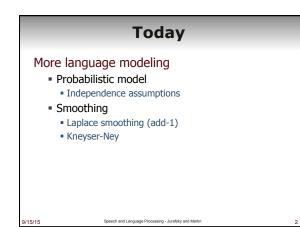
## Natural Language Processing

Lecture 7-9/15/2015

Jim Martin



### **Markov Assumption**

So for each component in the product replace with the approximation (assuming a prefix of N - 1)  $% \left( 1-\frac{1}{2}\right) =0$ 

$$P(w_n \mid w_1^{n-1}) \approx P(w_n \mid w_{n-N+1}^{n-1})$$

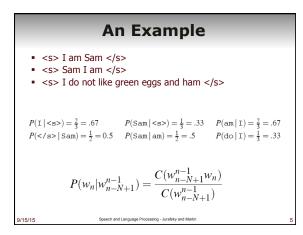
Big

9/15/15

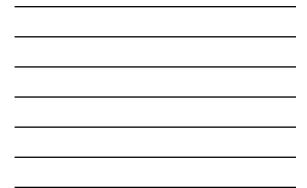
$$P(w_n | w_1^{n-1}) \approx P(w_n | w_{n-1})$$

**Estimating Bigram**  
**Probabilities**  
• The Maximum Likelihood Estimate (MLE)  

$$P(w_i | w_{i-1}) = \frac{Count(w_{i-1}, w_i)}{Count(w_{i-1})}$$
  
9(15/15)

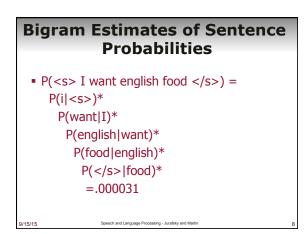


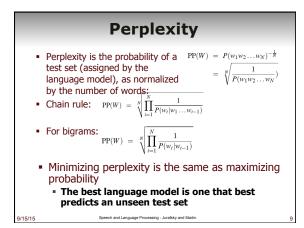
Bigram Counts									
<ul> <li>Ou</li> </ul>	t of	9222	sent	ence	6  V  s 827 tim	es			
	i	want	to	eat	chinese	food	lunch	spend	
1	5	827	0	9	0	0	0	-	
1	5	827	0	9	0	0	0	2	
ı want	2	827 0	0 608	1	6	0 6	5	2 1	
•			Ŭ	-	Ŭ	Ŭ.	· ·	2 1 211	
want	2	0	608	1	6	6	5	1	
want to	2 2	0 0	608 4	1 686	6 2	6 0	5 6	1 211	
want to eat	2 2	0 0 0	608 4 2	1 686 0	6 2 16	6 0 2	5 6	1 211 0	
want to eat chinese	2 2 0 1	0 0 0 0	608 4 2 0	1 686 0 0	6 2 16	6 0 2 82	5 6 42 1	1 211 0 0	

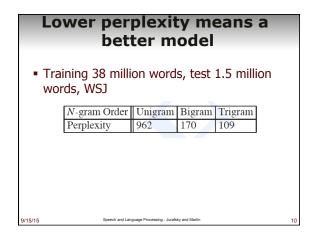


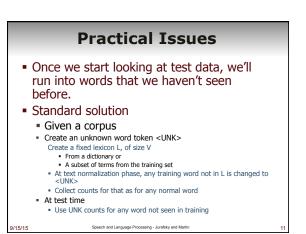
	ide big nts to							n
i	want	to	eat	chines				nd
2533	927	2417	746	158	109	3 341	278	3
	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	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











### **Practical Issues**

- Multiplying a bunch of small numbers between 0 and 1 is a really bad idea
  - Multiplication is slow

9/15/15

- And underflow is likely
- So do everything in log space
   Avoid underflow
  - Adding is faster than multiplying

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

### Smoothing

- Back to Shakespeare
  - Recall that 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)
  - Does that mean that any sentence that contains one of those bigrams should have a probability of 0?
  - For generation (shannon game) it means we'll never emit those bigrams
  - But for analysis it's problematic because if we run across a new bigram in the future then we have no choice but to assign it a probability of zero.

Speech and Language Processing - Jurafsky and Martin

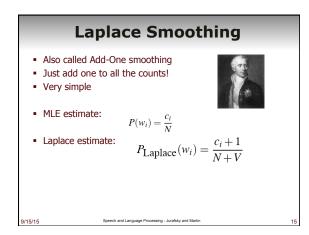
# **Zero Counts**

- Some of those zeros are really zeros...
  Things that really aren't ever going to happen
- On the other hand, some of them are just rare events. If the training corpus had been a little bigger they would have had a count
   What would that count be in all likelihood?
- Zipf's Law (long tail phenomenon):
  - A small number of events occur with high frequency
  - A large number of events occur with low frequency
     You can quickly collect statistics on the high frequency events
- You might have to wait an arbitrarily long time to get valid statistics on low frequency events Result:
- Our estimates are sparse! We have no counts at all for the vast number of things we want to estimate!
- Answer:

9/15/15

9/15/15

Estimate the likelihood of unseen (zero count) N-grams!



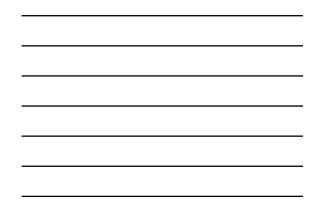
	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0000	- 16	2	42	0
chinese	1	0	0	0	0	82	1	õ
food	15	0	15	0	1	4	0	õ
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0
		× .	1	× .	×	-	Ŭ.	× .



	i	want	to	eat	chinese	food	lunch	spen
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



L	Laplace-Smoothed Bigram Probabilities									
	$P^*($	$[w_n w_n]$	_1) =	$\frac{C(w_n)}{C(w_n)}$	$(-1w_n)$ (n-1) +	$+1 \over V$				
	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		
9/15/15		Spe	ech and Language	e Processing - Jura	fsky and Martin			18		



	R	ecol	nsti	tute	d Co	oun	ts	
	$c^*(w_n$	$-1w_n$	$= \frac{[C(1)]}{[C(1)]}$	$\frac{w_{n-1}w_n}{C(1)}$	$(v_{n-1}) + 1] \times (v_{n-1}) + (v_{n-1}) +$	$\frac{C(w_r)}{V}$	<u>n-1)</u>	
	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
9/15/15		Spe	ech and Languag	e Processing - Jun	afsky and Martin			19



Reconstituted Counts (2)										
	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		
5/15		Speed	h and Languag	e Processing -	Juralsky and Mart	in				





- Because of this Laplace smoothing not often used for language models, as we have much better methods
- Despite its flaws Laplace (add-1) is still used to smooth other probabilistic models in NLP and IR, especially
  - For pilot studies
     In document classification
  - In document classification
    In domains where the number of zeros isn't so huge.

9/15/15

Speech and Language Processing - Jurafsky and Martin

チ

#### **Better Smoothing**

- Two key ideas
  - Use less context if the counts are missing for longer contexts
  - Use the count of things we've seen once to help estimate the count of things we've never seen

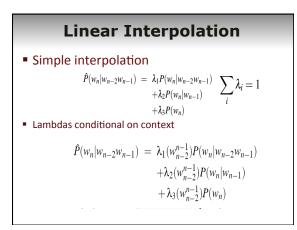
Speech and Language Processing - Jurafsky and Martin

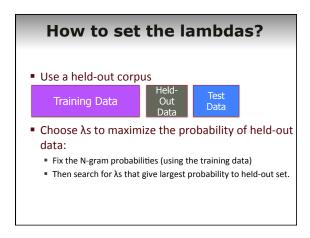
### **Backoff and Interpolation**

- Sometimes it helps to use less context
   Condition on less context for contexts you haven't learned much about
- Backoff

9/15/15

- use trigram if you have good evidence,
- otherwise bigram, otherwise unigram
- Interpolation
  - mix unigram, bigram, trigram
- Interpolation works better





## Types, Tokens and Fish

- Much of what's coming up was first studied by biologists who are often faced with 2 related problems
  - Determining how many species occupy a particular area (types)
  - And determining how many individuals of a given species are living in a given area (tokens)

Speech and Language Processing - Jurafsky and Marti

### **One Fish Two Fish**

Imagine you are fishing
 There are 8 species: carp, perch, whitefish, trout, salmon, eel, catfish, bass

9/15/15

9/15/15

- Not sure where this fishing hole is...
- You have caught up to now
- 10 carp, 3 perch, 2 whitefish, 1 trout, 1 salmon, 1 eel = 18 fish
- How likely is it that the next fish to be caught is an eel?
- How likely is it that the next fish caught will be a member of newly seen species?
- Now how likely is it that the next fish caught will be an eel?

Slide adapted from Josh Goodman Speech and Language Processing - Juratsky and Martin

#### **Fish Lesson**

- We need to steal part of the observed probability mass to give it to the as yet unseen N-Grams. Questions are:
  - How much to steal
  - How to redistribute it

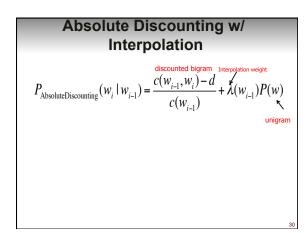
9/15/15

9/15/15

# **Absolute Discounting**

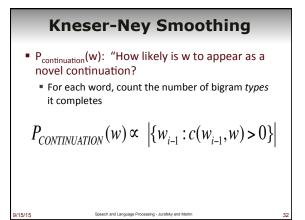
Speech and Language Processing - Jurafsky and Martin

- Just subtract a fixed amount from all the observed counts (call that d).
- Redistribute it proportionally based on observed data



### **Kneser-Ney Smoothing**

- Better estimate for probabilities of lower-order unigrams!
  - Shannon game: I can't see without my reading Fighting?
  - "Francisco" is more common than "glasses"
  - ... but "Francisco" frequently follows "San"
- So P(w) isn't what we want



### **Kneser-Ney Smoothing**

• Normalize by the total number of word bigram types to get a true probability

$$P_{CONTINUATION}(w) = \frac{\left| \{ w_{i-1} : c(w_{i-1}, w) > 0 \} \right|}{\left| \{ (w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0 \} \right|}$$

**Kneser-Ney Smoothing**  

$$P_{KN}(w_i | w_{i-1}) = \frac{\max(c(w_{i-1}, w_i) - d, 0)}{c(w_{i-1})} + \lambda(w_{i-1})P_{CONTINUATION}(w_i)$$

$$\lambda \text{ is a normalizing constant; the probability mass we've discounted}$$

$$\lambda(w_{i-1}) = \frac{d}{c(w_{i-1})} | \{w: c(w_{i-1}, w) > 0\} |$$

$$\sum_{i=0}^{n} \frac{d}{v_{i}(w_{i-1})} | \{w: c(w_{i-1}, w_{i-1}) > 0\} |$$

$$\sum_{i=0}^{n} \frac{d}{v_{i}(w_{i-1})} | w_{i-1} = \frac{1}{v_{i}(w_{i-1})} |$$