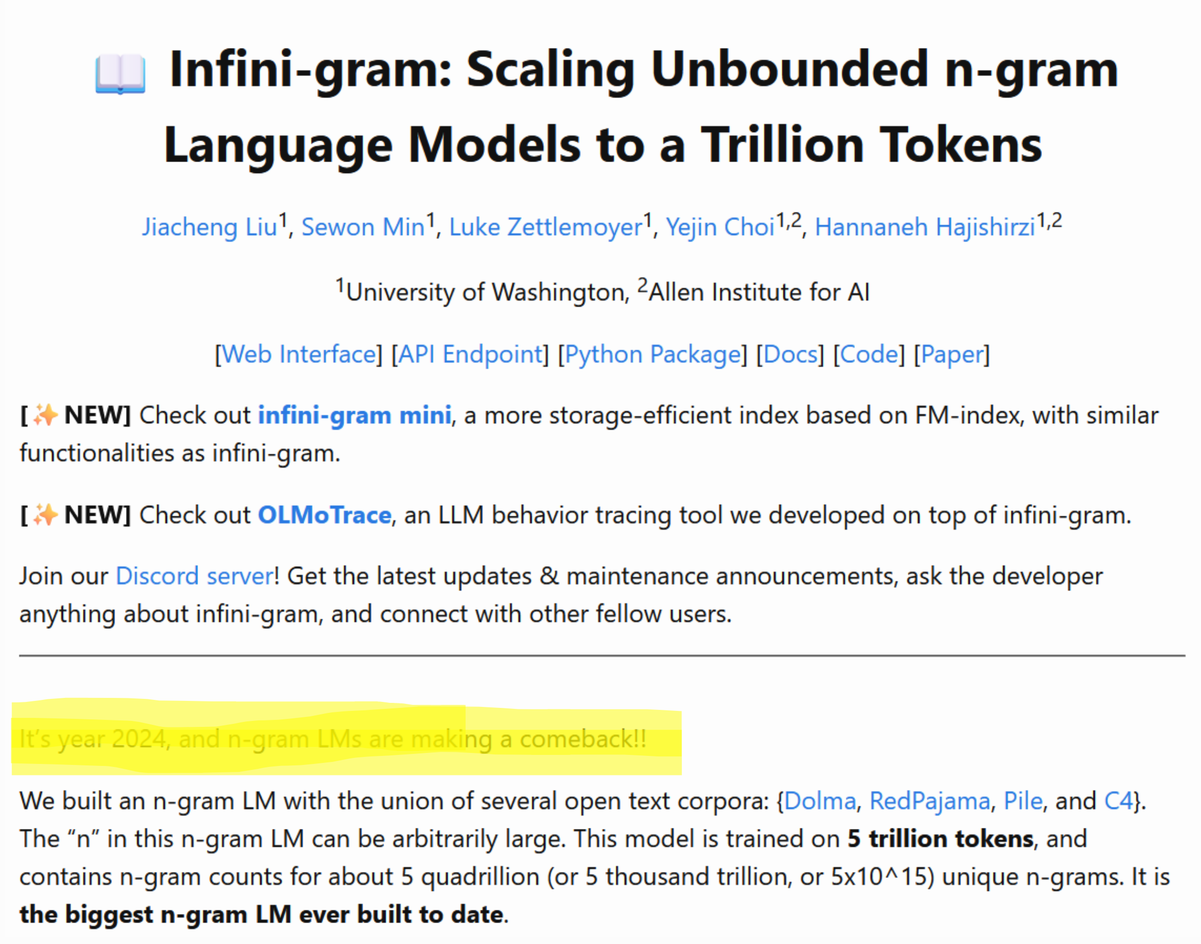


# 📖 Infini-gram: Scaling Unbounded n-gram Language Models to a Trillion Tokens

Jiacheng Liu[1], Sewon Min[1], Luke Zettlemoyer[1], Yejin Choi[1,2], Hannaneh Hajishirzi[1,2]

[1]University of Washington, [2]Allen Institute for AI

[Web Interface] [API Endpoint] [Python Package] [Docs] [Code] [Paper]

[✨ **NEW]** Check out **infini-gram mini**, a more storage-efficient index based on FM-index, with similar functionalities as infini-gram.

[✨ **NEW]** Check out **OLMoTrace**, an LLM behavior tracing tool we developed on top of infini-gram.

Join our Discord server! Get the latest updates & maintenance announcements, ask the developer anything about infini-gram, and connect with other fellow users.

---

It's year 2024, and n-gram LMs are making a comeback!!

We built an n-gram LM with the union of several open text corpora: {Dolma, RedPajama, Pile, and C4}. The "n" in this n-gram LM can be arbitrarily large. This model is trained on **5 trillion tokens**, and contains n-gram counts for about 5 quadrillion (or 5 thousand trillion, or 5x10^15) unique n-grams. It is **the biggest n-gram LM ever built to date**.

BABE, MY LIFE IS A MARKOV CHAIN
GIVEN THE PRESENT, THE FUTURE DOES NOT DEPEND ON THE PAST
quickmeme.com

# CS 1671 / CS 2071 / ISSP 2071
# Human Language Technologies

## Session 7: N-gram language models

Michael Miller Yoder

February 4, 2026

University of Pittsburgh | School of Computing and Information

## Course logistics: quiz

- Next in-class quiz is next class session, **Mon Feb 9**
  - Session 6: J+M 5.3-5.4, **11.1.1**
  - Session 7 (today): J+M 3-3.6.2, 3.8
- Conceptual, not programming
- Lowest quiz score in the course will be dropped
- If you won't be in class, let me know and I can accommodate

## Course logistics: project

- Project idea form to submit project ideas is **due tomorrow, Thu Feb 5**
- Take a look at the example projects on the project website. You can submit one or more of those for the form, or submit your own idea!
- Have a potential project idea that involves deriving insight from a dataset of text, or building an NLP system that can do something with text? You can submit it!
  - Ideas do not need to be well-formed
  - Ideas that have data already available are more realistic
- You will later choose from an **anonymized** list of project ideas on Project Match Day, Feb 11

## Course logistics: homework

- Homework 1 has been released. Is **due next Thu Feb 12 at 11:59pm**
- Homework assignments are programming-based
- Homework 1 covers text processing and regular expressions in Python

## Course logistics: Discord server

- I’ve created a Discord server for the class for in-class questions and discussion of assessments (homework, projects, etc)
- Invite link: https://discord.gg/AbVVBm9C
    - If it has expired, reach out to Michael
- **Please change your server nickname to match your full name as it appears on Canvas (including your first and last name)**
    - To do this, right click on the server icon in the server list, then click "Edit Per-server Profile". Then edit the "Server Nickname" field.

## Overview: N-gram language models

- Coding activity from last time: clickbait n-gram document representations
- Language modeling
- N-gram language models
- Sampling sentences from n-gram language models
- Estimating n-gram probabilities
- Perplexity and evaluating language models
- Handling zeros in n-gram language models
- Coding activity: build your own n-gram language model!

## Coding activity:
## clickbait n-gram document representations

## N-gram document representations on JupyterHub

1. Go to this nbgitpuller link (also available on course website)
2. Log in with your Pitt username if necessary
3. Start a server with **TEACH – 6 CPUs, 48 GB**
4. Load custom environment at **/ix1/cs1671-2026s/class_env**
    1. If you have multiple accounts on the CRCD, put in **cs1671-2026s** for Account
5. This should pull the cs1671_spring2026_jupyterhub folder into your JupyterLab
6. Open **session6_clickbait_ngrams.ipynb**

# Structure of this course

| | | | |
|---|---|---|---|
| MODULE 1 | Prerequisite skills for NLP | text normalization, linear alg., prob., machine learning | |
| | Approaches | How text is represented | NLP tasks |
| MODULE 2 | statistical machine learning | n-grams | language modeling<br>text classification |
| MODULE 3 | | | |
| MODULE 4 | | | language modeling<br>text classification<br>sequence labeling |
| MODULE 5 | NLP applications and ethics | machine translation, chatbots, information retrieval, bias | |

## Introduction to language models

## Language Models Estimate the Probability of Sequences

Which of these sentences would you be more likely to observe in an English corpus?

- Hugged I big brother my.
- I hugged my large brother.
- I hugged my big brother.

*Slide credit: David Mortensen*

## Language Models Estimate the Probability of Sequences

Which of following word would be most likely to come after "David hates visiting New..."

- York
- California
- giggled

*Slide credit: David Mortensen*

These are actually instances of the same problem: the language modeling problem!

Slide credit: David Mortensen

## Language Modeling is Tremendously Useful

LMs (language models) are at the center of NLP today and have many different applications

- Machine Translation
  P(high winds tonight) > P(large winds tonight)
- Spelling Correction
  P(about fifteen **minutes** from) > P(about fifteen **minuets** from)
- Text Input Methods
  P(i cant believe how hot you **are**) > P(i cant believe how hot you **art**)
- Speech Recognition
  P(recognize speech) > P(wreck a nice beach)

*Slide credit: David Mortensen*

## The Goal of Language Modeling

Compute the probability of a sequence of words/tokens/characters:

$$P(\mathbf{w}) = P(w_1, w_2, w_3, w_5, \ldots, w_n)$$

$$P(\text{I, hugged, my, big, brother})$$

This is related to next-word prediction:

$$P(w_t | w_1 w_2 \ldots w_{t-1})$$

$$P(\text{York} | \text{David, hates, going, to, New})$$

Do you compute either of these? Then you're in luck:

**You are a language model!**

*Slide credit: David Mortensen*

## N-gram language models

## The Chain Rule Helps Us Compute Joint Probabilities

The definition of conditional probability is

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

which can be rewritten as

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

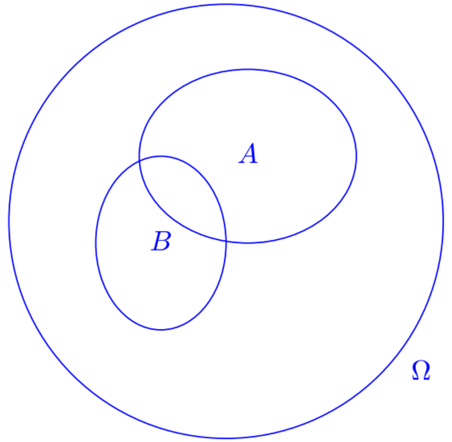


Figure 4.1: Events on the dart board

*Slide credit: David Mortensen*

## The Chain Rule Helps Us Compute Joint Probabilities

If we add more variables, we see the following pattern:

$$\begin{aligned} P(A, B, C) &= P(A)P(B|A)P(C|A, B) \\ P(A, B, C, D) &= P(A)P(B|A)P(C|A, B)P(D|A, B, C) \end{aligned}$$

which can be generalized as

$$P(x_1, x_2, x_3, \ldots, x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1, x_2)\ldots P(x_n|x_1, \ldots, x_{n-1})$$

**The Chain Rule!**

*Slide credit: David Mortensen*

## The chain rule to compute the joint probability of words in a sentence

$$P(w_1, w_2, w_3, \ldots, w_n) = \prod_i^n P(w_i | w_1 w_2 \ldots w_{i-1})$$

$$\begin{aligned} &P(\text{now is the winter of our discontent}) = \\ &P(\text{now}) \times P(\text{is}|\text{now}) \times \\ &P(\text{the}|\text{now is}) \times P(\text{winter}|\text{now is the}) \times \\ &P(\text{of}|\text{now is the winter}) \times \\ &P(\text{our}|\text{now is the winter of}) \times \\ &P(\text{discontent}|\text{now is the winter of our}) \end{aligned}$$

*Slide credit: David Mortensen*

## How Are We Estimating these Probabilities?

Could we just count and divide?

$$P(\text{discontent}|\text{now is the winter of our}) = \frac{\text{Count(now is the winter of our discontent)}}{\text{Count(now is the winter of our)}}$$

But this can’t be a valid estimate! “now is the winter of our” is going to very rare in corpora. It isn’t going to be a good estimate of its true probability.

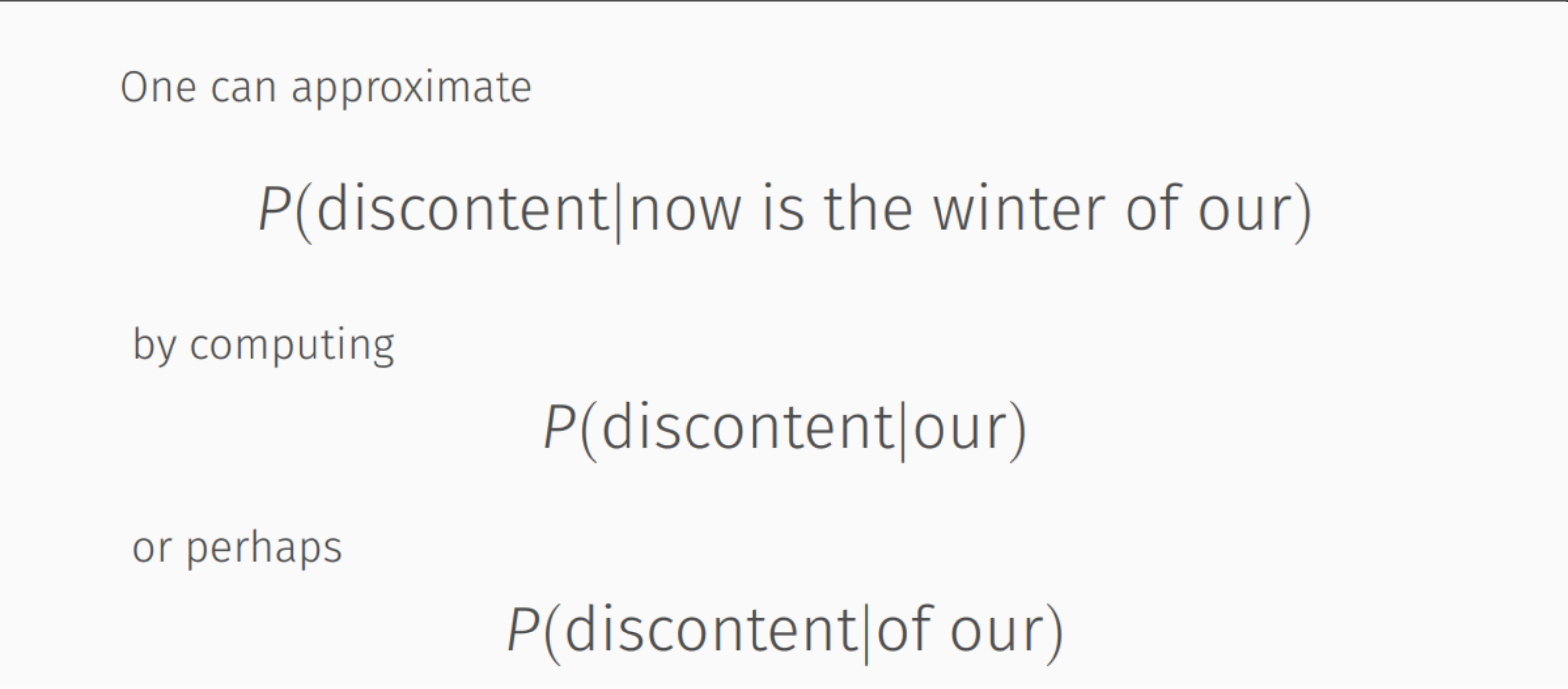
*Slide credit: David Mortensen*

## This May not Seem Very Helpful

Is $P(\text{discontent}|\text{now is the winter of our})$ really easier to compute than $P(\text{now is the winter of our discontent})$?

How can the chain rule help us? We can **cheat**.

## Markov Showed that You Could Make a Simplifying Assumption



One can approximate

$$P(\text{discontent}|\text{now is the winter of our})$$

by computing

$$P(\text{discontent}|\text{our})$$

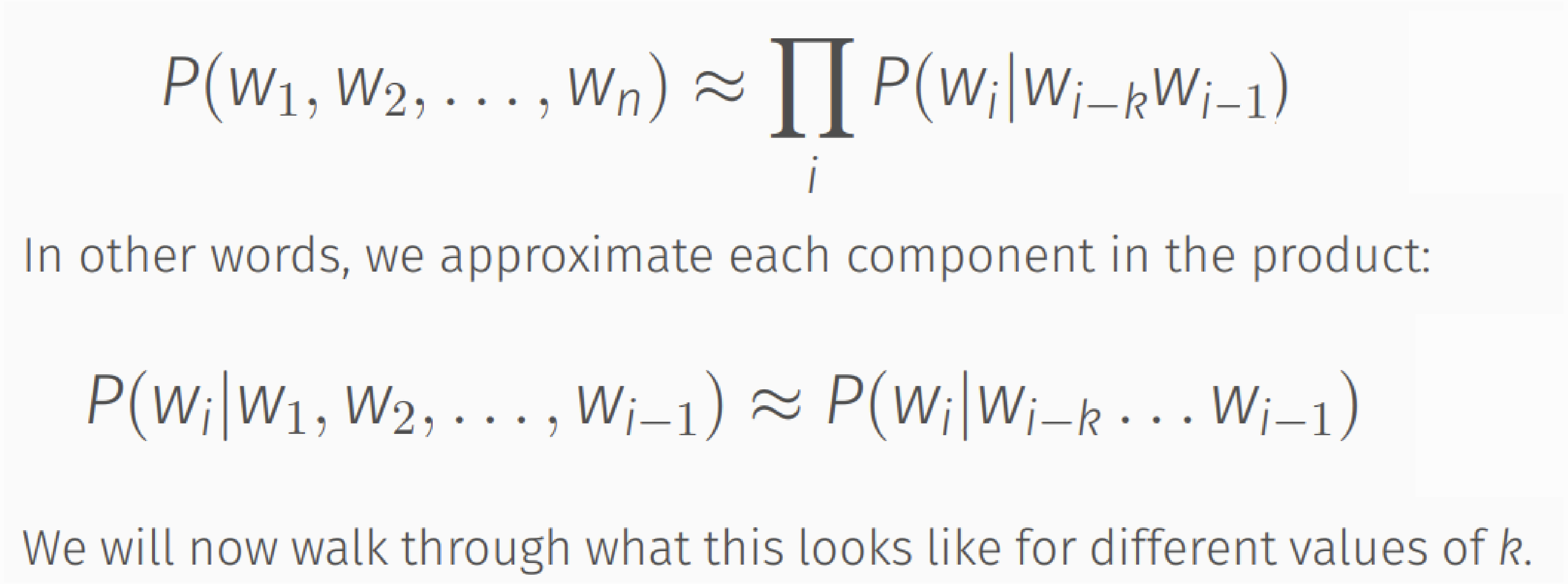
or perhaps

$$P(\text{discontent}|\text{of our})$$

- We only get an estimate this way, but we can obtain it by only counting simpler things: “our discontent”, “discontent”, “of our”, etc
- N-gram language modeling is a generalization of this observation

*Slide credit: David Mortensen*

## This assumption is the Markov assumption



$$P(w_1, w_2, \ldots, w_n) \approx \prod_i P(w_i | w_{i-k} w_{i-1})$$

In other words, we approximate each component in the product:

$$P(w_i | w_1, w_2, \ldots, w_{i-1}) \approx P(w_i | w_{i-k} \ldots w_{i-1})$$

We will now walk through what this looks like for different values of $k$.

*Slide credit: David Mortensen*

## Sampling sentences from language models

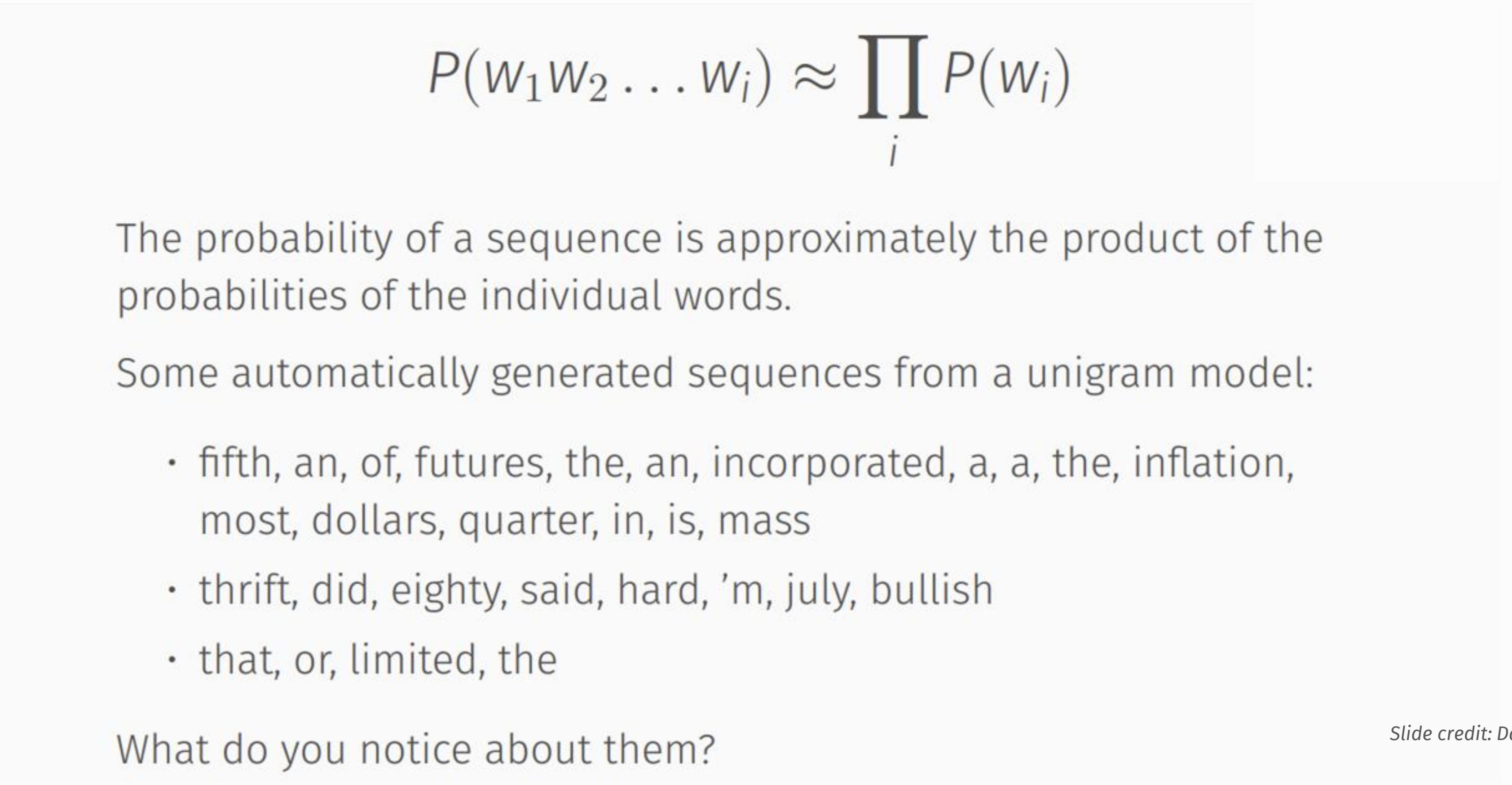
# The Shannon Visualization Method

- Choose a random bigram (<s>, w) according to its probability
- Now choose a random bigram (w, x) according to its probability
- And so on until we choose </s>
- Then string the words together

```
<s> I
    I want
      want to
           to eat
              eat Chinese
                  Chinese food
                          food  </s>
I want to eat Chinese food
```

*Slide adapted from Jurafsky & Martin*

# Unigram model



$$P(w_1 w_2 \ldots w_i) \approx \prod_i P(w_i)$$

The probability of a sequence is approximately the product of the probabilities of the individual words.

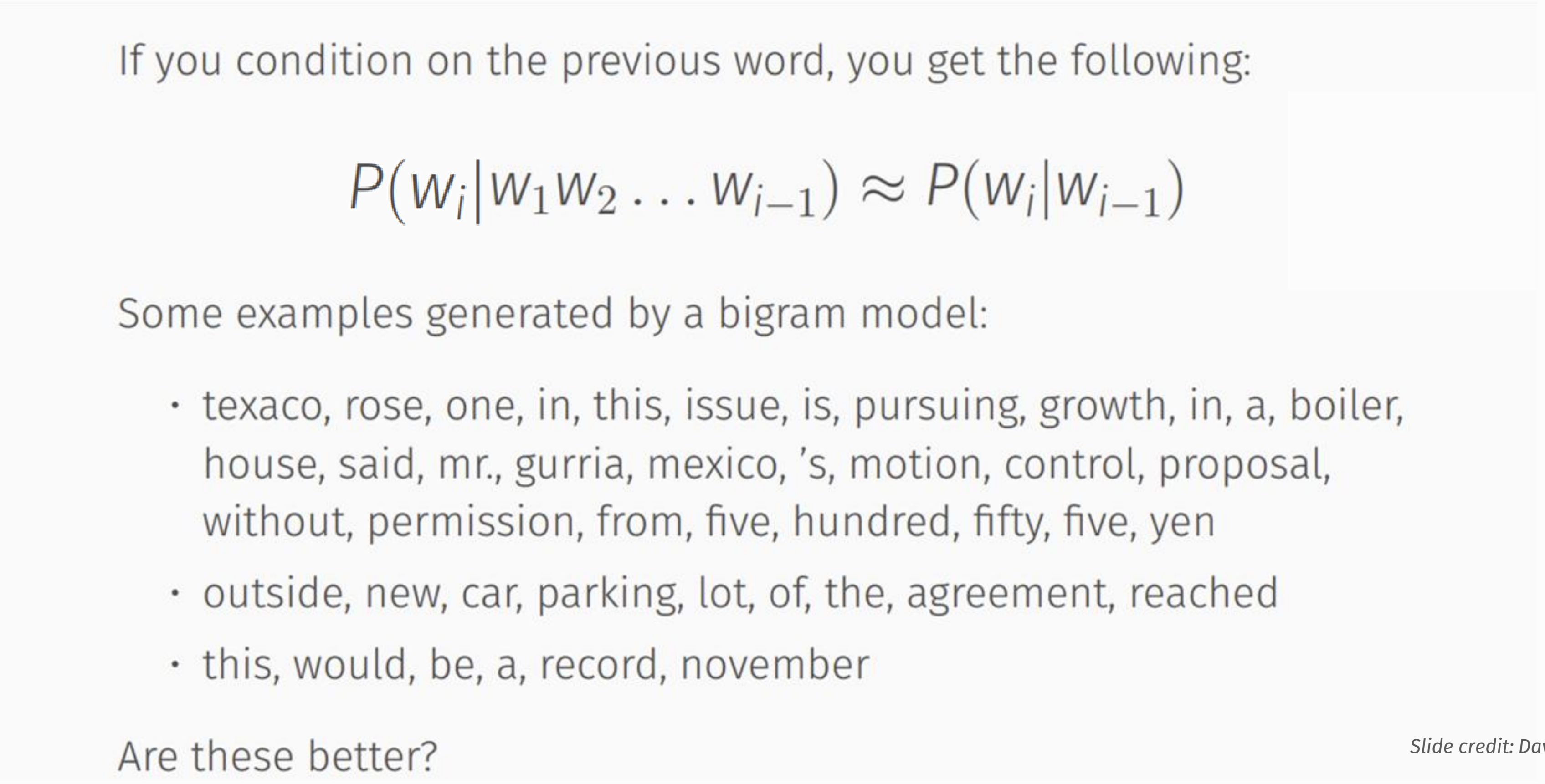
Some automatically generated sequences from a unigram model:

- fifth, an, of, futures, the, an, incorporated, a, a, the, inflation, most, dollars, quarter, in, is, mass
- thrift, did, eighty, said, hard, 'm, july, bullish
- that, or, limited, the

What do you notice about them?

*Slide credit: David Mortensen*

# Bigram model



If you condition on the previous word, you get the following:

$$P(w_i | w_1 w_2 \ldots w_{i-1}) \approx P(w_i | w_{i-1})$$

Some examples generated by a bigram model:

- texaco, rose, one, in, this, issue, is, pursuing, growth, in, a, boiler, house, said, mr., gurria, mexico, 's, motion, control, proposal, without, permission, from, five, hundred, fifty, five, yen
- outside, new, car, parking, lot, of, the, agreement, reached
- this, would, be, a, record, november

Are these better?

*Slide credit: David Mortensen*

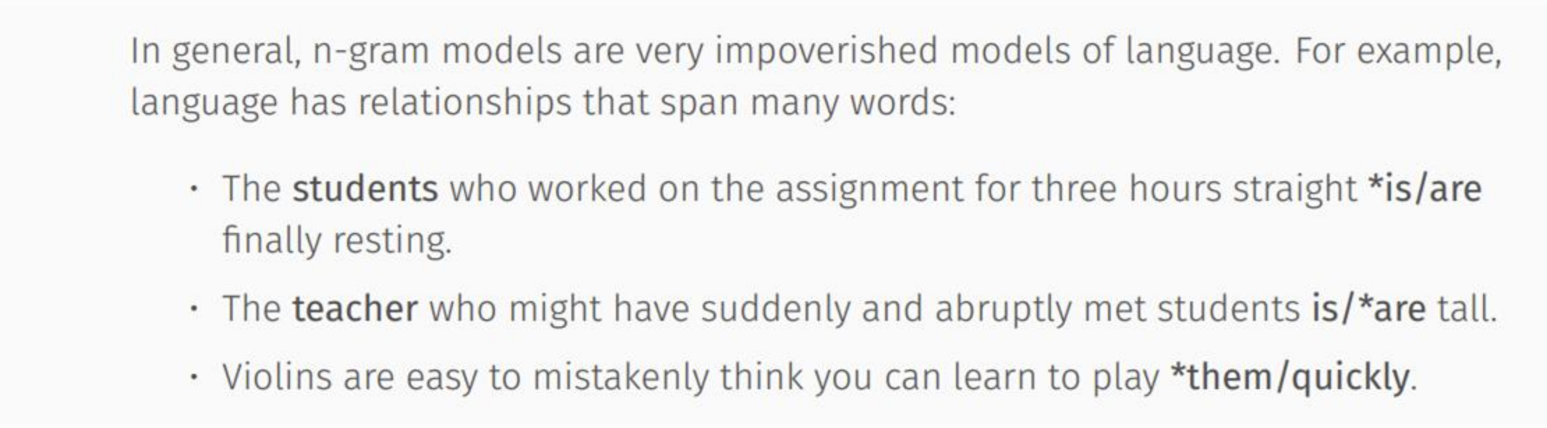
## The Trigram Model

The trigram model is just like the bigram model, only with a larger *k*:

$$P(w_i|w_1 w_2 \ldots w_{i-1}) \approx P(w_i|w_{i-2} w_{i-1})$$

The output of a trigram language model is generally **much** better than that of a bigram model **provided the training corpus is large enough**. Why do you need a larger corpus to train a trigram corpus than a bigram or unigram corpus?

## N-gram models have trouble with long-range dependencies



In general, n-gram models are very impoverished models of language. For example, language has relationships that span many words:

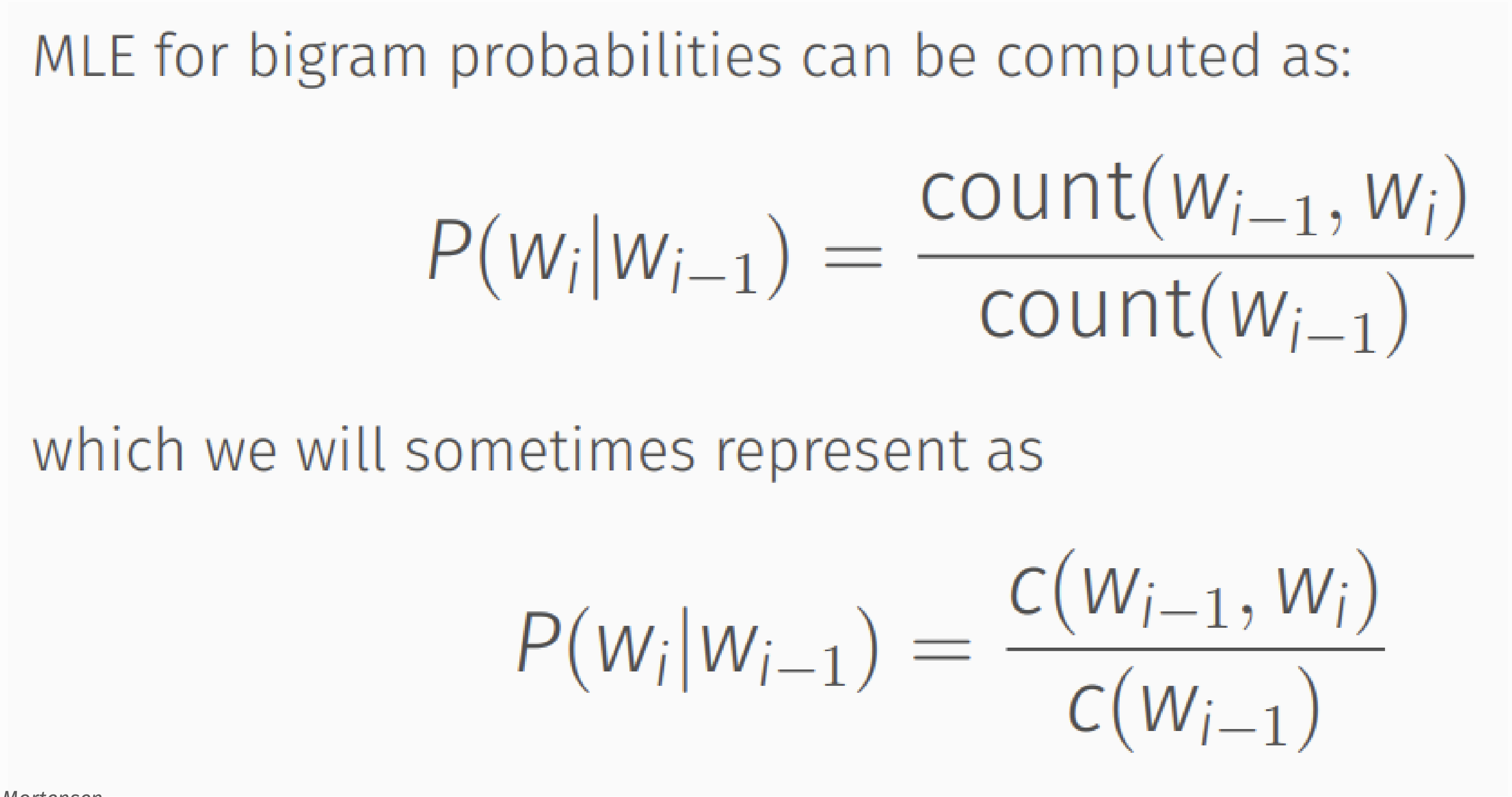
- The **students** who worked on the assignment for three hours straight ***is/are** finally resting.
- The **teacher** who might have suddenly and abruptly met students **is/*are** tall.
- Violins are easy to mistakenly think you can learn to play ***them/quickly**.

## Ngram LMs Are Often Adequate

**Nevertheless, for many applications, ngram models are good enough** (and they're super fast and efficient)

## Estimating n-gram probabilities

## Estimating bigram probabilities with the maximum likelihood estimate (MLE)



MLE for bigram probabilities can be computed as:

$$P(w_i|w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}$$

which we will sometimes represent as

$$P(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

*Slide credit: David Mortensen*

## An example

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

<s> I am Sam </s>

<s> Sam I am </s>

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

$P(\texttt{I} \mid \texttt{<s>}) =$ $P(\texttt{Sam} \mid \texttt{<s>}) =$ $P(\texttt{am} \mid \texttt{I}) =$

$P(\texttt{</s>} \mid \texttt{Sam}) =$ $P(\texttt{Sam} \mid \texttt{am}) =$ $P(\texttt{do} \mid \texttt{I}) =$

## More examples: Berkeley Restaurant Project sentences

can you tell me about any good cantonese restaurants close by

mid priced thai food is what i’m looking for

tell me about chez panisse

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

## Bigram estimates of sentence probabilities

From a corpus, you could estimate probabilities of bigrams and then calculate probabilities of new sentences:

P(<s> I want english food </s>) =

P(I|<s>)

× P(want|I)

× P(english|want)

× P(food|english)

× P(</s>|food)

= .000031

*Slide adapted from Jurafsky & Martin*

## Multiplication Considered Harmful

Doing computation in log space is preferred for language models

- **Avoid underflow** Multiplying small probabilities by small probabilities results in *very small* numbers, which is problematic
- **Optimize computation** Addition is cheaper than multiplication

$$\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

*Slide adapted from David Mortensen*

## Perplexity and evaluating language models

## The Evaluation Process for ML Models

**The goal of LM evaluation:**

- Does our model prefer good sentences to bad sentences?
- Specifically, does it assign higher probabilities to the good/grammatical/frequently observed ones and lower probabilities to the bad/ungrammatical/seldom observed ones?

**In ML evaluation, we divide our data into three sets: `train`, `dev`, and `test`.**

- We train the model's parameters on the `train` set
- We tune the model's hyperparameters (if appropriate) on the `dev` set (which should not overlap with the `train` set
- We test the model on the `test` set, which should not overlap with `train` or `dev`

An evaluation metric tells us how well our model has done on `test`.

*Slide credit: David Mortensen*

## We Can Evaluate Models Intrinsically or Extrinsically

- **Extrinsic Evaluation** means asking how much the model contributes to a larger task or goal. We may evaluate an LM based on how much it improves machine translation over a BASELINE.
- **Intrinsic Evaluation** means measuring some property of the model directly. We may quantify the probability that an LM assigns to a corpus of text.

In general, EXTRINSIC EVALUATION is better, but more expensive and time-consuming.

*Slide credit: David Mortensen*

## Extrinsic Evaluation of LMs

**Best evaluation for comparing models A and B**

- Put each model in a task (spelling corrector, speech recognizer, MT system)
- Run the task, get an accuracy for A and for B
  - How many misspelled words corrected properly?
  - How many sentences translated correctly?
- Compare scores for A and B

**This takes a lot of time to set up and can be expensive to carry out.**

*Slide credit: David Mortensen*

# Perplexity is an intrinsic metric for language modeling

Perplexity evaluates the probability assigned by a model **to a collection of test documents, controlling for length** and is, thus, useful for evaluating LMs.

A better model of a text is one which assigns a higher probability to words that actually occur in the test set. Better models result in lower perplexities.

However:

- It is a rather crude instrument
- It sometimes correlates only weakly with performance on downstream tasks
- It's only useful for pilot experiments
- But it's cheap and easy to compute, so it's important to understand

*Slide credit: David Mortensen*

## Lower perplexity = better model

Training 38 million words, test 1.5 million words, WSJ

| N-gram Order | Unigram | Bigram | Trigram |
|---|---|---|---|
| Perplexity | 962 | 170 | 109 |

*Slide adapted from Jurafsky & Martin*

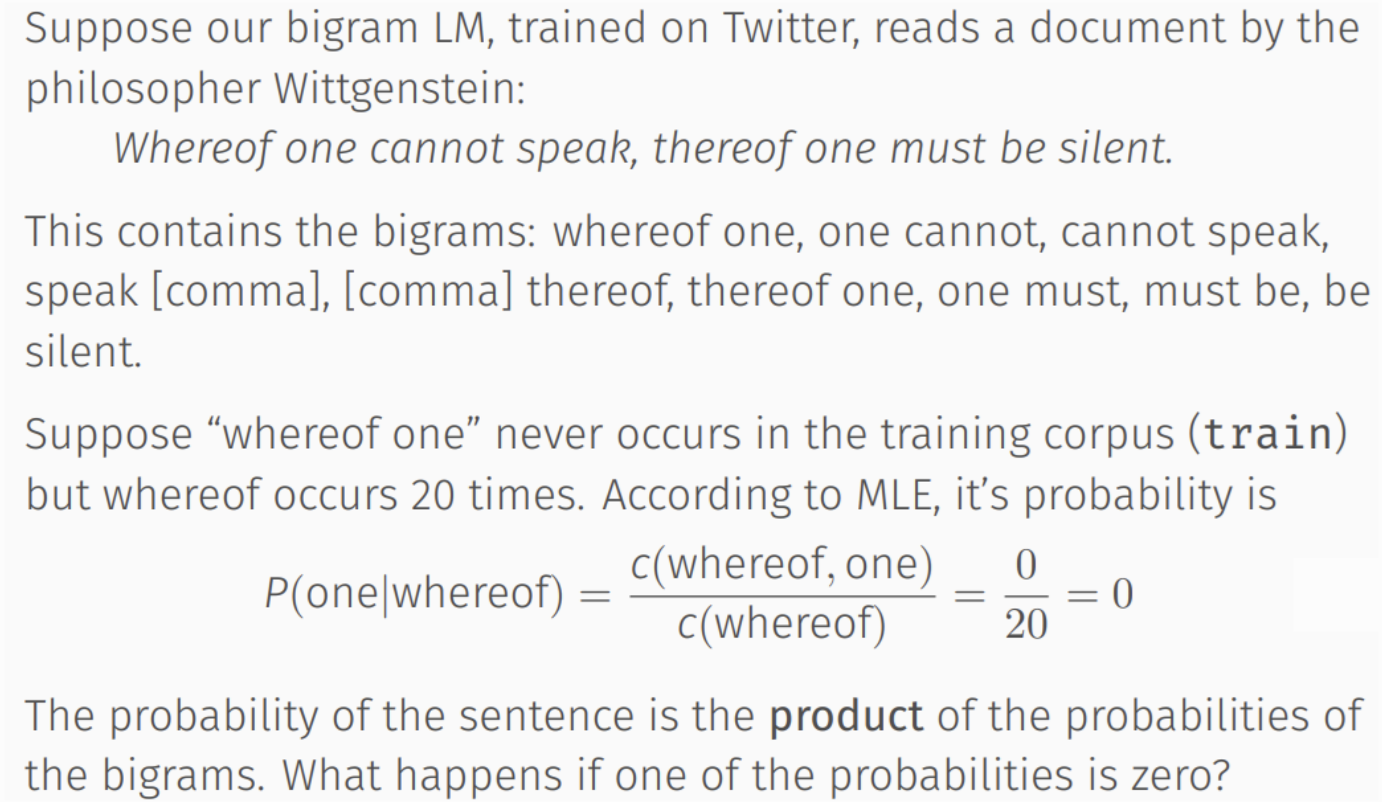
## The problem of zeros in n-gram language models

## The Perils of Overfitting

N-grams only work well for word prediction if the test corpus looks like the training corpus

- In real life, it often doesn't
- We need to train robust models that generalize!
    - One kind of generalization: Zeros!
    - Things that don't ever occur in the training set but occur in the test set

## N-grams in the test set that weren’t in the training set



Suppose our bigram LM, trained on Twitter, reads a document by the philosopher Wittgenstein:

*Whereof one cannot speak, thereof one must be silent.*

This contains the bigrams: whereof one, one cannot, cannot speak, speak [comma], [comma] thereof, thereof one, one must, must be, be silent.

Suppose “whereof one” never occurs in the training corpus (`train`) but whereof occurs 20 times. According to MLE, it’s probability is

$$P(\text{one}|\text{whereof}) = \frac{c(\text{whereof}, \text{one})}{c(\text{whereof})} = \frac{0}{20} = 0$$

The probability of the sentence is the **product** of the probabilities of the bigrams. What happens if one of the probabilities is zero?

## Strategies for handling zeros in n-gram LMs

- Laplace and Lidstone smoothing: simply add a small pseudocount to all possible n-grams in the vocabulary so none are 0
- There are more advanced methods that work better in practice, including interpolation and backoff

## Coding activity: build your own n-gram LM

## N-gram document representations on JupyterHub

1. Go to this nbgitpuller link (also available on course website)
2. Start a server with **TEACH – 6 CPUs, 48 GB**
3. Load custom environment at **/ix1/cs1671-2026s/class_env**
    1. If you have multiple accounts on the CRCD, put in **cs1671-2026s** for Account
4. This should pull the cs1671_spring2026_jupyterhub folder into your JupyterLab
5. Open **session7_ngram_lm.ipynb**

## *Questions?*