CSCI 5832
Natural Language Processing

Lecture 7

Jim Martin

Today 2/6

- **Review N-Gram language models**
- **Add-1 smoothing**
- **Good-Turing smoothing**

Basic Idea

- We're interested in assigning probabilities to sentences (or more generally to sequences of events).
- We'll treat sequences as conjunctions of random events and make a particular set of conditional independence assumptions.
- These assumptions will allow us to break the sequence down into components for which we can gather the necessary statistics.

Chain Rule

- Recall the definition of conditional probabilities [illegible]
- Rewriting [illegible]
- Or… [illegible]
- Or… [illegible]

Example

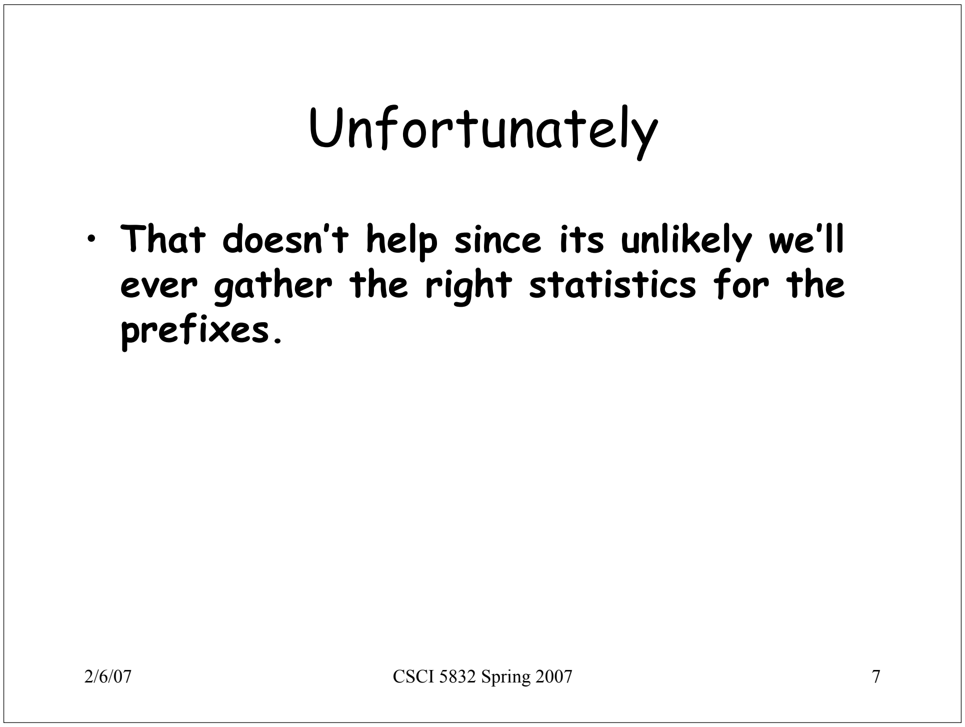
- **The big red dog**
- **P(The)*P(big|the)*P(red|the big)*P(dog|the big red)**
- **Better P(The| <Beginning of sentence>) written as**
  P(The | <S>)

General Case

- **The word sequence from position 1 to n is** [illegible]
- **So the probability of a sequence is**

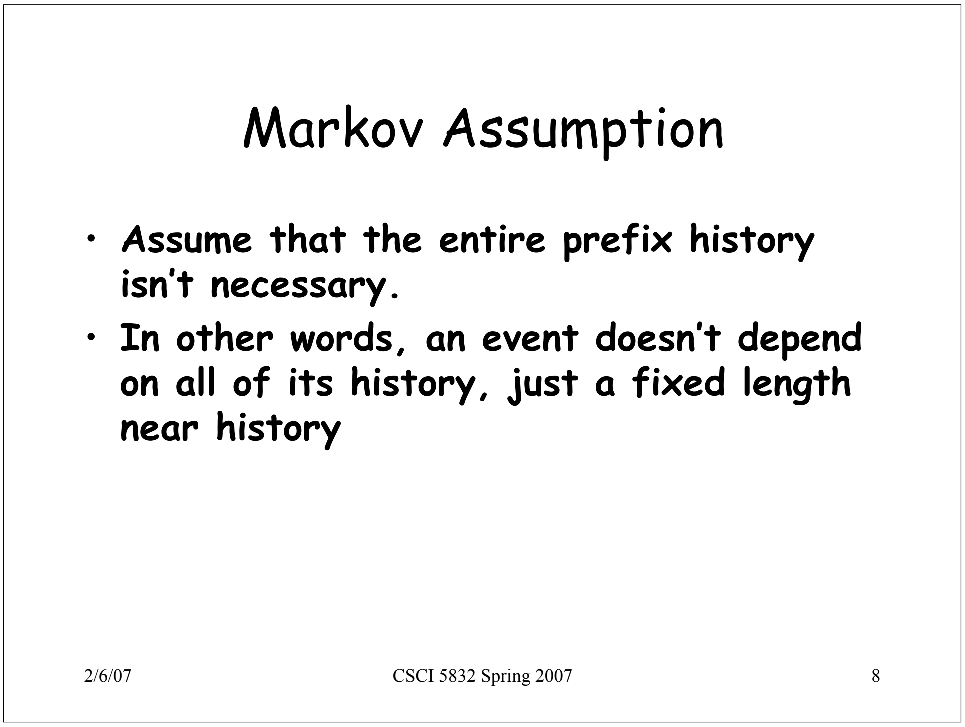
[illegible]

[illegible]



Unfortunately

- That doesn’t help since its unlikely we’ll ever gather the right statistics for the prefixes.



Markov Assumption

- Assume that the entire prefix history isn’t necessary.
- In other words, an event doesn’t depend on all of its history, just a fixed length near history

Markov Assumption

- So for each component in the product replace each with its with the approximation (assuming a prefix of N)

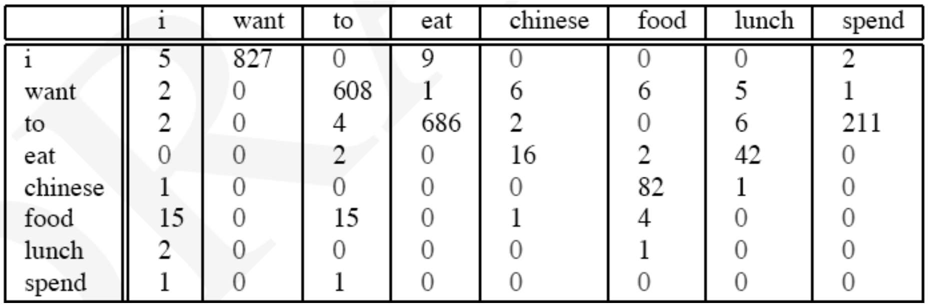
[illegible]

N-Grams
The big red dog

- Unigrams: P(dog)
- Bigrams: P(dog|red)
- Trigrams: P(dog|big red)
- Four-grams: P(dog|the big red)

In general, we'll be dealing with
P(Word| Some fixed prefix)

BERP Table: 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

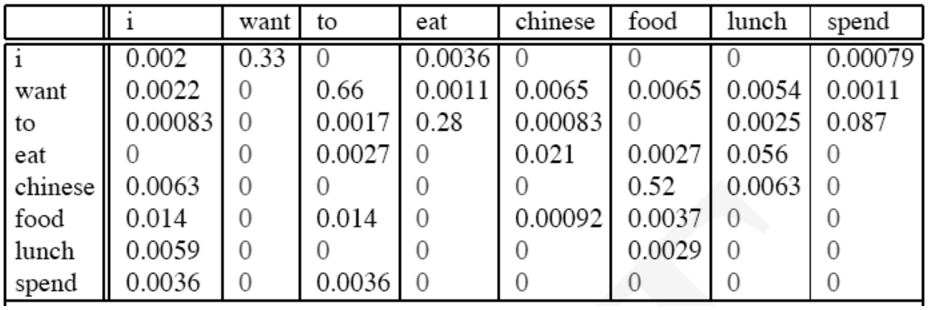
Counts/Bigram Probs

- **Recall… if we want P(want | I) that’s the**

P(I want)/P(I) and that’s just

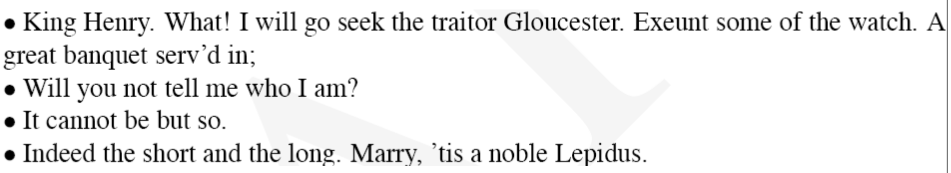
Count(I want)/Count(I)

BERP Table: Bigram Probabilities



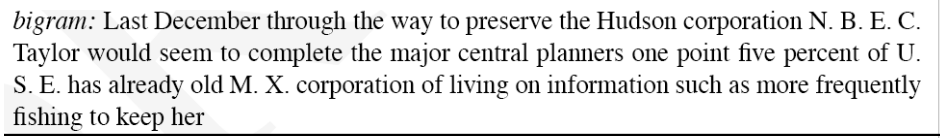
	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Shakespeare: 4-Grams



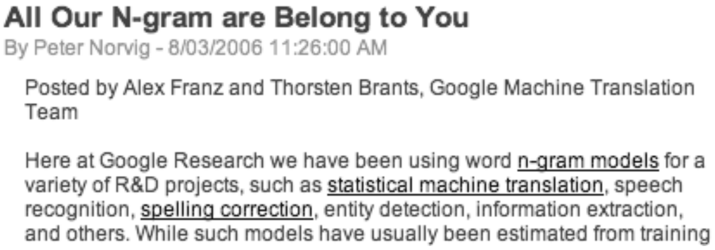
- King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in;
- Will you not tell me who I am?
- It cannot be but so.
- Indeed the short and the long. Marry, 'tis a noble Lepidus.

WSJ: Bigrams



bigram: Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her

Google N-Gram Release



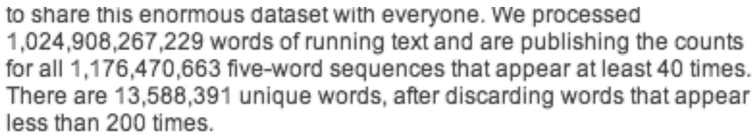
All Our N-gram are Belong to You

By Peter Norvig - 8/03/2006 11:26:00 AM

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

Here at Google Research we have been using word n-gram models for a variety of R&D projects, such as statistical machine translation, speech recognition, spelling correction, entity detection, information extraction, and others. While such models have usually been estimated from training

Google N-Gram Release



to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.

Google N-Gram Release

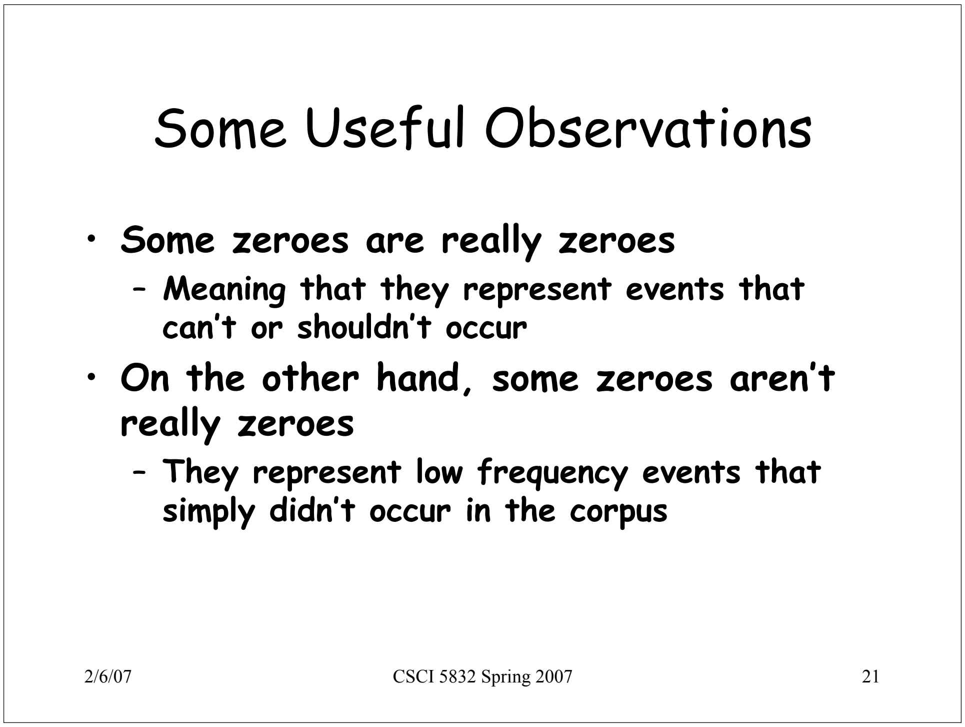
- **serve as the incoming 92**
- **serve as the incubator 99**
- **serve as the independent 794**
- **serve as the index 223**
- **serve as the indication 72**
- **serve as the indicator 120**
- **serve as the indicators 45**
- **serve as the indispensable 111**
- **serve as the indispensible 40**
- **serve as the individual 234**

Question

- **What the heck is that a model of?**

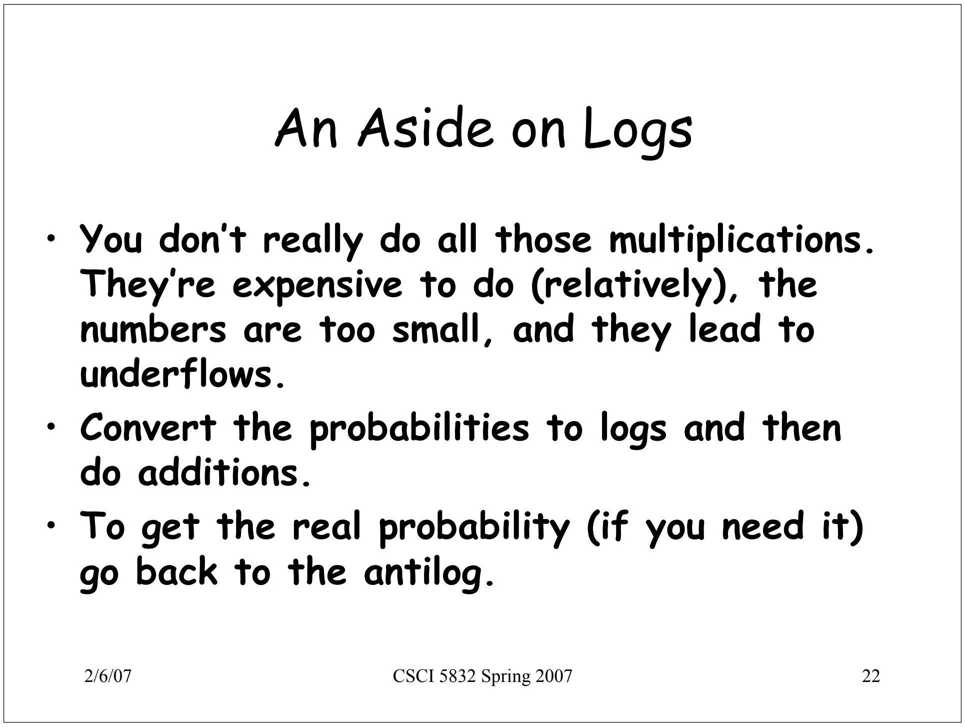
Some Useful Observations

- **A small number of events occur with high frequency**
  - **You can collect reliable statistics on these events with relatively small samples**
  - **Generally you should believe these numbers**
- **A large number of events occur with small frequency**
  - **You might have to wait a long time to gather statistics on the low frequency events**
  - **You should treat these numbers with skepticism**



Some Useful Observations

- Some zeroes are really zeroes
  - Meaning that they represent events that can’t or shouldn’t occur
- On the other hand, some zeroes aren’t really zeroes
  - They represent low frequency events that simply didn’t occur in the corpus



An Aside on Logs

- You don’t really do all those multiplications. They’re expensive to do (relatively), the numbers are too small, and they lead to underflows.
- Convert the probabilities to logs and then do additions.
- To get the real probability (if you need it) go back to the antilog.

Problem

- Let's assume we're using N-grams
- How can we assign a probability to a sequence where one of the component n-grams has a value of zero
- Assume all the words are known and have been seen
  - Go to a lower order n-gram
  - Back off from bigrams to unigrams
  - Replace the zero with something else

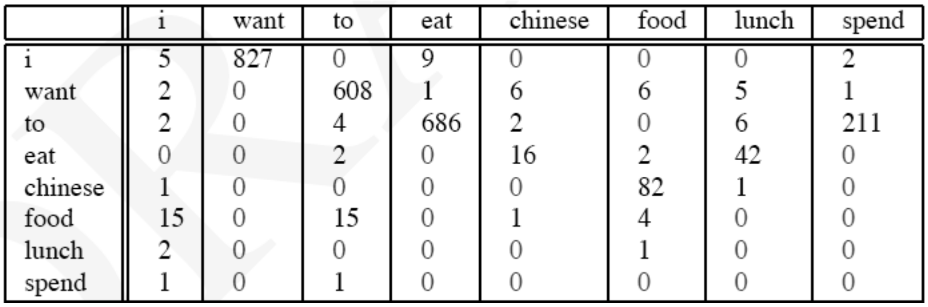
Smoothing Solutions

- Lots of solutions… All based on different intuitions about how to think about events that haven't occurred (yet).
- They range from the very simple to very convoluted. We'll cover
  - Add 1
  - Good-Turing

Add-One (Laplace)

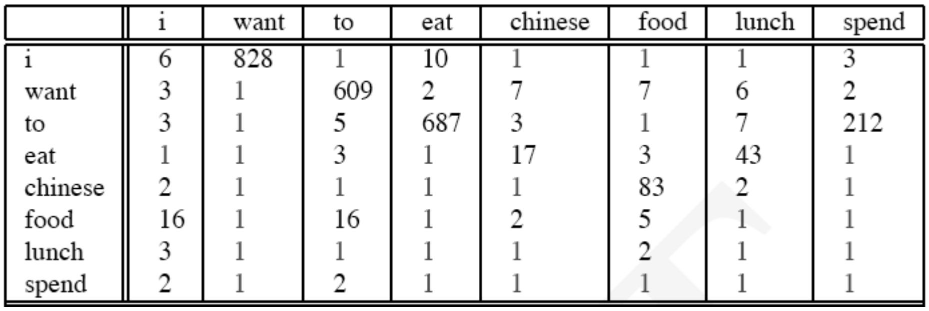
- **Make the zero counts 1.**
- **Rationale: They're just events you haven't seen yet. If you had seen them, chances are you would only have seen them once… so make the count equal to 1.**
- **Caveat: Other than the name there's no reason to add 1, you can just as easily add some other fixed amount.**

Original BERP Counts



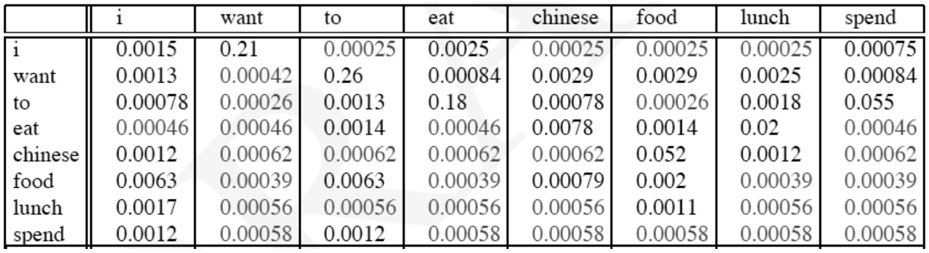
	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Add-1 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

Add-One Smoothed BERP Bigram Probs



	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

BERP Table: Original Bigram 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

Add-One Comments

- **Pros**
  - **Easy**
- **Cons**
  - **Doesn’t work very well.**
  - **Technical: Moves too much of the probability mass to the zero events and away from the events that actually occurred.**
  - **Intuitive: Makes too many of the zeroes too big, making the things that occurred look less likely than they really are.**

Better Approaches

- Good-Turing, Witten-Bell, Kneiser-Ney
- Think about events that have never happened in the same vein as things that have happened once…
- Why?
  - Well but for dumb luck they might have happened
  - And from what we know about the Zipf-like distribution of things, they likely would only have occurred once.

Fishing Metaphor

- You're out fishing in a lake with 7 kinds of fish.
- You've caught
  - 10 carp
  - 3 perch
  - 2 whitefish
  - 1 trout
  - 1 salmon
  - 1 eel

What's the probability that the next fish caught will be from an unseen species?

Fishing Metaphor

- Well if it was it would then be a species with a count of 1.
- There were 3 events like this from the total of 18 events.
- So let's make the probability of a new species showing up be 3/18.
  - That is use the prob of the 1s to reestimate the prob of the 0s.

But

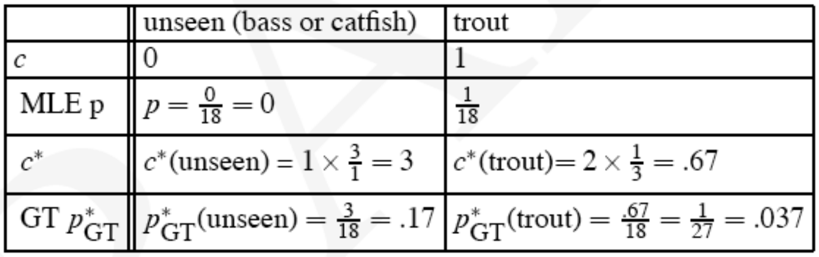
- But now what's the probability of a trout? Can't still be 1/18. We stole too much of the probability mass to give to the zeroes. It has to be lower.
- So if the 0s are like 1s then what are the 1s like?

Good-Turing

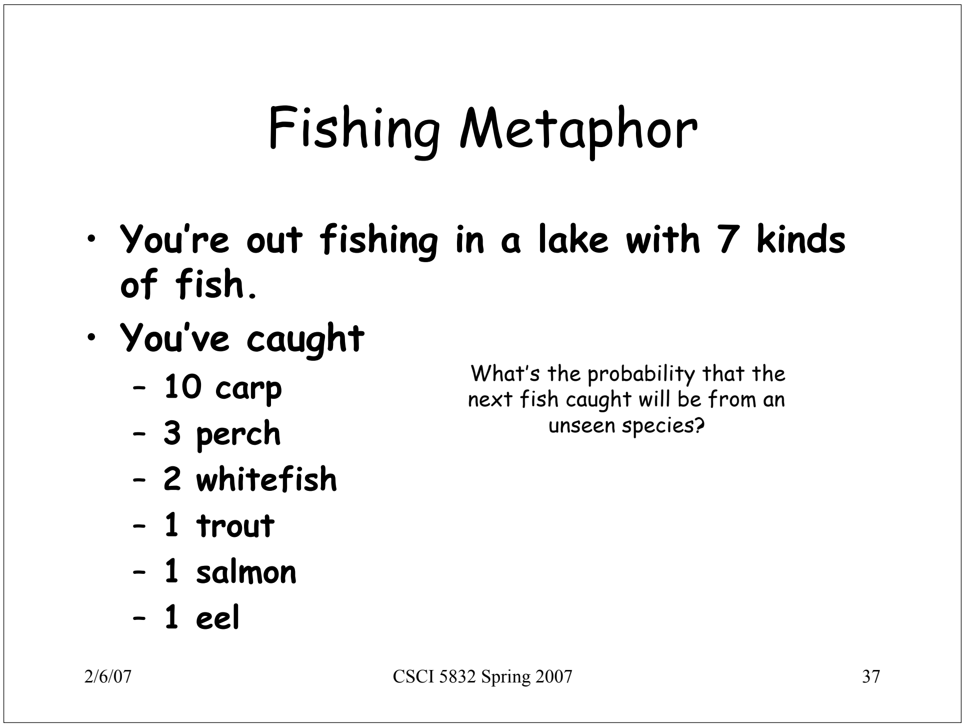
The reestimated count for a given bucket is

$$c^* = (c+1)\frac{N_{c+1}}{N_c}$$

Good Turing



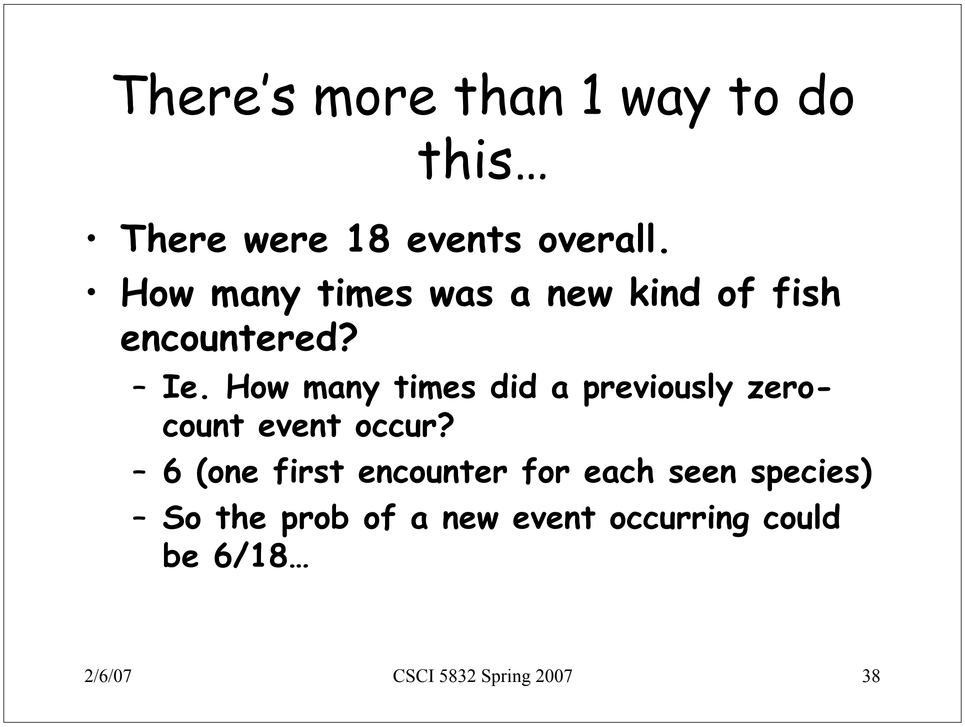
	unseen (bass or catfish)	trout
c	0	1
MLE p	$p = \frac{0}{18} = 0$	$\frac{1}{18}$
c^*	$c^*(\text{unseen}) = 1 \times \frac{3}{1} = 3$	$c^*(\text{trout}) = 2 \times \frac{1}{3} = .67$
GT p^*_{GT}	$p^*_{\text{GT}}(\text{unseen}) = \frac{3}{18} = .17$	$p^*_{\text{GT}}(\text{trout}) = \frac{.67}{18} = \frac{1}{27} = .037$



Fishing Metaphor

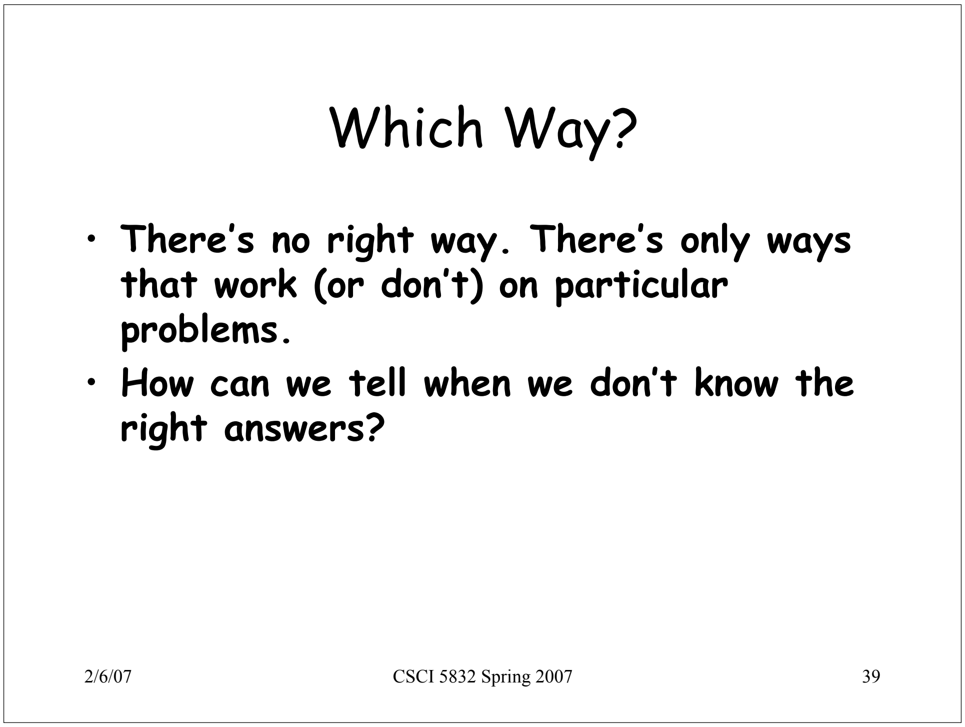
- You're out fishing in a lake with 7 kinds of fish.
- You've caught
  - 10 carp
  - 3 perch
  - 2 whitefish
  - 1 trout
  - 1 salmon
  - 1 eel

What's the probability that the next fish caught will be from an unseen species?



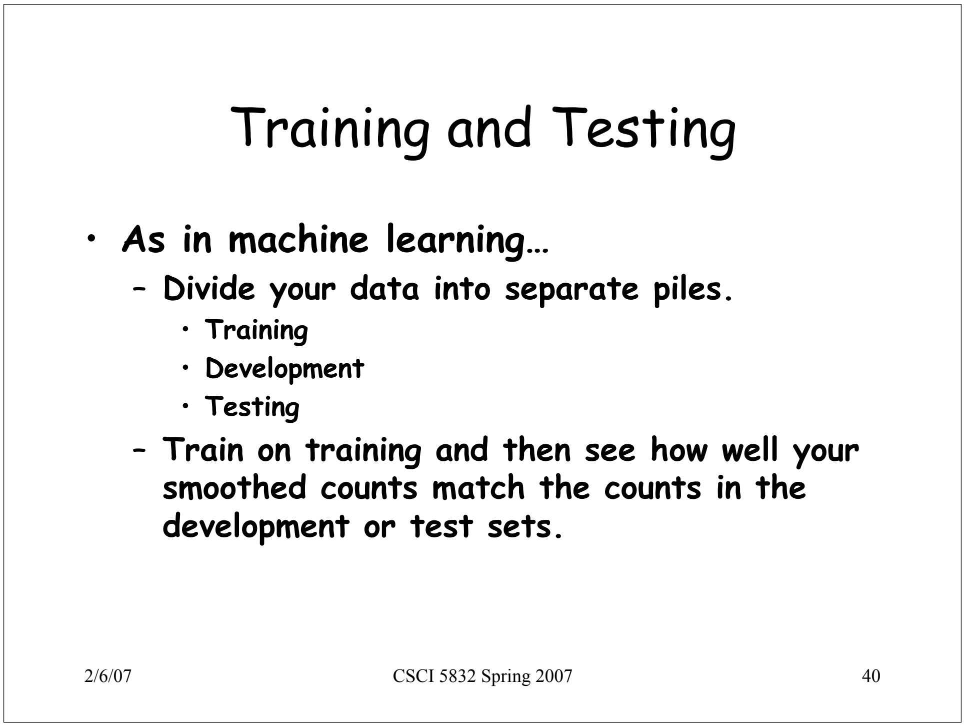
There's more than 1 way to do this…

- There were 18 events overall.
- How many times was a new kind of fish encountered?
  - Ie. How many times did a previously zero-count event occur?
  - 6 (one first encounter for each seen species)
  - So the prob of a new event occurring could be 6/18…



Which Way?

- There’s no right way. There’s only ways that work (or don’t) on particular problems.
- How can we tell when we don’t know the right answers?



Training and Testing

- As in machine learning…
  - Divide your data into separate piles.
    - Training
    - Development
    - Testing
  - Train on training and then see how well your smoothed counts match the counts in the development or test sets.

Next time

- **Thursday we'll do backoff and start on Chapter 5 (parts of speech and part of speech tagging.)**