Lecture 2:
n-gram Language Models

Instructor: Swabha Swayamdipta
USC CSCI 544 Applied NLP
Aug 29, Fall 2024

Some slides adapted from Dan Jurafsky and Chris Manning

Announcements

+

Recap

Logistics and Announcements

Logistics and Announcements

- Syllabus changes (see website) based on requests
    - e.g. Quiz 4 date changed to accommodate Grace Hopper Conference attendance
    - Project Dates have changed to give you more time for the status report and presentations
    - Add / drop dates for class: Sep 6 and project team formation deadline: Sep 10

Logistics and Announcements

- Syllabus changes (see website) based on requests
    - e.g. Quiz 4 date changed to accommodate Grace Hopper Conference attendance
    - Project Dates have changed to give you more time for the status report and presentations
    - Add / drop dates for class: Sep 6 and project team formation deadline: Sep 10
- Class Project
    - Forming Groups: Watch out for Brightspace Announcement
        - Change: Will allow some groups of 6, but higher expectations from these groups
        - Will only accommodate a **maximum** of 52 teams
    - CARC access - We are working on it!

Logistics and Announcements

- Syllabus changes (see website) based on requests
  - e.g. Quiz 4 date changed to accommodate Grace Hopper Conference attendance
  - Project Dates have changed to give you more time for the status report and presentations
  - Add / drop dates for class: Sep 6 and project team formation deadline: Sep 10
- Class Project
  - Forming Groups: Watch out for Brightspace Announcement
    - Change: Will allow some groups of 6, but higher expectations from these groups
    - Will only accommodate a **maximum** of 52 teams
  - CARC access - We are working on it!
- Next Week:
  - Tue: HW1 released
  - Thu: Quiz 1 (Bring your laptop!)

Logistics and Announcements

- Syllabus changes (see website) based on requests
  - e.g. Quiz 4 date changed to accommodate Grace Hopper Conference attendance
  - Project Dates have changed to give you more time for the status report and presentations
  - Add / drop dates for class: Sep 6 and project team formation deadline: Sep 10
- Class Project
  - Forming Groups: Watch out for Brightspace Announcement
    - Change: Will allow some groups of 6, but higher expectations from these groups
    - Will only accommodate a **maximum** of 52 teams
  - CARC access - We are working on it!
- Next Week:
  - Tue: HW1 released
  - Thu: Quiz 1 (Bring your laptop!)
- Brightspace Discussions: Start a new thread under Activities / Discussions / Forums / Topics (e.g. Group Creation)

Logistics and Announcements

- Syllabus changes (see website) based on requests
  - e.g. Quiz 4 date changed to accommodate Grace Hopper Conference attendance
  - Project Dates have changed to give you more time for the status report and presentations
  - Add / drop dates for class: Sep 6 and project team formation deadline: Sep 10
- Class Project
  - Forming Groups: Watch out for Brightspace Announcement
    - Change: Will allow some groups of 6, but higher expectations from these groups
    - Will only accommodate a **maximum** of 52 teams
  - CARC access - We are working on it!
- Next Week:
  - Tue: HW1 released
  - Thu: Quiz 1 (Bring your laptop!)
- Brightspace Discussions: Start a new thread under Activities / Discussions / Forums / Topics (e.g. Group Creation)
- Missing Class? Report in advance using the form (pinned to Brightspace Announcements)

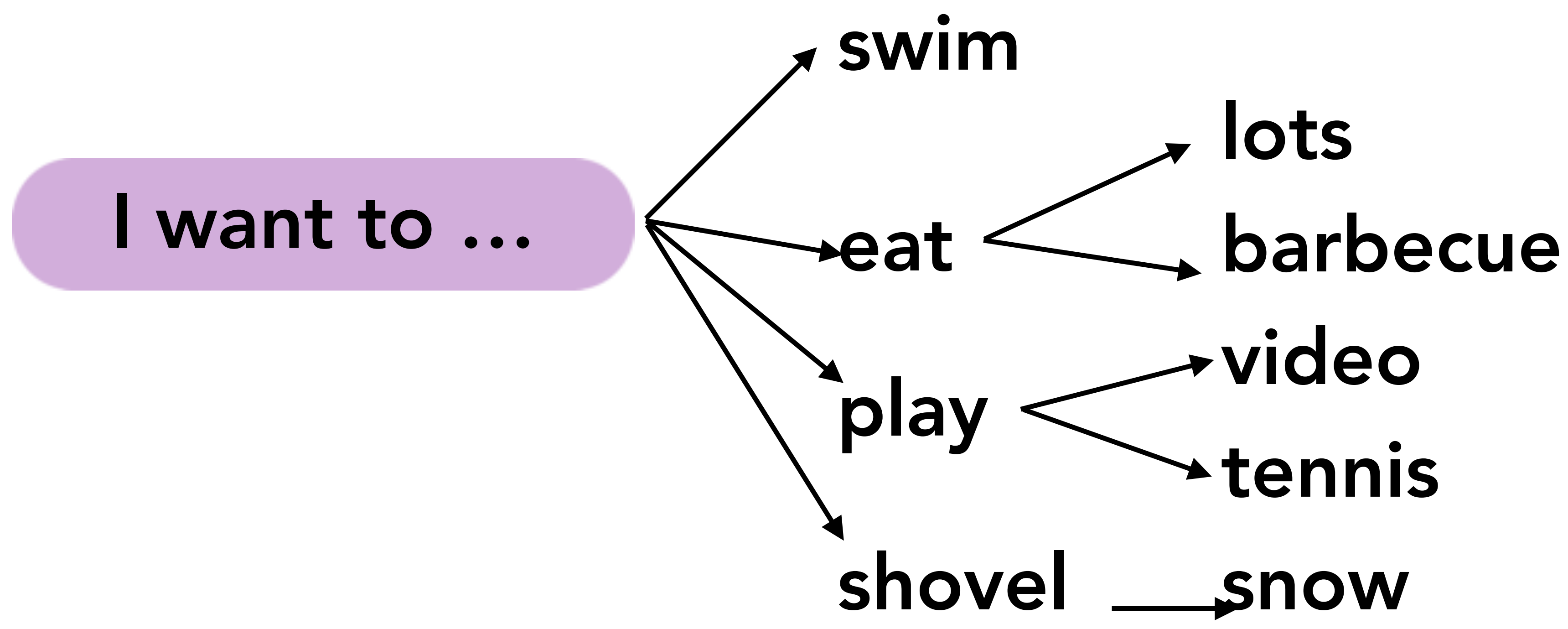
Logistics and Announcements

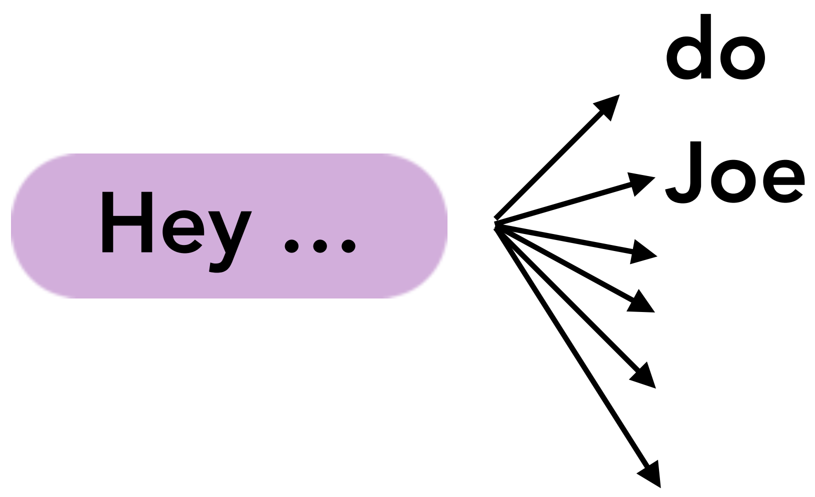
- Syllabus changes (see website) based on requests
  - e.g. Quiz 4 date changed to accommodate Grace Hopper Conference attendance
  - Project Dates have changed to give you more time for the status report and presentations
  - Add / drop dates for class: Sep 6 and project team formation deadline: Sep 10
- Class Project
  - Forming Groups: Watch out for Brightspace Announcement
    - Change: Will allow some groups of 6, but higher expectations from these groups
    - Will only accommodate a **maximum** of 52 teams
  - CARC access - We are working on it!
- Next Week:
  - Tue: HW1 released
  - Thu: Quiz 1 (Bring your laptop!)
- Brightspace Discussions: Start a new thread under Activities / Discussions / Forums / Topics (e.g. Group Creation)
- Missing Class? Report in advance using the form (pinned to Brightspace Announcements)
- Lecture Slides: Available after class

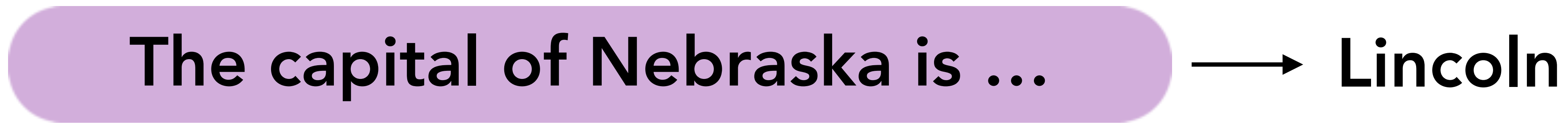
Logistics and Announcements

- Syllabus changes (see website) based on requests
  - e.g. Quiz 4 date changed to accommodate Grace Hopper Conference attendance
  - Project Dates have changed to give you more time for the status report and presentations
  - Add / drop dates for class: Sep 6 and project team formation deadline: Sep 10
- Class Project
  - Forming Groups: Watch out for Brightspace Announcement
    - Change: Will allow some groups of 6, but higher expectations from these groups
    - Will only accommodate a **maximum** of 52 teams
  - CARC access - We are working on it!
- Next Week:
  - Tue: HW1 released
  - Thu: Quiz 1 (Bring your laptop!)
- Brightspace Discussions: Start a new thread under Activities / Discussions / Forums / Topics (e.g. Group Creation)
- Missing Class? Report in advance using the form (pinned to Brightspace Announcements)
- Lecture Slides: Available after class
- Interest in research in my lab

Building a Language Model


I want to …
swim
eat
lots
barbecue
play
video
tennis
shovel
snow

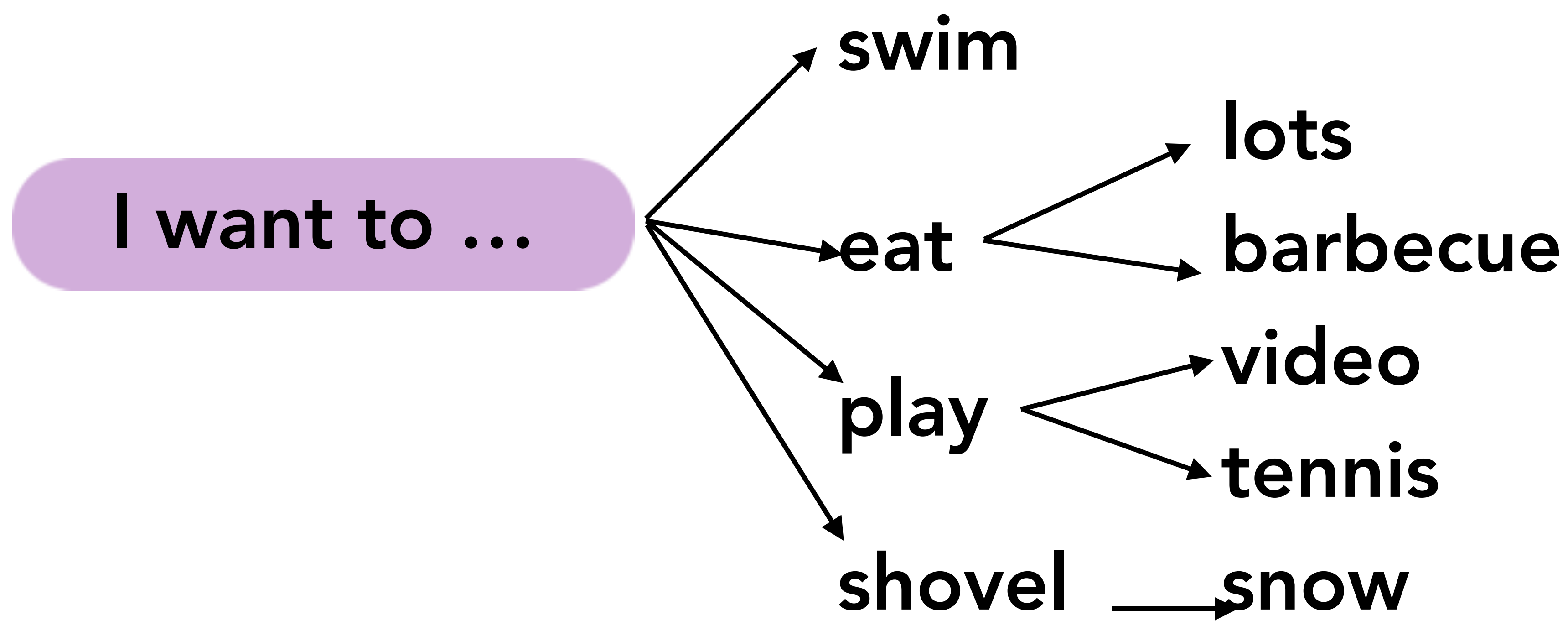

Hey …
do
Joe


The capital of Nebraska is …
Lincoln

- Task: Given a sequence of words so far (the **context**), predict what comes next

- We never know for sure what comes next, but we can still make good guesses!

Building a Language Model



I want to …

swim

eat

lots

barbecue

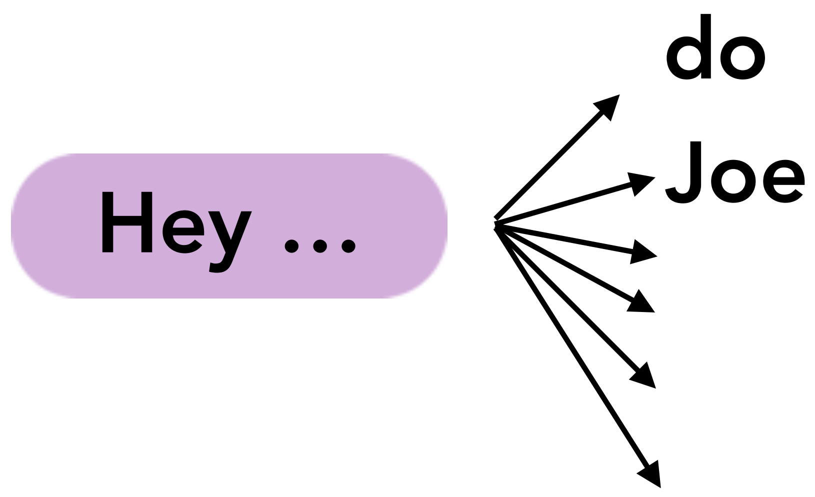
play

video

tennis

shovel

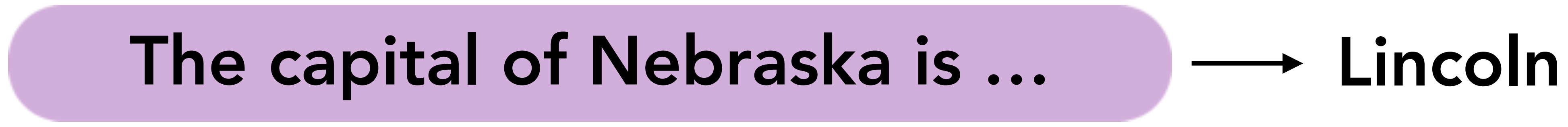
snow



Hey …

do

Joe



The capital of Nebraska is …

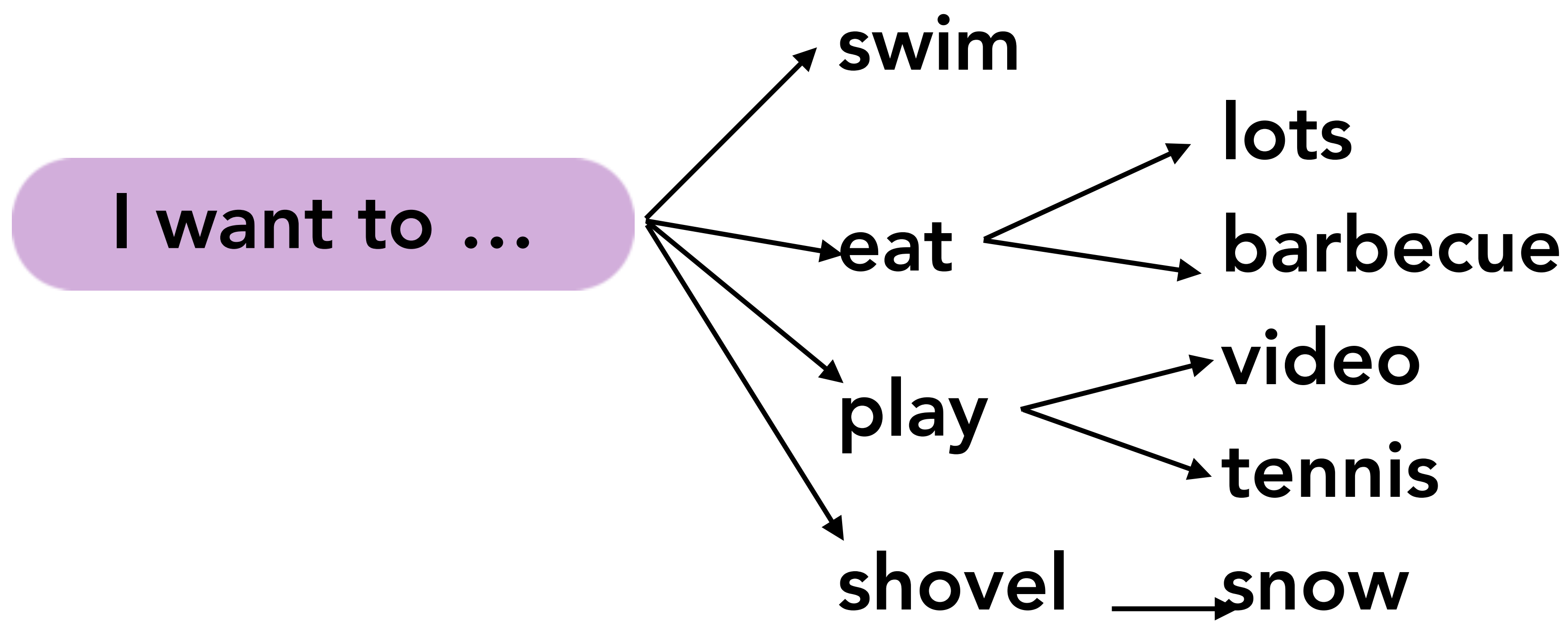
Lincoln

- Task: Given a sequence of words so far (the **context**), predict what comes next

- We never know for sure what comes next, but we can still make good guesses!

Certain sentence constructions are more likely than others, due to grammaticality, obscurity or commonness

Building a Language Model



I want to …

swim

eat

lots

barbecue

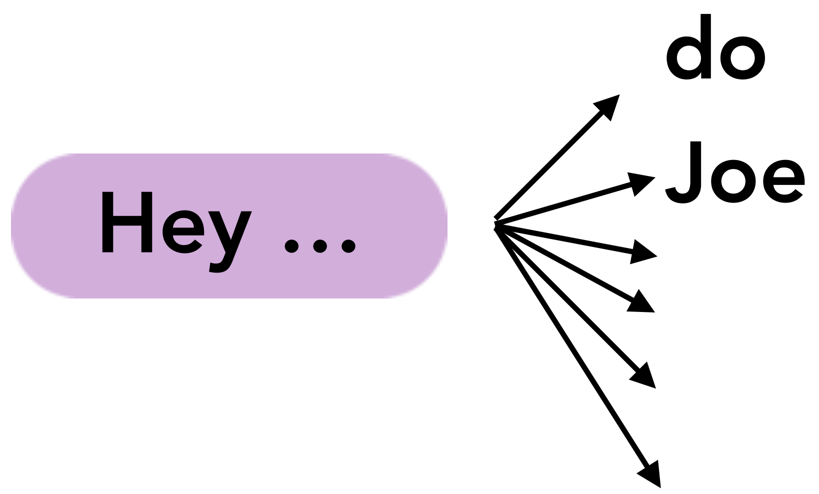
play

video

tennis

shovel

snow



Hey …

do

Joe

The capital of Nebraska is …

Lincoln

- Task: Given a sequence of words so far (the **context**), predict what comes next

- We never know for sure what comes next, but we can still make good guesses!

Certain sentence constructions are more likely than others, due to grammaticality, obscurity or commonness

Sentences have different probabilities!

Lecture Outline

1. Announcements + Recap
2. Probabilistic Language Models
3. n-gram Language Models
4. Evaluation and Perplexity
5. Generating from an n-gram Language Model
   i. Zeroes
6. Smoothing

Probabilistic Language Models!

Assign a probability to a sentence

Probabilistic Language Modeling

Probabilistic Language Modeling

Goal: compute the probability of a sentence or sequence of words:

Probabilistic Language Modeling

Goal: compute the probability of a sentence or sequence of words:

$$P(\mathbf{w}) = P(w_1, w_2, w_3, w_4, w_5, \ldots w_n)$$

Probabilistic Language Modeling

Goal: compute the probability of a sentence or sequence of words:

$$P(\mathbf{w}) = P(w_1, w_2, w_3, w_4, w_5, \ldots w_n)$$

Related task: probability of an upcoming word: $P(w_n \mid w_1, w_2, w_3, w_4, \ldots w_{n-1})$

Probabilistic Language Modeling

Goal: compute the probability of a sentence or sequence of words:

$$P(\mathbf{w}) = P(w_1, w_2, w_3, w_4, w_5, \ldots w_n)$$

Related task: probability of an upcoming word: $P(w_n \mid w_1, w_2, w_3, w_4, \ldots w_{n-1})$

A model that assigns probabilities to sequences of words (e.g., either of these: $P(\mathbf{w})$ or $P(w_n \mid w_1, w_2, \ldots w_{n-1})$) is called a language model

Probabilistic Language Modeling

Goal: compute the probability of a sentence or sequence of words:

$$P(\mathbf{w}) = P(w_1, w_2, w_3, w_4, w_5, \ldots w_n)$$

Difference

Related task: probability of an upcoming word: $P(w_n \mid w_1, w_2, w_3, w_4, \ldots w_{n-1})$

A model that assigns probabilities to sequences of words (e.g., either of these: $P(\mathbf{w})$ or $P(w_n \mid w_1, w_2, \ldots w_{n-1})$) is called a language model "its water is so transparent that you can see the bottom" "its water is so transparent that you can see the bottom"

$$P(\text{its water is so transparent that you can see the bottom})$$

“its water is so transparent that you can see the bottom”

$$P(\text{its water is so transparent that you can see the bottom})$$

$$P(\text{its, water, is, so, transparent, that, you, can, see, the, bottom})$$

How to compute $P(W)$?

“its water is so transparent that you can see the bottom”

P(its water is so transparent that you can see the bottom)

P(its, water, is, so, transparent, that, you, can, see, the, bottom)

How to compute this joint probability, $P(\mathbf{w}) = P(w_1, w_2, w_3, w_4, w_5, \ldots w_n)$?

e.g. P(its, water, is, so, transparent, that)

How to compute $P(W)$?

"its water is so transparent that you can see the bottom"

P(its water is so transparent that you can see the bottom)

P(its, water, is, so, transparent, that, you, can, see, the, bottom)

How to compute this joint probability, $P(\mathbf{w}) = P(w_1, w_2, w_3, w_4, w_5, \ldots w_n)$?

e.g. P(its, water, is, so, transparent, that)

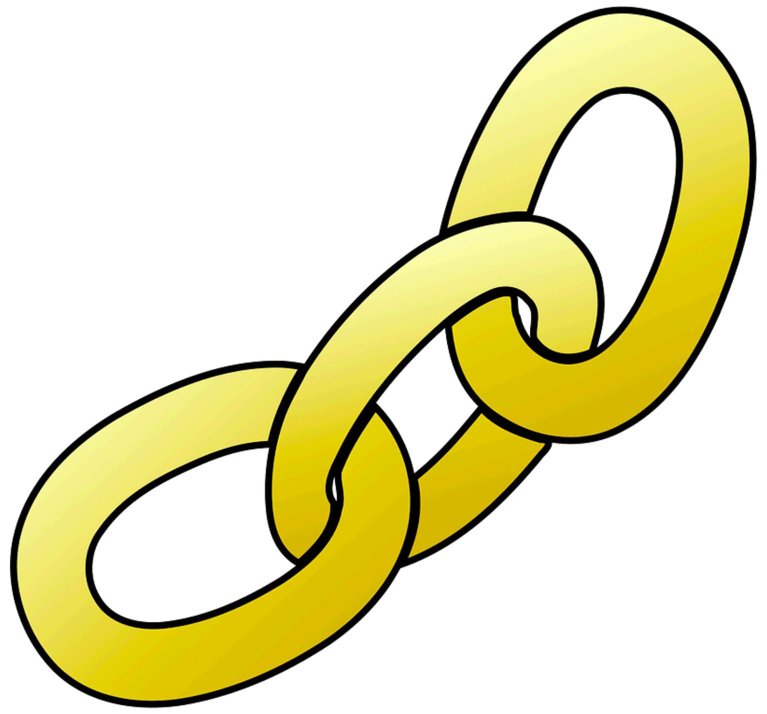
Intuition: let's rely on the Chain Rule of Probability

Chain Rule for words in a sentence

$P(\text{its water is so transparent}) =$

Chain Rule for words in a sentence

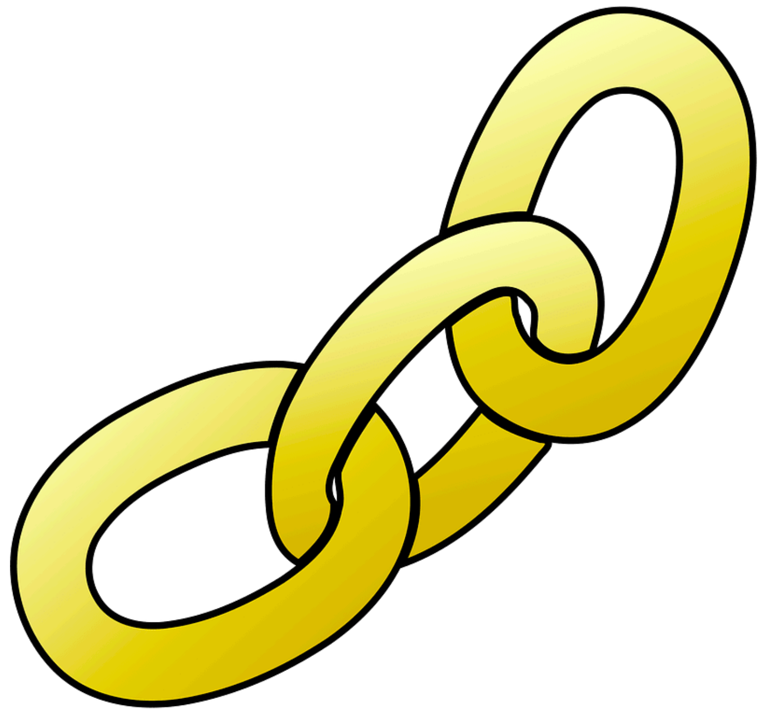
$$P(w_1, w_2, \ldots w_n) = \prod_{i=1}^{n} P(w_i \mid w_{i-1} \ldots w_1)$$



P(its water is so transparent) =

Chain Rule for words in a sentence

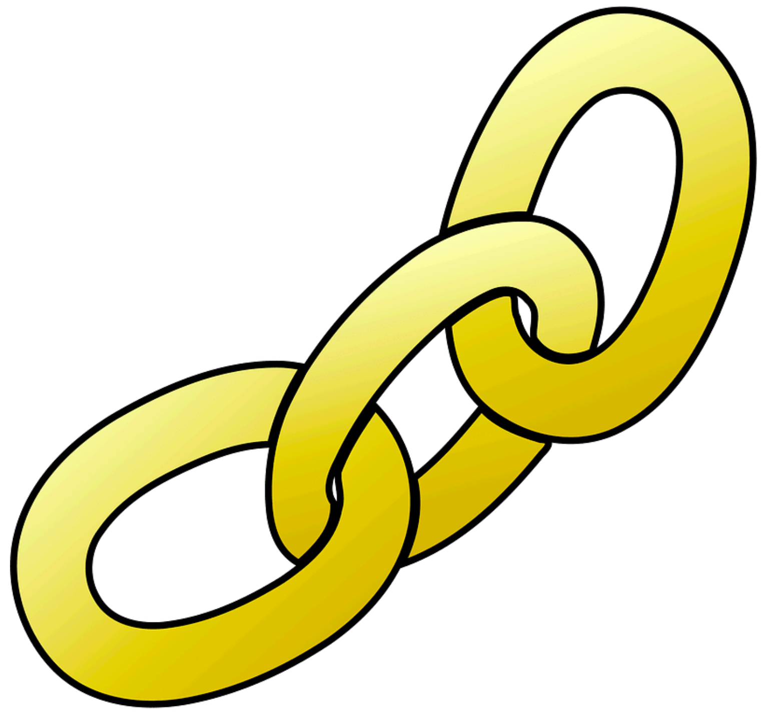
$$P(w_1, w_2, \ldots w_n) = \prod_{i=1}^{n} P(w_i \mid w_{i-1} \ldots w_1)$$



$$\begin{aligned} P(\text{its water is so transparent}) = \; & P(\text{its}) \times \\ & P(\text{water} \mid \text{its}) \times \\ & P(\text{is} \mid \text{its water}) \times \\ & P(\text{so} \mid \text{its water is}) \times \\ & P(\text{transparent} \mid \text{its water is so}) \end{aligned}$$

Chain Rule for words in a sentence

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



$$\begin{aligned} P(\text{its water is so transparent}) = \; & P(\text{its}) \times \\ & P(\text{water} \mid \text{its}) \times \\ & P(\text{is} \mid \text{its water}) \times \\ & P(\text{so} \mid \text{its water is}) \times \\ & P(\text{transparent} \mid \text{its water is so}) \end{aligned}$$

Ordering matters in language!

Why Probabilistic Models?

Why Probabilistic Models?

Why would you want to predict upcoming words,
or assign probabilities to sentences?

Why Probabilistic Models?

Why would you want to predict upcoming words, or assign probabilities to sentences?

- Probabilities are essential for language generation

Why Probabilistic Models?

Why would you want to predict upcoming words, or assign probabilities to sentences?

- Probabilities are essential for language generation
- Any task in which we have to identify words in noisy, ambiguous input, like speech recognition

I will be back soonish

I will be bassoon dish

Why Probabilistic Models?

Why would you want to predict upcoming words, or assign probabilities to sentences?

- Probabilities are essential for language generation
- Any task in which we have to identify words in noisy, ambiguous input, like speech recognition
- For writing tools like spelling correction or grammatical error correction

I will be back soonish

I will be bassoon dish

Your so silly

You’re so silly

Everything has improve

Everything has improved

Probabilistic Language Models

Machine Translation:

- P(high winds tonight) $>$ P(large winds tonight)

Spell Correction:

- P(I'm about fifteen minuets away) $<$ P(I'm about fifteen minutes away)

Speech Recognition:

- P(I saw a van) $>>$ P(eyes awe of an)

Summarization, question-answering, etc., etc.!!

Probabilistic Language Models

Machine Translation:

- P(high winds tonight) $>$ P(large winds tonight)

Spell Correction:

- P(I'm about fifteen minuets away) $<$ P(I'm about fifteen minutes away)

Speech Recognition:

- P(I saw a van) $>>$ P(eyes awe of an)

Summarization, question-answering, etc., etc.!!

But how to learn these probabilities?

CoolClips.com

Suppose we have a biased coin that's heads with probability p.

Suppose we have a biased coin that's heads with probability p.

Suppose we flip the coin four times and see (H, H, H, T). What is p?

Probability Estimation via Statistical Modeling

Suppose we have a biased coin that's heads with probability p.

Suppose we flip the coin four times and see (H, H, H, T). What is p?

We don't know what p is — could be 0.5! But $p = 3/4 = 0.75$ maximizes the probability of data sequence (H,H,H,T)

Probability Estimation via Statistical Modeling

Suppose we have a biased coin that's heads with probability p.

Suppose we flip the coin four times and see (H, H, H, T). What is p?

We don't know what p is — could be 0.5! But $p = 3/4 = 0.75$ maximizes the probability of data sequence (H,H,H,T)

maximum likelihood estimate

Probability Estimation via Statistical Modeling

Suppose we have a biased coin that's heads with probability p.

Suppose we flip the coin four times and see (H, H, H, T). What is p?

We don't know what p is — could be 0.5! But $p = 3/4 = 0.75$ maximizes the probability of data sequence (H,H,H,T)

maximum likelihood estimate

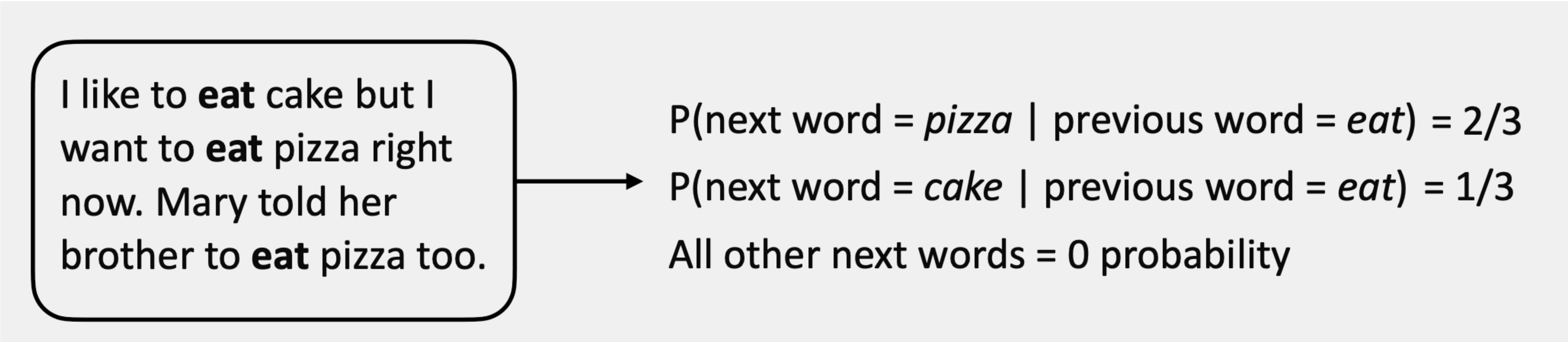
The probability of the data is $ppp(1-p)$: if you take the derivative and set it equal to zero and find $p = 0.75$

n-gram Language Model

The decision for what words occur after a word w is exactly the same as the biased coin, but with ***many*** possible outcomes (as many as all the words) instead of 2

n-gram Language Model

The decision for what words occur after a word w is exactly the same as the biased coin, but with ***many*** possible outcomes (as many as all the words) instead of 2



I like to **eat** cake but I want to **eat** pizza right now. Mary told her brother to **eat** pizza too.

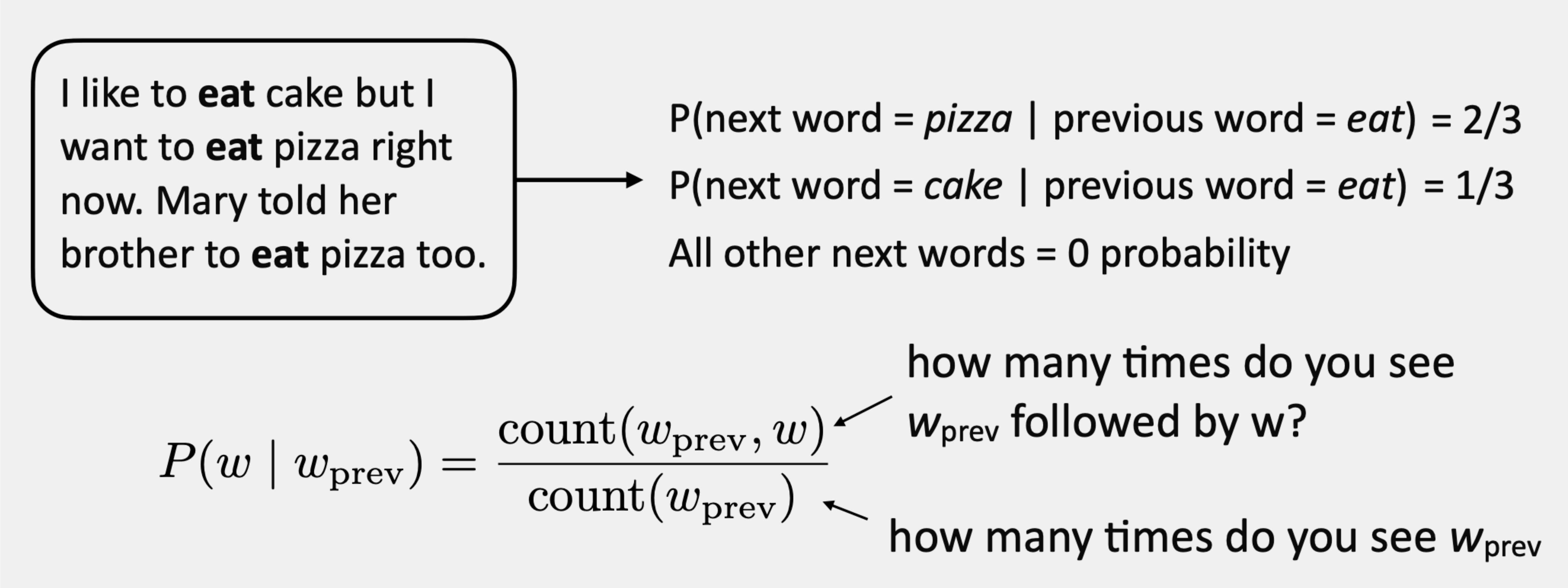
P(next word = *pizza* | previous word = *eat*) = 2/3

P(next word = *cake* | previous word = *eat*) = 1/3

All other next words = 0 probability

n-gram Language Model

The decision for what words occur after a word w is exactly the same as the biased coin, but with ***many*** possible outcomes (as many as all the words) instead of 2



I like to **eat** cake but I want to **eat** pizza right now. Mary told her brother to **eat** pizza too.

P(next word = *pizza* | previous word = *eat*) = 2/3

P(next word = *cake* | previous word = *eat*) = 1/3

All other next words = 0 probability

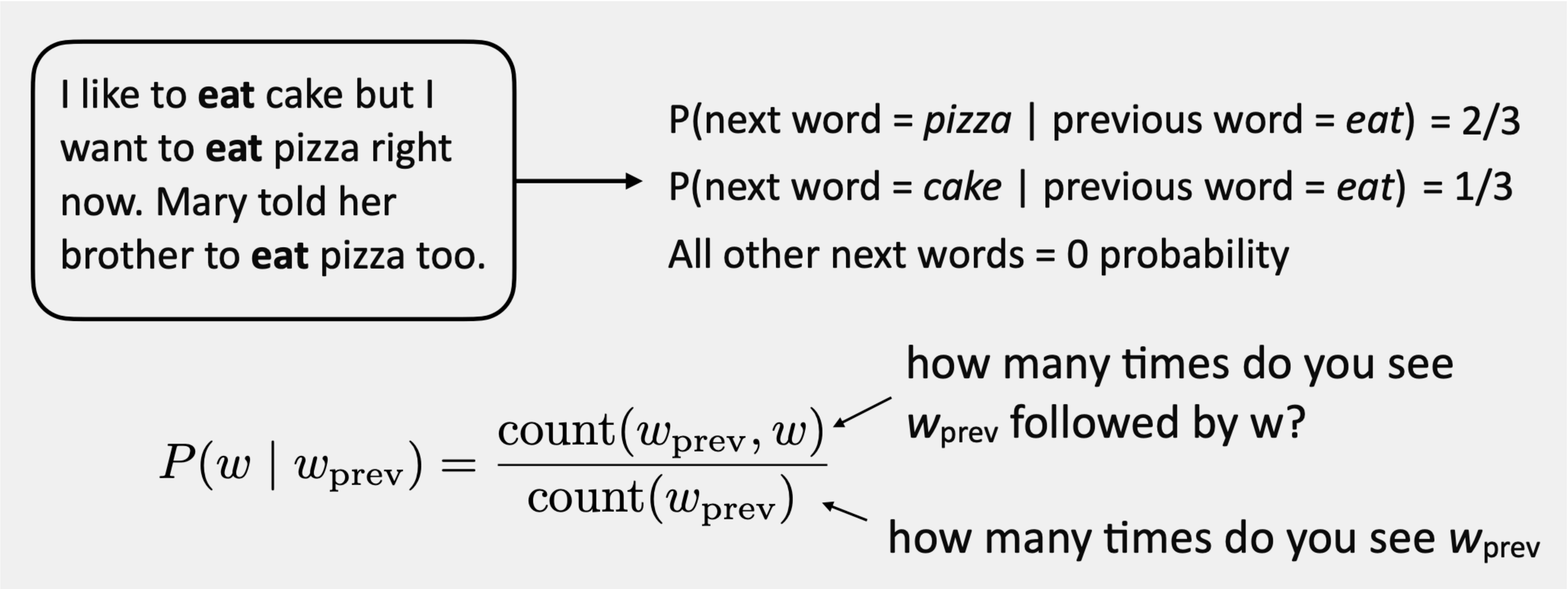
$$P(w \mid w_{\text{prev}}) = \frac{\text{count}(w_{\text{prev}}, w)}{\text{count}(w_{\text{prev}})}$$

how many times do you see w_{prev} followed by w?

how many times do you see w_{prev}

n-gram Language Model

The decision for what words occur after a word w is exactly the same as the biased coin, but with ***many*** possible outcomes (as many as all the words) instead of 2



I like to **eat** cake but I want to **eat** pizza right now. Mary told her brother to **eat** pizza too.

P(next word = *pizza* | previous word = *eat*) = 2/3

P(next word = *cake* | previous word = *eat*) = 1/3

All other next words = 0 probability

$$P(w \mid w_{\text{prev}}) = \frac{\text{count}(w_{\text{prev}}, w)}{\text{count}(w_{\text{prev}})}$$

how many times do you see w_{prev} followed by w?

how many times do you see w_{prev}

Vocabulary

How to estimate the probability of the next word?

$$P(\text{that} \mid \text{its water is so transparent}) = \frac{Count(\text{its water is so transparent that})}{Count(\text{its water is so transparent})}$$

How to estimate the probability of the next word?

$$P(\text{that} \mid \text{its water is so transparent}) = \frac{Count(\text{its water is so transparent that})}{Count(\text{its water is so transparent})}$$

Could we just count and divide?

How to estimate the probability of the next word?

$$P(\text{that} \mid \text{its water is so transparent}) = \frac{Count(\text{its water is so transparent that})}{Count(\text{its water is so transparent})}$$

Could we just count and divide?

No! Too many possible sentences!

We’ll never see enough data for estimating these

Simplifying Assumption:

Simplifying Assumption:

$$P(\text{that} \,|\, \text{its water is so transparent }) \approx P(\text{that} \,|\, \text{transparent})$$

Markov Assumption

Simplifying Assumption:

$$P(\text{that} \mid \text{its water is so transparent}) \approx P(\text{that} \mid \text{transparent})$$

Andrei Markov

Markov Assumption

Andrei Markov

Simplifying Assumption:

$$P(\text{that} \mid \text{its water is so transparent }) \approx P(\text{that} \mid \text{transparent})$$

Or maybe…

$$P(\text{that} \mid \text{its water is so transparent}) \approx P(\text{that} \mid \text{so transparent})$$

Markov Assumption contd.

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

Markov Assumption contd.

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

In other words, we approximate each component in the product such that it is only conditioned on the previous k elements

Markov Assumption contd.

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

In other words, we approximate each component in the product such that it is only conditioned on the previous k elements

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

Markov Assumption contd.

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

In other words, we approximate each component in the product such that it is only conditioned on the previous k elements

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

$(k+1)$-th order Markov assumption

Mini Recap: Probabilistic Modeling

Mini Recap: Probabilistic Modeling

- What is a probabilistic language model?

Mini Recap: Probabilistic Modeling

- What is a probabilistic language model?
- Why would we need one?

Mini Recap: Probabilistic Modeling

- What is a probabilistic language model?
- Why would we need one?
- How do we estimate one?

Mini Recap: Probabilistic Modeling

- What is a probabilistic language model?
- Why would we need one?
- How do we estimate one?
- How do we simplify the estimation problem?

Mini Recap: Probabilistic Modeling

- What is a probabilistic language model?
- Why would we need one?
- How do we estimate one?
- How do we simplify the estimation problem?
- Next: a simple probabilistic language model

Lecture Outline

1. Announcements + Recap
2. Probabilistic Language Models
3. n-gram Language Models
4. Evaluation and Perplexity
5. Generating from an n-gram Language Model
    i. Zeroes
6. Smoothing

n-gram Language Models

simplest probabilistic model

Simplest Case: Unigram model

Simplest Case: Unigram model

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

Simplest Case: Unigram model

$$P(w_1, w_2, \dots w_n) \approx \prod_i P(w_i)$$

Some automatically generated sentences 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

Bigram Model

Condition on the previous word:

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

Bigram Model

Condition on the previous word:

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

Some automatically generated sentences from a bigram model

Bigram Model

Condition on the previous word:

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

Some automatically generated sentences from 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

n-gram Language Models

Can extend to trigrams, 4-grams, 5-grams, …

In general this is an insufficient model of language

n-gram Language Models

Can extend to trigrams, 4-grams, 5-grams, …

In general this is an insufficient model of language

“The computer which I had just put into the machine room on the fifth floor crashed.”

n-gram Language Models

Can extend to trigrams, 4-grams, 5-grams, …

In general this is an insufficient model of language

"The computer which I had just put into the machine room on the fifth floor crashed."

Long-distance / Long-range dependencies

n-gram Language Models

Can extend to trigrams, 4-grams, 5-grams, …

In general this is an insufficient model of language

> “The computer which I had just put into the machine room on the fifth floor crashed.”

Long-distance / Long-range dependencies

But we can often get away with n-gram models, where n is a small number

Estimating bigram probabilities

The maximum likelihood estimate

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

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

Estimating bigram probabilities

The maximum likelihood estimate

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

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

What happens when $i = 1$?

Estimating bigram probabilities

The maximum likelihood estimate

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

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

What happens when $i = 1$?

Special edge case tokens: <s> and </s> for beginning of sentence and end of sentence, respectively

An example

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

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>

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>

$$P(\texttt{I} \mid \texttt{<s>}) = \tfrac{2}{3} = .67 \qquad P(\texttt{Sam} \mid \texttt{<s>}) = \tfrac{1}{3} = .33 \qquad P(\texttt{am} \mid \texttt{I}) = \tfrac{2}{3} = .67$$

$$P(\texttt{</s>} \mid \texttt{Sam}) = \tfrac{1}{2} = 0.5 \qquad P(\texttt{Sam} \mid \texttt{am}) = \tfrac{1}{2} = .5 \qquad P(\texttt{do} \mid \texttt{I}) = \tfrac{1}{3} = .33$$

Larger Example:
Berkeley Restaurant Project (BRP)

- 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

Total: 9222 similar sentences

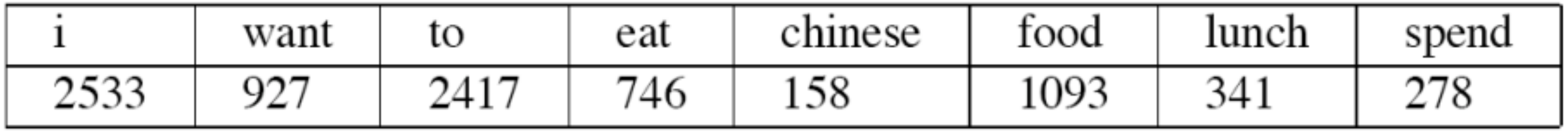
BRP: Raw Counts

Out of 9222 sentences

BRP: Raw Counts

Out of 9222 sentences

Unigrams

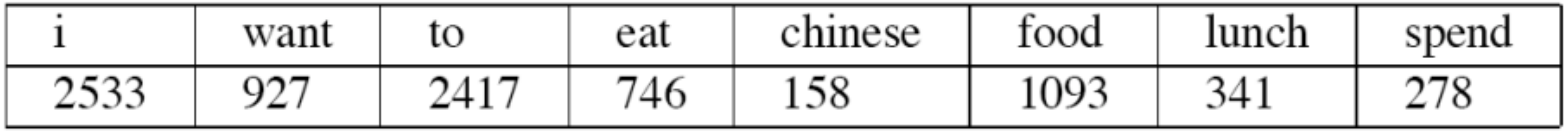


i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

BRP: Raw Counts

Out of 9222 sentences

Unigrams

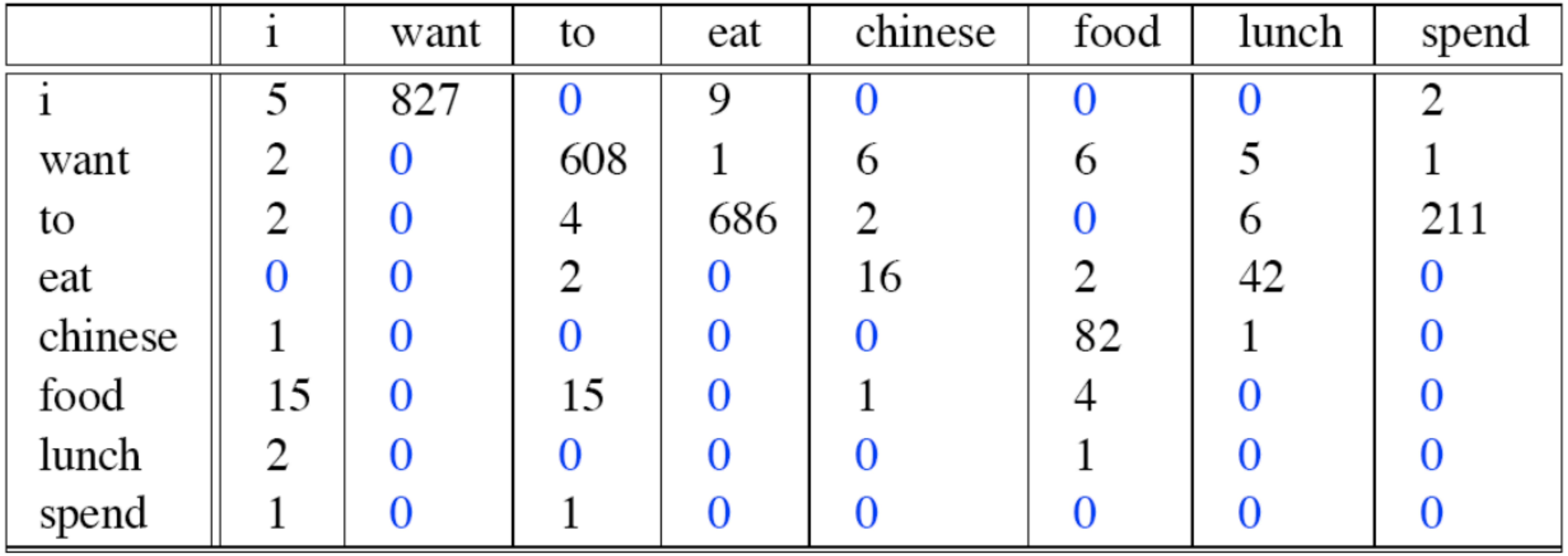


i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Next Word

Bigrams

History



	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

BRP: Bigram Probabilities

Bigram Probabilities: Raw bigram counts normalized by unigram counts

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

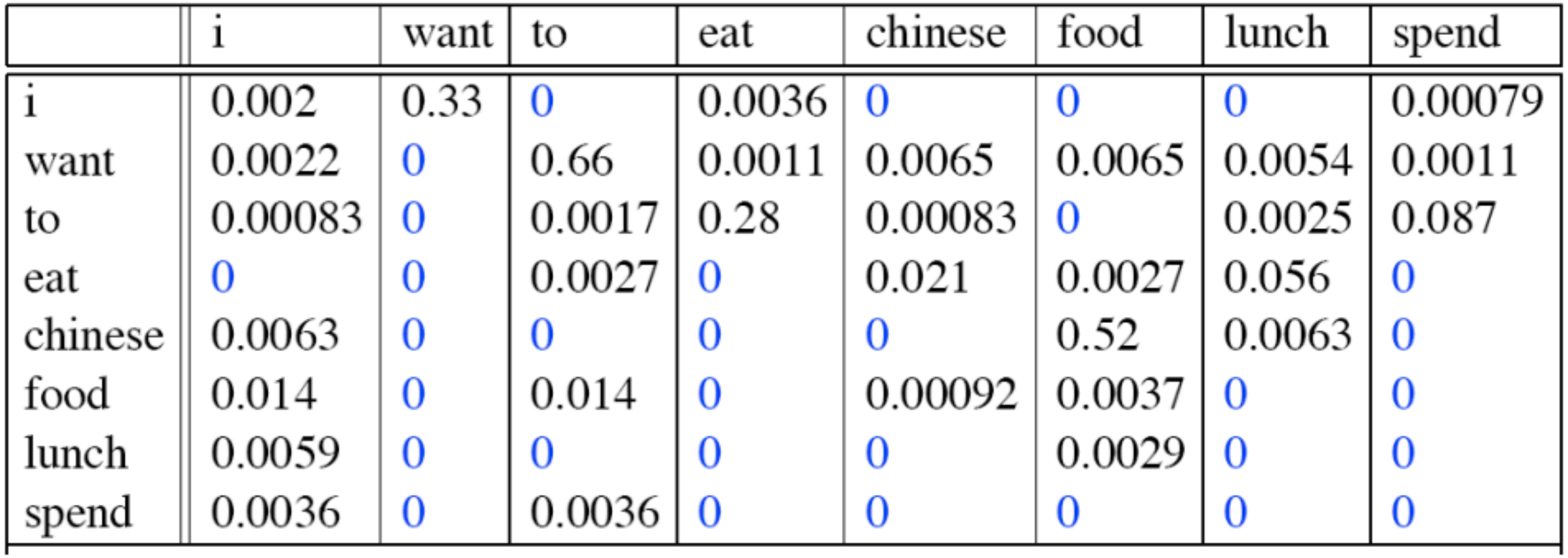
BRP: Bigram Probabilities

Bigram Probabilities: Raw bigram counts normalized by unigram counts

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

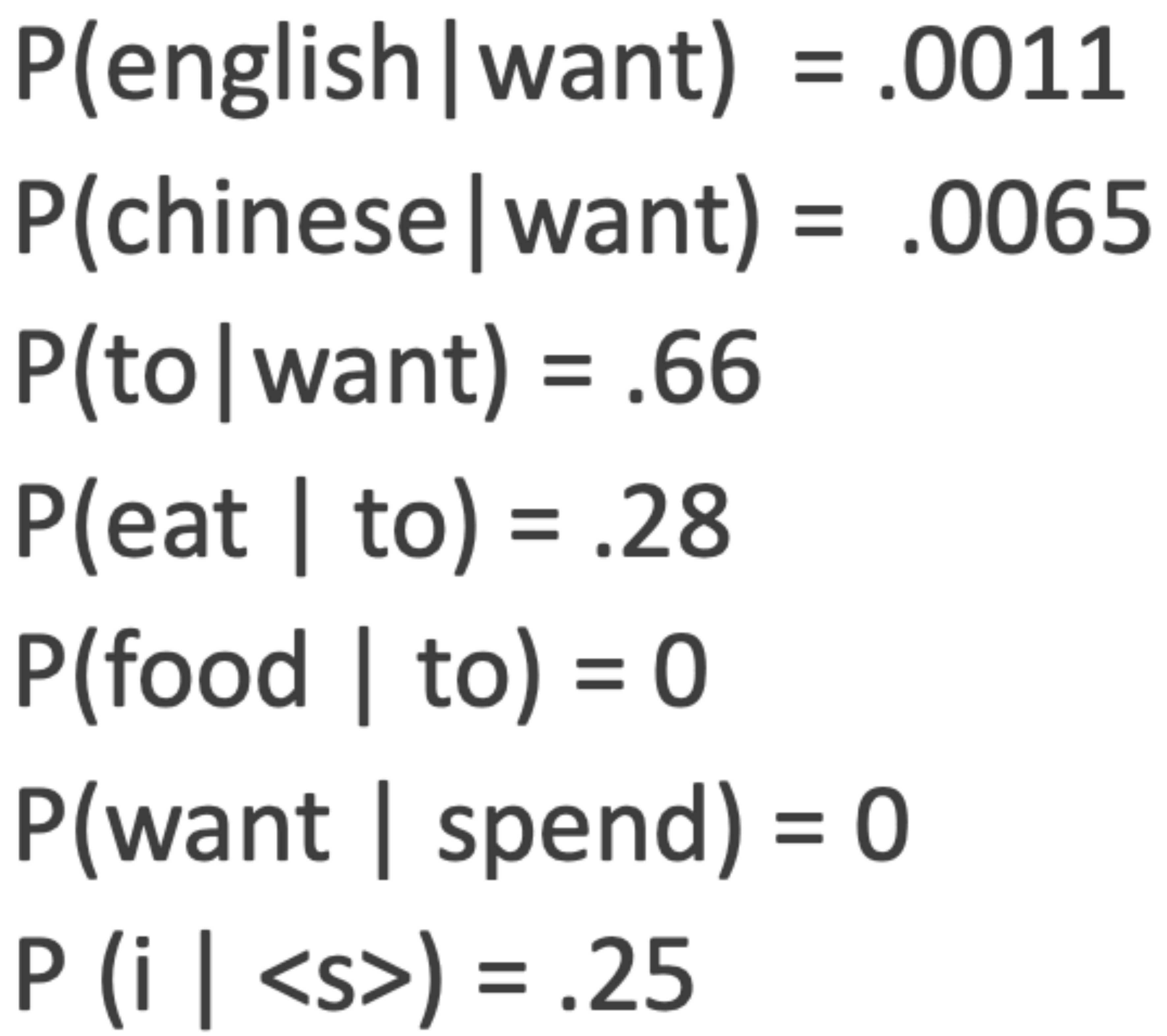
w_i

w_{i-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

What kinds of knowledge?



$P(\text{english}|\text{want}) = .0011$

$P(\text{chinese}|\text{want}) = .0065$

$P(\text{to}|\text{want}) = .66$

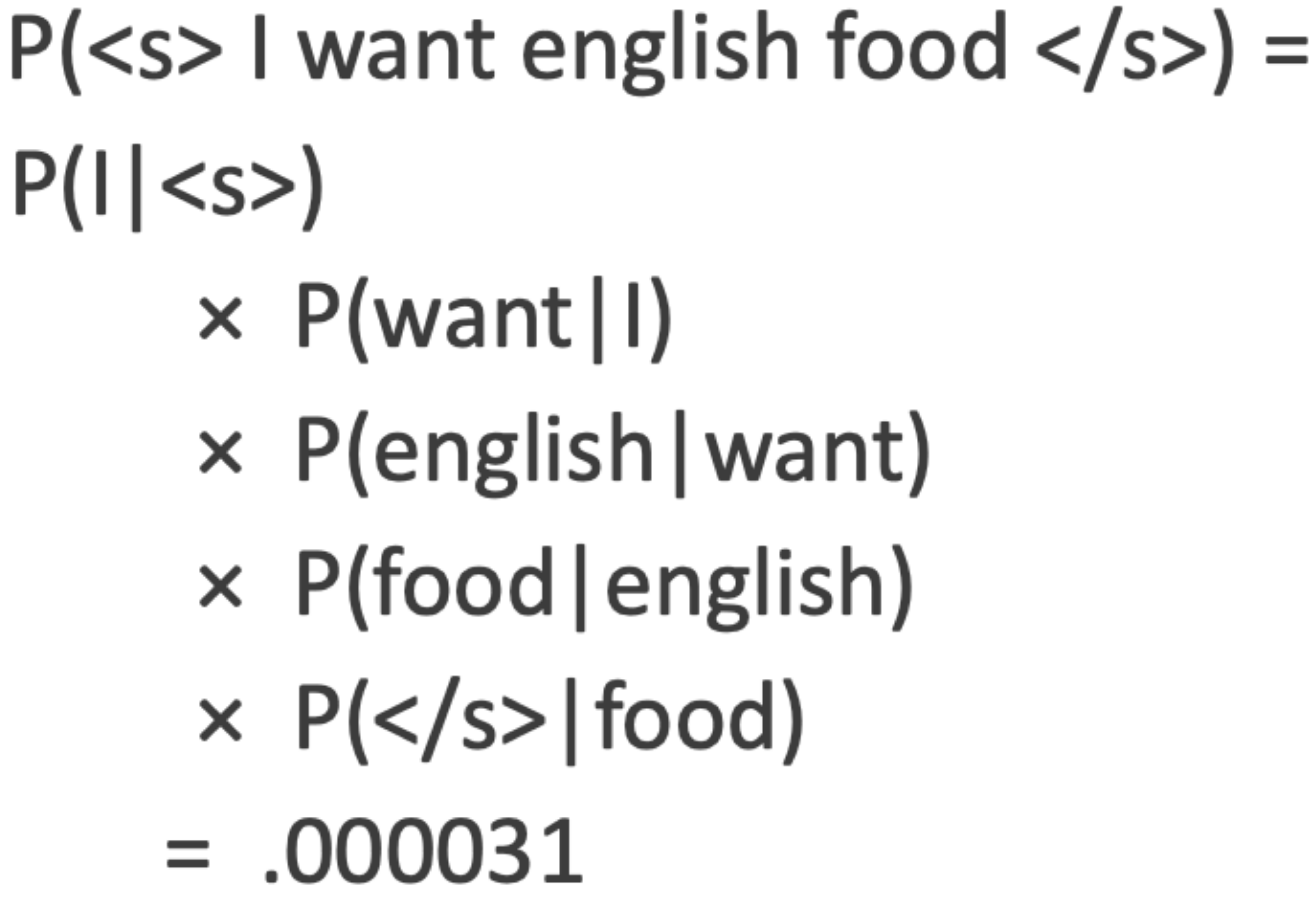
$P(\text{eat} \mid \text{to}) = .28$

$P(\text{food} \mid \text{to}) = 0$

$P(\text{want} \mid \text{spend}) = 0$

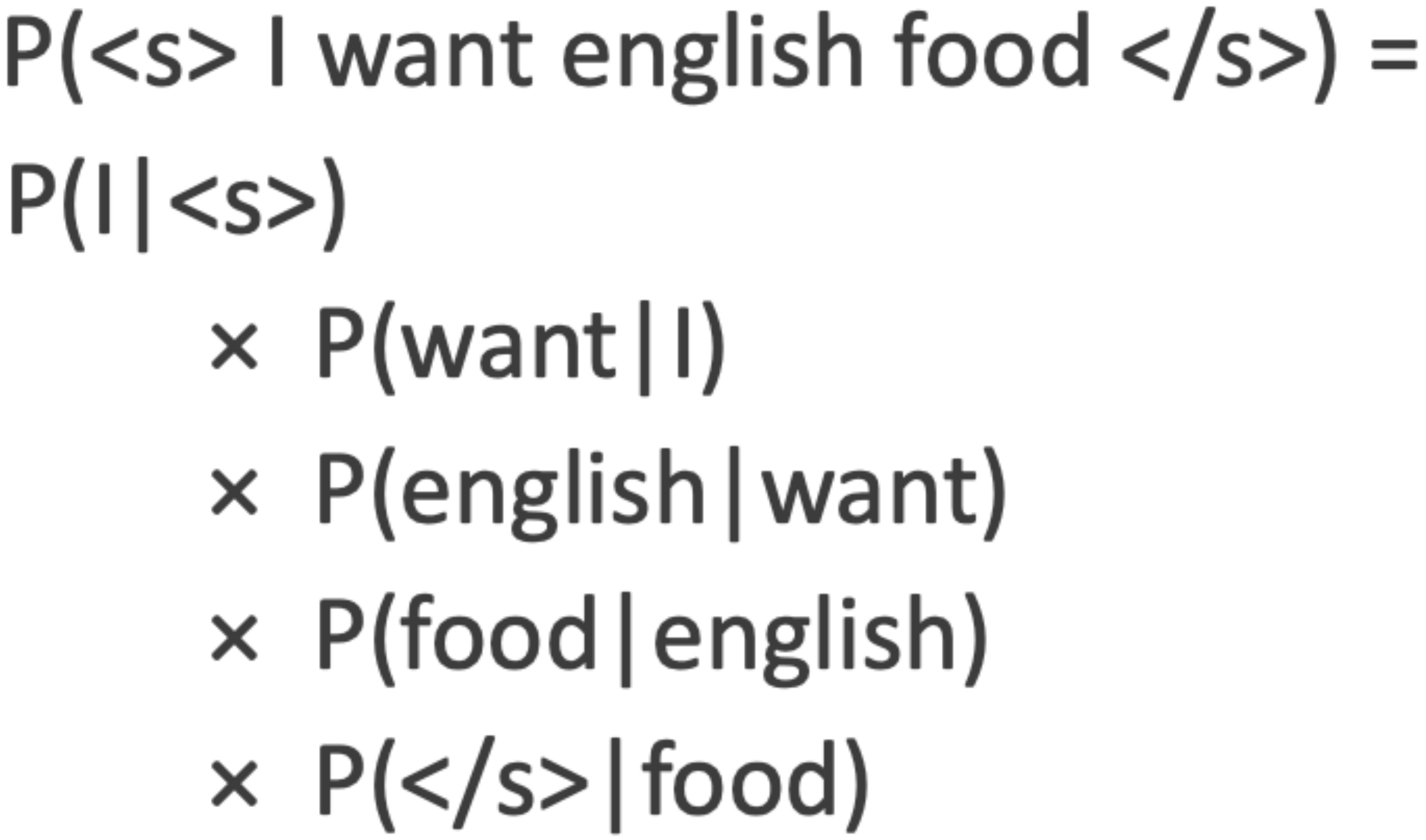
$P(\text{i} \mid \text{<s>}) = .25$

Bigram estimates of sentence probabilities



$$
\begin{aligned}
&P(\text{<s> I want english food </s>}) = \\
&P(\text{I}|\text{<s>}) \\
&\quad \times P(\text{want}|\text{I}) \\
&\quad \times P(\text{english}|\text{want}) \\
&\quad \times P(\text{food}|\text{english}) \\
&\quad \times P(\text{</s>}|\text{food}) \\
&= .000031
\end{aligned}
$$

Bigram estimates of sentence probabilities



$$
\begin{aligned}
&P(\text{<s> I want english food </s>}) = \\
&P(\text{I}|\text{<s>}) \\
&\quad \times P(\text{want}|\text{I}) \\
&\quad \times P(\text{english}|\text{want}) \\
&\quad \times P(\text{food}|\text{english}) \\
&\quad \times P(\text{</s>}|\text{food})
\end{aligned}
$$

$$\boxed{= .000031}$$

Quite low…

Underflow Issues

We do everything in log space

- Avoid underflow
- Adding is faster than multiplying

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

Lecture Outline

1. Announcements + Recap
2. Probabilistic Language Models
3. n-gram Language Models
4. Evaluation and Perplexity
5. Generating from an n-gram Language Model
   i. Zeroes
6. Smoothing

Evaluation and Perplexity

How good is a language model?

How good is a language model?

Does our language model prefer good sentences to bad ones?

How good is a language model?

Does our language model prefer good sentences to bad ones?

- Key Idea: Assign higher probability to “real” or “frequently observed” sentences than “ungrammatical” or “rarely observed” sentences?

How good is a language model?

Does our language model prefer good sentences to bad ones?

- Key Idea: Assign higher probability to “real” or “frequently observed” sentences than “ungrammatical” or “rarely observed” sentences?
  - In practice we don’t explicitly need to do the latter!

How good is a language model?

Does our language model prefer good sentences to bad ones?

- Key Idea: Assign higher probability to “real” or “frequently observed” sentences than “ungrammatical” or “rarely observed” sentences?
  - In practice we don’t explicitly need to do the latter!

We train parameters of our model on a **training set**.

How good is a language model?

Does our language model prefer good sentences to bad ones?

- Key Idea: Assign higher probability to “real” or “frequently observed” sentences than “ungrammatical” or “rarely observed” sentences?
  - In practice we don’t explicitly need to do the latter!

We train parameters of our model on a **training set**.

We test the model’s performance on data we haven’t seen.

How good is a language model?

Does our language model prefer good sentences to bad ones?

- Key Idea: Assign higher probability to “real” or “frequently observed” sentences than “ungrammatical” or “rarely observed” sentences?
  - In practice we don’t explicitly need to do the latter!

We train parameters of our model on a **training set**.

We test the model’s performance on data we haven’t seen.

- A **test set** is an unseen dataset that is different from our training set, totally unused.

How good is a language model?

Does our language model prefer good sentences to bad ones?

- Key Idea: Assign higher probability to “real” or “frequently observed” sentences than “ungrammatical” or “rarely observed” sentences?
  - In practice we don’t explicitly need to do the latter!

We train parameters of our model on a **training set**.

We test the model’s performance on data we haven’t seen.

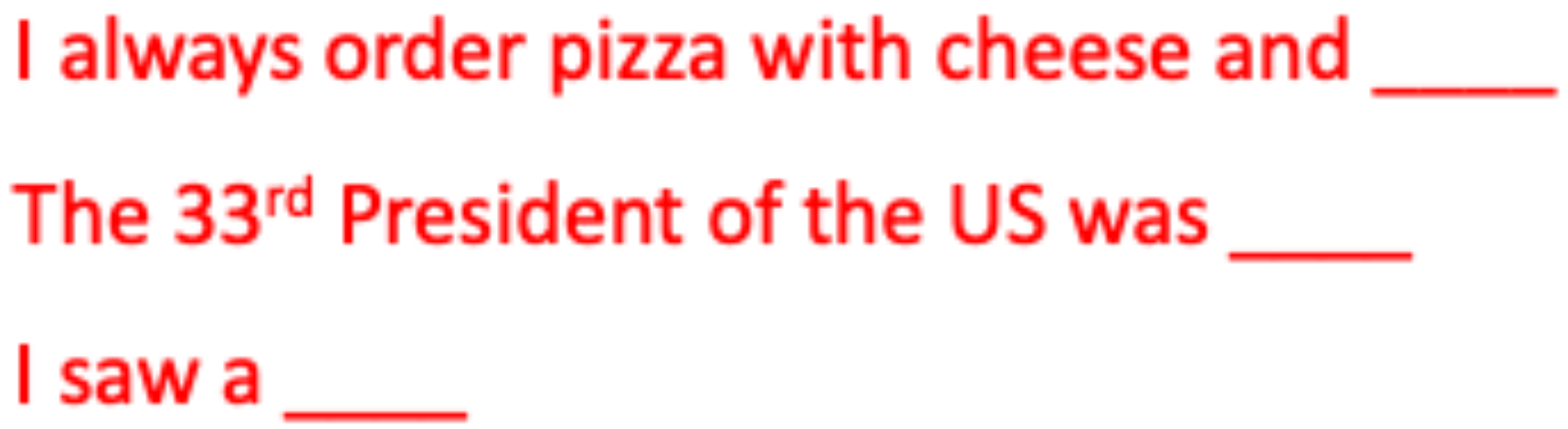
- A **test set** is an unseen dataset that is different from our training set, totally unused.
- An **evaluation metric** tells us how well our model does on the test set.

Intuition of Perplexity

The **Shannon Game**: How well can we predict the next word?

Intuition of Perplexity

The **Shannon Game**: How well can we predict the next word?



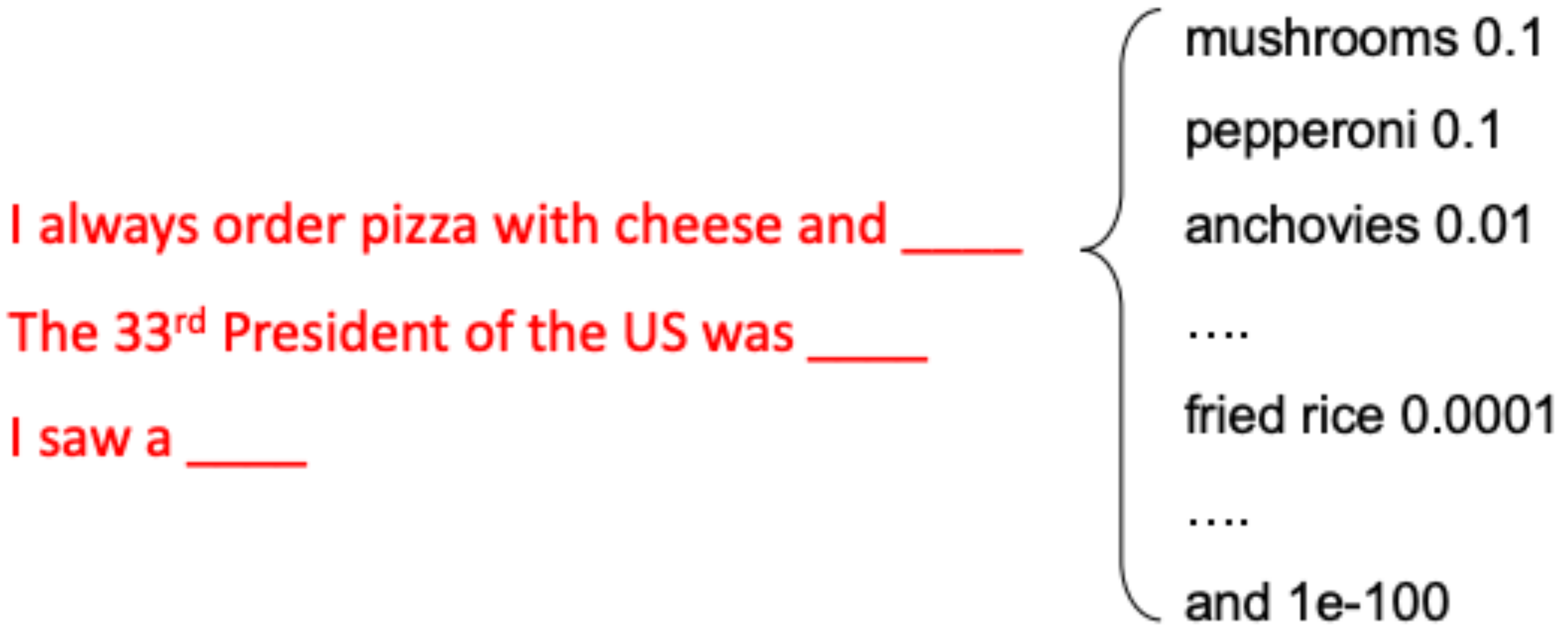
I always order pizza with cheese and ____

The 33rd President of the US was ____

I saw a ____

Intuition of Perplexity

The **Shannon Game**: How well can we predict the next word?



I always order pizza with cheese and ____

The 33rd President of the US was ____

I saw a ____

mushrooms 0.1

pepperoni 0.1

anchovies 0.01

….

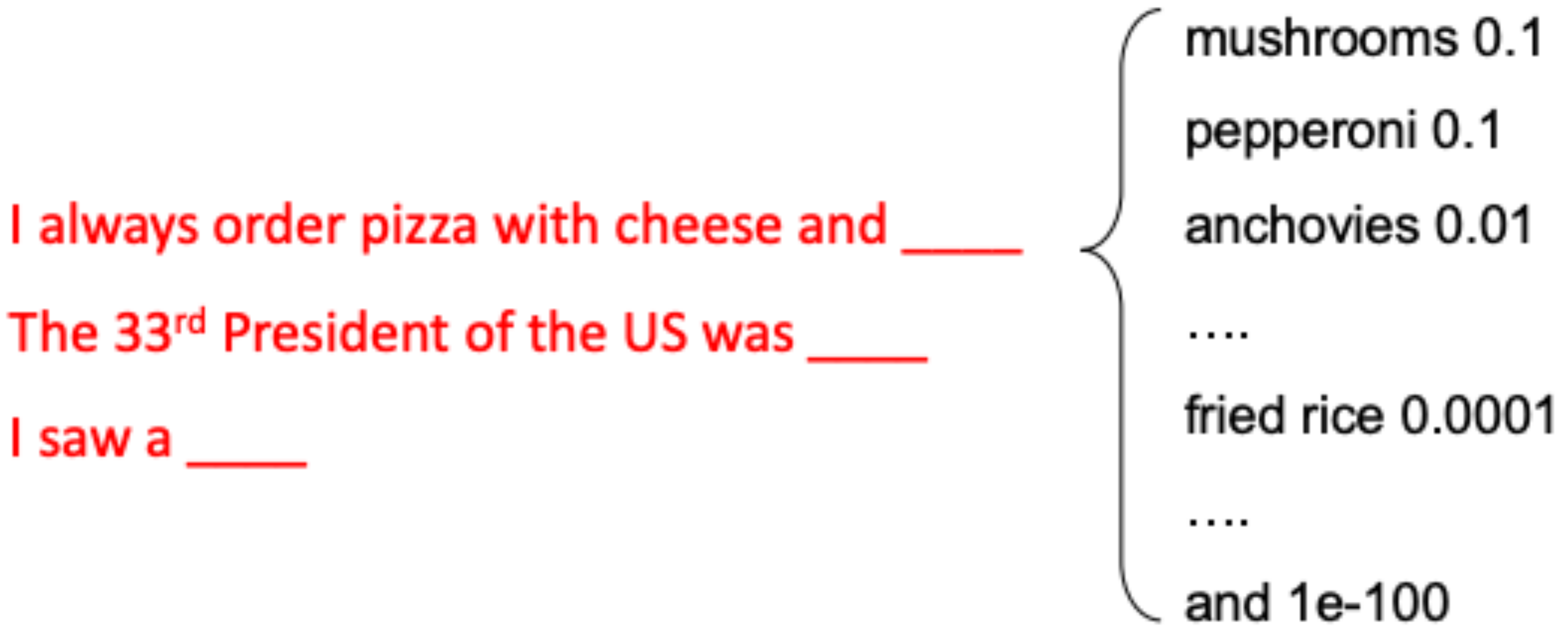
fried rice 0.0001

….

and 1e-100

Intuition of Perplexity

The **Shannon Game**: How well can we predict the next word?



I always order pizza with cheese and ____

The 33rd President of the US was ____

I saw a ____

mushrooms 0.1

pepperoni 0.1

anchovies 0.01

….

fried rice 0.0001

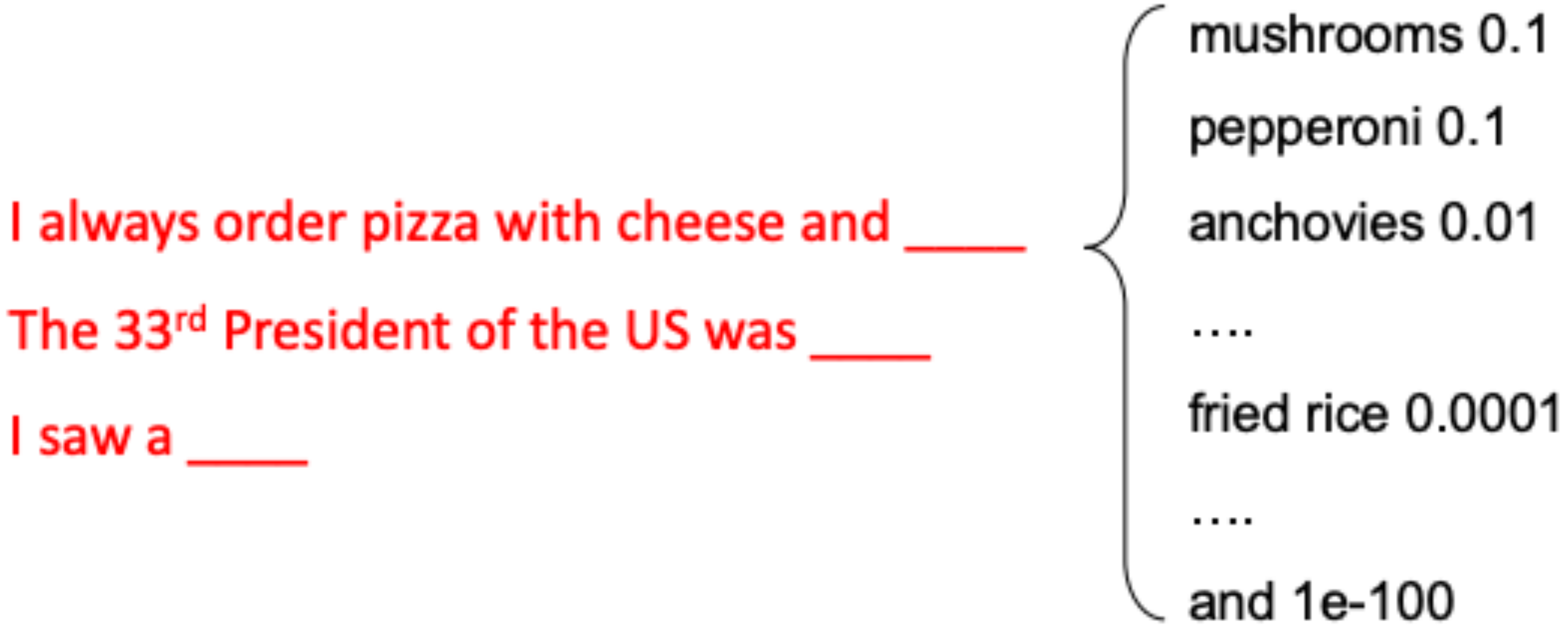
….

and 1e-100

Unigrams are terrible at this game!

Intuition of Perplexity

The **Shannon Game**: How well can we predict the next word?



I always order pizza with cheese and ____

The 33rd President of the US was ____

I saw a ____

mushrooms 0.1

pepperoni 0.1

anchovies 0.01

….

fried rice 0.0001

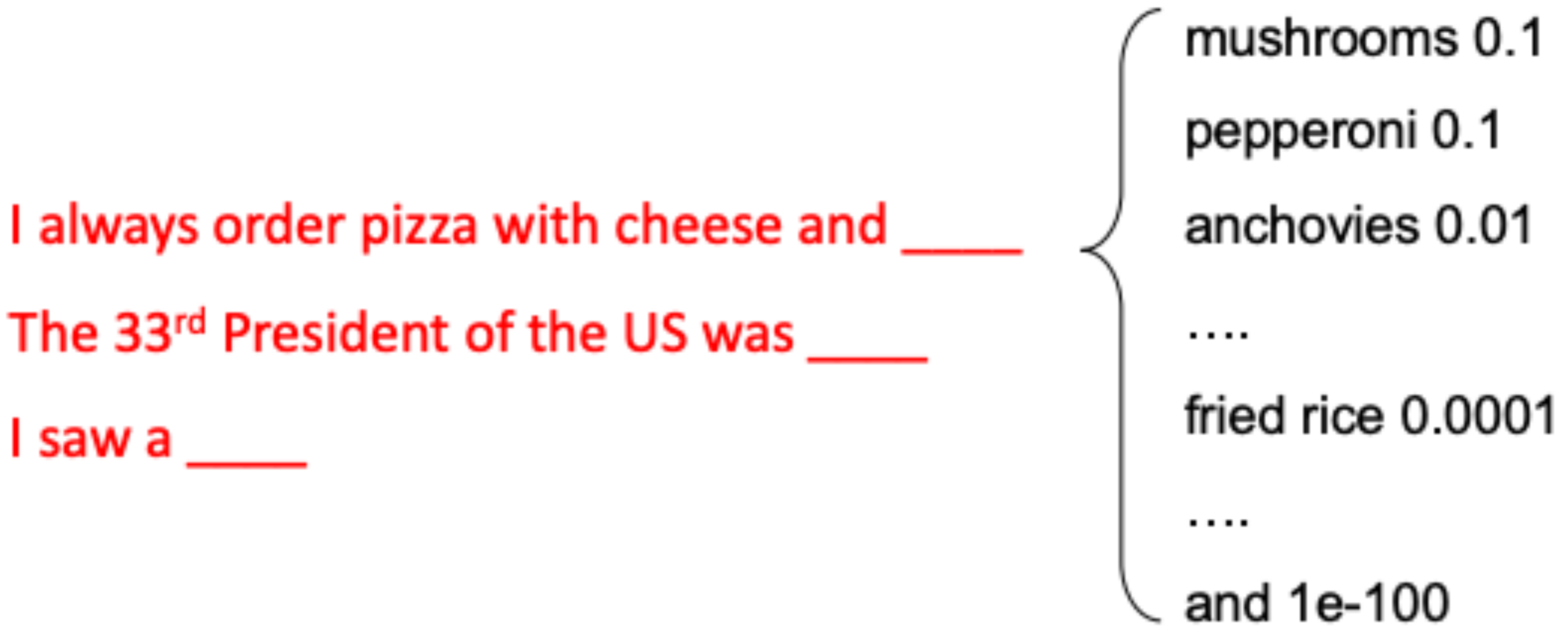
….

and 1e-100

Unigrams are terrible at this game!

Intuition of Perplexity

The **Shannon Game**: How well can we predict the next word?



I always order pizza with cheese and ____

The 33rd President of the US was ____

I saw a ____

mushrooms 0.1

pepperoni 0.1

anchovies 0.01

….

fried rice 0.0001

….

and 1e-100

Unigrams are terrible at this game!

A better model of a text is one which assigns a higher probability to the word that actually occurs

Perplexity

The best language model is one that best predicts an unseen test set

- Gives the highest P(sentence), for most sentences acceptable to humans

Perplexity

The best language model is one that best predicts an unseen test set

- Gives the highest P(sentence), for most sentences acceptable to humans

Perplexity is the inverse probability of the test set, normalized by the number of words

Perplexity

The best language model is one that best predicts an unseen test set

- Gives the highest P(sentence), for most sentences acceptable to humans

$$PPL(\mathbf{w}) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

Perplexity is the inverse probability of the test set, normalized by the number of words

$$PPL(\mathbf{w}) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$PPL(\mathbf{w}) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

$$PPL(\mathbf{w}) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

Chain rule:

$$PPL(\mathbf{w}) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

Chain rule:

$$= \sqrt[N]{\frac{1}{\prod_i P(w_i \mid w_1 \ldots w_{i-1})}}$$

$$PPL(\mathbf{w}) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

Chain rule:

$$= \sqrt[N]{\frac{1}{\prod_i P(w_i \mid w_1 \ldots w_{i-1})}}$$

Applying Markov's assumption for bigrams:

$$PPL(\mathbf{w}) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

Chain rule:

$$= \sqrt[N]{\frac{1}{\prod_i P(w_i \mid w_1 \ldots w_{i-1})}}$$

Applying Markov's assumption for bigrams:

$$= \sqrt[N]{\frac{1}{\prod_i P(w_i \mid w_{i-1})}}$$

$$PPL(\mathbf{w}) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

Minimizing perplexity is the same as maximizing probability

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

Chain rule:

$$= \sqrt[N]{\frac{1}{\prod_i P(w_i \mid w_1 \ldots w_{i-1})}}$$

Applying Markov's assumption for bigrams:

$$= \sqrt[N]{\frac{1}{\prod_i P(w_i \mid w_{i-1})}}$$

Perplexity Example

Perplexity Example

Let’s suppose a sentence of length 50 consisting of random digits

Perplexity Example

Let’s suppose a sentence of length 50 consisting of random digits

$$P(w) = \frac{1}{10}$$

Perplexity Example

Let's suppose a sentence of length 50 consisting of random digits

$$P(w) = \frac{1}{10}$$

What is the perplexity of this sentence according to a model that assigns uniform probability to each digit?

$$PPL(\mathbf{w}) = P(w_1 w_2 \ldots w_N)^{\frac{-1}{N}}$$

Perplexity Example

Let’s suppose a sentence of length 50 consisting of random digits

$$P(w) = \frac{1}{10}$$

What is the perplexity of this sentence according to a model that assigns uniform probability to each digit?

$$PPL(\mathbf{w}) = P(w_1 w_2 \ldots w_N)^{\frac{-1}{N}}$$

$$= (\frac{1}{10}^{50})^{-\frac{1}{50}}$$

Perplexity Example

Let's suppose a sentence of length 50 consisting of random digits

$$P(w) = \frac{1}{10}$$

What is the perplexity of this sentence according to a model that assigns uniform probability to each digit?

$$PPL(\mathbf{w}) = P(w_1 w_2 \ldots w_N)^{\frac{-1}{N}}$$

$$= (\frac{1}{10}^{50})^{-\frac{1}{50}}$$

$$= 10$$

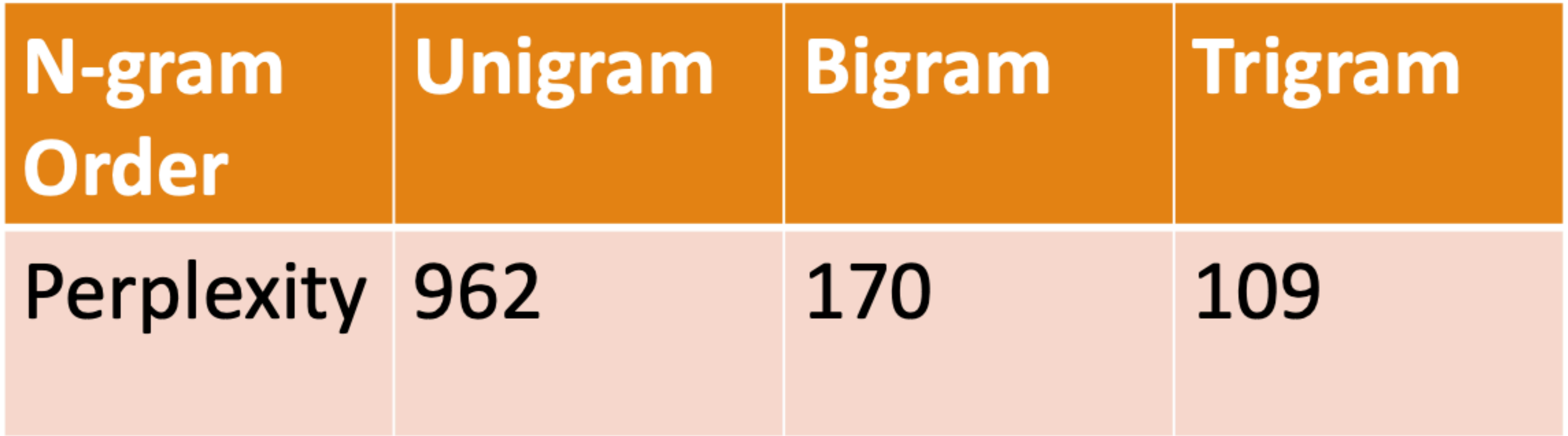
Lower perplexity = better model!

Lower perplexity = better model!

Training 38 million words, test 1.5 million words, from the Wall Street Journal

Lower perplexity = better model!

Training 38 million words, test 1.5 million words, from the Wall Street Journal



N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Lower perplexity = better model!

Training 38 million words, test 1.5 million words, from the Wall Street Journal

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Lower perplexity = better model!

Training 38 million words, test 1.5 million words, from the Wall Street Journal

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

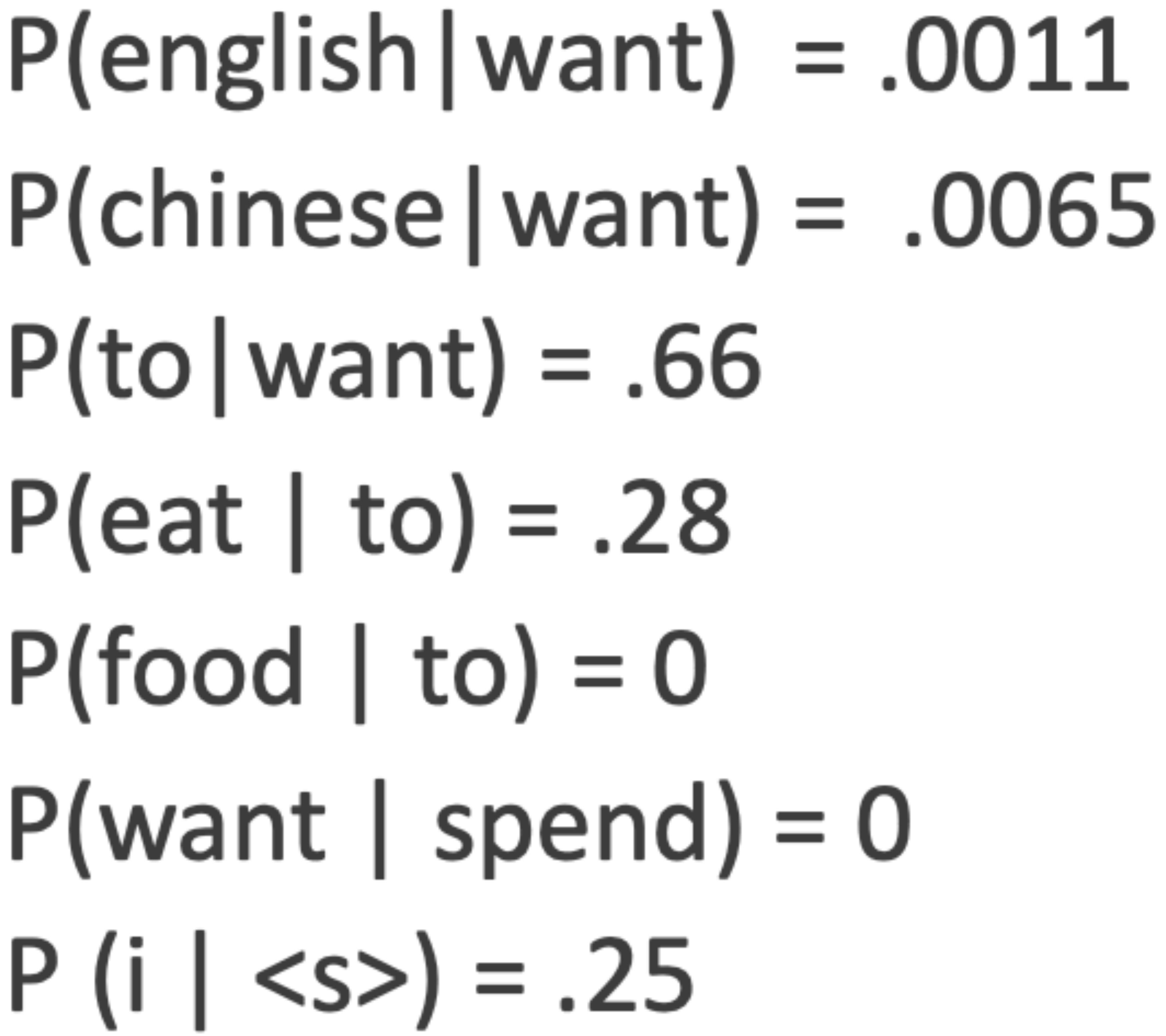
What are the two things that might affect perplexity?

Lecture Outline

1. Announcements + Recap
2. Probabilistic Language Models
3. n-gram Language Models
4. Evaluation and Perplexity
5. Generating from an n-gram Language Model
    i. Zeroes
6. Smoothing

Generating from an n-gram model and Zeros

Recall: BRP



$P(\text{english}|\text{want}) = .0011$

$P(\text{chinese}|\text{want}) = .0065$

$P(\text{to}|\text{want}) = .66$

$P(\text{eat} \mid \text{to}) = .28$

$P(\text{food} \mid \text{to}) = 0$

$P(\text{want} \mid \text{spend}) = 0$

$P(\text{i} \mid \text{<s>}) = .25$

Recall: BRP

P(english|want) = .0011

P(chinese|want) = .0065

P(to|want) = .66

P(eat | to) = .28

P(food | to) = 0

P(want | spend) = 0

P (i | <s>) = .25

How can we generate sentences from this bigram model?

Generating from a bigram model

Generating from a bigram model

- Choose a random bigram (<s>, w) according to its probability

Generating from a bigram model

`<s> I`

- Choose a random bigram (<s>, w) according to its probability

Generating from a bigram model

```
<s> I
    I want
```

- Choose a random bigram (<s>, w) according to its probability
- Now choose a random bigram (w, x) according to its probability

Generating from a bigram model

- 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>

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

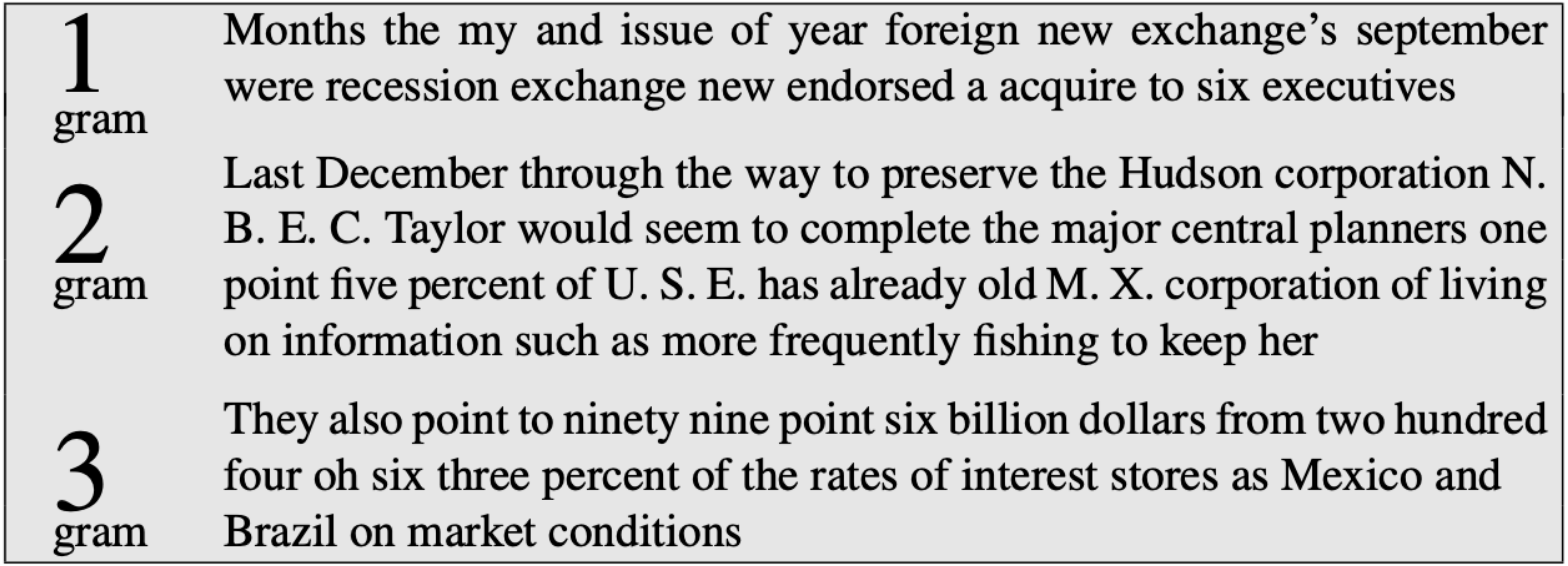
Generating from a bigram model

- 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>
```

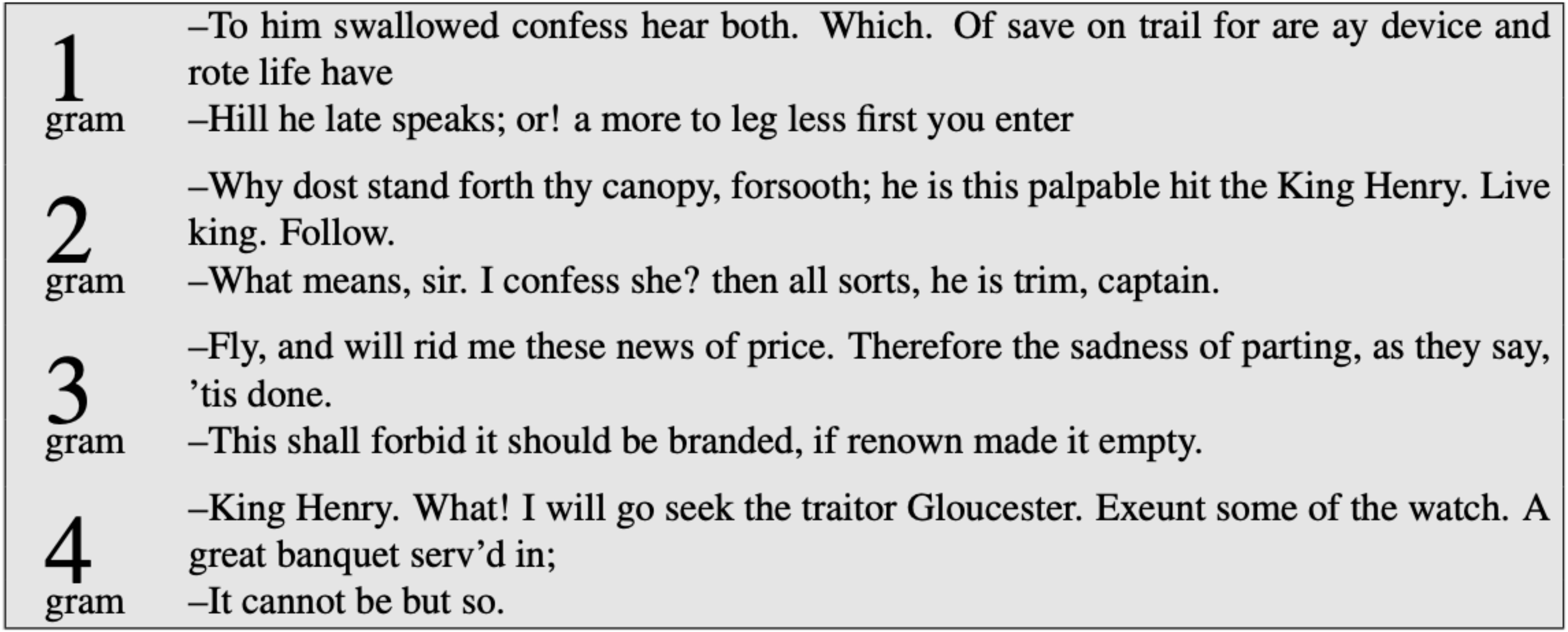
I want to eat Chinese food

The WSJ is no Shakespeare!



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

Shakespearean n-grams



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.

Shakespeare as a corpus

Shakespeare as a corpus

N=884,647 tokens, V=29,066

Shakespeare as a corpus

N=884,647 tokens, V=29,066

Shakespeare produced 300,000 bigram types out of V^2= 844 million possible bigrams

Shakespeare as a corpus

N=884,647 tokens, V=29,066

Shakespeare produced 300,000 bigram types out of V^2= 844 million possible bigrams

So 99.96% of the possible bigrams were never seen (have zero entries in the table)

Shakespeare as a corpus

N=884,647 tokens, V=29,066

Shakespeare produced 300,000 bigram types out of V^2= 844 million possible bigrams

So 99.96% of the possible bigrams were never seen (have zero entries in the table)

4-grams (quadrigrams) are rarer still…

Shakespeare as a corpus

N=884,647 tokens, V=29,066

Shakespeare produced 300,000 bigram types out of V^2= 844 million possible bigrams

So 99.96% of the possible bigrams were never seen (have zero entries in the table)

4-grams (quadrigrams) are rarer still…

Most n-grams are never seen!

Shakespeare as a corpus

N=884,647 tokens, V=29,066

Shakespeare produced 300,000 bigram types out of V^2= 844 million possible bigrams

So 99.96% of the possible bigrams were never seen (have zero entries in the table)

4-grams (quadrigrams) are rarer still…

What's coming out looks like Shakespeare because it is Shakespeare!

Most n-grams are never seen!

So why not just sample from very high order n-gram models? Do we even need GPT-style LLMs?

So why not just sample from very high order n-gram models? Do we even need GPT-style LLMs?

The successes we are seeing here is a phenomena commonly known as overfitting

Overfitting bad!

Overfitting bad!

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

Overfitting bad!

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

- In real life, it often doesn’t

Overfitting bad!

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**!
  - Technical terms for “doing well on the test data” or “doing well on any test data”

Overfitting bad!

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**!
  - Technical terms for “doing well on the test data” or “doing well on any test data”
- One kind of generalization: Zeros!
  - Things that don’t ever occur in the training set
    - But occur in the test set

Training set:

… denied the allegations

… denied the reports

… denied the claims

… denied the request

Training set:

… denied the allegations

… denied the reports

… denied the claims

… denied the request

Test set

… denied the offer

… denied the loan

Training set:

… denied the allegations

… denied the reports

… denied the claims

… denied the request

Test set

… denied the offer

… denied the loan

$P\,(\textbf{offer} \mid \textbf{denied the}) =$

Zeros

Training set:

… denied the allegations

… denied the reports

… denied the claims

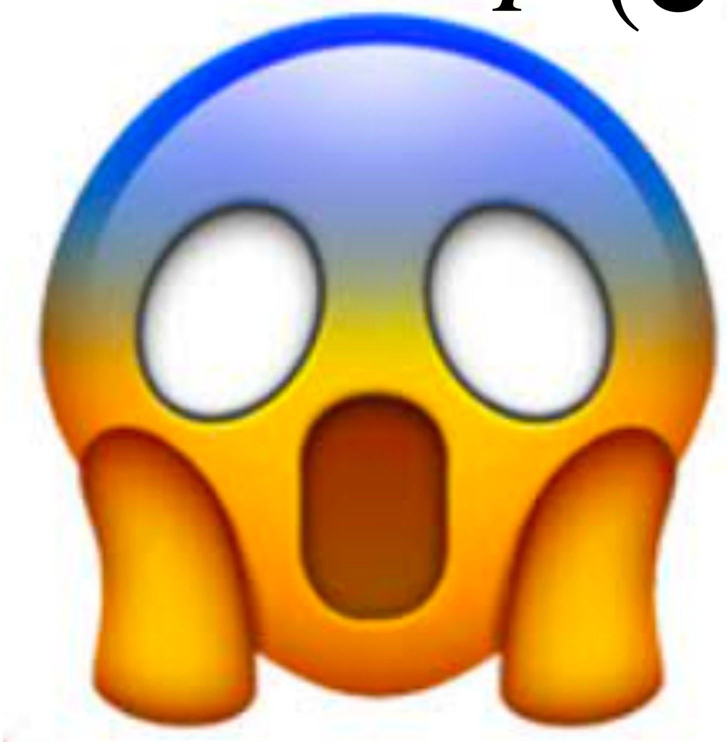
… denied the request

Test set

… denied the offer

… denied the loan

$P\,(\textbf{offer}\,|\,\textbf{denied the}) = \;\; 0$



will assign 0 probability to the test set!

Zeros

Training set:

… denied the allegations

… denied the reports

… denied the claims

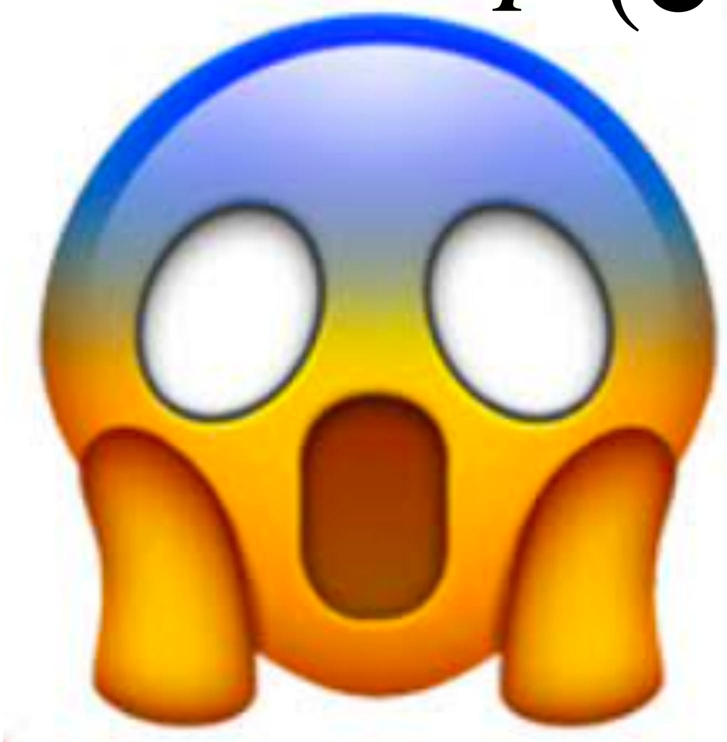
… denied the request

Test set

… denied the offer

… denied the loan

$$P\left(\textbf{offer} \,|\, \textbf{denied the}\right) = 0$$



will assign 0 probability to the test set!

What happens to perplexity??

One solution: the UNK token

One solution: the UNK token

Problem: Word “offer” didn’t appear in the train set…many words like “Swayamdipta” won’t appear in most training sets!

One solution: the UNK token

Problem: Word “offer” didn’t appear in the train set…many words like “Swayamdipta” won’t appear in most training sets!

These are known as **OOV** for “out of vocabulary”, or **unknown tokens**

One solution: the UNK token

A token is a technical term in NLP for what is commonly referred to as a word

Problem: Word “offer” didn’t appear in the train set…many words like “Swayamdipta” won’t appear in most training sets!

These are known as **OOV** for “out of vocabulary”, or **unknown tokens**

One solution: the UNK token

A token is a technical term in NLP for what is commonly referred to as a word

Problem: Word "offer" didn't appear in the train set…many words like "Swayamdipta" won't appear in most training sets!

These are known as **OOV** for "out of vocabulary", or **unknown tokens**

One way to handle OOV tokens is by adding a pseudo-word called <UNK>

One solution: the UNK token

A token is a technical term in NLP for what is commonly referred to as a word

Problem: Word "offer" didn't appear in the train set…many words like "Swayamdipta" won't appear in most training sets!

These are known as **OOV** for "out of vocabulary", or **unknown tokens**

One way to handle OOV tokens is by adding a pseudo-word called <UNK>

We can replace all words that occur fewer than n times in the training set—where n is some small number—by <UNK> and re-estimate the counts and probabilities

One solution: the UNK token

A token is a technical term in NLP for what is commonly referred to as a word

Problem: Word "offer" didn't appear in the train set…many words like "Swayamdipta" won't appear in most training sets!

These are known as **OOV** for "out of vocabulary", or **unknown tokens**

One way to handle OOV tokens is by adding a pseudo-word called <UNK>

We can replace all words that occur fewer than n times in the training set—where n is some small number—by <UNK> and re-estimate the counts and probabilities

When not done carefully, may artificially lower perplexity

One solution: the UNK token

A token is a technical term in NLP for what is commonly referred to as a word

Problem: Word "offer" didn't appear in the train set…many words like "Swayamdipta" won't appear in most training sets!

These are known as **OOV** for "out of vocabulary", or **unknown tokens**

One way to handle OOV tokens is by adding a pseudo-word called <UNK>

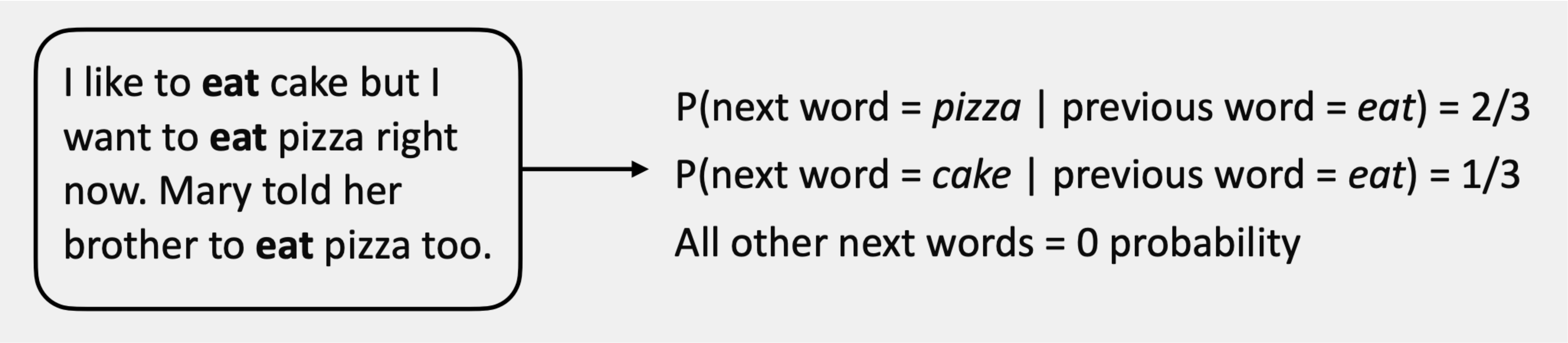
We can replace all words that occur fewer than n times in the training set—where n is some small number—by <UNK> and re-estimate the counts and probabilities

When not done carefully, may artificially lower perplexity

Lecture Outline

1. Announcements + Recap
2. Probabilistic Language Models
3. *n*-gram Language Models
4. Evaluation and Perplexity
5. Generating from an n-gram Language Model
    i. Zeroes
6. Smoothing
    i. Add-one / Laplace
    ii. Interpolation

Intuition for Smoothing



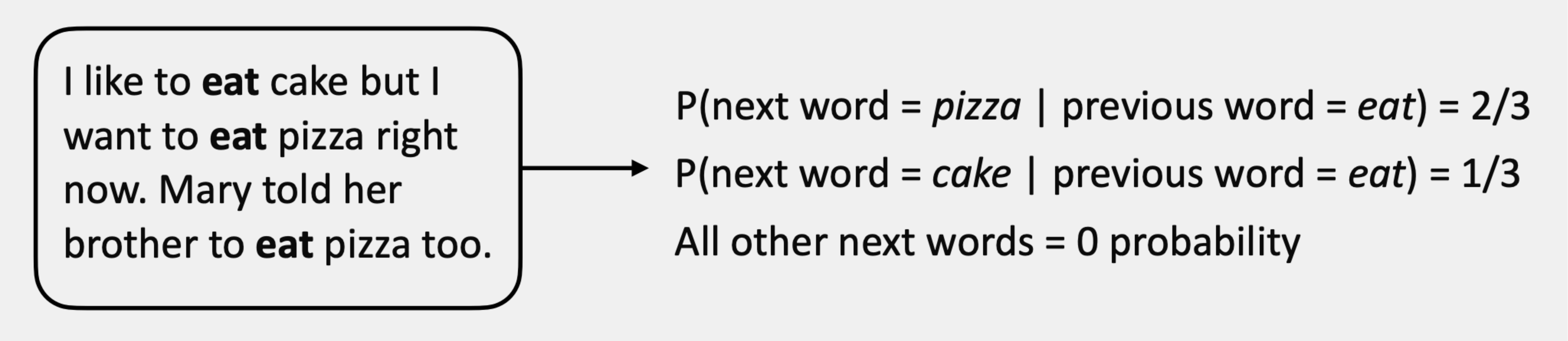
I like to **eat** cake but I want to **eat** pizza right now. Mary told her brother to **eat** pizza too.

P(next word = *pizza* | previous word = *eat*) = 2/3

P(next word = *cake* | previous word = *eat*) = 1/3

All other next words = 0 probability

Intuition for Smoothing



I like to **eat** cake but I want to **eat** pizza right now. Mary told her brother to **eat** pizza too.

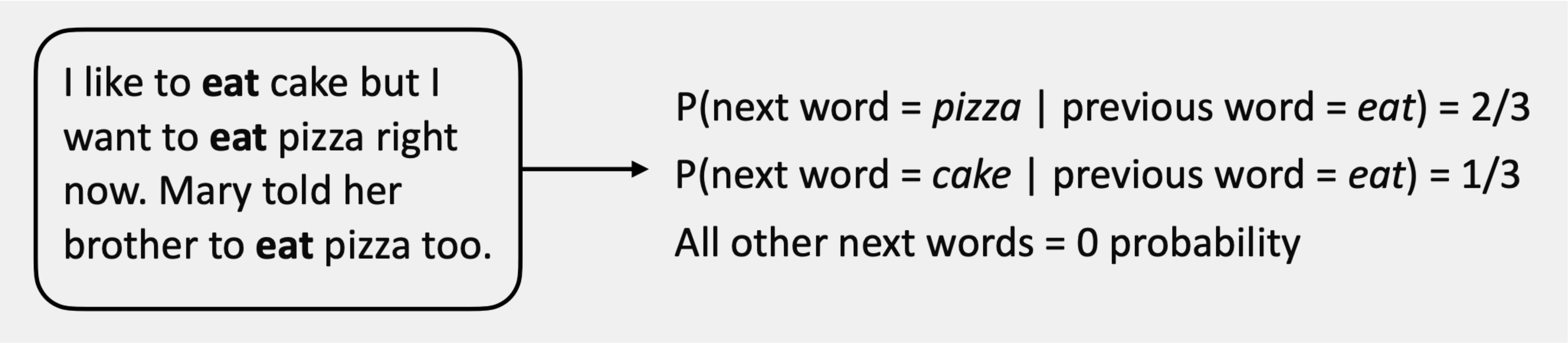
P(next word = *pizza* | previous word = *eat*) = 2/3

P(next word = *cake* | previous word = *eat*) = 1/3

All other next words = 0 probability

- Types: I, like, to, eat, cake, but, want, pizza, right, now, ., Mary, told, her, brother, too

Intuition for Smoothing



I like to **eat** cake but I want to **eat** pizza right now. Mary told her brother to **eat** pizza too.

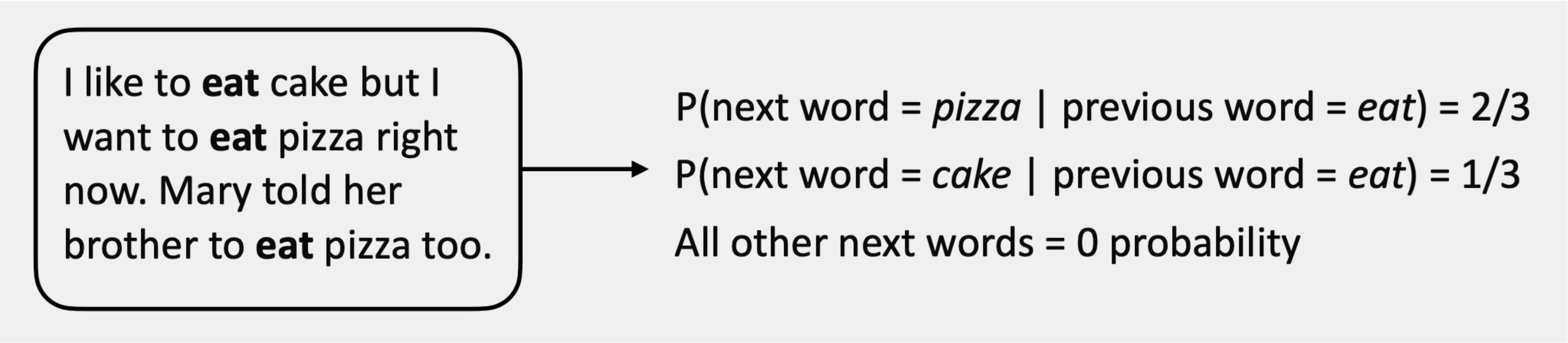
P(next word = *pizza* | previous word = *eat*) = 2/3

P(next word = *cake* | previous word = *eat*) = 1/3

All other next words = 0 probability

- Types: I, like, to, eat, cake, but, want, pizza, right, now, ., Mary, told, her, brother, too
  - $|V| = ?$ $|V_{\text{bigrams}}| = ?$

Intuition for Smoothing



I like to **eat** cake but I want to **eat** pizza right now. Mary told her brother to **eat** pizza too.

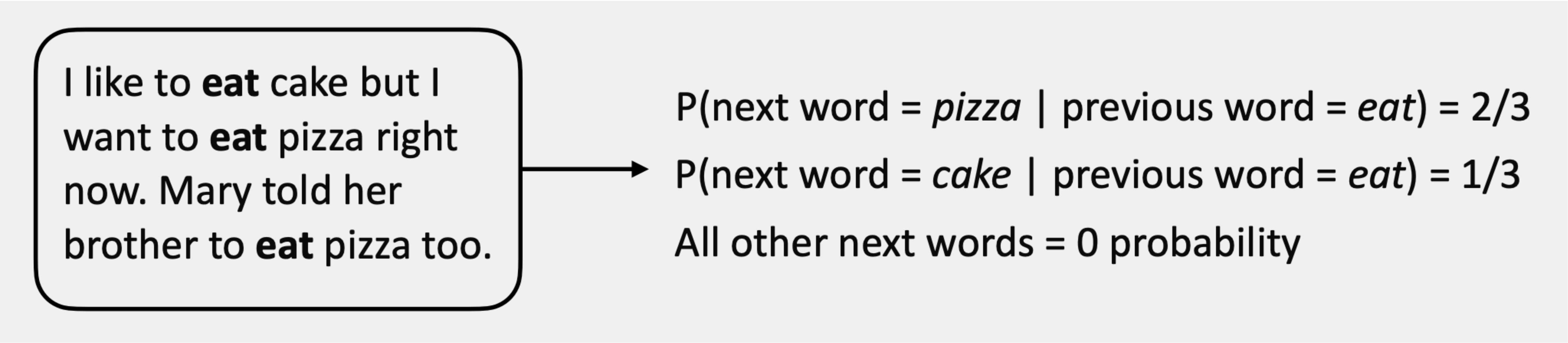
P(next word = *pizza* | previous word = *eat*) = 2/3

P(next word = *cake* | previous word = *eat*) = 1/3

All other next words = 0 probability

- Types: I, like, to, eat, cake, but, want, pizza, right, now, ., Mary, told, her, brother, too
  - $|V| = ?$ $|V_{\text{bigrams}}| = ?$
- All other vocabulary tokens getting 0 probability just doesn't seem right. We want to assign some probability to other words

Intuition for Smoothing



I like to **eat** cake but I want to **eat** pizza right now. Mary told her brother to **eat** pizza too.

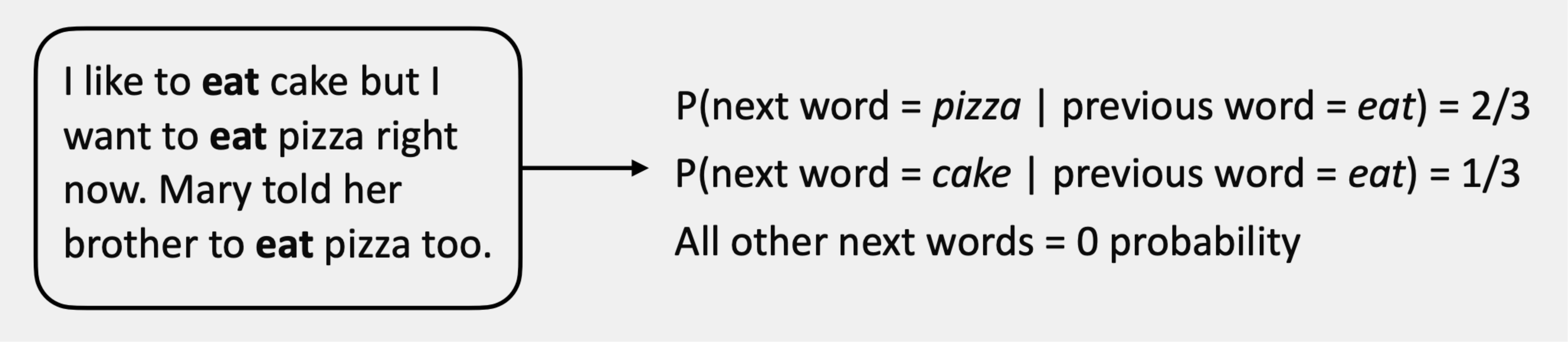
P(next word = *pizza* | previous word = *eat*) = 2/3

P(next word = *cake* | previous word = *eat*) = 1/3

All other next words = 0 probability

- Types: I, like, to, eat, cake, but, want, pizza, right, now, ., Mary, told, her, brother, too
    - $|V| = ?$ $|V_{\text{bigrams}}| = ?$
- All other vocabulary tokens getting 0 probability just doesn't seem right. We want to assign some probability to other words
- We want to **smooth the distribution from our counts**

Intuition for Smoothing



I like to **eat** cake but I want to **eat** pizza right now. Mary told her brother to **eat** pizza too.

P(next word = *pizza* | previous word = *eat*) = 2/3

P(next word = *cake* | previous word = *eat*) = 1/3

All other next words = 0 probability

- Types: I, like, to, eat, cake, but, want, pizza, right, now, ., Mary, told, her, brother, too
  - $|V| = ?$ $|V_{\text{bigrams}}| = ?$
- All other vocabulary tokens getting 0 probability just doesn't seem right. We want to assign some probability to other words
- We want to **smooth the distribution from our counts**

What does a count distribution look like?

Zipf's Law

Zipf, G. K. (1949). Human behavior and the principle of least effort.

Zipf's Law

The distribution over words resembles that of a power law:

Zipf, G. K. (1949). Human behavior and the principle of least effort.

Zipf’s Law

The distribution over words resembles that of a power law:

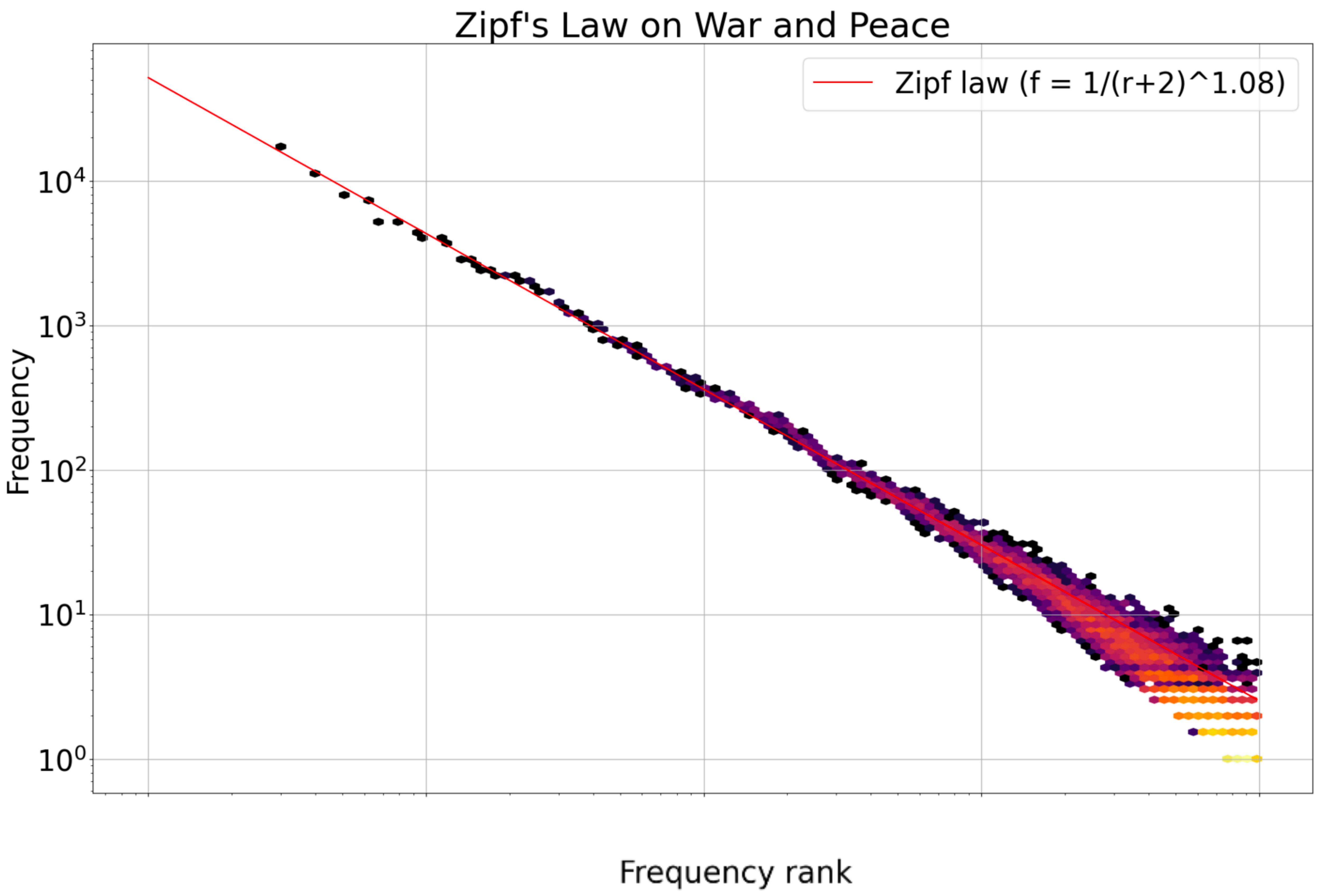
- there will be a few words that are very frequent, and a long tail of words that are rare

Zipf, G. K. (1949). Human behavior and the principle of least effort.

Zipf’s Law

The distribution over words resembles that of a power law:

- there will be a few words that are very frequent, and a long tail of words that are rare
- $freq_w(r) \approx r^{-s}$, where s is a constant

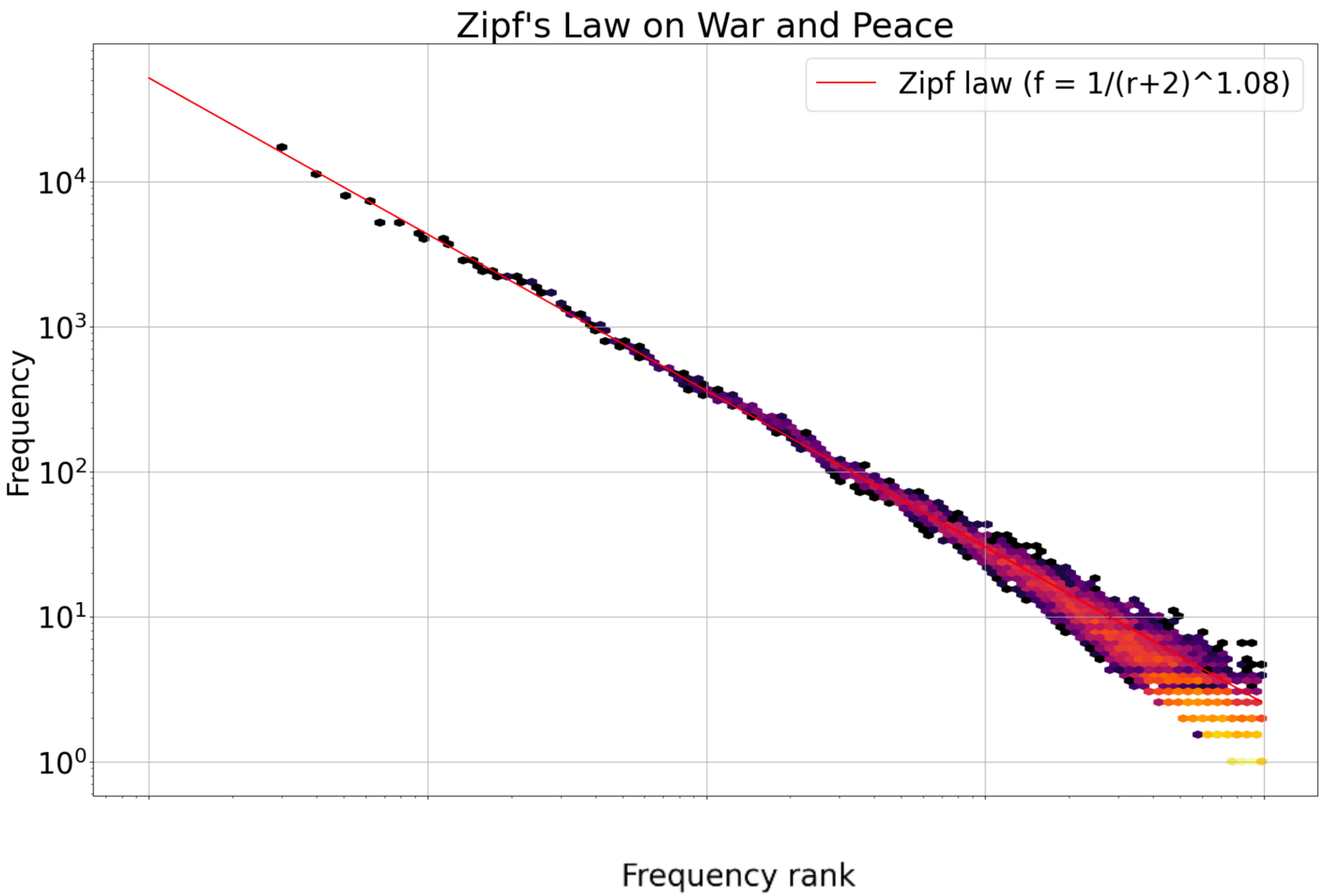


Zipf, G. K. (1949). Human behavior and the principle of least effort.

Zipf’s Law

The distribution over words resembles that of a power law:

- there will be a few words that are very frequent, and a long tail of words that are rare
- $freq_w(r) \approx r^{-s}$, where s is a constant



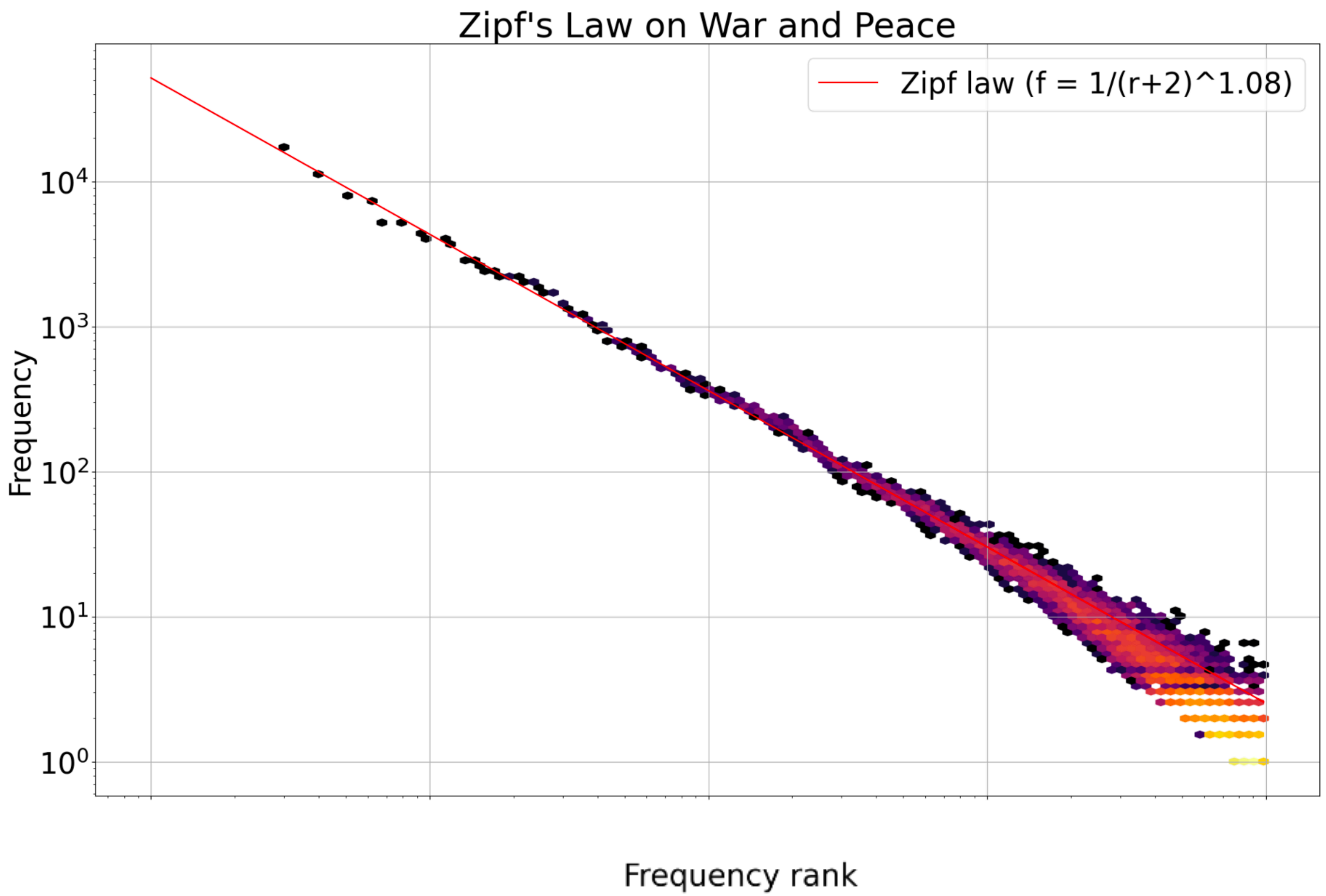
Zipf, G. K. (1949). Human behavior and the principle of least effort.

Zipf’s Law

The distribution over words resembles that of a power law:

- there will be a few words that are very frequent, and a long tail of words that are rare
- $freq_w(r) \approx r^{-s}$, where s is a constant

NLP algorithms must be especially robust to observations that do not occur or rarely occur in the training data



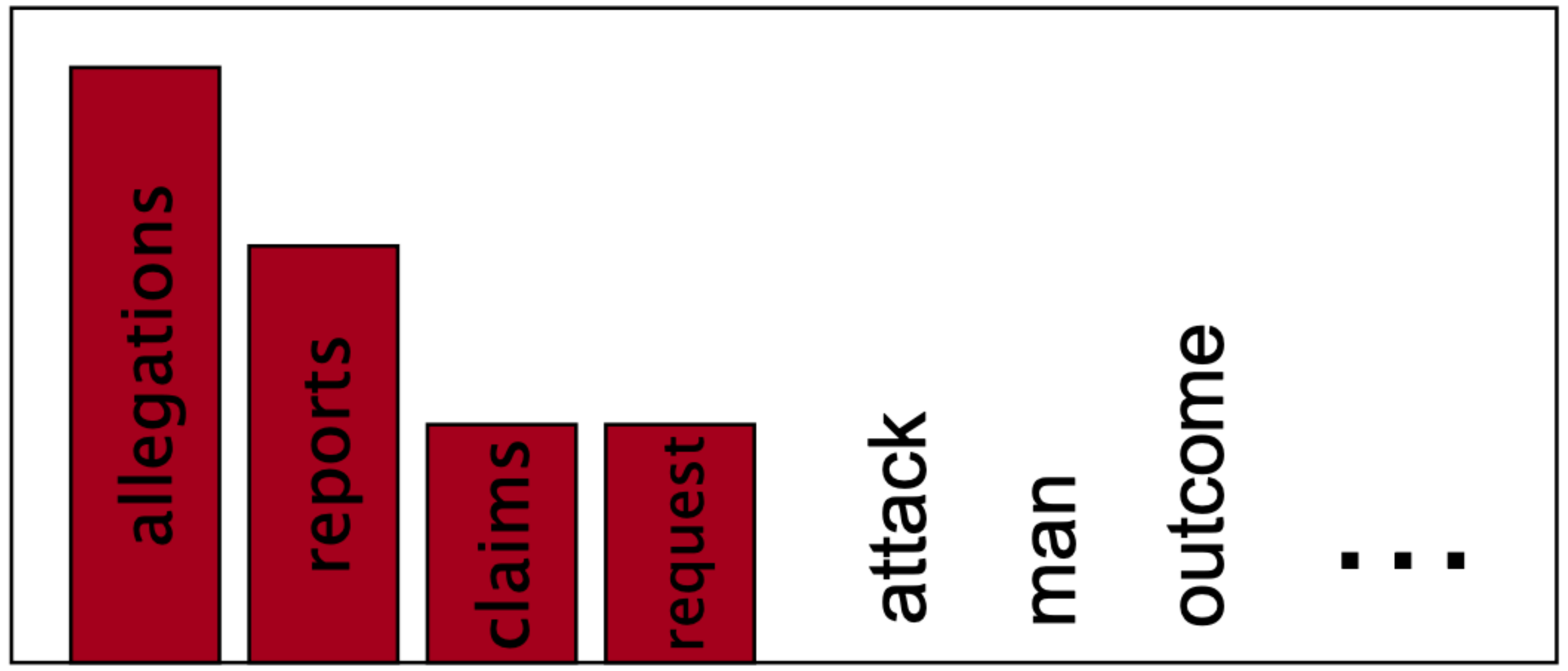
Zipf, G. K. (1949). Human behavior and the principle of least effort.

Smoothing ~ Massaging Probability Masses

Smoothing ~ Massaging Probability Masses

When we have sparse statistics: $Count(w \mid \text{denied the})$

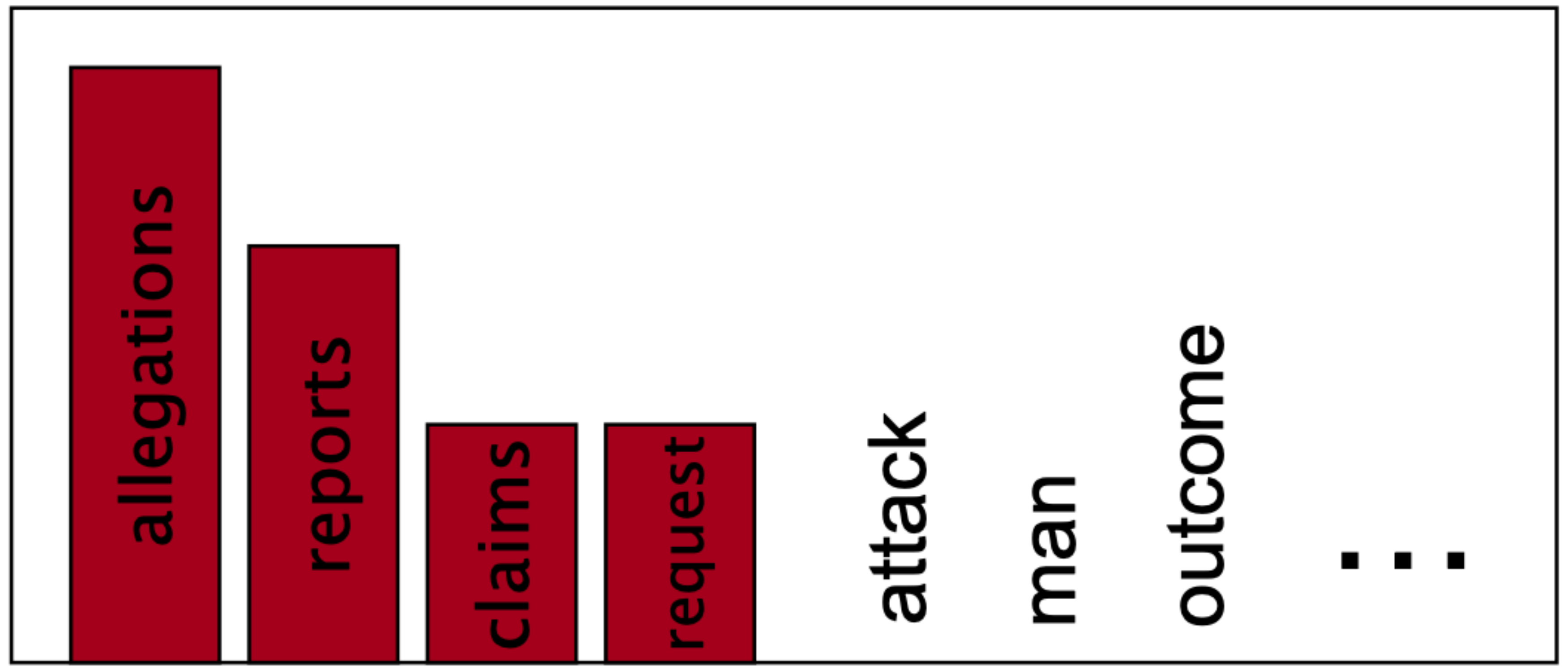
3 allegations
2 reports
1 claims
1 request
7 total



Smoothing ~ Massaging Probability Masses

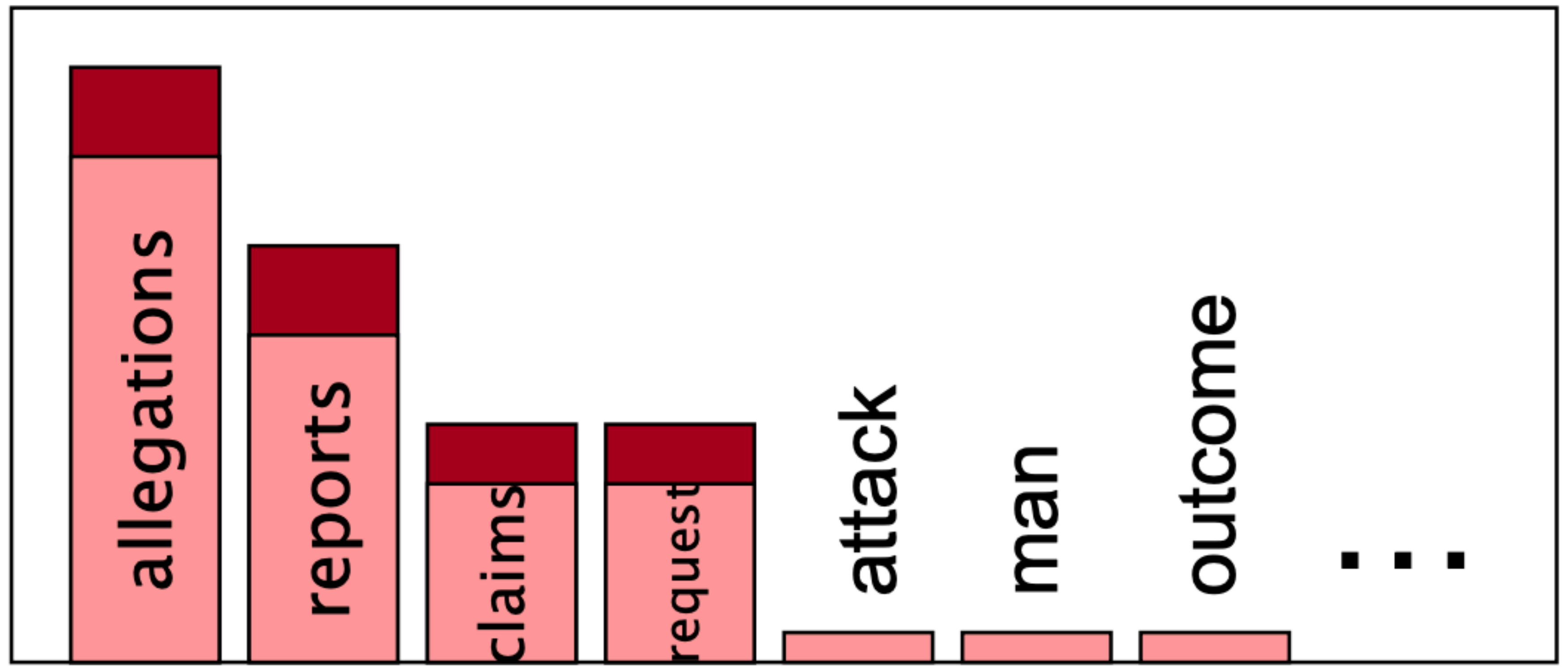
When we have sparse statistics: $Count(w \mid \text{denied the})$

3 allegations
2 reports
1 claims
1 request
7 total



Steal probability mass to generalize better: $Count(w \mid \text{denied the})$

2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other
7 total


allegations
reports
claims
request
attack
man
outcome
…