Department of Linguistics and Translation
香港城市大學
City University of Hong Kong

Computational Linguistics LT3233

Jixing Li

Lecture 3: N-gram Models

Slides adapted from Dan Jurafsky, Jordan Boyd-Graber

Lecture plan

- What are n-gram models?
- How to evaluate n-gram models?
- Short break (15 mins)
- Hands-on exercises

The Shannon game

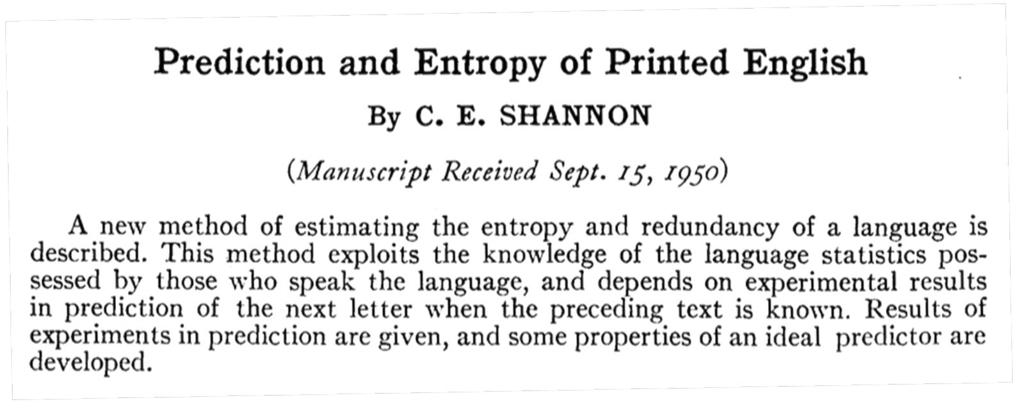
In the Shannon game, I show you a bit of English, and you need to tell me what the next character is going to be

I have a sad story to tell you

It may hurt your feelings a bit

Last night I walked into my bathroom

And stepped in a pile of shaving cream



Prediction and Entropy of Printed English

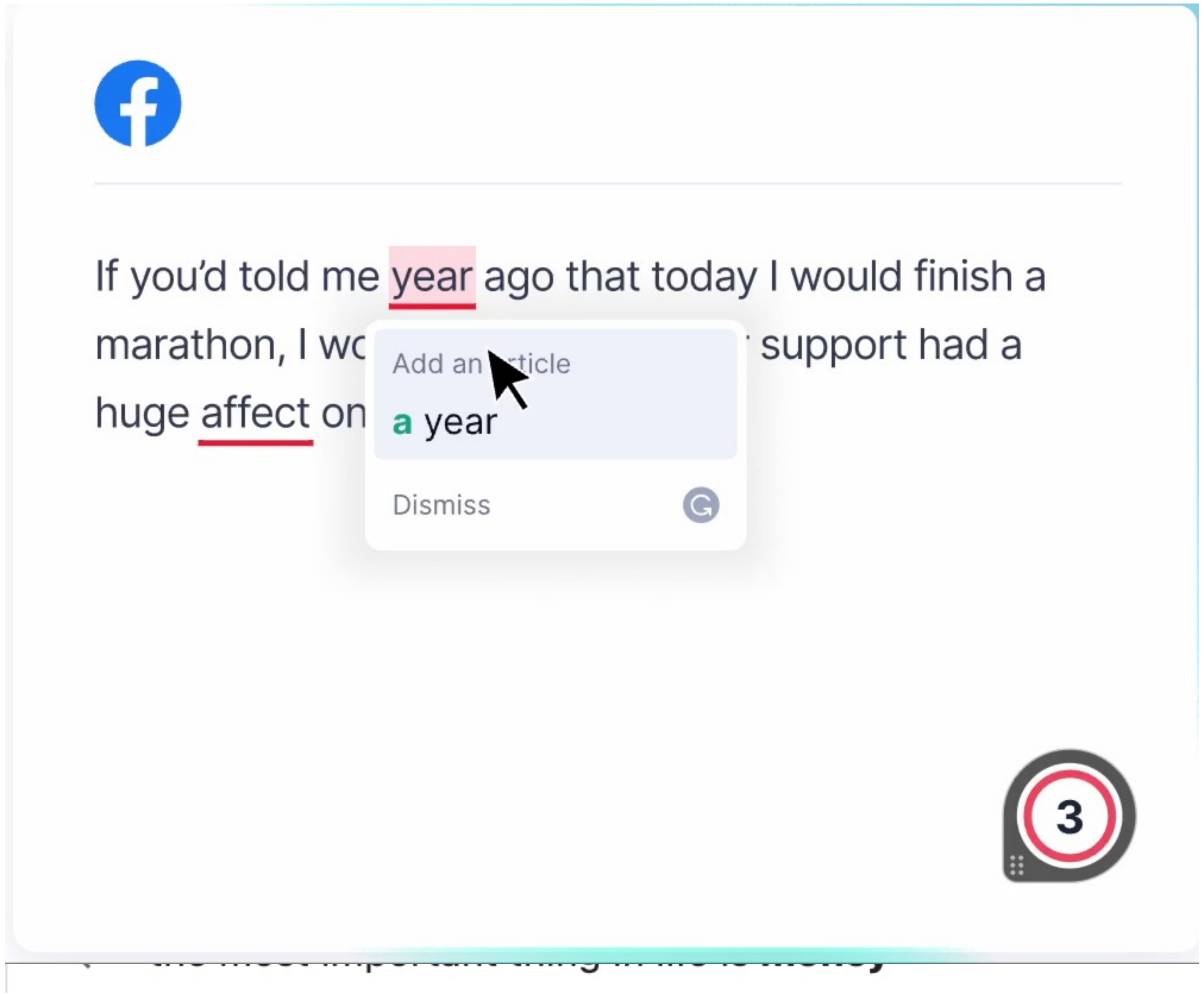
By C. E. SHANNON

(*Manuscript Received Sept. 15, 1950*)

A new method of estimating the entropy and redundancy of a language is described. This method exploits the knowledge of the language statistics possessed by those who speak the language, and depends on experimental results in prediction of the next letter when the preceding text is known. Results of experiments in prediction are given, and some properties of an ideal predictor are developed.

Claude Shannon (1916-2001)

Computers play the Shannon Game



If you'd told me year ago that today I would finish a marathon, I wo support had a huge affect on

Add an article

a year

Dismiss

3

Probabilistic language models

Goal: Compute the probability of a sequence of words

P(*a pile of shaving cream***)** **or** **P(***cream*|*a pile of shaving***)**

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

→ Language Model (LM)

- **Spell correction:**
  P(*a huge* ***effect*** *on***) > P(***a huge* ***affect*** *on***)**
- **Speech recognition:**
  P(*I saw a van***) > P(***eyes awe of an***)**
- **Summarization, question-answering, etc.**

How to compute P(*W*)

P(*a pile of shaving cream***)**

The Chain Rule: $P(A,B) = P(A)P(B|A)$

More variables:

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

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

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

Chain rule applied to compute joint probability of words in sentence

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

$$\begin{aligned}\mathbf{P}(\textit{a pile of shaving cream}) = {} & \mathbf{P}(\textit{a}) \times \\ & \mathbf{P}(\textit{pile}|\textit{a}) \times \\ & \mathbf{P}(\textit{of}|\textit{a pile}) \times \\ & \mathbf{P}(\textit{shaving}|\textit{a pile of}) \times \\ & \mathbf{P}(\textit{cream}|\textit{a pile of shaving})\end{aligned}$$

$\mathbf{P}(\textit{I walked into my bathroom}) = \,?$

How to estimate these probabilities

Divide and count?

P(*a pile of shaving cream***) = P(***a***) x P(***pile*|*a***) x P(***of*|*a pile***) x P(***shaving*|*a pile of***) x P(***cream*|*a pile of shaving***)**

$$= \frac{\text{Count}(a)}{\#\ \text{number of words}} \times \frac{\text{Count}(a\ pile)}{\text{Count}(a)} \times \frac{\text{Count}(a\ pile\ of)}{\text{Count}(a\ pile)} \times$$

Brown Corpus $\frac{21881}{1161192}$ $\frac{7}{21881}$ $\frac{6}{7}$

$$\frac{\text{Count}(a\ pile\ of\ shaving)}{\text{Count}(a\ pile\ of\ )} \times \frac{\text{Count}(a\ pile\ of\ shaving\ cream)}{\text{Count}(a\ pile\ of\ shaving)}$$

$\frac{0}{7}$ $\frac{0}{0}$ **Sparse data problem!**

Markov assumption

Simplifying assumption:

$\mathbf{P}(\textit{cream}|\textit{a pile of shaving}) \approx \mathbf{P}(\textit{cream}|\textit{shaving})$

Or maybe:

$\mathbf{P}(\textit{cream}|\textit{a pile of shaving}) \approx \mathbf{P}(\textit{cream}|\textit{of shaving})$

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

Andrey Markov
(1856-1922)

Simplest case: Unigram model

$$\mathbf{P}(w_1, w_2, w_3, \ldots w_n) \approx \prod_{i=1}^{n} \mathbf{P}(w_i)$$

$$\mathbf{P}(\textit{a pile of shaving cream}) \approx$$
$$\mathbf{P}(\textit{a})\mathbf{P}(\textit{pile})\mathbf{P}(\textit{of})\mathbf{P}(\textit{shaving})\mathbf{P}(\textit{cream})$$

Bigram model

Condition on the previous words:

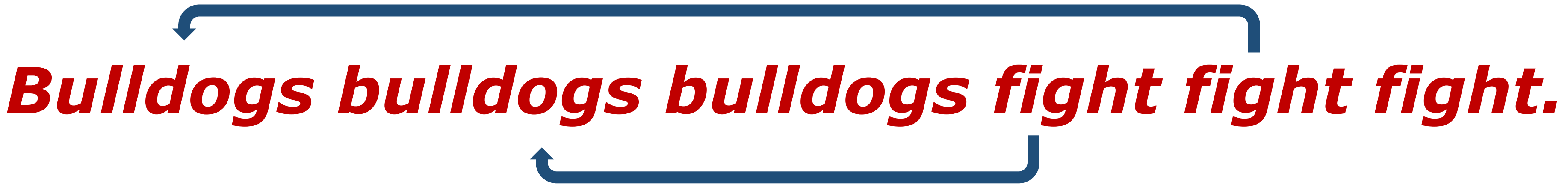
$$\mathbf{P}(w_1, w_2, w_3, \ldots w_n) \approx \prod_{i=1}^{n} \mathbf{P}(w_i | w_{i-1})$$

$$\mathbf{P}(\textit{a pile of shaving cream}) \approx$$
$$\mathbf{P}(\textit{a})\mathbf{P}(\textit{pile}|\textit{a})\mathbf{P}(\textit{of}|\textit{pile})\mathbf{P}(\textit{shaving}|\textit{of})\mathbf{P}(\textit{cream}|\textit{shaving})$$

N-gram models

We can extend to trigrams, 4-grams, 5-grams...

In general, this is an insufficient model of language



Bulldogs bulldogs bulldogs fight fight fight.

The bulldogs that are fought by the bulldogs that the bulldogs fight, fight.

→ **long-distance dependencies**

Chomsky: language is not Markovian

Estimating bigram probabilities

The Maximum Likelihood Estimate

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

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

$\mathbf{P}(I|<s>)=\frac{2}{3}=0.67$ $\mathbf{P}(am|I)= \frac{2}{3} =0.67$ $\mathbf{P}(Sam|am)= \frac{1}{2} =0.5$

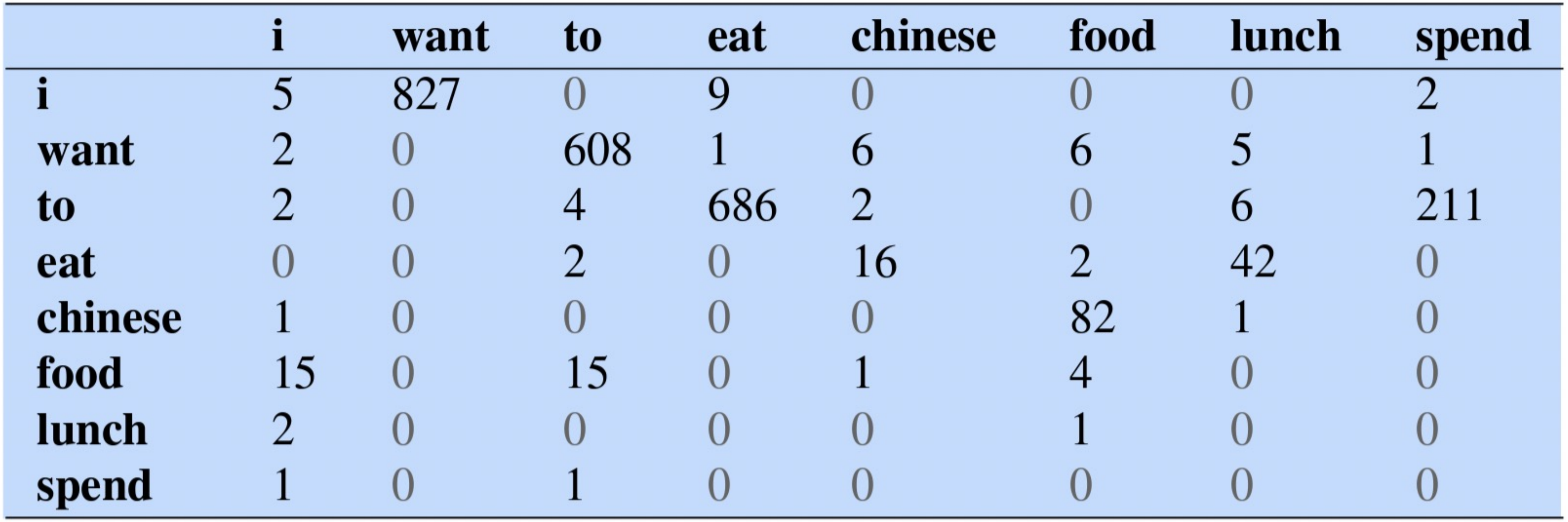
$\mathbf{P}(</s>|Sam)= \frac{1}{2} =0.5$ $\mathbf{P}(Sam|<s>)= \frac{1}{3} =0.33$ $\mathbf{P}(do|I)= \frac{1}{3} =0.33$

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*

Raw bigram counts

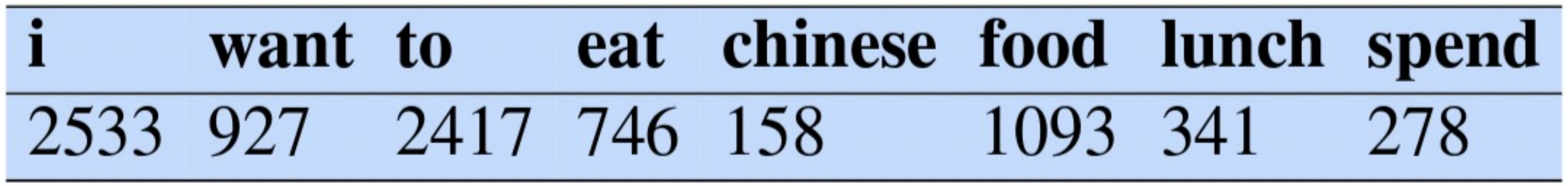
Out of 9222 sentences



	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

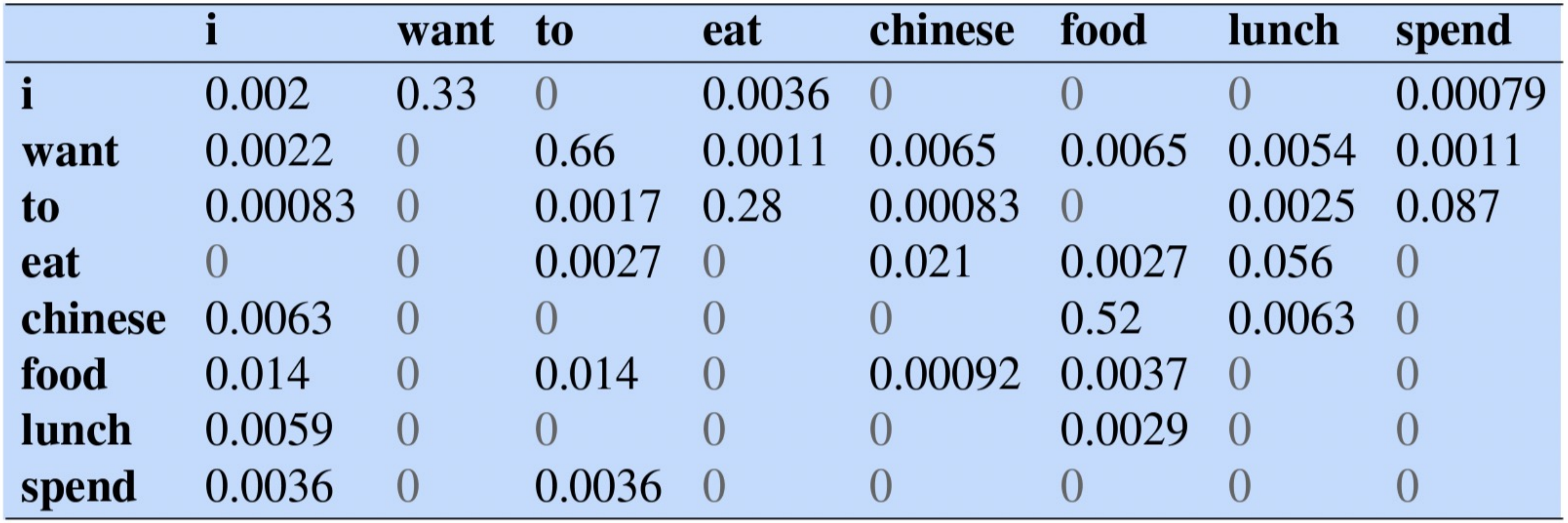
Raw bigram probabilities

Unigram count



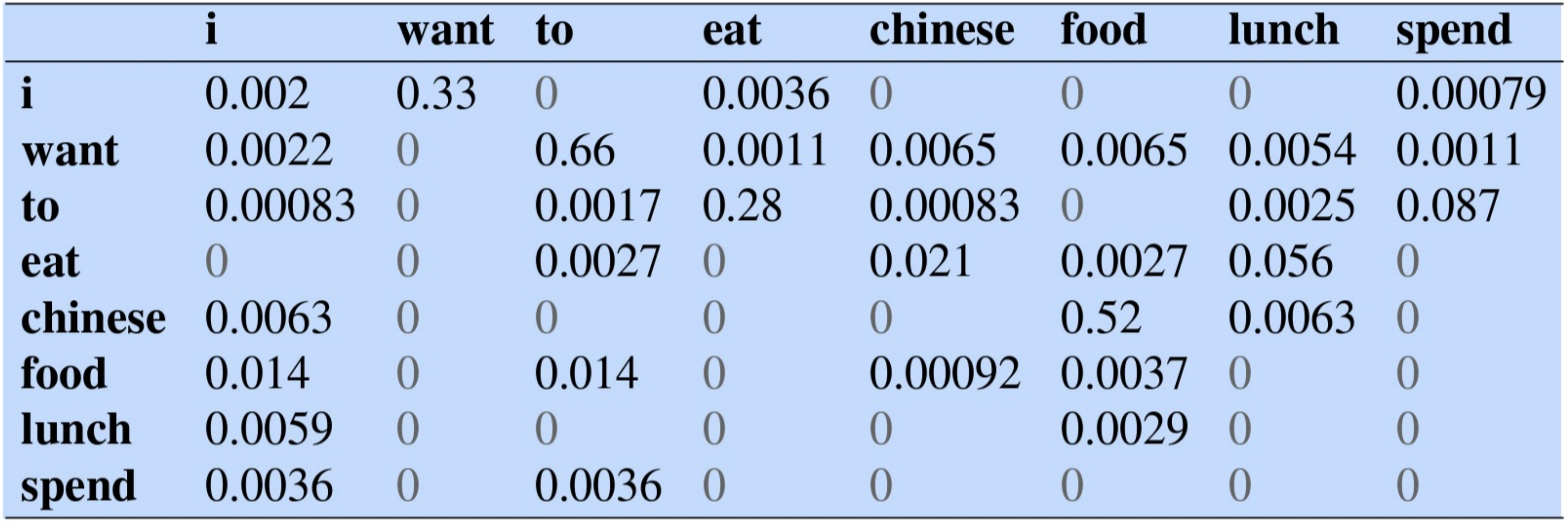
i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Normalized by unigrams



	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

Bigram estimates of sentence probabilities



	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

P(<s> *i eat chinese food* </s>**)** = **P(***i*|<s>**)** x **P(***eat*|*i***)** x **P(***chinese*|*eat***)** x **P(***food*|*chinese***)** x **P(**</s>|food**)**

= 0.25 x ? x ? x ? x 0.68 **= 0.00000668**

What kinds of knowledge?

P(*english*|*want*) = .0011

P(*chinese*|*want*) = .0065

P(*to*|*want*) = .66

P(*want*|*spend*) = 0

P(*food*|*to*) = 0

five steps to food safety

world knowledge

grammar

contingent zero

Practical issues

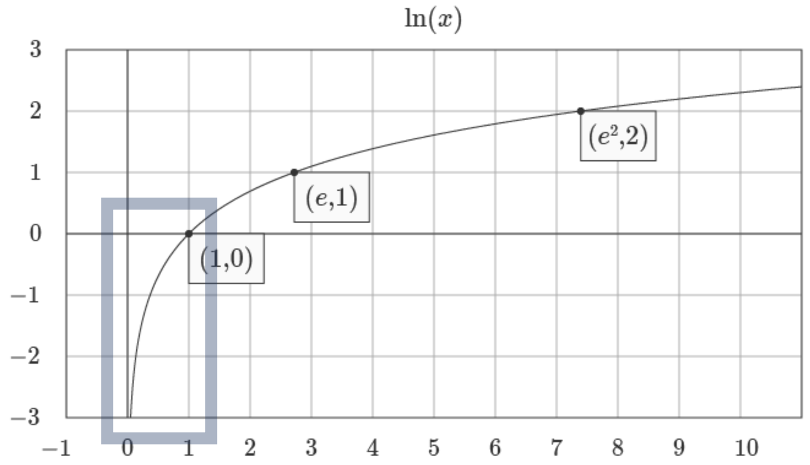
We do everything in log space

- Avoid arithmetic underflow

  P(<s> *i eat chinese food* </s>**)** = **P(***i*|<*s*>**)** x **P(***eat*|*i***)** x **P(***chinese*|*eat***)** x **P(***food*|*chinese***)** x **P(**</s>|food**)**

  = 0.25 x 0.0036 x 0.021 x 0.52 x 0.68

  = 0.00000668



- Adding is faster than multiplying

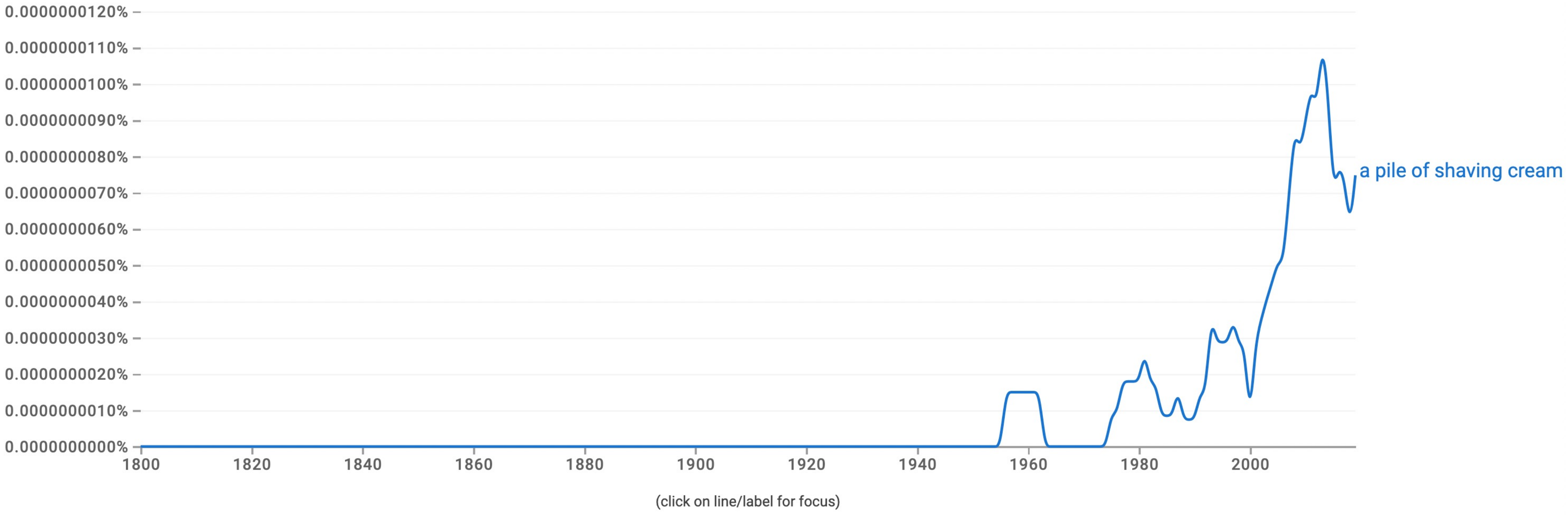
  log(P(<s> *i eat chinese food* </s>**))** = **log(P(***i*|<*s*>**)** x **P(***eat*|*i***)** x **P(***chinese*|*eat***)** x **P(***food*|*chinese***)** x **P(**</s>|food**))**

  = log(0.25**)** + **log(**0.0036**)** + **log(**0.021**)** + **log(**0.52**)** + **log(**0.68**)**

  = -11.92

Google n-gram

https://books.google.com/ngrams/



Evaluation: How good is our model?

The **Shannon Game** again:

I always order pizza with cheese and ▮

mushrooms	**0.08**
pepperoni	**0.1**
anchovies	**0.01**
…	
fried rice	**0.0001**
…	
and	**1e-100**

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

Extrinsic evaluation of N-gram models

Best evaluation for comparing models A and B:

- Put each model in a task
  - spelling corrector, speech recognizer, machine translation system
- Run the task, get an accuracy for A and for B
  - How many misspelled words corrected properly
  - How many words translated correctly
- Compare accuracy for A and B

Intrinsic evaluation of N-gram models

- Extrinsic evaluation is time-consuming
    - can take days or weeks

- So we sometimes use intrinsic evaluation: **perplexity**

- Bad approximation
    - unless the test data looks just like the training data
    - So generally only useful in pilot experiments

Perplexity

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

$$\mathbf{PP}(\textit{a pile of shaving cream})$$

$$= \sqrt[5]{\frac{1}{\mathbf{P}(\textit{a pile of shaving cream})}}$$

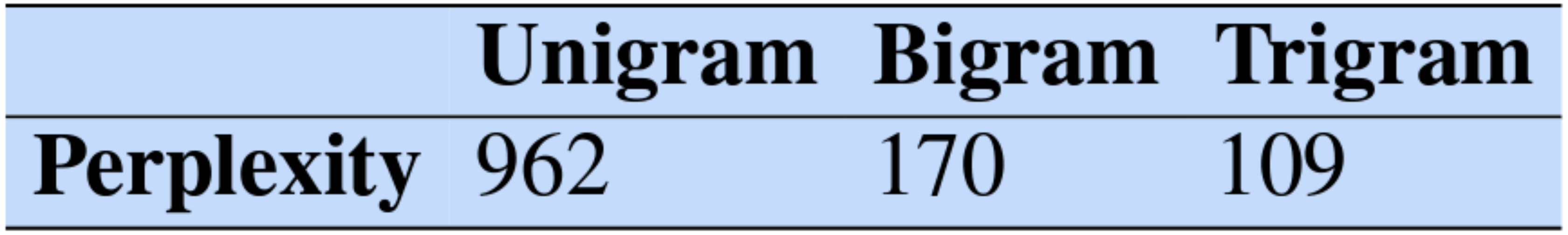
$$\approx \sqrt[5]{\frac{1}{\mathbf{P}(a)\mathbf{P}(pile|a)\mathbf{P}(of|pile)\mathbf{P}(shaving|of)\mathbf{P}(cream|shaving)}}$$

$$\mathbf{PP}(W) = \mathbf{P}(w_1, w_2, w_3, \ldots w_n)^{-\frac{1}{n}} \quad = \sqrt[n]{\frac{1}{\mathbf{P}(w_1, w_2, w_3, \ldots w_n)}}$$

$$= \sqrt[n]{\prod_{i=1}^{n} \frac{1}{\mathbf{P}(w_1, w_2, w_3, \ldots w_n)}} \quad \approx \sqrt[n]{\prod_{\boldsymbol{i-1}}^{\boldsymbol{n}} \frac{1}{\mathbf{P}(w_i|w_{i-1})}}$$

Lower perplexity = better model

Training 38 million words, test 1.5 million words,WSJ



	Unigram	Bigram	Trigram
Perplexity	962	170	109

To do

- **Leave a comment for Lecture 3!**
  https://jixing-li.github.io/comments.html
- Do HW2
- Optional reading: **NTLK** Ch2:2.4; **SLP** Ch3.1-3.2.