

N-Gram Language Models

CS 485, Fall 2024
Applications of Natural Language Processing

Brendan O'Connor
College of Information and Computer Sciences
University of Massachusetts Amherst

[slides from SLP3 and Mohit Iyyer]

goal: assign probability to a piece of text

- Goal #1: assign probability to a piece of text
 - translation:
 - $P(i \text{ flew to the movies}) <<<< P(i \text{ went to the movies})$
 - speech recognition:
 - $P(i \text{ saw a van}) >>> P(\text{eyes awe of an})$
 - text classification (NB):
 - $P(i \text{ am so mad!!} \mid [\text{author is happy}]) < P(i \text{ am so mad!!} \mid [\text{author is not happy}])$
- Goal #2: generate text from this defined probability distribution

Probabilistic Language Modeling

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

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

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

How to compute $P(W)$

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

Reminder: The Chain Rule

- Recall the definition of conditional probabilities

$$P(B|A) = P(A,B)/P(A) \quad \text{Rewriting: } P(A,B) = P(A)P(B|A)$$

- 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

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

LM as next-word prob

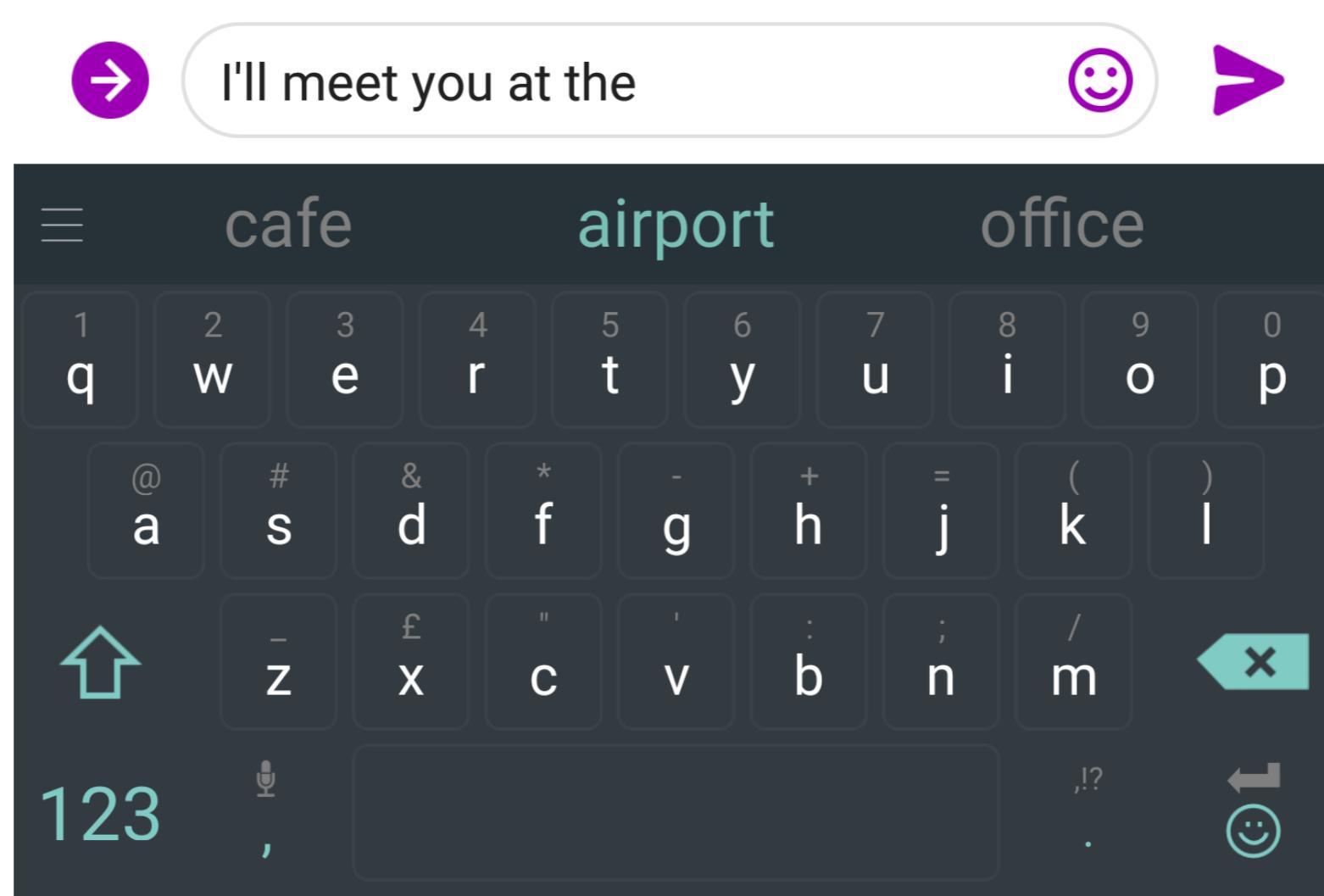
The Chain Rule applied to compute joint probability of words in sentence

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

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

let's try one step!

LM as next-word prob



How to estimate these probabilities

- Could we just count and divide?

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

How to estimate these probabilities

- Could we just count and divide?

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

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

Markov Assumption

- Simplifying assumption:



Andrei Markov (1856~1922)

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

- Or maybe

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

Markov Assumption

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

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

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

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

Approximating Shakespeare

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have
2 gram	–Hill he late speaks; or! a more to leg less first you enter –Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.
3 gram	–What means, sir. I confess she? then all sorts, he is trim, captain. –Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.
4 gram	–This shall forbid it should be branded, if renown made it empty. –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.

N-gram models

- Can extend n-grams to higher n...
- N-gram models are surprisingly useful; state of the art for >50 years!
- But this is an insufficient model of language
 - Long-distance dependencies
 - Language is compositional
 - Generalize effects beyond distinct words

Estimating bigram probabilities

- The Maximum Likelihood Estimate (MLE)
 - relative frequency based on the empirical counts on a training set

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

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

c – count

An example

$$P(w_i | w_{i-1}) = \frac{\underset{\text{MLE}}{C(w_{i-1}, w_i)}}{C(w_{i-1})} \begin{array}{l} \text{<s> I am Sam </s>} \\ \text{<s> Sam I am </s>} \\ \text{<s> I do not like green eggs and ham </s>} \end{array}$$

$$P(\text{I} | \text{<s>}) = \frac{2}{3} = .67$$

$$P(\text{Sam} | \text{<s>}) = ???$$

$$P(\text{</s>} | \text{Sam}) = \frac{1}{2} = 0.5$$

$$P(\text{Sam} | \text{am}) = ???$$

An example

$$P(w_i | w_{i-1}) = \frac{\underset{\text{MLE}}{C(w_{i-1}, w_i)}}{C(w_{i-1})} \begin{array}{l} \text{<s> I am Sam </s>} \\ \text{<s> Sam I am </s>} \\ \text{<s> I do not like green eggs and ham </s>} \end{array}$$

$$P(\text{I} | \text{<s>}) = \frac{2}{3} = .67$$

$$P(\text{</s>} | \text{Sam}) = \frac{1}{2} = 0.5$$

$$P(\text{Sam} | \text{<s>}) = \frac{1}{3} = .33$$

$$P(\text{Sam} | \text{am}) = \frac{1}{2} = .5$$

$$P(\text{am} | \text{I}) = \frac{2}{3} = .67$$

$$P(\text{do} | \text{I}) = \frac{1}{3} = .33$$

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

- 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

$P(< s > | \text{I want english food } < /s >) =$

$P(\text{I} | < s >)$

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

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

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

$\times P(< /s > | \text{food})$

$= .000031$

these probabilities get super tiny when we have longer inputs w/ more infrequent words... how can we get around this?

What kinds of knowledge?

- $P(\text{english} \mid \text{want}) = .0011$ about the world
- $P(\text{chinese} \mid \text{want}) = .0065$
- $P(\text{to} \mid \text{want}) = .66$ grammar – infinitive verb
- $P(\text{eat} \mid \text{to}) = .28$
- $P(\text{food} \mid \text{to}) = 0$???
- $P(\text{want} \mid \text{spend}) = 0$ grammar
- $P(\text{i} \mid \langle s \rangle) = .25$

Evaluation

- Does the model give higher probability to real text?
 - ...in the test set (*why?*)

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 _____*
 - 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
 - compute per word log likelihood
(M words, m test sentence s_i)

mushrooms 0.1
pepperoni 0.1
anchovies 0.01
....
fried rice 0.0001
....
and 1e-100



Claude Shannon
(1916~2001)

$$l = \frac{1}{M} \sum_{i=1}^m \log p(s_i)$$

Perplexity

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

- Gives the highest $P(\text{sentence})$

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

$$\begin{aligned} PP(W) &= P(w_1 w_2 \dots w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_N)}} \end{aligned}$$

Chain rule:

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

For bigrams:

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

Minimizing perplexity is the same as maximizing probability

Perplexity and log-likelihood

Lower perplexity = better model

- Training 38 million words, test 1.5 million words, Wall Street Journal

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

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?

$$\begin{aligned} \text{PP}(W) &= P(w_1 w_2 \dots w_N)^{-\frac{1}{N}} \\ &= \left(\frac{1}{10}\right)^N)^{-\frac{1}{N}} \\ &= \frac{1}{10}^{-1} \\ &= 10 \end{aligned}$$

N-gram sparsity (Shakespeare 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)
- Quadrigrams worse: What's coming out looks like Shakespeare because it *is* Shakespeare

Zeros

Training set:

- ... denied the allegations
- ... denied the reports
- ... denied the claims
- ... denied the request

- Test set

- ... denied the offer
 - ... denied the loan

$$P(\text{"offer" } | \text{denied the}) = 0$$

how does this affect perplexity?

The intuition of smoothing (from Dan Klein)

- When we have sparse statistics:

$P(w | \text{denied the})$

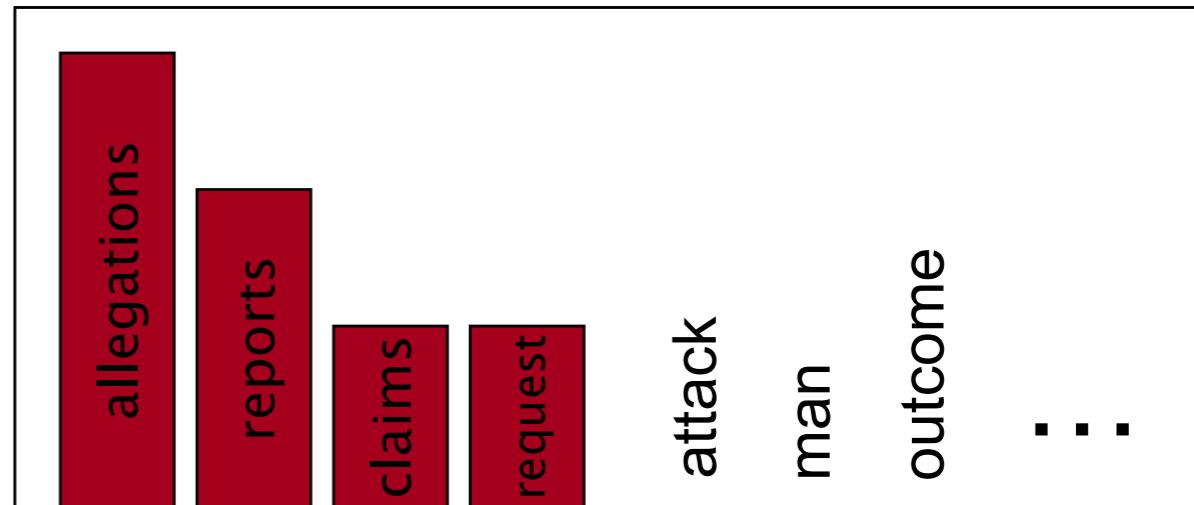
3 allegations

2 reports

1 claims

1 request

7 total



- Steal probability mass to generalize better

$P(w | \text{denied the})$

2.5 allegations

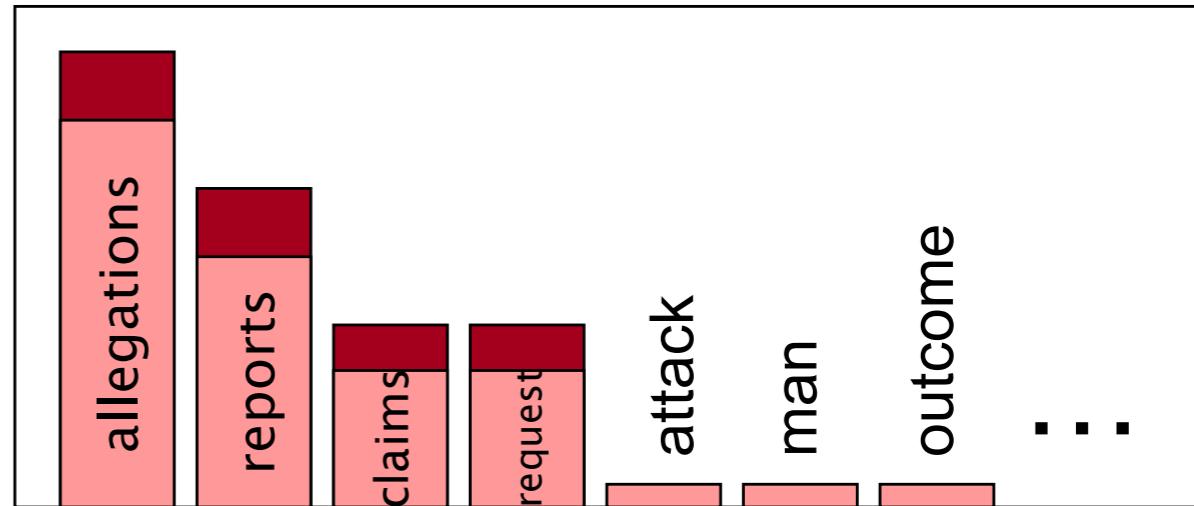
1.5 reports

0.5 claims

0.5 request

2 other

7 total



Add-one estimation (again!)

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

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

- MLE estimate:

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

- Add-1 estimate:

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:
 - We'll see better methods
- But add-1 is used to smooth other NLP models
 - For text classification
 - In domains where the number of zeros isn't so huge.

