}{C(x_{i-n+1}, \ldots, x_{i-1})}$$

$$= \frac{C(x_{i-1}, x_i)}{C(x_{i-1})}$$

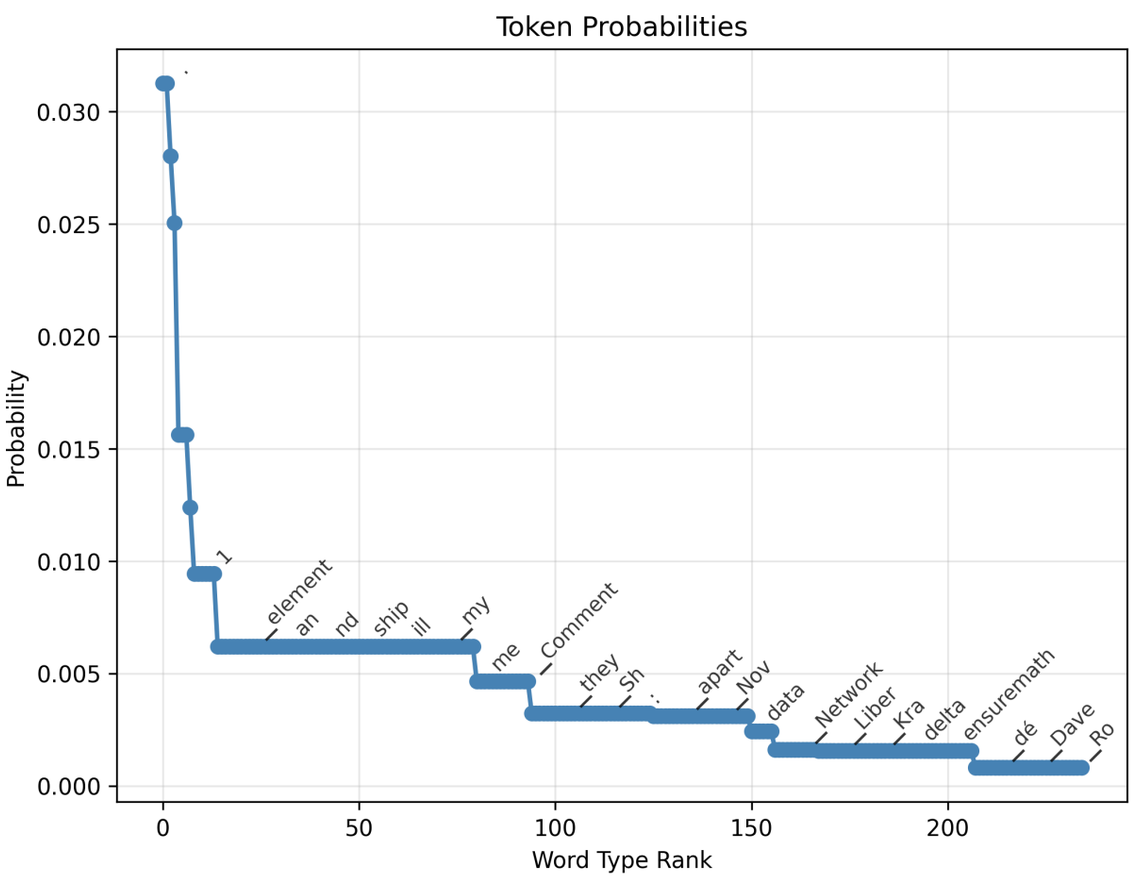
$$p(\overline{x}) \approx \prod_{i=1}^{|\overline{x}|} \frac{C(x_{i-1}, x_i)}{C(x_{i-1})}$$

Bigram Language Model ($n = 2$)

- First sample a start token using the empirical first-token distribution:

$$p(X_1 = x) = \frac{C(\texttt{BOS}, x)}{C(\texttt{BOS})} = \frac{C(\texttt{BOS}, x)}{|\mathcal{D}|}$$

$$p(X \mid \texttt{BOS})$$



my

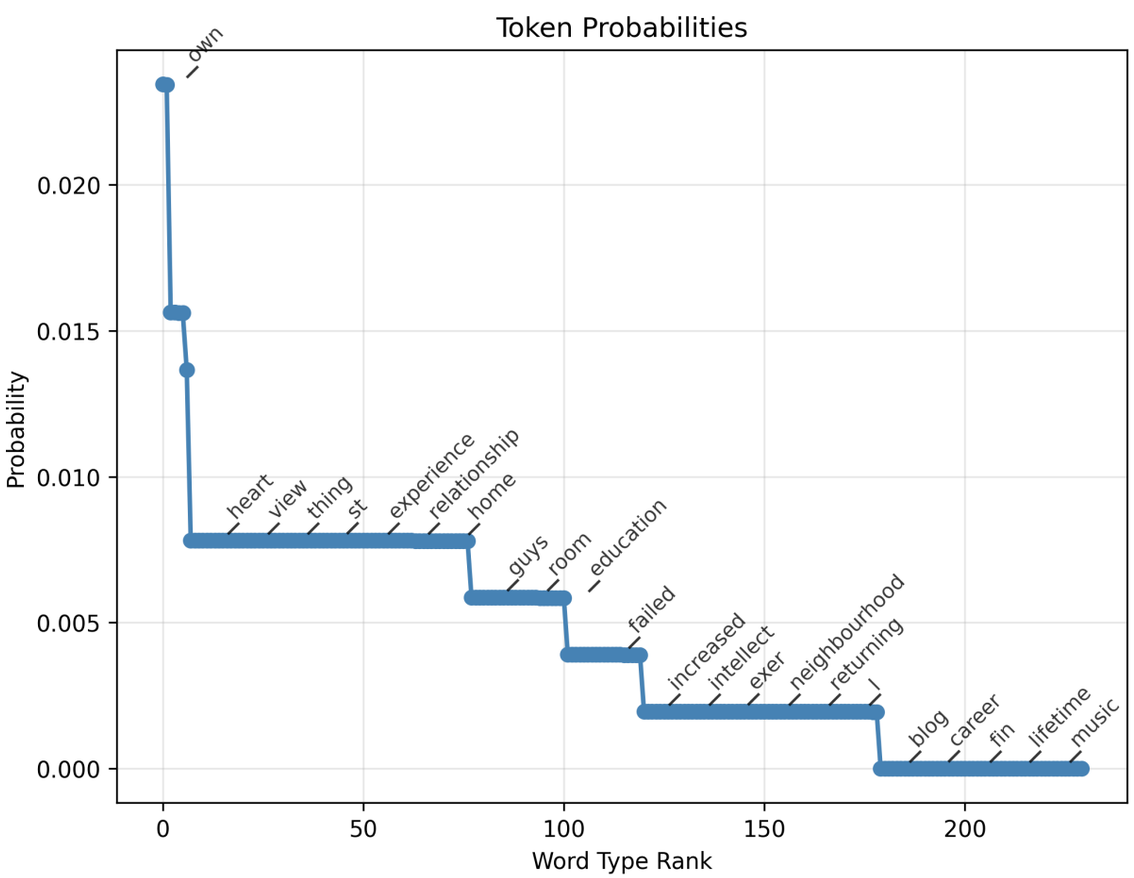
Infini-gram (Liu et al. 2024) stats, vibecoded with Claude

Bigram Language Model ($n = 2$)

- Then sample conditioned on the previous word:

$$p(X_2 = x) = \frac{C(\text{my}, x)}{C(\text{my})}$$

$$p(X \mid \text{my})$$



my
mother

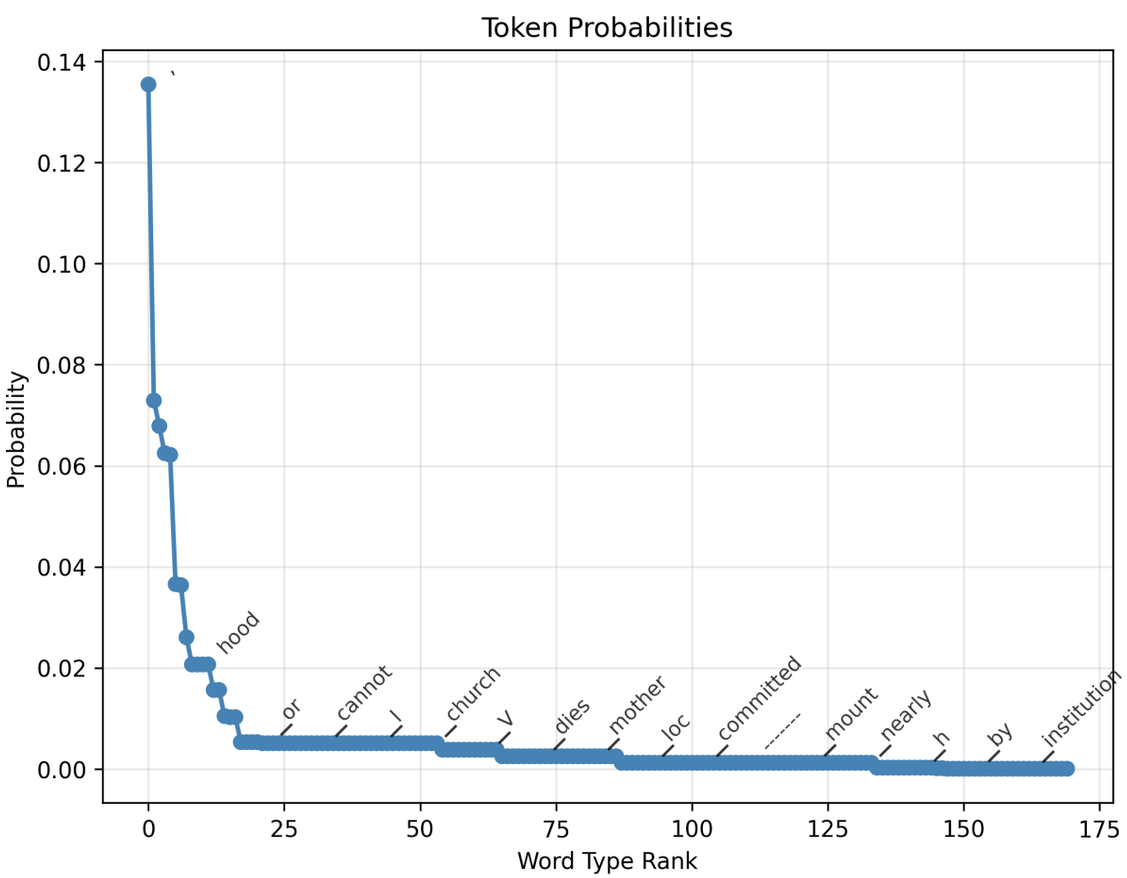
Infini-gram (Liu et al. 2024) stats, vibecoded with Claude

Bigram Language Model
(*n* = 2)

- Then sample conditioned on the previous word:

$$p(X_3 = x) = \frac{C(\text{mother}, x)}{C(\text{mother})}$$

$$p(X \mid \text{mother})$$



my
mother
and

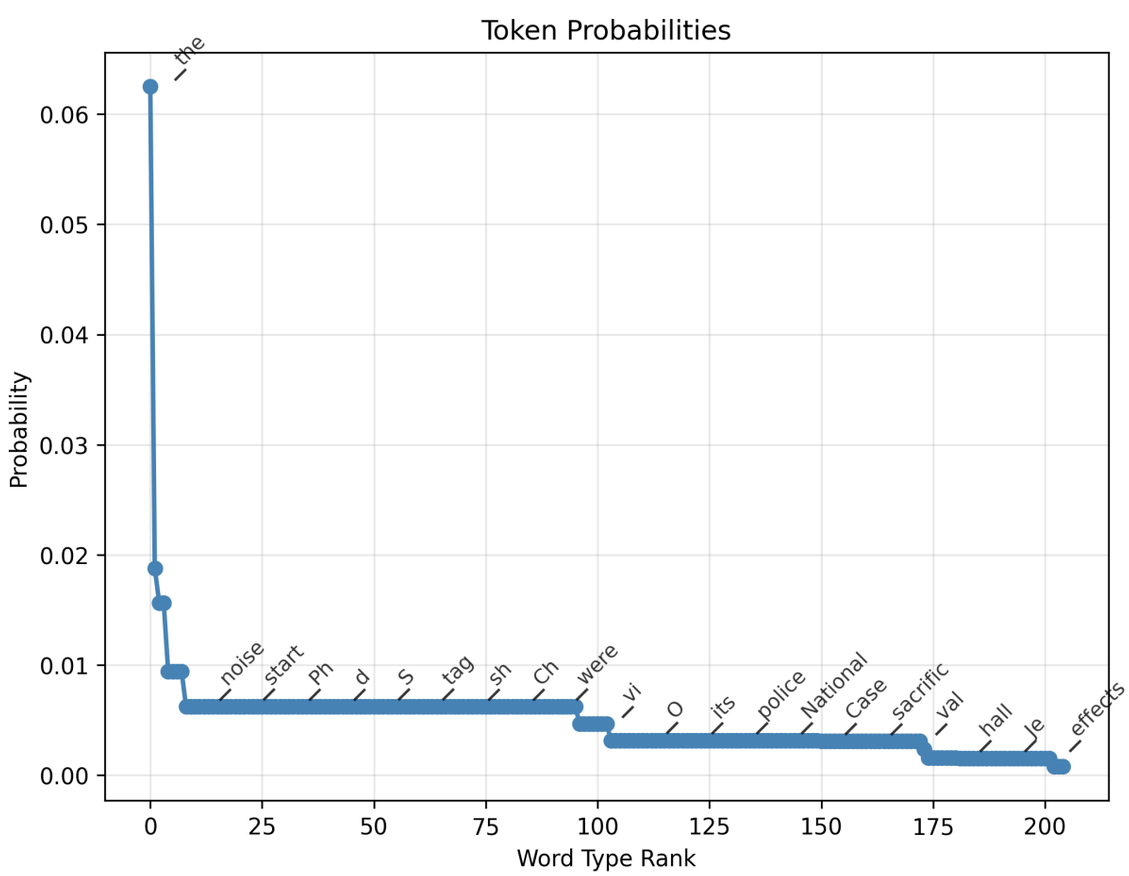
Infini-gram (Liu et al. 2024) stats, vibecoded with Claude

Bigram Language Model ($n = 2$)

- Then sample conditioned on the previous word:

$$p(X_4 = x) = \frac{C(\text{and}, x)}{C(\text{and})}$$

$$p(X \mid \text{and})$$



my
mother
and
my

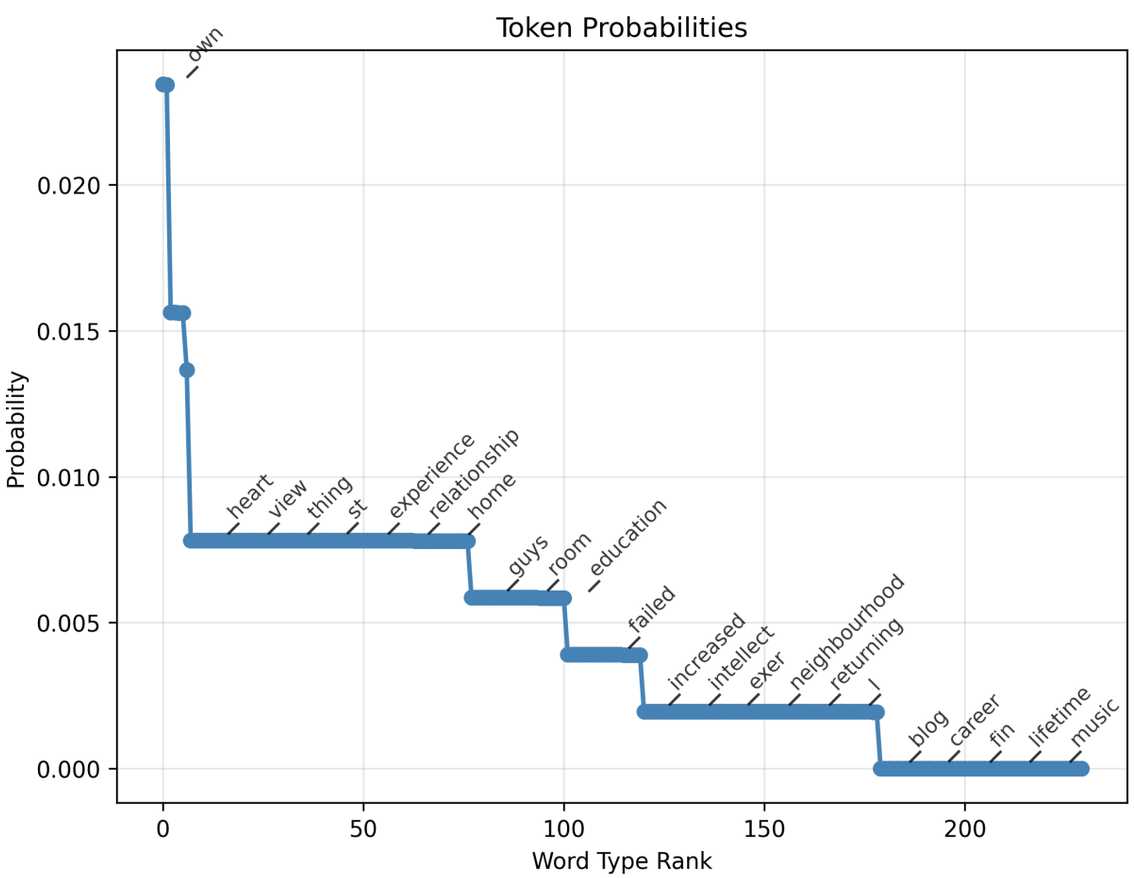
Infini-gram (Liu et al. 2024) stats, vibecoded with Claude

Bigram Language Model ($n = 2$)

- Then sample conditioned on the previous word:

$$p(X_5 = x) = \frac{C(\text{my}, x)}{C(\text{my})}$$

$$p(X \mid \text{my})$$



my
mother
and
my
mother
...

Infini-gram (Liu et al. 2024) stats, vibecoded with Claude

Generalization of *n*-gram Language Model

$$p(X_i = x) \approx \frac{C(x_{i-n+1}, \ldots, x_{i-1}, x)}{C(x_{i-n+1}, \ldots, x_{i-1})}$$

n=1				
never				
Comment				
has				
in				
.				
“				
t				
view				
C				
never Comment has in . “t view C				

Sampled from Infini-gram (Liu et al. 2024)

Generalization of *n*-gram Language Model

$$p(X_i = x) \approx \frac{C(x_{i-n+1}, \ldots, x_{i-1}, x)}{C(x_{i-n+1}, \ldots, x_{i-1})}$$

n=1	n=2			
never	view			
Comment	-find			
has	a			
in	place			
.	of			
“	a			
t	human			
view	intelligence			
C	brief			
never Comment has in . “t view C	***viewfind a place of a human intelligence brief***			

Sampled from Infini-gram (Liu et al. 2024)

Generalization of *n*-gram Language Model

$$p(X_i = x) \approx \frac{C(x_{i-n+1}, \ldots, x_{i-1}, x)}{C(x_{i-n+1}, \ldots, x_{i-1})}$$

n=1	n=2	n=3		
never	view	were		
Comment	-find	e		
has	a	-colog		
in	place	-ical		
.	of	topics		
“	a	related		
t	human	T		
view	intelligence	-weet		
C	brief	niet		
never Comment has in . “t view C	***viewfind a place of a human intelligence brief***	***were ecological topics related to Tweet niet***		

Sampled from Infini-gram (Liu et al. 2024)

Generalization of *n*-gram Language Model

$$p(X_i = x) \approx \frac{C(x_{i-n+1}, \ldots, x_{i-1}, x)}{C(x_{i-n+1}, \ldots, x_{i-1})}$$

n=1	n=2	n=3	n=4	
never	view	were	dropped	
Comment	-find	e	her	
has	a	-colog	off	
in	place	-ical	at	
.	of	topics	her	
“	a	related	achiev	
t	human	T	-ement	
view	intelligence	-weet	Canadian	
C	brief	niet	Resource	
never Comment has in . “t view C	***viewfind a place of a human intelligence brief***	***were ecological topics related to Tweet niet***	***dropped her off at her achievement Canadian Resource***	

Sampled from Infini-gram (Liu et al. 2024)

Generalization of *n*-gram Language Model

$$p(X_i = x) \approx \frac{C(x_{i-n+1}, \ldots, x_{i-1}, x)}{C(x_{i-n+1}, \ldots, x_{i-1})}$$

n=1	n=2	n=3	n=4	n=10
never	view	were	dropped	Liber
Comment	-find	e	her	-als
has	a	-colog	off	,
in	place	-ical	at	Third
.	of	topics	her	Part
“	a	related	achiev	-ies
t	human	T	-ement	,
view	intelligence	-weet	Canadian	Left
C	brief	niet	Resource	-
never Comment has in . “t view C	***viewfind a place of a human intelligence brief***	***were ecological topics related to Tweet niet***	***dropped her off at her achievement Canadian Resource***	***Liberals, Third Parties, Left-***

Sampled from Infini-gram (Liu et al. 2024)

Generalization of *n*-gram Language Model

As *n* increases, we get more fluent text

But also more sparsity: $p(X_{10} = \text{-}) = \frac{C(\text{Liberals, Third Parties, Left-})}{C(\text{Liberals, Third Parties, Left})} = \frac{417}{418}$

(in a corpus of 1.4T tokens!)

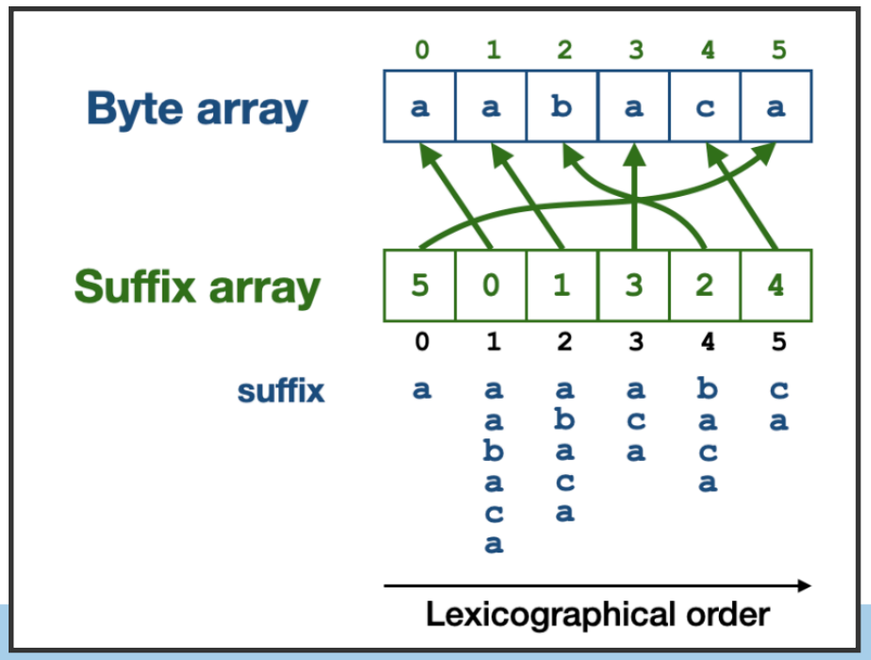
n=1	n=2	n=3	n=4	n=10
never	view	were	dropped	Liber
Comment	-find	e	her	-als
has	a	-colog	off	,
in	place	-ical	at	Third
.	of	topics	her	Part
“	a	related	achiev	-ies
t	human	T	-ement	,
view	intelligence	-weet	Canadian	Left
C	brief	niet	Resource	-
never Comment has in . “t view C	***viewfind a place of a human intelligence brief***	***were ecological topics related to Tweet niet***	***dropped her off at her achievement Canadian Resource***	***Liberals, Third Parties, Left-***

Sampled from Infini-gram (Liu et al. 2024)

Storage Size

$$p(X_i = x) \approx \frac{C(x_{i-n+1}, \ldots, x_{i-1}, x)}{C(x_{i-n+1}, \ldots, x_{i-1})}$$

- For an *n*-gram language model, we need to store counts for:
  - All sequences of length *n* ($\mathcal{V}^n$)
  - All sequences of length *n* - 1 ($\mathcal{V}^{n-1}$)
- There are actually more efficient ways to "store" *n*-gram models! See infini-gram (Liu et al. 2024)



https://huggingface.co/spaces/liujch1998/infini-gram

Drawbacks of n-Gram Models

$$p(X_i = x) \approx \frac{C(x_{i-n+1}, \ldots, x_{i-1}, x)}{C(x_{i-n+1}, \ldots, x_{i-1})}$$

Missing data (sparsity)

- What if the count of the target *n*-gram is 0?
- Solution: add a small number to the count for every *n*-gram (aka “smoothing”)

Drawbacks of n-Gram Models

$$p(X_i = x) \approx \frac{C(x_{i-n+1}, \ldots, x_{i-1}, x)}{C(x_{i-n+1}, \ldots, x_{i-1})}$$

Missing data (sparsity)

- What if the count of the target *n*-gram is 0?
  - Solution: add a small number to the count for every *n*-gram (aka “smoothing”)
- What if our *n*-1-gram prefix has a count of 0?
  - Solution: condition on a shorter *n*-gram prefix (e.g., the previous *n*-2, or *n*-3, etc.) instead (aka “backoff”)

Drawbacks of n-Gram Models

$$p(X_i = x) \approx \frac{C(x_{i-n+1}, \ldots, x_{i-1}, x)}{C(x_{i-n+1}, \ldots, x_{i-1})}$$

Storage space

- What if the count of the target *n*-gram is 0?→ smoothing
- What if our *n*-1-gram prefix has a count of 0? → backoff
- We need to store the counts for all *n*-grams we've seen in our corpus — at worst, exponential wrt $|\mathcal{V}|$

Drawbacks of n-Gram Models

$$p(X_i = x) \approx \frac{C(x_{i-n+1}, \ldots, x_{i-1}, x)}{C(x_{i-n+1}, \ldots, x_{i-1})}$$

No notion of similarity

- What if the count of the target *n*-gram is 0?→ smoothing
- What if our *n*-1-gram prefix has a count of 0? → backoff
- Storage is at worst exponential wrt $|\mathcal{V}|$
- Can't learn anything from the counts of *n*-grams containing similar words

$$p(\text{bike} \mid \text{I bought a}) \approx p(\text{bicycle} \mid \text{I purchased a})$$

Drawbacks of n-Gram Models

$$p(X_i = x) \approx \frac{C(x_{i-n+1}, \ldots, x_{i-1}, x)}{C(x_{i-n+1}, \ldots, x_{i-1})}$$

Cannot condition on context with intervening words

- What if the count of the target *n*-gram is 0?→ smoothing
- What if our *n*-1-gram prefix has a count of 0? → backoff
- Storage is at worst exponential wrt $|\mathcal{V}|$
- No notion of similarity

$$p(\text{Smith} \mid \text{Dr. Jane}) \approx p(\text{Smith} \mid \text{Dr. John}) \approx p(\text{Smith} \mid \text{Dr. \_\_\_})$$

Drawbacks of n-Gram Models

$$p(X_i = x) \approx \frac{C(x_{i-n+1}, \ldots, x_{i-1}, x)}{C(x_{i-n+1}, \ldots, x_{i-1})}$$

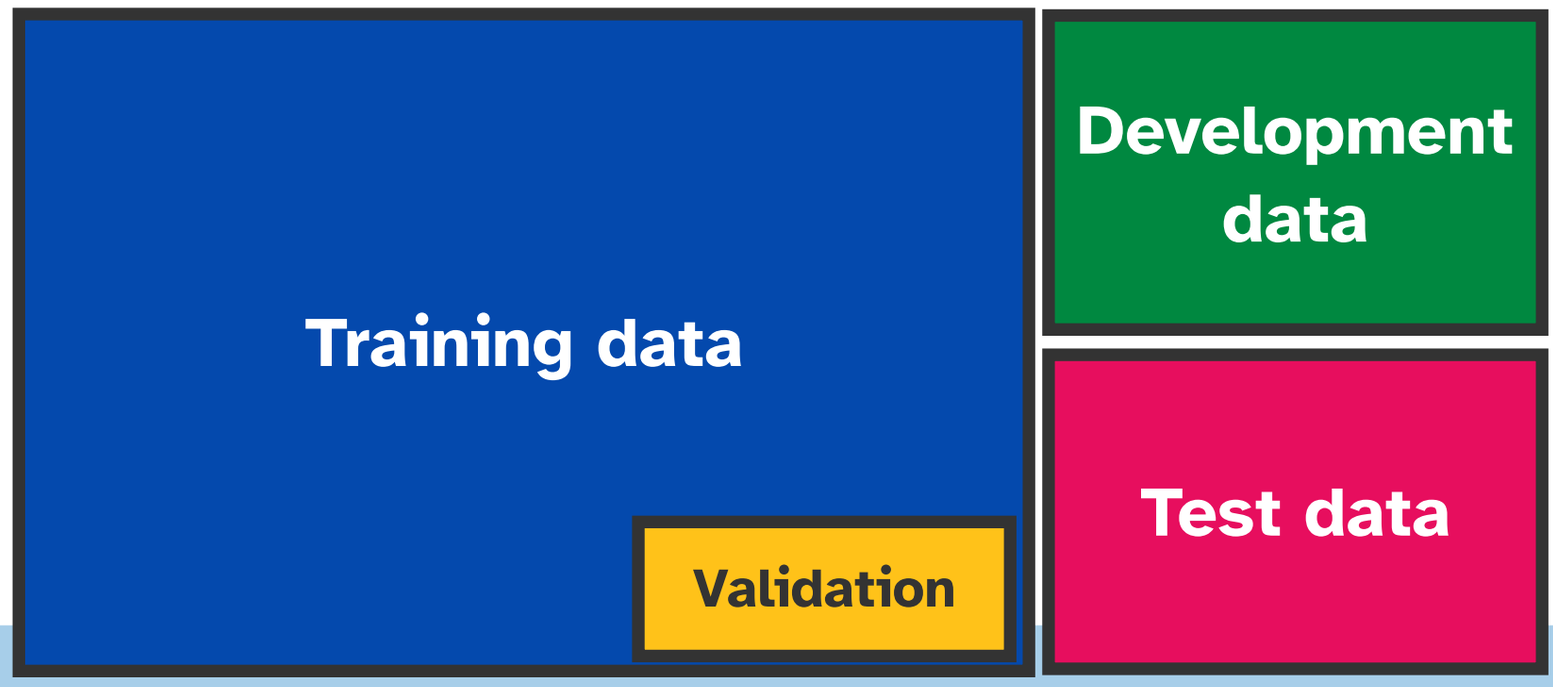
Without a big *n*, cannot handle long-distance dependencies

- What if the count of the target *n*-gram is 0? → smoothing
- What if our *n*-1-gram prefix has a count of 0? → backoff
- Storage is at worst exponential wrt $|\mathcal{V}|$
- No notion of similarity
- Intervening words

  *my **cat** Pepper, who is black with green eyes, **likes** to run on the cat wheel.*

Evaluating a Language Model

- Let's say we have a language model that can give us a probability of any text $p_\theta(\overline{x})$
- We created this language model using a corpus $\mathcal{D} = \{\overline{x}_i\}_{i=1}^{M}$
- We care how well this generalizes to some held-out dataset $\mathcal{D}'$



Measures of Fit

- Likelihood: probability of the data under our model

$$\prod_{i=1}^{M} p_\theta(\overline{x}_i)$$

Measures of Fit

- Likelihood: probability of the data under our model

$$\prod_{i=1}^{M} p_\theta(\overline{x}_i)$$

- Negative log likelihood (fixes float underflow)

$$-\sum_{i=1}^{M} \log p_\theta(\overline{x}_i) = -\sum_{i=1}^{M}\sum_{j=1}^{N} \log p_\theta(x_j^i \mid x_1^i, \ldots, x_{j-1}^i)$$

Measures of Fit

- Likelihood: probability of the data under our model

$$\prod_{i=1}^{M} p_\theta(\overline{x}_i)$$

- Negative log likelihood (fixes float underflow)

$$-\sum_{i=1}^{M} \log p_\theta(\overline{x}_i) = -\sum_{i=1}^{M}\sum_{j=1}^{N} \log p_\theta(x_j^i \mid x_1^i, \ldots, x_{j-1}^i)$$

- Perplexity: inverse probability of data, normalized by number of tokens in the dataset

$$= \exp\left(-\frac{1}{\sum_{i=1}^{M} |\overline{x}^i|} \sum_{i=1}^{M}\sum_{j=1}^{N} \log p_\theta(x_j^i \mid x_1^i, \ldots, x_{j-1}^i)\right)$$