LING/C SC/PSYC 438/538

Lecture 22

Sandiway Fong

Last Time

- *Gentle introduction to probability*
- **Important notions**:
  - sample space
  - events
  - rule of counting
  - weighted events: probability
  - conditional probability: P(A|B)
  - *the importance of conditional probability or expectation in language*
    - Just then, the white rabbit
    - the
    - expectation is p(*rabbit*|*white*) > p(*the*|*white*) (*conditional*)
    - but p(*the*) > p(*rabbit*) (*unconditional*)

Language Models and N-grams

- **given a word sequence**
  - $w_1\ w_2\ w_3\ ...\ w_n$
- **chain rule**
  - ***how to compute the probability of a sequence of words***
  - $p(w_1\ w_2) = p(w_1)\ p(w_2|w_1)$
  - $p(w_1\ w_2\ w_3) = p(w_1)\ p(w_2|w_1)\ p(w_3|w_1 w_2)$
  - ...
  - $p(w_1\ w_2\ w_3 ... w_n) = p(w_1)\ p(w_2|w_1)\ p(w_3|w_1 w_2) ...\ p(w_n|w_1 ... w_{n-2}\ w_{n-1})$

- **note**
  - It's not easy to collect (meaningful) statistics on $p(w_n|w_{n-1} w_{n-2} ... w_1)$ for all possible word sequences

Language Models and N-grams

- **Given a word sequence**
  - $w_1\ w_2\ w_3\ \ldots\ w_n$
- **Bigram approximation**
  - *just look at the previous word only* (*not all the proceedings words*)
  - **Markov Assumption**: **finite length history**
  - 1st order Markov Model
  - $p(w_1\ w_2\ w_3 \ldots w_n) = p(w_1)\ p(w_2|w_1)\ p(w_3|w_1 w_2)\ \ldots p(w_n|w_1 \ldots w_{n-3} w_{n-2} w_{n-1})$
  - $\mathbf{p(w_1\ w_2\ w_3 \ldots w_n) \approx p(w_1)\ p(w_2|w_1)\ p(w_3|w_2) \ldots p(w_n|w_{n-1})}$
- **note**
  - $p(w_n|w_{n-1})$ is a lot easier to collect data for (and thus estimate well) than $p(w_n|w_1 \ldots w_{n-2}\ w_{n-1})$

Language Models and N-grams

- **Trigram approximation**
  - 2nd order Markov Model
  - *just look at the preceding two words only*
  - $p(w_1\ w_2\ w_3\ w_4 \ldots w_n) = p(w_1)\ p(w_2|w_1)\ p(w_3|w_1 w_2)\ p(w_4|w_1 w_2 w_3) \ldots p(w_n|w_1 \ldots w_{n-3} w_{n-2} w_{n-1})$
  - $\mathbf{p(w_1\ w_2\ w_3 \ldots w_n) \approx p(w_1)\ p(w_2|w_1)\ p(w_3|w_1 w_2) p(w_4|w_2 w_3) \ldots p(w_n\ |\ w_{n-2}\ w_{n-1})}$
- **note**
  - $p(w_n|w_{n-2}w_{n-1})$ is a lot easier to estimate well than $p(w_n|w_1 \ldots w_{n-2}\ w_{n-1})$ but harder than $p(w_n|w_{n-1})$

Language Models and N-grams

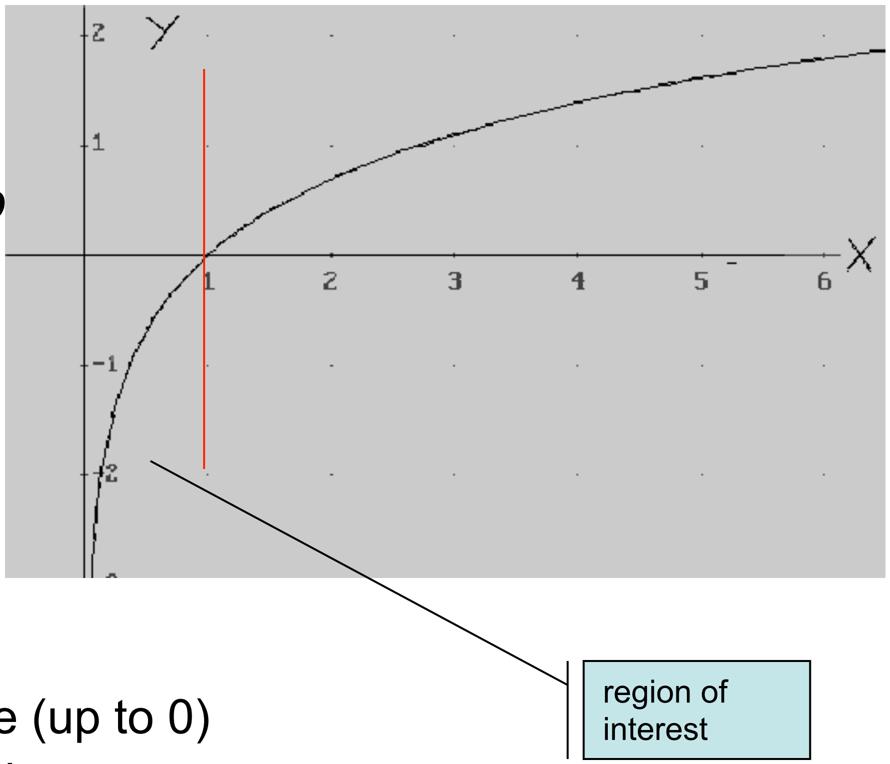
- **estimating from corpora**
  - *how to compute bigram probabilities*
  - $p(w_n|w_{n-1}) = f(w_{n-1}w_n)/f(w_{n-1}w)$ w is any word
  - Since $f(w_{n-1}w) = f(w_{n-1})$ $f(w_{n-1})$ = unigram frequency for w_{n-1}
  - $p(w_n|w_{n-1}) = f(w_{n-1}w_n)/f(w_{n-1})$ **relative frequency**

- Note:
  - The technique of estimating (true) probabilities using a relative frequency measure over a training corpus is known as **maximum likelihood estimation** (MLE)

Language Models and N-grams

- **Typical Practice:**
  - **Logprob calculations used**
- **Question**:
  - *Why sum negative log of prob*
- **Answer (Part 2)**:
  - A = BC
  - log(A) = log(B) + log(C)
  - probabilities are in range (0, 1]
  - Note:
    - want probabilities to be non-zero
    - log(0) = -∞
  - log of probabilites will be negative (up to 0)
  - take negative to make them positive

log function



Motivation for smoothing

- **Smoothing**: *avoid zero probability estimates*
- Consider

$$p(w_1\, w_2\, w_3 \ldots w_n) \approx p(w_1)\, p(w_2|w_1)\, p(w_3|w_2) \ldots p(w_n|w_{n-1})$$

- what happens when any individual probability component is zero?
  - **Arithmetic multiplication law**: $0 \times X = 0$
  - ***very brittle!***
- even in a very large corpus, many possible n-grams over vocabulary space will have zero frequency
  - *particularly so for larger n-grams*

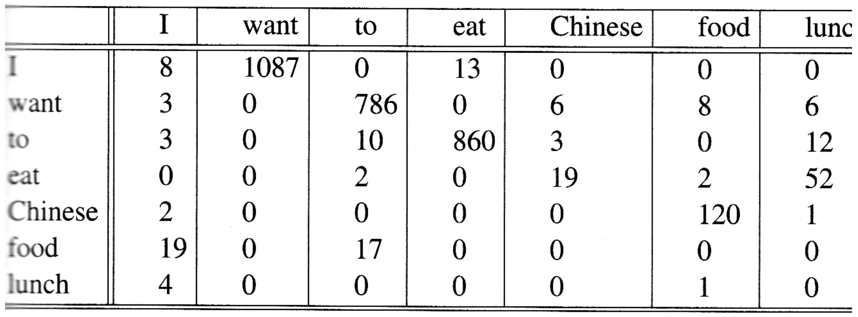
Language Models and N-grams

- **Example**:

w_n

w_{n-1}

$w_{n-1}w_n$ bigram frequencies



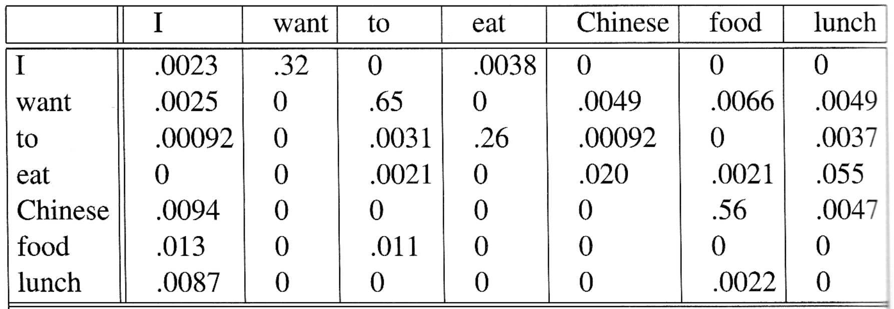
	I	want	to	eat	Chinese	food	lunc
I	8	1087	0	13	0	0	0
want	3	0	786	0	6	8	6
to	3	0	10	860	3	0	12
eat	0	0	2	0	19	2	52
Chinese	2	0	0	0	0	120	1
food	19	0	17	0	0	0	0
lunch	4	0	0	0	0	1	0

Figure 6.4 Bigram counts for seven of the words (out of 1616 total word types) in the Berkeley Restaurant Project corpus of ≈10,000 sentences.

I	3437
want	1215
to	3256
eat	938
Chinese	213
food	1506
lunch	459

unigram frequencies

bigram probabilities



	I	want	to	eat	Chinese	food	lunch
I	.0023	.32	0	.0038	0	0	0
want	.0025	0	.65	0	.0049	.0066	.0049
to	.00092	0	.0031	.26	.00092	0	.0037
eat	0	0	.0021	0	.020	.0021	.055
Chinese	.0094	0	0	0	0	.56	.0047
food	.013	0	.011	0	0	0	0
lunch	.0087	0	0	0	0	.0022	0

Figure 6.5 Bigram probabilities for seven of the words (out of 1616 total word types) in the Berkeley Restaurant Project corpus of ≈10,000 sentences.

sparse matrix

zeros render probabilities unusable

(*we'll need to add fudge factors - i.e. do **smoothing***)

Smoothing and N-grams

- **sparse dataset means zeros are a problem**
  - Zero probabilities are a problem
    - p(w_1 w_2 w_3...w_n) ≈ p(w_1) p(w_2|w_1) p(w_3|w_2)...p(w_n|w_{n-1}) **bigram model**
    - *one zero and the whole product is zero*
  - Zero frequencies are a problem
    - p(w_n|w_{n-1}) = f($w_{n-1}w_n$)/f(w_{n-1}) **relative frequency**
    - *bigram* f($w_{n-1}w_n$) *doesn't exist in dataset*

- **smoothing**
  - refers to ways of assigning zero probability n-grams a non-zero value
  - we'll look at two ways here **(just one of them today**)

Smoothing and N-grams

- **Add-One Smoothing**
  - add 1 to **all** frequency counts
  - *simple and no more zeros (but there are better methods)*

- **unigram**
  - $p(w) = f(w)/N$ (*before* Add-One)
    - N = size of corpus
  - **$p(w) = (f(w)+1)/(N+V)$** (*with* Add-One)
  - $f^*(w) = (f(w)+1)^*N/(N+V)$ (*with* Add-One)
    - V = number of distinct words in corpus
    - N/(N+V) normalization factor adjusting for the effective increase in the corpus size caused by Add-One

must rescale so that total probability mass stays at 1

- **bigram**
  - $p(w_n|w_{n-1}) = f(w_{n-1}w_n)/f(w_{n-1})$ (*before* Add-One)
  - **$p(w_n|w_{n-1}) = (f(w_{n-1}w_n)+1)/(f(w_{n-1})+V)$** (*after* Add-One)
  - $f^*(w_{n-1}\, w_n) = (f(w_{n-1}\, w_n)+1)^*\, f(w_{n-1}) /(f(w_{n-1})+V)$ (*after* Add-One)

Smoothing and N-grams

- **Add-One Smoothing**
  - add 1 to all frequency counts
- **bigram**
  - $p(w_n|w_{n-1}) = (f(w_{n-1}w_n)+1)/(f(w_{n-1})+V)$
  - $(f(w_{n-1}\,w_n)+1)^*\ f(w_{n-1})\ /(f(w_{n-1})+V)$
- **frequencies**

	I	want	to	eat	Chinese	food	lunch
I	8	1087	0	13	0	0	0
want	3	0	786	0	6	8	6
to	3	0	10	860	3	0	12
eat	0	0	2	0	19	2	52
Chinese	2	0	0	0	0	120	1
food	19	0	17	0	0	0	0
lunch	4	0	0	0	0	1	0

= figure 6.4

	I	want	to	eat	Chinese	food	lunch
I	6.12	740.05	0.68	9.52	0.68	0.68	0.68
want	1.72	0.43	337.76	0.43	3.00	3.86	3.00
to	2.67	0.67	7.35	575.41	2.67	0.67	8.69
eat	0.37	0.37	1.10	0.37	7.35	1.10	19.47
Chinese	0.35	0.12	0.12	0.12	0.12	14.09	0.23
food	9.65	0.48	8.68	0.48	0.48	0.48	0.48
lunch	1.11	0.22	0.22	0.22	0.22	0.44	0.22

= figure 6.8

Remarks:
perturbation problem

add-one causes large changes in some frequencies due to relative size of V (1616)

want to: 786 $\Rightarrow$ 338

Smoothing and N-grams

- **Add-One Smoothing**
  - *add 1 to all frequency counts*
- **bigram**
  - p(w_n|w_{n-1}) = (f($w_{n-1}w_n$)+1)/(f(w_{n-1})+V)
  - (f(w_{n-1} w_n)+1)* f(w_{n-1}) /(f(w_{n-1})+V)
- **Probabilities**

Remarks:
perturbation problem

similar changes in probabilities

	I	want	to	eat	Chinese	food	lunch
I	0.00233	0.31626	0.00000	0.00378	0.00000	0.00000	0.00000
want	0.00247	0.00000	0.64691	0.00000	0.00494	0.00658	0.00494
to	0.00092	0.00000	0.00307	0.26413	0.00092	0.00000	0.00369
eat	0.00000	0.00000	0.00213	0.00000	0.02026	0.00213	0.05544
Chinese	0.00939	0.00000	0.00000	0.00000	0.00000	0.56338	0.00469
food	0.01262	0.00000	0.01129	0.00000	0.00000	0.00000	0.00000
lunch	0.00871	0.00000	0.00000	0.00000	0.00000	0.00218	0.00000

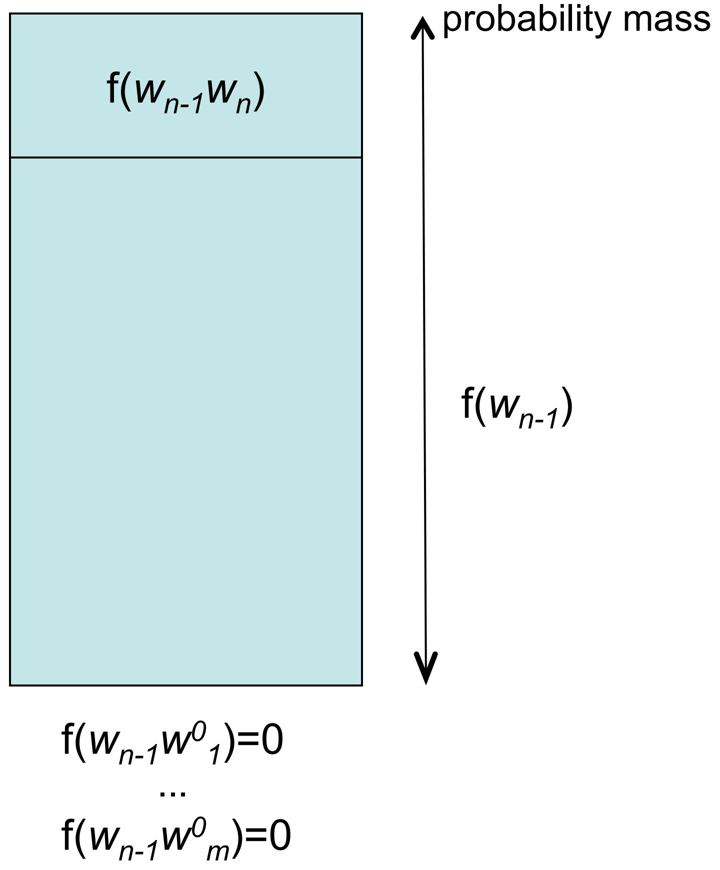
= figure 6.5

	I	want	to	eat	Chinese	food	lunch
I	0.00178	0.21532	0.00020	0.00277	0.00020	0.00020	0.00020
want	0.00141	0.00035	0.27799	0.00035	0.00247	0.00318	0.00247
to	0.00082	0.00021	0.00226	0.17672	0.00082	0.00021	0.00267
eat	0.00039	0.00039	0.00117	0.00039	0.00783	0.00117	0.02075
Chinese	0.00164	0.00055	0.00055	0.00055	0.00055	0.06616	0.00109
food	0.00641	0.00032	0.00577	0.00032	0.00032	0.00032	0.00032
lunch	0.00241	0.00048	0.00048	0.00048	0.00048	0.00096	0.00048

= figure 6.7

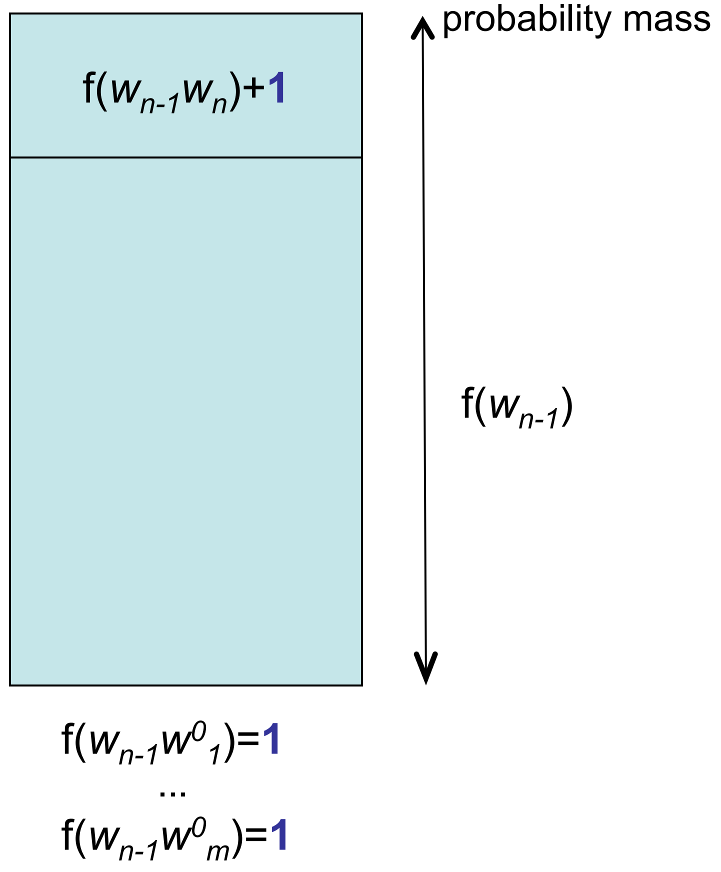
Smoothing and N-grams

- *let's illustrate the problem*
  - take the bigram case:
  - $w_{n-1}w_n$
  - $p(w_n|w_{n-1}) = f(w_{n-1}w_n)/f(w_{n-1})$
  - suppose there are cases
  - $w_{n-1}w^0_1$ that don't occur in the corpus



Smoothing and N-grams

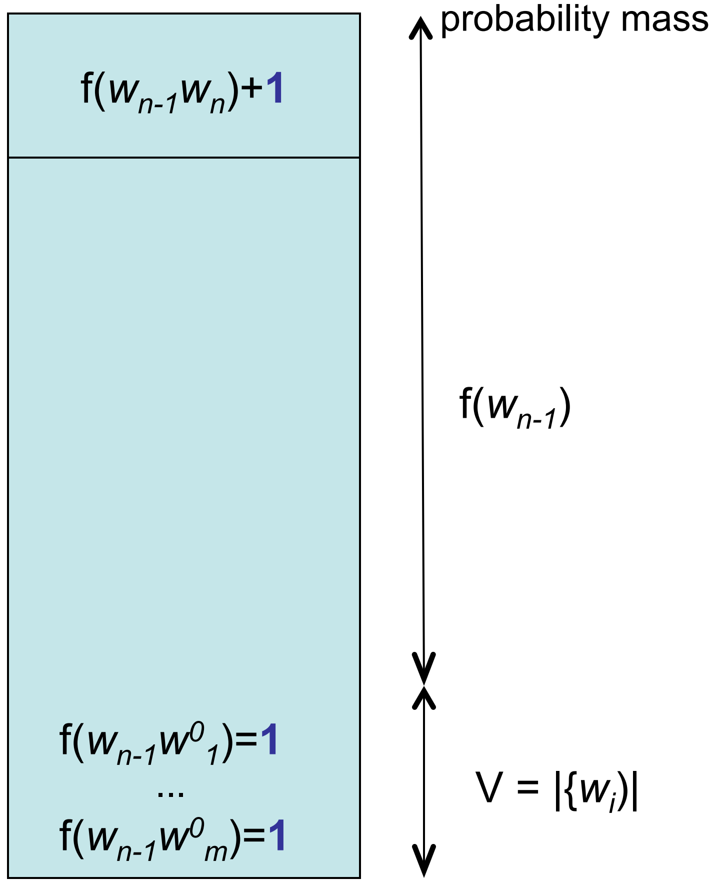
- *add-one*
  - *"give everyone 1"*



Smoothing and N-grams

- *add-one*
  - *"give everyone 1"*

- *redistribution of probability mass*
  - p(w_n|w_{n-1}) = (f($w_{n-1}w_n$)+1)/(f(w_{n-1})+V)



Smoothing and N-grams

- Excel spreadsheet available
  – addone.xls

Smoothing and N-grams

- **Witten-Bell Smoothing**
  - equate zero frequency items with frequency 1 items
  - use frequency of things seen once to estimate frequency of things we haven't seen yet
  - *smaller impact than Add-One*
- **unigram**
  - a zero frequency word (unigram) is "an event that hasn't happened yet"
  - count the number of (*different*) words (T) we've observed in the corpus
  - p(w) = T/(Z*(N+T))
    - w is a word with zero frequency
    - Z = number of zero frequency words
    - N = size of corpus
- **bigram**
  - $p(w_n|w_{n-1}) = f(w_{n-1}w_n)/f(w_{n-1})$ (*original*)
  - $\mathbf{p(w_n|w_{n-1}) = T(w_{n-1})/(Z(w_{n-1})*(T(w_{n-1})+N(w_{n-1}))}$ for zero bigrams (*after* Witten-Bell)
    - $T(w_{n-1})$ = number of (seen) bigrams beginning with w_{n-1}
    - $Z(w_{n-1})$ = number of unseen bigrams beginning with w_{n-1}
    - $Z(w_{n-1})$ = (possible bigrams beginning with w_{n-1}) – (the ones we've seen)
    - $Z(w_{n-1}) = V - T(w_{n-1})$
  - $T(w_{n-1})/ Z(w_{n-1}) * f(w_{n-1})/(f(w_{n-1})+ T(w_{n-1}))$ estimated zero bigram frequency
  - $\mathbf{p(w_n|w_{n-1}) = f(w_{n-1}w_n)/(f(w_{n-1})+T(w_{n-1}))}$ for non-zero bigrams (*after* Witten-Bell)

Smoothing and N-grams

- **Witten-Bell Smoothing**
  - use frequency of things seen once to estimate frequency of things we haven't seen yet
- **bigram**
  - $T(w_{n-1}) / Z(w_{n-1}) * f(w_{n-1})/(f(w_{n-1}) + T(w_{n-1}))$ estimated zero bigram frequency
    - $T(w_{n-1})$ = number of bigrams beginning with w_{n-1}
    - $Z(w_{n-1})$ = number of unseen bigrams beginning with w_{n-1}

	I	want	to	eat	Chinese	food	lunch
I	8	1087	0	13	0	0	0
want	3	0	786	0	6	8	6
to	3	0	10	860	3	0	12
eat	0	0	2	0	19	2	52
Chinese	2	0	0	0	0	120	1
food	19	0	17	0	0	0	0
lunch	4	0	0	0	0	1	0

= figure 6.4

Remark:
smaller changes

	I	want	to	eat	Chinese	food	lunch
I	7.785	1057.763	0.061	12.650	0.061	0.061	0.061
want	2.823	0.046	739.729	0.046	5.647	7.529	5.647
to	2.885	0.084	9.616	826.982	2.885	0.084	11.539
eat	0.073	0.073	1.766	0.073	16.782	1.766	45.928
Chinese	1.828	0.011	0.011	0.011	0.011	109.700	0.914
food	18.019	0.051	16.122	0.051	0.051	0.051	0.051
lunch	3.643	0.026	0.026	0.026	0.026	0.911	0.026

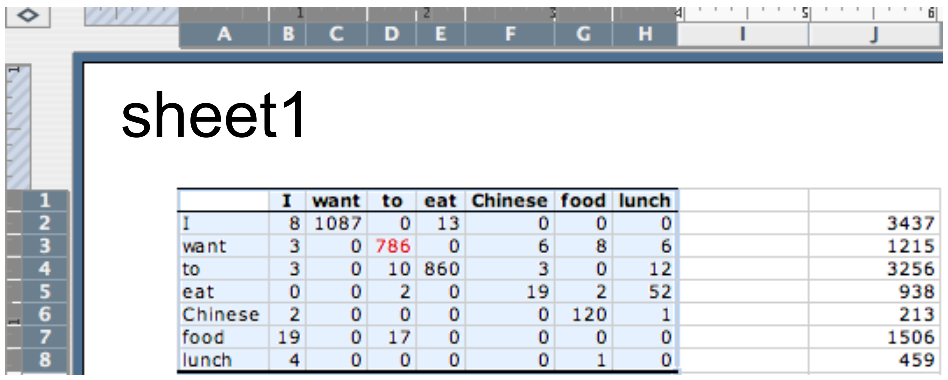
= figure 6.9

Smoothing and N-grams

- Witten-Bell excel spreadsheet:
  - `wb.xls`

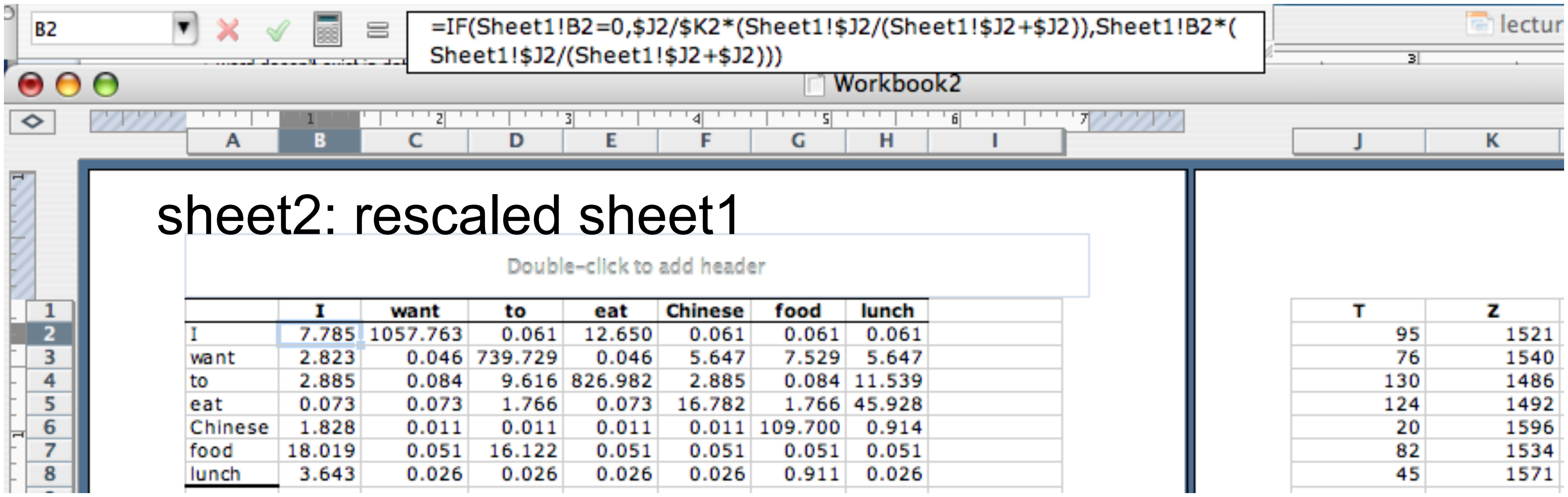
Smoothing and N-grams

- **Witten-Bell Smoothing**
- ***Implementation* (*Excel*)**
  - one line formula



sheet1

	I	want	to	eat	Chinese	food	lunch		
I	8	1087	0	13	0	0	0		3437
want	3	0	786	0	6	8	6		1215
to	3	0	10	860	3	0	12		3256
eat	0	0	2	0	19	2	52		938
Chinese	2	0	0	0	0	120	1		213
food	19	0	17	0	0	0	0		1506
lunch	4	0	0	0	0	1	0		459



B2

```
=IF(Sheet1!B2=0,$J2/$K2*(Sheet1!$J2/(Sheet1!$J2+$J2)),Sheet1!B2*(
Sheet1!$J2/(Sheet1!$J2+$J2)))
```

Workbook2

sheet2: rescaled sheet1

Double-click to add header

	I	want	to	eat	Chinese	food	lunch
I	7.785	1057.763	0.061	12.650	0.061	0.061	0.061
want	2.823	0.046	739.729	0.046	5.647	7.529	5.647
to	2.885	0.084	9.616	826.982	2.885	0.084	11.539
eat	0.073	0.073	1.766	0.073	16.782	1.766	45.928
Chinese	1.828	0.011	0.011	0.011	0.011	109.700	0.914
food	18.019	0.051	16.122	0.051	0.051	0.051	0.051
lunch	3.643	0.026	0.026	0.026	0.026	0.911	0.026

T	Z
95	1521
76	1540
130	1486
124	1492
20	1596
82	1534
45	1571

Language Models and N-grams

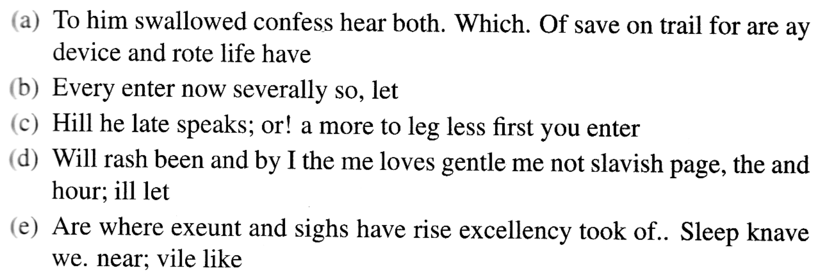
- **N-gram models**
  - they’re technically easy to compute
    - (*in the sense that lots of training data are available*)
  - but just how good are these n-gram language models?
  - and what can they show us about language?

Language Models and N-grams

approximating Shakespeare

- generate random sentences using n-grams
- train on complete *Works of Shakespeare*

• **Unigram** (pick random, unconnected words)



(a) To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have

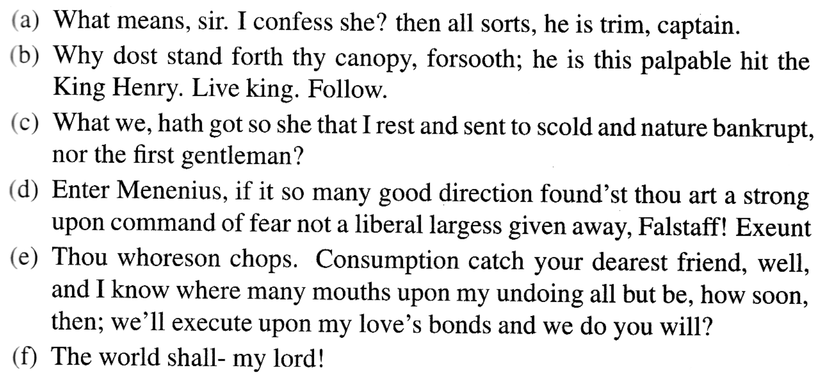
(b) Every enter now severally so, let

(c) Hill he late speaks; or! a more to leg less first you enter

(d) Will rash been and by I the me loves gentle me not slavish page, the and hour; ill let

(e) Are where exeunt and sighs have rise excellency took of.. Sleep knave we. near; vile like

• **Bigram**



(a) What means, sir. I confess she? then all sorts, he is trim, captain.

(b) Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.

(c) What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman?

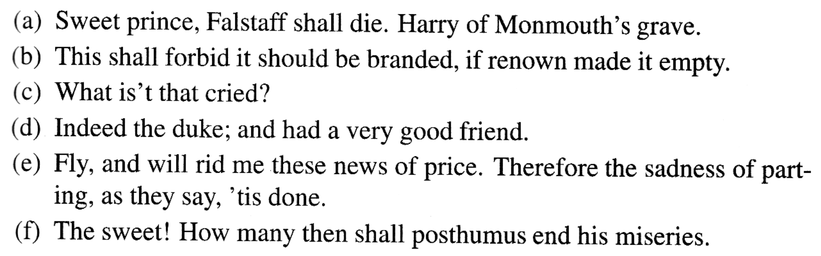
(d) Enter Menenius, if it so many good direction found'st thou art a strong upon command of fear not a liberal largess given away, Falstaff! Exeunt

(e) Thou whoreson chops. Consumption catch your dearest friend, well, and I know where many mouths upon my undoing all but be, how soon, then; we'll execute upon my love's bonds and we do you will?

(f) The world shall- my lord!

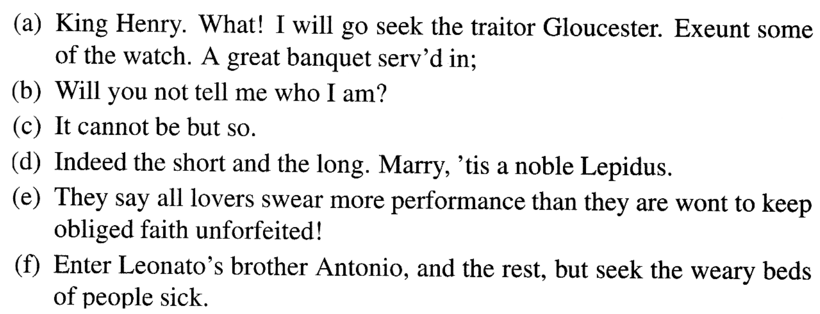
Language Models and N-grams

- Approximating Shakespeare (section 6.2)
  - generate random sentences using n-grams
  - train on complete *Works of Shakespeare*
- **Trigram**



(a) Sweet prince, Falstaff shall die. Harry of Monmouth's grave.
(b) This shall forbid it should be branded, if renown made it empty.
(c) What is't that cried?
(d) Indeed the duke; and had a very good friend.
(e) Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.
(f) The sweet! How many then shall posthumus end his miseries.

- **Quadrigram**



(a) King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in;
(b) Will you not tell me who I am?
(c) It cannot be but so.
(d) Indeed the short and the long. Marry, 'tis a noble Lepidus.
(e) They say all lovers swear more performance than they are wont to keep obliged faith unforfeited!
(f) Enter Leonato's brother Antonio, and the rest, but seek the weary beds of people sick.

Remarks:
dataset size problem

training set is small
884,647 words
29,066 different words

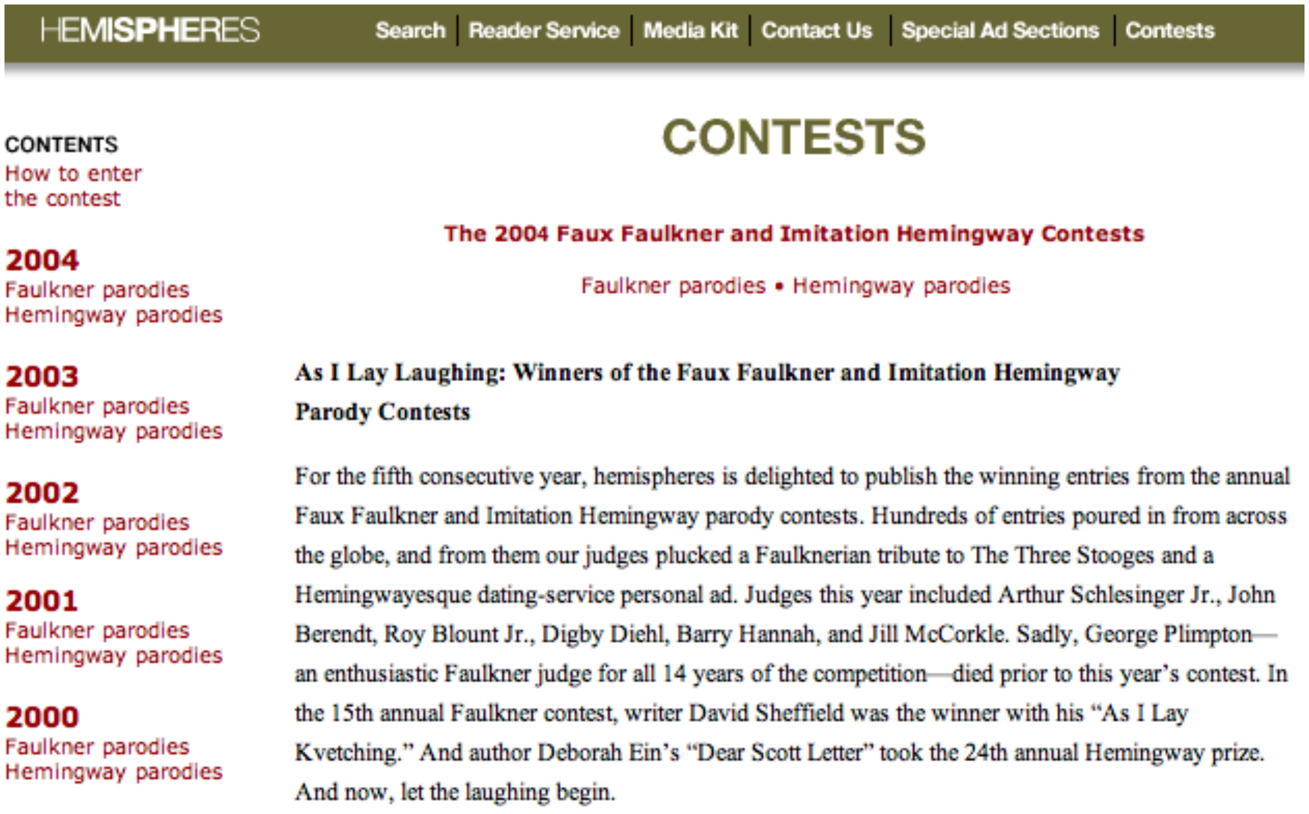
$29{,}066^2 = 844{,}832{,}356$
possible bigrams

for the random sentence generator, this means very limited choices for possible continuations, which means program can't be very innovative for higher *n*

Possible Application?

Language Models and N-grams

- **Aside**: http://hemispheresmagazine.com/contests/2004/intro.htm


HEMISPHERES
Search | Reader Service | Media Kit | Contact Us | Special Ad Sections | Contests

CONTENTS
How to enter the contest

2004
Faulkner parodies
Hemingway parodies

2003
Faulkner parodies
Hemingway parodies

2002
Faulkner parodies
Hemingway parodies

2001
Faulkner parodies
Hemingway parodies

2000
Faulkner parodies
Hemingway parodies

CONTESTS

The 2004 Faux Faulkner and Imitation Hemingway Contests

Faulkner parodies • Hemingway parodies

As I Lay Laughing: Winners of the Faux Faulkner and Imitation Hemingway Parody Contests

For the fifth consecutive year, hemispheres is delighted to publish the winning entries from the annual Faux Faulkner and Imitation Hemingway parody contests. Hundreds of entries poured in from across the globe, and from them our judges plucked a Faulknerian tribute to The Three Stooges and a Hemingwayesque dating-service personal ad. Judges this year included Arthur Schlesinger Jr., John Berendt, Roy Blount Jr., Digby Diehl, Barry Hannah, and Jill McCorkle. Sadly, George Plimpton—an enthusiastic Faulkner judge for all 14 years of the competition—died prior to this year's contest. In the 15th annual Faulkner contest, writer David Sheffield was the winner with his "As I Lay Kvetching." And author Deborah Ein's "Dear Scott Letter" took the 24th annual Hemingway prize. And now, let the laughing begin.

Language Models and N-grams

- **N-gram models + smoothing**
  - one consequence of smoothing is that
  - every possible concatentation or sequence of words has a non-zero probability

Colorless green ideas

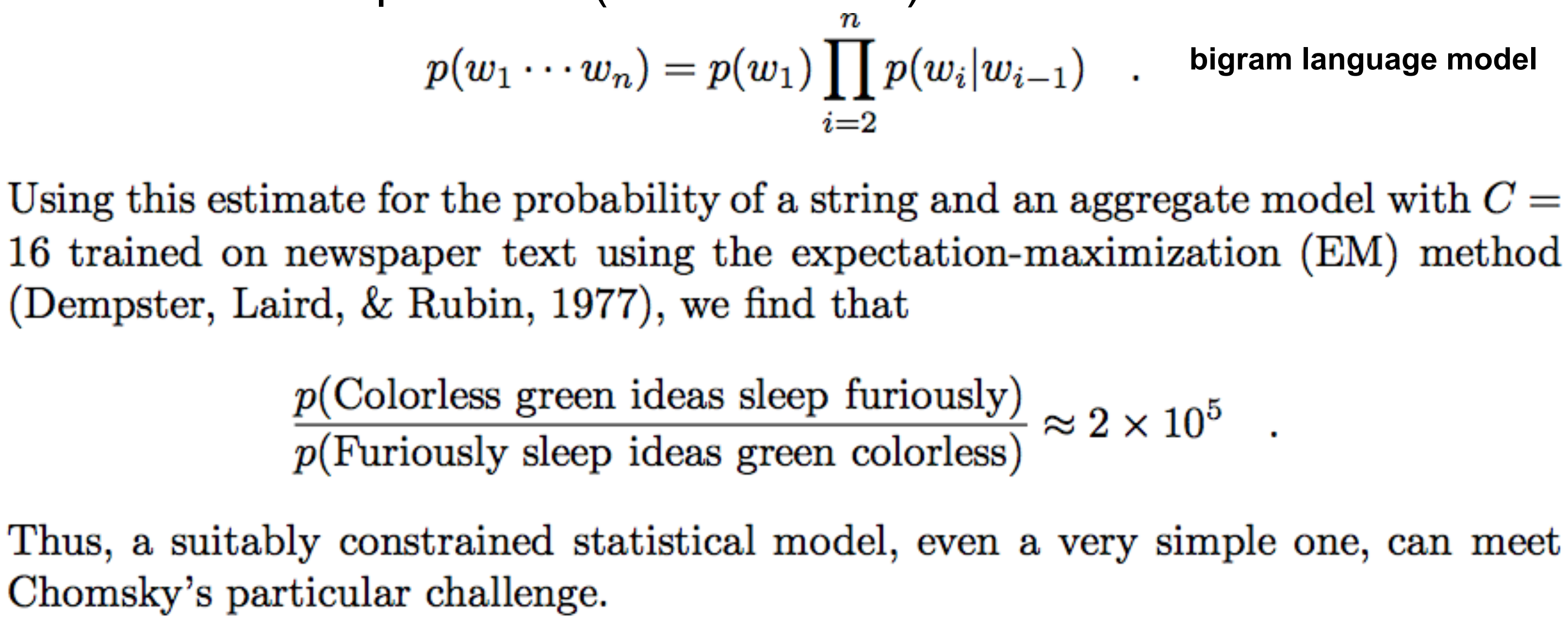
- **examples**
  - (1) **colorless green ideas** sleep furiously
  - (2) furiously sleep ideas green colorless
- **Chomsky** (1957):
  - . . . It is fair to assume that neither sentence (1) nor (2) (nor indeed any part of these sentences) has ever occurred in an English discourse. Hence, **in any statistical model for grammaticalness**, these sentences will be ruled out on identical grounds as equally `remote' from English. Yet (1), though nonsensical, is grammatical, while (2) is not.
- **idea**
  - (1) is syntactically valid, (2) is word salad
- Statistical Experiment (Pereira 2002)

Colorless green ideas

- **examples**
  - (1) **colorless green ideas** sleep furiously
  - (2) furiously sleep ideas green colorless
- Statistical Experiment (Pereira 2002)

w_{i-1}	w_i

bigram language model



$$p(w_1 \cdots w_n) = p(w_1) \prod_{i=2}^{n} p(w_i | w_{i-1}) \quad .$$

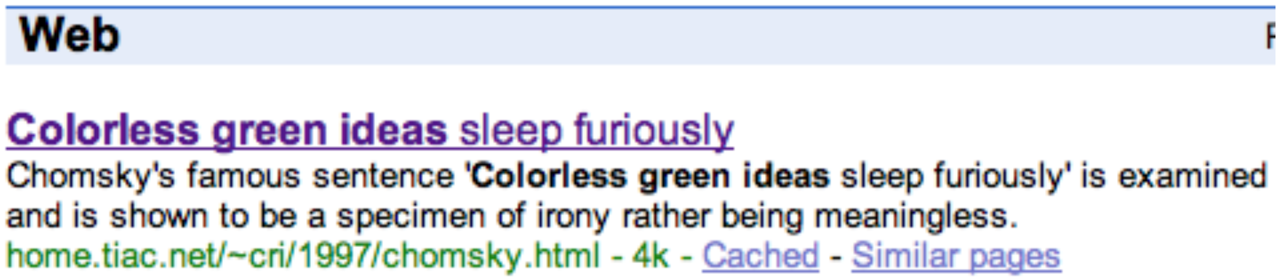
Using this estimate for the probability of a string and an aggregate model with $C = 16$ trained on newspaper text using the expectation-maximization (EM) method (Dempster, Laird, & Rubin, 1977), we find that

$$\frac{p(\text{Colorless green ideas sleep furiously})}{p(\text{Furiously sleep ideas green colorless})} \approx 2 \times 10^5 \quad .$$

Thus, a suitably constrained statistical model, even a very simple one, can meet Chomsky's particular challenge.

Interesting things to Google™

- **example**
  - **colorless green ideas** sleep furiously
- **First hit**



Web

Colorless green ideas sleep furiously
Chomsky's famous sentence '**Colorless green ideas** sleep furiously' is examined and is shown to be a specimen of irony rather being meaningless.
home.tiac.net/~cri/1997/chomsky.html - 4k - Cached - Similar pages

Interesting things to Google™

- **example**
  - **colorless green ideas** sleep furiously
- **first hit**
  - **compositional semantics**
  - a **green idea** is, according to well established usage of the word "green" is one that is an idea that is **new and untried**.
  - again, a **colorless idea** is one **without vividness, dull and unexciting**.
  - so it follows that a **colorless green idea** is a **new, untried idea that is without vividness, dull and unexciting**.
  - to **sleep** is, among other things, is to be in a state of dormancy or inactivity, or in a state of unconsciousness.
  - to **sleep furiously** may seem a puzzling turn of phrase but one reflects that **the mind in sleep often indeed moves furiously with ideas and images flickering in and out**.

Interesting things to Google

- **example**
  - **colorless green ideas** sleep furiously
- **another hit**: (*a story*)
  - "So this is our ranking system," said Chomsky. "As you can see, the highest rank is yellow."
  - "And the **new ideas**?"
  - "The **green ones**? Oh, **the green ones don't get a color until they've had some seasoning**. These ones, anyway, are still too angry. **Even when they're asleep, they're furious**. We've had to kick them out of the dormitories - they're just unmanageable."
  - "So where are they?"
  - "Look," said Chomsky, and pointed out of the window. There below, on the lawn, the colorless green ideas slept, furiously.

More on N-grams

- How to degrade gracefully when we don’t have evidence
  - Backoff
  - Deleted Interpolation
- N-grams and Spelling Correction

Backoff

- **idea**
  - Hierarchy of approximations
  - trigram > bigram > unigram
  - *degrade gracefully*
- Given a word sequence fragment:
  - $\ldots w_{n-2}\ w_{n-1}\ w_n \ldots$
- **preference rule**

1. $p(w_n \mid w_{n-2}\, w_{n-1})$ if $f(w_{n-2} w_{n-1}\, w_n) \neq 0$
2. $\alpha_1 p(w_n \mid w_{n-1})$ if $f(w_{n-1}\, w_n) \neq 0$
3. $\alpha_2 p(w_n)$

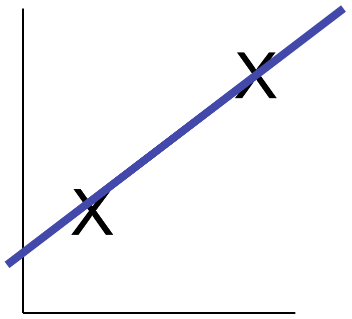
- notes:
  - α_1 and α_2 are fudge factors to ensure that probabilities still sum to 1

Backoff

- **preference rule**
  1. p($w_n | w_{n-2}\, w_{n-1}$) if f($w_{n-2} w_{n-1}\, w_n$) $\neq 0$
  2. α_1p($w_n | w_{n-1}$) if f($w_{n-1}\, w_n$) $\neq 0$
  3. α_2p(w_n)
- **problem**
  - if f($w_{n-2} w_{n-1}\, w_n$) = 0, we use one of the estimates from (2) or (3)
  - assume the backoff value is non-zero
  - then we are introducing non-zero probability for p($w_n | w_{n-2}\, w_{n-1}$) – which is zero in the corpus
  - then this adds “probability mass” to p($w_n | w_{n-2}\, w_{n-1}$) which is not in the original system
  - *therefore, we have to be careful to juggle the probabilities to still sum to 1*

Deleted Interpolation

- *fundamental idea of interpolation*
- **equation**: (*trigram*)
  - **p**($w_n | w_{n-2}\, w_{n-1}$) =
    - λ_1p($w_n | w_{n-2}\, w_{n-1}$) +
    - λ_2p($w_n | w_{n-2}$) +
    - λ_3p(w_n)



- **Note**:
  - λ_1, λ_2 and λ_3 are fudge factors to ensure that probabilities still sum to 1