DATA ANALYSIS AND VISUALIZATION

INSTRUCTOR: UMME AMMARAH

NATURAL LANGUAGE PROCESSING (NLP)

LANGUAGE MODELING

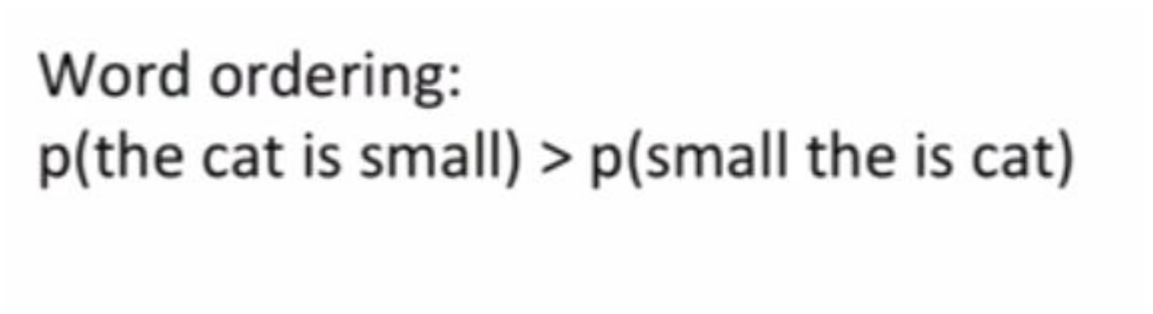
N-GRAM

- An N-gram is a sequence of N tokens (or words).

- Example: “I love reading blogs about data science on Towards Science.”
- 1-gram (unigram): “I”, “love”, “reading”, “blogs”, “about”, “data”, “science”, “on”, “Towards”, “Science”.
- 2-gram (bigram): “I love”, “love reading”, or “Towards Science”
- 3-gram (trigram): “I love reading”, “about data science” or “on Towards Science”.

LANGUAGE MODEL

- A language model learns to predict the probability of a sequence of words.



Word ordering:
p(the cat is small) > p(small the is cat)

APPLICATIONS

- Speech recognition
- Machine translation
- Part-of-speech tagging
- Parsing
- Optical Character Recognition
- Handwriting recognition
- Information retrieval

TYPES

- **Statistical Language Models**
- These models use traditional statistical techniques like N-grams, Hidden Markov Models (HMM) and certain linguistic rules to learn the probability distribution of words

- **Neural Language Models**
- They use different kinds of Neural Networks to model language

PROBABILISTIC LANGUAGE MODELS

- Today's goal: assign a probability to a sentence
  - Machine Translation:
    - P(**high** winds tonite) > P(**large** winds tonite)
  - Spell Correction
    - The office is about fifteen **minuets** from my house
      - P(about fifteen **minutes** from) > P(about fifteen **minuets** from)
  - Speech Recognition
    - P(I saw a van) >> P(eyes awe of an)
  - + Summarization, question-answering, etc., etc.!!

Why?

PROBABILISTIC LANGUAGE MODELING

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

  $P(W) = P(w_1,w_2,w_3,w_4,w_5\ldots w_n)$

- Related task: probability of an upcoming word:

  $P(w_5|w_1,w_2,w_3,w_4)$

- A model that computes either of these:

  $P(W)$ or $P(w_n|w_1,w_2\ldots w_{n-1})$ is called a **language model**.

- Better: **the grammar** But **language model** or **LM** is standard

HOW TO COMPUTE P(W)

- How to compute this joint probability:
  - P(its, water, is, so, transparent, that)
- Intuition: let’s rely on the Chain Rule of Probability

REMINDER: THE CHAIN RULE

- Recall the definition of conditional probabilities

  $\mathbf{p(B|A) = P(A,B)/P(A)}$ Rewriting: $\mathbf{P(A,B) = P(A)P(B|A)}$

- More variables:

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

- The Chain Rule in General

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

THE CHAIN RULE APPLIED TO COMPUTE JOINT PROBABILITY OF WORDS IN SENTENCE

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

P("its water is so transparent") =

P(its) × P(water|its) × P(is|its water)

× P(so|its water is) × P(transparent|its water is so)

HOW TO ESTIMATE THESE PROBABILITIES

- Could we just count and divide?

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

- No! Too many possible sentences!
- We’ll never see enough data for estimating these

MARKOV ASSUMPTION

- Simplifying assumption:

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

- Or maybe

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

MARKOV ASSUMPTION

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

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

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

SIMPLEST CASE: UNIGRAM MODEL

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

BIGRAM MODEL

- Condition on the previous word:

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

N-GRAM MODELS

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

- In general this is an insufficient model of language
  - because language has **long-distance dependencies**:

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

- But we can often get away with N-gram models

ESTIMATING N-GRAM PROBABILITIES

ESTIMATING BIGRAM PROBABILITIES

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

- The Maximum Likelihood Estimate

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

AN EXAMPLE

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

Laplace:
P(wi|wi-1) = c(wi-1,wi)+1 /c(wi-1) +V

v= 12

$\overset{1}{\text{<s>}}\ \overset{2}{\text{I}}\ \overset{3}{\text{am}}\ \overset{4}{\text{Sam}}\ \overset{5}{\text{</s>}}$

<s> Sam I am </s>

$\text{<s> I}\ \overset{6}{\text{do}}\ \overset{7}{\text{not}}\ \overset{8}{\text{like}}\ \overset{9}{\text{green}}\ \overset{10}{\text{eggs}}\ \overset{11}{\text{and}}\ \overset{12}{\text{ham}}\ \text{</s>}$

P(I|<S>) =2+1 / 3+12 =3/15 = 1/5 = 0.2

0+1/1+12 =1/13>0

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

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

MORE EXAMPLES: BERKELEY RESTAURANT PROJECT SENTENCES

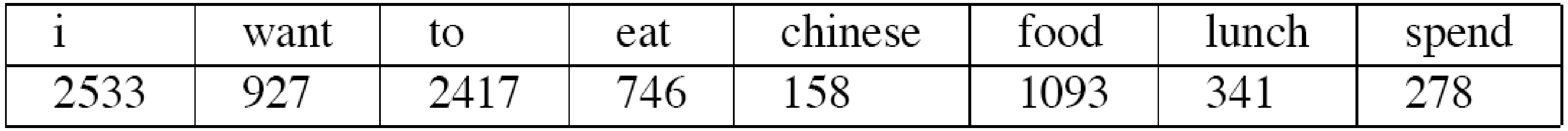
- can you tell me about any good cantonese restaurants close by
- mid priced thai food is what i’m looking for
- tell me about chez panisse
- can you give me a listing of the kinds of food that are available
- i’m looking for a good place to eat breakfast
- when is caffe venezia open during the day

RAW BIGRAM COUNTS

	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

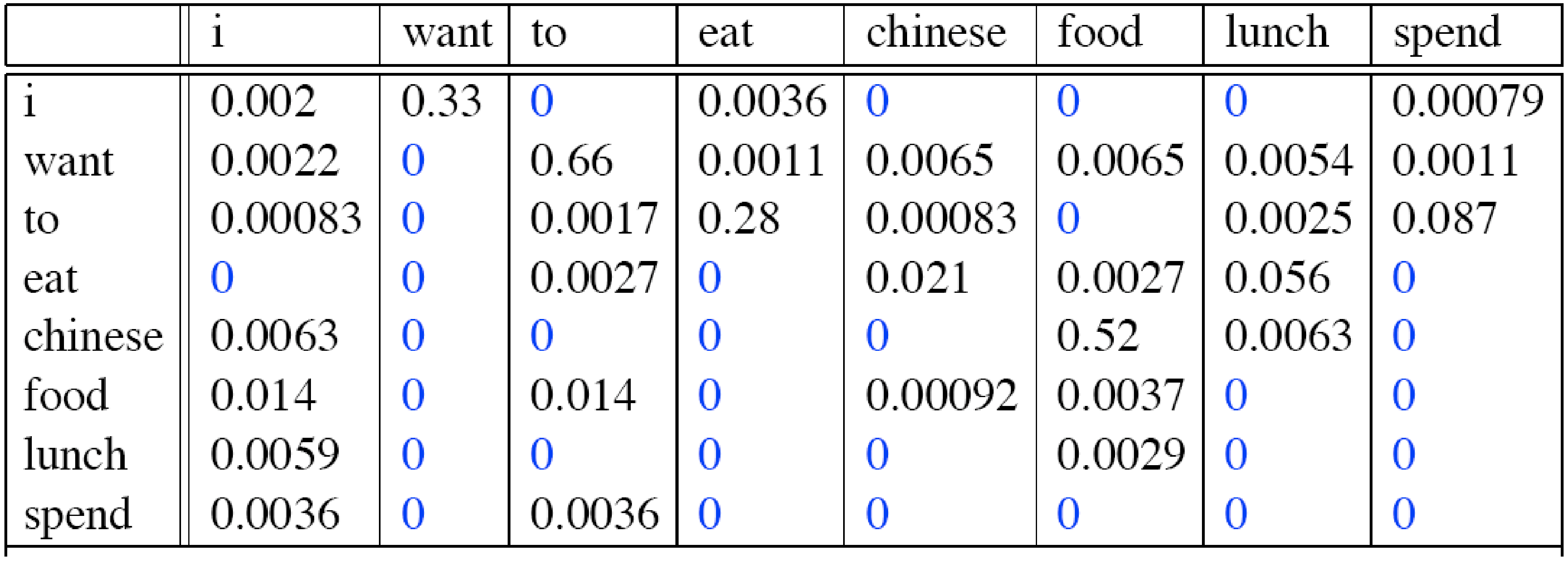
RAW BIGRAM PROBABILITIES

- Normalize by unigrams:



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

- Result:



	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

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

P(I|<s>)

× P(want|I)

× P(english|want)

× P(food|english)

× P(</s>|food)

= .000031

PRACTICAL ISSUES

- We do everything in log space
  - Avoid underflow
  - (also adding is faster than multiplying)

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

EVALUATION AND PERPLEXITY

EVALUATION: HOW GOOD IS OUR MODEL?

- Does our language model prefer good sentences to bad ones?
  - Assign higher probability to "real" or "frequently observed" sentences
    - Than "ungrammatical" or "rarely observed" sentences?
- 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.

EXTRINSIC EVALUATION OF N-GRAM MODELS

- 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 words translated correctly
  - Compare accuracy for A and B

DIFFICULTY OF EXTRINSIC EVALUATION OF N-GRAM MODELS

- Time-consuming; can take days or weeks
- So
  - Sometimes use **intrinsic** evaluation: **perplexity**
  - Bad approximation
    - unless the test data looks **just** like the training data
    - So **generally only useful in pilot experiments**
  - But is helpful to think about.

INTUITION OF PERPLEXITY

- The **Shannon Game**:

  I always order pizza with cheese and ____

  - How well can we predict the next word?

  The 33rd President of the US was ____

  I saw a ____

  - Unigrams are terrible at this game. (Why?)

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

mushrooms 0.1

pepperoni 0.1

anchovies 0.01

….

fried rice 0.0001

….

and 1e-100

PERPLEXITY

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

- Gives the highest P(sentence)

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

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

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

Chain rule:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \ldots w_{i-1})}}$$

For bigrams:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}$$

Minimizing perplexity is the same as maximizing probability

PERPLEXITY AS BRANCHING FACTOR

- Let’s suppose a sentence consisting of random digits
- What is the perplexity of this sentence according to a model that assign P=1/10 to each digit?

$$\begin{aligned} \mathrm{PP}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\ &= (\frac{1}{10}^{N})^{-\frac{1}{N}} \\ &= \frac{1}{10}^{-1} \\ &= 10 \end{aligned}$$

LOWER PERPLEXITY = BETTER MODEL

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

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
```

APPROXIMATING 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.

ZEROS

- Training set:

  … denied the allegations

  … denied the reports

  … denied the claims

  … denied the request

  P("offer" | denied the) = 0

- Test set

  … denied the offer

  … denied the loan

ZERO PROBABILITY BIGRAMS

- Bigrams with zero probability
  - mean that we will assign 0 probability to the test set!
- And hence we cannot compute perplexity (can’t divide by 0)!

UNKNOWN WORDS

- How to handle words that our model had never seen before?
- It depends on type of system:
  - Closed Vocabulary System

unk

  - Open Vocabulary System

CLOSED VOCABULARY SYSTEM

- Language tasks in which this can’t happen because we know all the words that can occur, no unknown words

OPEN VOCABULARY SYSTEM

- In other cases we have to deal with words we haven't seen before, called unknown words or out of vocabulary (OOV) words.
- Percentage of OOV words that appear in test set is called OOV rate.
- We add <UNK> word and model these potential words to it.
- 2 ways to deal with model with <UNK> word

FIRST APPROACH

- Turn the problem back into closed vocabulary by choosing a fixed vocabulary in advance.
- Convert in the training set any word that is not in this set (any OOV word) to the unknown token <UNK> in text normalization step.
- Estimate probabilities for <UNK> from its counts.

SECOND APPROACH

- We don’t have prior vocabulary in advance.
- Replace the words in training set by <UNK> based on their frequency.
- Proceed to train the language model as before treating <UNK> like a regular word.

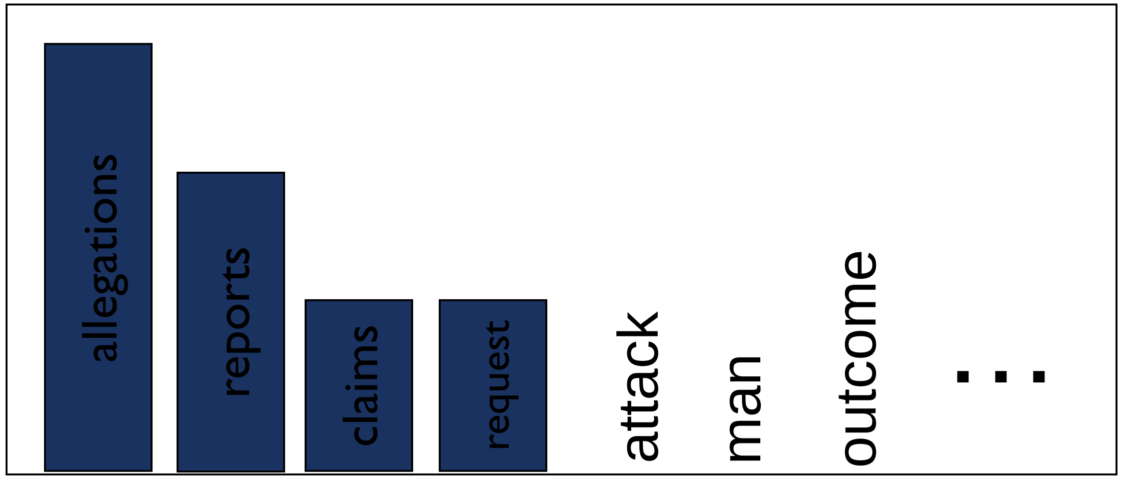
SMOOTHING: ADD-ONE (LAPLACE) SMOOTHING

THE INTUITION OF SMOOTHING (FROM DAN KLEIN)

- When we have sparse statistics:

P(w | denied the)

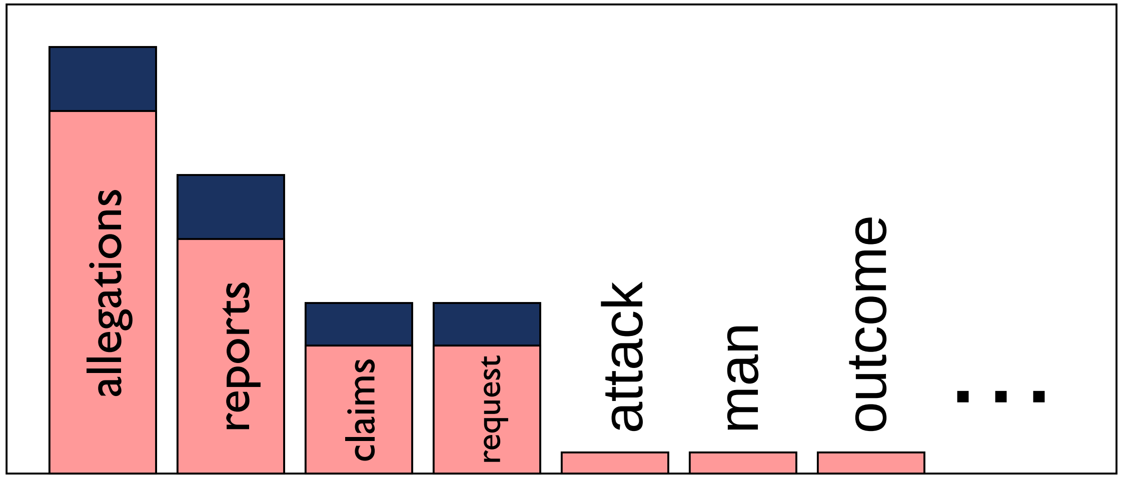
3	allegations
2	reports
1	claims
1	request
7	total



- Steal probability mass to generalize better

P(w | denied the)

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



ADD-ONE ESTIMATION

- Also called Laplace smoothing
- Pretend we saw each word one more time than we did
- Just add one to all the counts!

- MLE estimate:

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

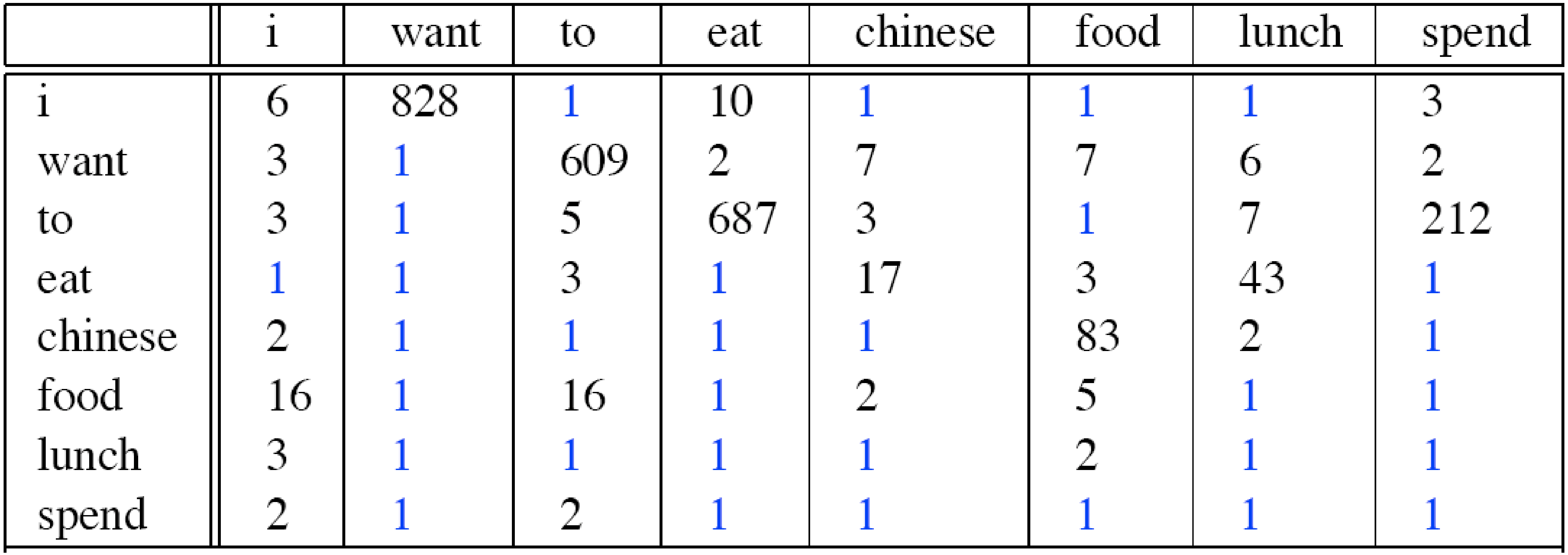
- Add-1 estimate:

$$P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$$

MAXIMUM LIKELIHOOD ESTIMATES

- The maximum likelihood estimate
  - of some parameter of a model M from a training set T
  - maximizes the likelihood of the training set T given the model M
- Suppose the word “bagel” occurs 400 times in a corpus of a million words
- What is the probability that a random word from some other text will be “bagel”?
- MLE estimate is 400/1,000,000 = .0004
- This may be a bad estimate for some other corpus
  - But it is the **estimate** that makes it **most likely** that “bagel” will occur 400 times in a million word corpus.

BERKELEY RESTAURANT CORPUS: 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

LAPLACE-SMOOTHED BIGRAMS

$$P^*(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

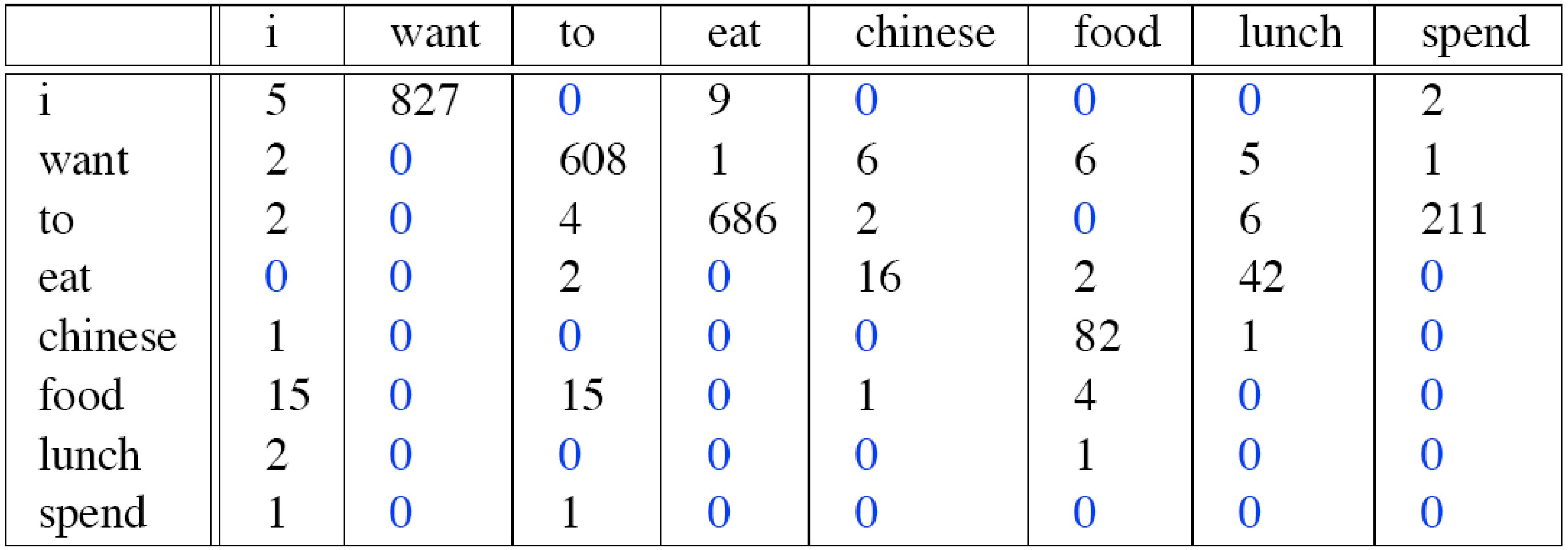
	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

RECONSTITUTED COUNTS

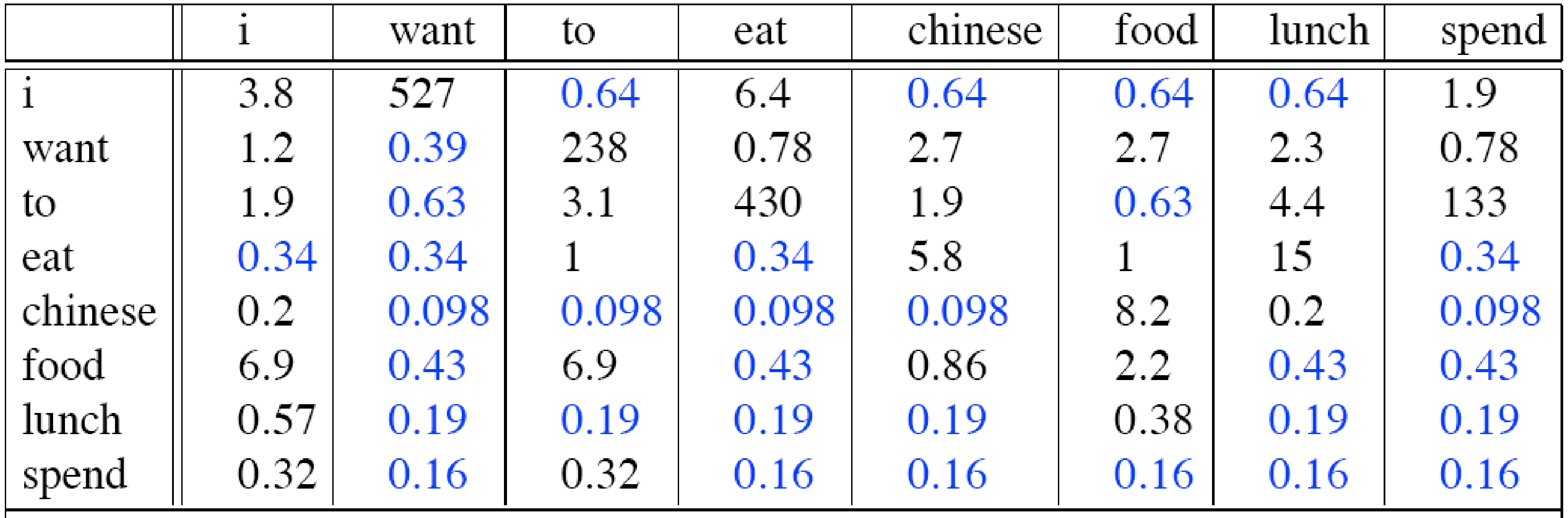
$$c^*(w_{n-1}w_n) = \frac{[C(w_{n-1}w_n)+1] \times C(w_{n-1})}{C(w_{n-1})+V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

COMPARE WITH RAW BIGRAM COUNTS



	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



	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

ADD-1 ESTIMATION IS A BLUNT INSTRUMENT

- So add-1 isn’t used for N-grams:
- But add-1 is used to smooth other NLP models
  - For text classification
  - In domains where the number of zeros isn’t so huge.