

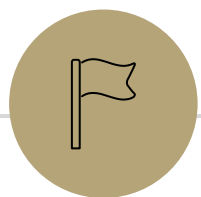
# Applications: N-Grams

Slides adapted from Yulia Tsvetkov's 25au offering of CSE 447 with many figures from Jurafsky and Martin ch. 3

CSE 312 Autumn 2025  
Lecture 29

# Announcements

- Quiz retakes are happening in your assigned section tomorrow
  - You must fill out the quiz retake form on Ed by noon today if you want to be able to retake any of the quizzes
- HW8 is due tonight!
- Our Final is next Wednesday (12/10) at 8:30 am
  - Final exam logistics and practice exams are posted under the "Exams" tab on the course website
  - There will be a Final Review Session this Sunday (12/7) on Zoom. More logistics will be posted on Ed.



# The Language Modeling Problem

# The Language Modeling problem

- Assign a probability to every sentence (or any string of words)
  - Finite vocabulary (e.g. words or characters)
  - Infinite set of sequences

# The Language Modeling problem

- Assign a probability to every sentence (or any string of words)
  - Finite vocabulary (e.g. words or characters) *{the, a, telescope, ...}*
  - Infinite set of sequences
    - *a telescope STOP*
    - *a STOP*
    - *the the the STOP*
    - *I saw a woman with a telescope STOP*
    - *STOP*
    - *...*

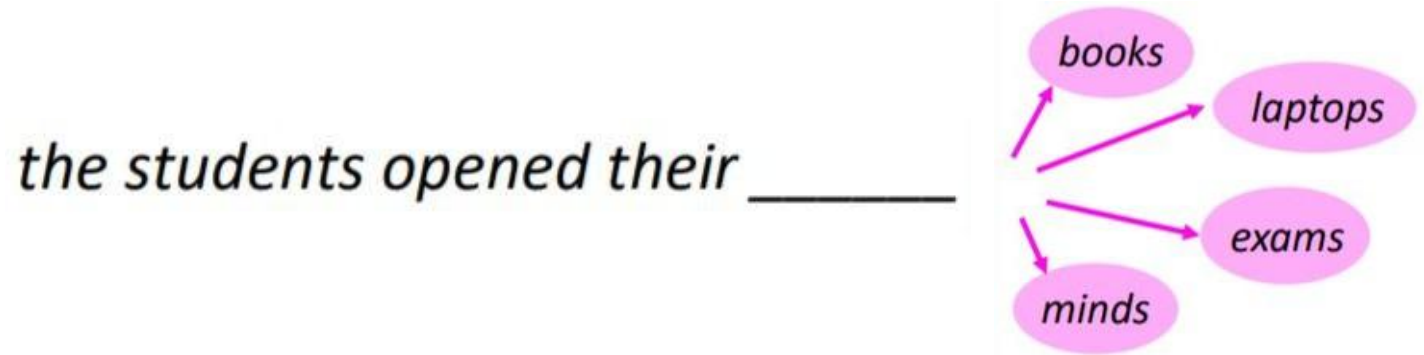
# Equivalent Definition

- **Language Modeling** is the task of predicting what word comes next

*the students opened their \_\_\_\_\_*


# Equivalent Definition

- **Language Modeling** is the task of predicting what word comes next



# Motivation: Autocorrect



what is the | 

what is the **weather**  
what is the **meaning of life**  
what is the **dark web**  
what is the **xfi**  
what is the **doomsday clock**  
what is the **weather today**  
what is the **keto diet**  
what is the **american dream**  
what is the **speed of light**  
what is the **bill of rights**



# Motivation: Text Generation



# Motivation: Text Generation



My current quest is for the Radiant Lexicon, a book that keeps being called the shiny one because that is easier for everyone's mouths to manage.

# Motivation: Text Generation



My current quest is for the Radiant Lexicon, a book that keeps being called the shiny one because that is easier for everyone's mouths to manage. Elminster once named it the sort of volume you truly should not approach, which only tilted my curiosity upward.

# Motivation: Text Generation



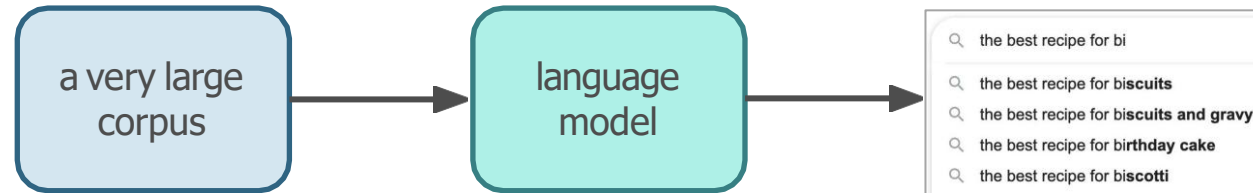
My current quest is for the Radiant Lexicon, a book that keeps being called the shiny one because that is easier for everyone's mouths to manage. Elminster once named it the sort of volume you truly should not approach, which only tilted my curiosity upward. You might wonder why I chase it, and honestly it feels more like it chases me, ever since the orb near my heart began attracting magic the way lanterns attract lost insects. The Lexicon sits in a ruin that cannot keep its hallways agreeing with themselves, shifting forward, then sideways, like the place is rehearsing its own confusion. Troublesome, yes, but so am I, and I imagine the ruin knows this already.

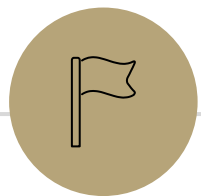
# Even More Motivation

- Machine translation
  - $p(\textit{strong winds}) > p(\textit{large winds})$
- Spelling correction
  - The office is about fifteen minuets from my house
  - $p(\textit{about fifteen minutes from}) > p(\textit{about fifteen minuets from})$
- Speech recognition
  - $p(\textit{I saw a van}) \gg p(\textit{eyes awe of an})$
- Summarization, question-answering, handwriting recognition, OCR, etc.

# Language Models

A **language model** is a machine learning model that predicts upcoming words based on the words that came before





# Formal Definition of Language Modeling

# The Language Modeling problem

- Create a probability distribution over all sequences of words
  - finite vocabulary:  $\Sigma$
  - infinite set of sequences:  $\Sigma^*$
  - Any sentence/sequence of words  $e = w_1 w_2 \dots w_n$  is an element of  $\Sigma^*$

$$\sum_{e \in \Sigma^*} P_{LM}(e) = 1$$

$$P_{LM}(e) \geq 0 \quad \forall e \in \Sigma^*$$



# The Language Modeling Problem



$$p(\textit{yield the magical artifact STOP}) = 10^{-8}$$
$$p(\textit{deliver the One Ring to Mount Doom STOP}) = 10^{-20}$$

# The Language Modeling problem

Let  $w_1 w_2 \dots w_n \in \Sigma^*$  be a sequence of words

How do we calculate  $P(w_1, w_2, \dots, w_n)$ ?

Using the chain rule:

$$\begin{aligned} P(w_1, w_2, \dots, w_n) &= P(w_1)P(w_2|w_1)P(w_3|w_2, w_1) \dots P(w_n|w_{n-1}, \dots, w_1) \\ &= \prod_{k=1}^n P(w_k|w_{1:k-1}) \end{aligned}$$

# The Language Modeling problem

Let  $w_1 w_2 \dots w_n \in \Sigma^*$  be a sequence of words

How do we calculate  $P(w_1, w_2, \dots, w_n)$ ?

Using the chain rule:

$$P(w_1, w_2, \dots, w_n) = P(w_1)P(w_2|w_1)P(w_3|w_2, w_1) \dots P(w_n|w_{n-1}, \dots, w_1)$$

$$= \prod_{k=1}^n P(w_k | w_{1:k-1})$$



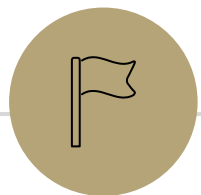
To solve the language modeling problem, all a language model needs to do is calculate the probability of the next word given the previous words (for each word in the sequence).

# Equivalent Definition

- For any sequence of words  $w_1 w_2 \dots w_{n-1} \in \Sigma^*$ , compute the probability distribution of the next word  $w_n$ , where  $w_n$  can be any word in our vocabulary  $\Sigma$ .

$$\sum_{w_n \in \Sigma} P_{LM}(w_n | w_{1:n-1}) = 1$$

$$P_{LM}(w_n | w_{1:n-1}) \geq 0 \quad \forall w_n \in \Sigma$$



# Our First Language Model

---

# Learning a Language Model

**Question:** How do we learn a Language Model?

(i.e. an algorithm or formula to produce  $P(w_1, \dots, w_n)$  or  $P(w_n | w_{1:n-1})$  for any sequence  $w_1 w_2 \dots w_n \in \Sigma^*$ )

# Our First Attempt

- Assume we have  $N$  training sentences
- Let  $w_1 w_2 \dots w_n$  be a sentence, and  $\text{count}(w_1, w_2, \dots, w_n)$  be the number of times it appeared in the training data.
- Define a language model:

$$P(w_1, \dots, w_n) = \frac{\text{count}(w_1, \dots, w_n)}{N}$$

# Our First Attempt

- Assume we have  $N$  training sentences
- Let  $w_1 w_2 \dots w_n$  be a sentence, and  $\text{count}(w_1, w_2, \dots, w_n)$  be the number of times it appeared in the training data.
- Define a language model:

$$P(w_1, \dots, w_n) = \frac{\text{count}(w_1, \dots, w_n)}{N}$$

Discuss: What are the drawbacks of this model?



# Our First Attempt

- Assume we have  $N$  training sentences
- Let  $w_1 w_2 \dots w_n$  be a sentence, and  $\text{count}(w_1, w_2, \dots, w_n)$  be the number of times it appeared in the training data.
- Define a language model:

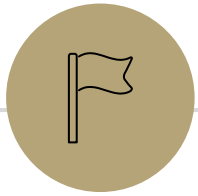
$$P(w_1, \dots, w_n) = \frac{\text{count}(w_1, \dots, w_n)}{N}$$

- No generalization!

# Our Second Attempt: N-grams

A solution that's similar but more robust than the first approach

- Question: How to learn a language model?
- Answer (pre-Deep Learning): learn an n-gram language model!



# N-gram Language Models

---

# N-grams

*“I have a dog whose name is Lucy. I have two cats, they like playing with Lucy.”*

- Definition: An n-gram is a chunk of n consecutive words.

# N-grams

*“I have a dog whose name is Lucy. I have two cats, they like playing with Lucy.”*

- **Definition:** An n-gram is a chunk of n consecutive words.
  - **unigrams:** {I, have, a, dog, whose, name, is, Lucy, two, cats, they, like, playing, with}

# N-grams

*“I have a dog whose name is Lucy. I have two cats, they like playing with Lucy.”*

- **Definition:** An n-gram is a chunk of n consecutive words.
  - **unigrams:** {I, have, a, dog, whose, name, is, Lucy, two, cats, they, like, playing, with}
  - **bigrams:** {I have, have a, a dog, dog whose, ... , with Lucy}

# N-grams

*“I have a dog whose name is Lucy. I have two cats, they like playing with Lucy.”*

- **Definition:** An n-gram is a chunk of n consecutive words.
  - **unigrams:** {I, have, a, dog, whose, name, is, Lucy, two, cats, they, like, playing, with}
  - **bigrams:** {I have, have a, a dog, dog whose, ... , with Lucy}
  - **trigrams:** {I have a, have a dog, a dog whose, ... , playing with Lucy}

# N-grams

*“I have a dog whose name is Lucy. I have two cats, they like playing with Lucy.”*

- **Definition:** An n-gram is a chunk of n consecutive words.
  - **unigrams:** {I, have, a, dog, whose, name, is, Lucy, two, cats, they, like, playing, with}
  - **bigrams:** {I have, have a, a dog, dog whose, ... , with Lucy}
  - **trigrams:** {I have a, have a dog, a dog whose, ... , playing with Lucy}
  - **four-grams:** {I have a dog, ... , like playing with Lucy}
  - ...



# N-grams

*“I have a dog whose name is Lucy. I have two cats, they like playing with Lucy.”*

- $w_1$  – a unigram
- $w_1 w_2$  – a bigram
- $w_1 w_2 w_3$  – a trigram
- $w_1 w_2 \dots w_n$  – an n-gram

# N-gram Language Models

A solution that's similar but more robust than the first approach

- **Question:** How to learn a language model?
- **Answer** (pre-Deep Learning): learn an **n-gram language model!**
- **Idea:** Collect statistics about how frequent different n-grams are and use these to predict next words.
  - For a given  $n$  and sequence  $w_1 \dots w_k$ , we approximate
$$P(w_k | w_{k-1}, \dots, w_1) \approx P(w_k | w_{k-1} \dots w_{k-n})$$
  - The relative n-gram frequencies are MLE estimates of the probabilities of n-grams appearing  $P(w_k | w_{k-n:k-1})$

# Unigram Language Model

$$P(w_k | w_{1:k-1}) \approx P(w_k)$$

$$\hat{P}(w) = \frac{\text{count}(w)}{\sum_{v \in \Sigma} \text{count}(v)}$$

# Unigram Language Model

*“I have a dog whose name is Lucy. I have two cats, they like playing with Lucy.”*

- $\sum_{w \in \Sigma} \text{count}(w) =$

$$P(w_k | w_{1:k-1}) \approx P(w_k)$$

- $P(\text{Lucy}) =$

- $P(\text{cats}) =$

$$\hat{P}(w) = \frac{\text{count}(w)}{\sum_{v \in \Sigma} \text{count}(v)}$$

# Unigram Language Model

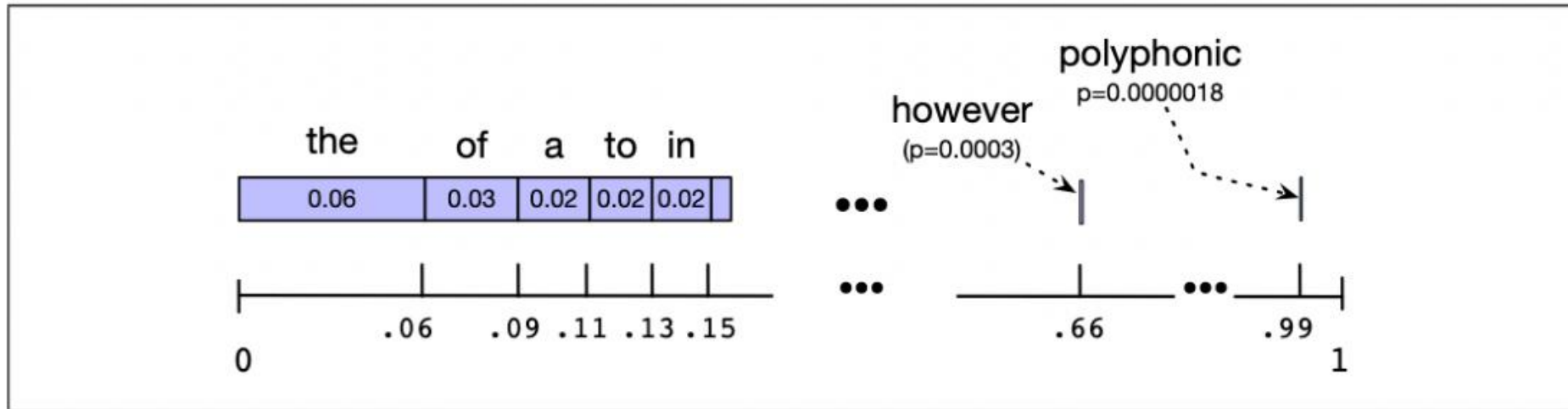
*“I have a dog whose name is Lucy. I have two cats, they like playing with Lucy.”*

- $\sum_{w \in \Sigma} \text{count}(w) = 17$
- $P(\text{Lucy}) = 2/17$
- $P(\text{cats}) = 1/17$

$$P(w_k | w_{1:k-1}) \approx P(w_k)$$

$$\hat{P}(w) = \frac{\text{count}(w)}{\sum_{v \in \Sigma} \text{count}(v)}$$

# Sampling from a unigram model



**Figure 3.3** A visualization of the sampling distribution for sampling sentences by repeatedly sampling unigrams. The blue bar represents the relative frequency of each word (we've ordered them from most frequent to least frequent, but the choice of order is arbitrary). The number line shows the cumulative probabilities. If we choose a random number between 0 and 1, it will fall in an interval corresponding to some word. The expectation for the random number to fall in the larger intervals of one of the frequent words (*the*, *of*, *a*) is much higher than in the smaller interval of one of the rare words (*polyphonic*).

# Bigram Language Model

$$P(w_k | w_{1:k-1}) \approx P(w_k | w_{k-1})$$

$$\hat{P}(w_2 | w_1) = \frac{\text{count}(w_1 w_2)}{\sum_{v \in \Sigma} \text{count}(w_1 v)} = \frac{\text{count}(w_1 w_2)}{\text{count}(w_1)}$$

We're now defining a separate conditional probability distribution for each word

# Bigram Language Model

*“I have a dog whose name is Lucy. I have two cats, they like playing with Lucy.”*

- $P(\text{have} \mid \text{I}) =$

$$P(w_k \mid w_{1:k-1}) \approx P(w_k \mid w_{k-1})$$

- $P(\text{two} \mid \text{have}) =$

$$\hat{P}(w_2 \mid w_1) = \frac{\text{count}(w_1 w_2)}{\text{count}(w_1)}$$

- $P(\text{eating} \mid \text{have}) =$



# Bigram Language Model

*“I have a dog whose name is Lucy. I have two cats, they like playing with Lucy.”*

- $P(\text{have} \mid \text{I}) = \frac{2}{2} = 1$

$$P(w_k \mid w_{1:k-1}) \approx P(w_k \mid w_{k-1})$$

- $P(\text{two} \mid \text{have}) = ?$

$$\hat{P}(w_2 \mid w_1) = \frac{\text{count}(w_1 w_2)}{\text{count}(w_1)}$$

- $P(\text{eating} \mid \text{have}) = ?$

# Bigram Language Model

*“I have a dog whose name is Lucy. I have two cats, they like playing with Lucy.”*

- $P(\text{have} \mid \text{I}) = \frac{2}{2} = 1$

$$P(w_k \mid w_{1:k-1}) \approx P(w_k \mid w_{k-1})$$

- $P(\text{two} \mid \text{have}) = \frac{1}{2} = 0.5$

$$\hat{P}(w_2 \mid w_1) = \frac{\text{count}(w_1 w_2)}{\text{count}(w_1)}$$

- $P(\text{eating} \mid \text{have}) = ?$

# Bigram Language Model

*“I have a dog whose name is Lucy. I have two cats, they like playing with Lucy.”*

- $P(\text{have} \mid \text{I}) = \frac{2}{2} = 1$

$$P(w_k \mid w_{1:k-1}) \approx P(w_k \mid w_{k-1})$$

- $P(\text{two} \mid \text{have}) = \frac{1}{2} = 0.5$

$$\hat{P}(w_2 \mid w_1) = \frac{\text{count}(w_1 w_2)}{\text{count}(w_1)}$$

- $P(\text{eating} \mid \text{have}) = \frac{0}{2} = 0$

# Trigram Language Model

$$P(w_k | w_{1:k-1}) \approx P(w_k | w_{k-2}, w_{k-1})$$

$$\hat{P}(w_3 | w_2, w_1) = \frac{\text{count}(w_1 w_2 w_3)}{\sum_{v \in \Sigma} \text{count}(w_1 w_2 v)} = \frac{\text{count}(w_1 w_2 w_3)}{\text{count}(w_1 w_2)}$$

We're now defining a separate conditional probability distribution over each possible word pair (bigrams)

# Trigram Language Model

*“I have a dog whose name is Lucy. I have two cats, they like playing with Lucy.”*

- $P(a \mid \text{I have}) = ?$
- $P(\text{several} \mid \text{I have}) = ?$

$$P(w_k \mid w_{1:k-1}) \approx P(w_k \mid w_{k-2}, w_{k-1})$$

$$\hat{P}(w_3 \mid w_2, w_1) = \frac{\text{count}(w_1 w_2 w_3)}{\text{count}(w_1 w_2)}$$

# Trigram Language Model

*“I have a dog whose name is Lucy. I have two cats, they like playing with Lucy.”*

- $P(a \mid \text{I have}) = \frac{1}{2} = 0.5$

$$P(w_k \mid w_{1:k-1}) \approx P(w_k \mid w_{k-2}, w_{k-1})$$

- $P(\text{several} \mid \text{I have}) = ?$

$$\hat{P}(w_3 \mid w_2, w_1) = \frac{\text{count}(w_1 w_2 w_3)}{\text{count}(w_1 w_2)}$$

# Trigram Language Model

*“I have a dog whose name is Lucy. I have two cats, they like playing with Lucy.”*

- $P(a \mid \text{I have}) = \frac{1}{2} = 0.5$

$$P(w_k \mid w_{1:k-1}) \approx P(w_k \mid w_{k-2}, w_{k-1})$$

- $P(\text{several} \mid \text{I have}) = \frac{0}{2} = 0$

$$\hat{P}(w_3 \mid w_2, w_1) = \frac{\text{count}(w_1 w_2 w_3)}{\text{count}(w_1 w_2)}$$

# Trigram Language Model

$$P(\text{the dog barks STOP}) =$$



# Trigram Language Model

$$P(\text{the dog barks STOP}) = P(\text{the} \mid *, *) \times$$

# Trigram Language Model

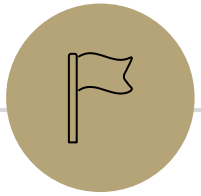
$$\begin{aligned} P(\text{the dog barks STOP}) = & P(\text{the} \mid *, *) \times \\ & P(\text{dog} \mid \text{the}, *) \times \\ & P(\text{barks} \mid \text{dog}, \text{the}) \times \\ & P(\text{STOP} \mid \text{barks}, \text{dog}) \end{aligned}$$

# General N-gram model

$$P(w_k | w_{1:k-1}) \approx P(w_k | w_{k-n:k-1})$$

$$\hat{P}(w_n | w_{n-1}, \dots, w_1) = \frac{\text{count}(w_1 w_2 \dots w_n)}{\sum_{v \in \Sigma} \text{count}(w_1 w_2 \dots w_{n-1} v)} = \frac{\text{count}(w_1 w_2 \dots w_n)}{\text{count}(w_1 w_2 \dots w_{n-1})}$$

We're now defining a separate conditional probability distribution over each possible word sequence of length  $n - 1$



## N-grams in action

---

# Berkeley restaurant project sentences

- can you tell me about any good cantonese restaurants close by
- mid priced that food is what i'm looking for
- tell me about chez pansies
- can you give me a listing of the kinds of food that are available
- i'm looking for a good place to eat breakfast
- when is caffe venezia open during the day

# Raw bigram counts

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

**Figure 3.1** Bigram counts for eight of the words (out of  $V = 1446$ ) in the Berkeley Restaurant Project corpus of 9332 sentences. Zero counts are in gray. Each cell shows the count of the column label word following the row label word. Thus the cell in row **i** and column **want** means that **want** followed **i** 827 times in the corpus.

# Bigram probabilities

$$P(w_k | w_{1:k-1}) \approx P(w_k | w_{k-1})$$

$$\hat{P}(w_2 | w_1) = \frac{\text{count}(w_1 w_2)}{\text{count}(w_1)}$$

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

**Figure 3.2** Bigram probabilities for eight words in the Berkeley Restaurant Project corpus of 9332 sentences. Zero probabilities are in gray.

# Bigram estimates of sentence probability

$$P(\langle s \rangle \text{ i want chinese food } \langle /s \rangle) = P(w_k | w_{1:k-1}) \approx P(w_k | w_{k-1})$$

$$P(\text{i} | \langle s \rangle)$$

$$\times P(\text{want} | \text{i})$$

$$\times P(\text{chinese} | \text{want})$$

$$\times P(\text{food} | \text{chinese})$$

$$\times P(\langle /s \rangle | \text{food})$$

$$= \dots$$

$$\hat{P}(w_2 | w_1) = \frac{\text{count}(w_1 w_2)}{\text{count}(w_1)}$$

	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

**Figure 3.2** Bigram probabilities for eight words in the Berkeley Restaurant Project corpus of 9332 sentences. Zero probabilities are in gray.



# Sampling from N-grams

N-gram samples (*Wall Street Journal*)

1 gram	Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
2 gram	
3 gram	

**Figure 3.5** Three sentences randomly generated from three n-gram models computed from 40 million words of the *Wall Street Journal*, lower-casing all characters and treating punctuation as words. Output was then hand-corrected for capitalization to improve readability.

# Sampling from N-grams

N-gram samples (*Wall Street Journal*)

1  
gram

Months the my and issue of year foreign new exchange's september  
were recession exchange new endorsed a acquire to six executives

2  
gram

Last December through the way to preserve the Hudson corporation N.  
B. E. C. Taylor would seem to complete the major central planners one  
point five percent of U. S. E. has already old M. X. corporation of living  
on information such as more frequently fishing to keep her

3  
gram

**Figure 3.5** Three sentences randomly generated from three n-gram models computed from 40 million words of the *Wall Street Journal*, lower-casing all characters and treating punctuation as words. Output was then hand-corrected for capitalization to improve readability.

# Sampling from N-grams

N-gram samples (*Wall Street Journal*)

1 gram	Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
2 gram	Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her
3 gram	They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

**Figure 3.5** Three sentences randomly generated from three n-gram models computed from 40 million words of the *Wall Street Journal*, lower-casing all characters and treating punctuation as words. Output was then hand-corrected for capitalization to improve readability.

# Sampling from N-grams

## N-gram samples (Shakespeare)

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	
3 gram	
4 gram	

**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.



# Sampling from N-grams

## N-gram samples (Shakespeare)

1 gram	<p>–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have</p> <p>–Hill he late speaks; or! a more to leg less first you enter</p>
2 gram	<p>–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.</p> <p>–What means, sir. I confess she? then all sorts, he is trim, captain.</p>
3 gram	
4 gram	

**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.

# Sampling from N-grams

## N-gram samples (Shakespeare)

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

**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.

# Sampling from N-grams

## N-gram samples (Shakespeare)

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.

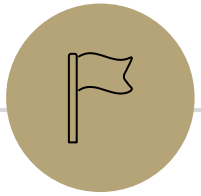
3  
gram

–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.

**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.



# Laplace Smoothing

---



$$P(w_k | w_{1:k-1}) \approx P(w_k | w_{k-1})$$

$$\hat{P}(w_2 | w_1) = \frac{\text{count}(w_1 w_2)}{\text{count}(w_1)}$$

# Bigram Counts

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

**Figure 3.1** Bigram counts for eight of the words (out of  $V = 1446$ ) in the Berkeley Restaurant Project corpus of 9332 sentences. Zero counts are in gray. Each cell shows the count of the column label word following the row label word. Thus the cell in row **i** and column **want** means that **want** followed **i** 827 times in the corpus.

# Bigram Counts with Laplace Smoothing

Add-one (Laplace) smoothed bigram counts

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

**Figure 3.6** Add-one smoothed bigram counts for eight of the words (out of  $V = 1446$ ) in the Berkeley Restaurant Project corpus of 9332 sentences. Previously-zero counts are in gray.

# N-gram model with Laplace Smoothing

$$P(w_k | w_{1:k-1}) \approx P(w_k | w_{k-n:k-1})$$

$$\hat{P}(w_n | w_{n-1}, \dots, w_1) = \frac{\text{count}(w_1 w_2 \dots w_n)}{\sum_{v \in \Sigma} \text{count}(w_1 w_2 \dots w_{n-1} v)} = \frac{\text{count}(w_1 w_2 \dots w_n)}{\text{count}(w_1 w_2 \dots w_{n-1})}$$

$$\hat{P}_{Laplace}(w_n | w_{n-1}, \dots, w_1) = \frac{1 + \text{count}(w_1 w_2 \dots w_n)}{|\Sigma| + \sum_{v \in \Sigma} \text{count}(w_1 w_2 \dots w_{n-1} v)} = \frac{1 + \text{count}(w_1 w_2 \dots w_n)}{|\Sigma| + \text{count}(w_1 w_2 \dots w_{n-1})}$$

# That's all folks!

TODOs:

- Fill out the quiz retake form ASAP if you haven't already
- Finish HW8
- Study for the final!