CSCI 5832
Natural Language Processing

Jim Martin

Lecture 7

2/7/08 1

Today 2/5

- Review LM basics
  - Chain rule
  - Markov Assumptions
- Why should you care?
- Remaining issues
  - Unknown words
  - Evaluation
  - Smoothing
  - Backoff and Interpolation

2/7/08 2

Language Modeling

- We want to compute P(w1,w2,w3,w4,w5…wn), the probability of a sequence
- Alternatively we want to compute P(w5|w1,w2,w3,w4,w5): the probability of a word given some previous words
- The model that computes P(W) or P(wn|w1,w2…wn-1) is called the language model.

2/7/08 3

Computing P(W)

- How to compute this joint probability:
  - P(“the”,”other”,”day”,”I”,”was”,”walking”,”along” ,”and”,”saw”,”a”,”lizard”)
- Intuition: let’s rely on the Chain Rule of Probability

2/7/08 4

The Chain Rule

- Recall the definition of conditional probabilities
- Rewriting: [illegible]

[illegible]

- More generally
- P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)
- In general
- $P(x_1,x_2,x_3,\ldots x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\ldots P(x_n|x_1\ldots x_{n-1})$

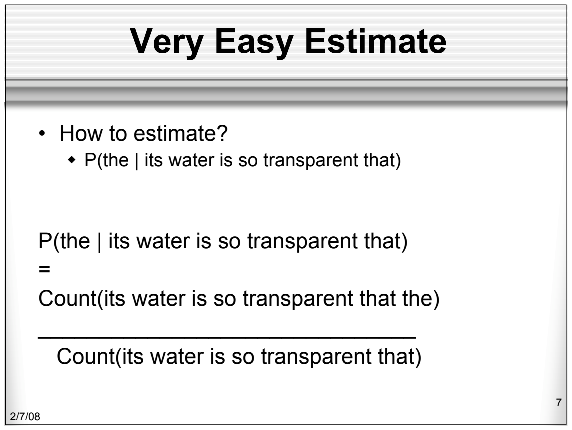
2/7/08 5

The Chain Rule

$$\begin{aligned} P(w_1^n) &= P(w_1)P(w_2|w_1)P(w_3|w_1^2)\ldots P(w_n|w_1^{n-1}) \\ &= \prod_{k=1}^{n} P(w_k|w_1^{k-1}) \end{aligned}$$

- P(“the big red dog was”)=
- P(the)*P(big|the)*P(red|the big)*P(dog|the big red)*P(was|the big red dog)

2/7/08 6

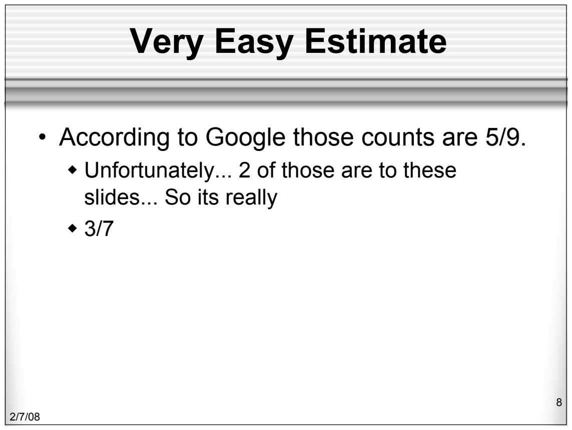


Very Easy Estimate

- How to estimate?
  - P(the | its water is so transparent that)

P(the | its water is so transparent that)
=
Count(its water is so transparent that the)
____
Count(its water is so transparent that)

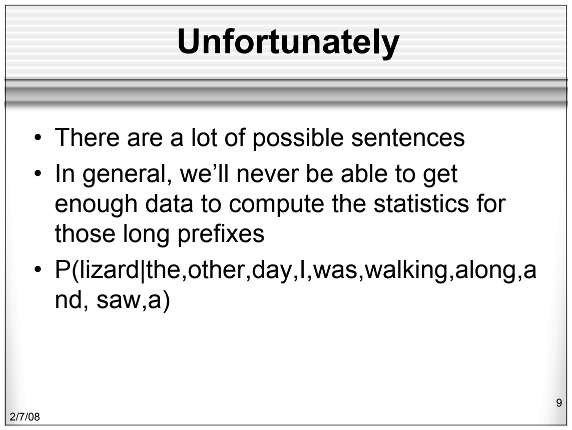
2/7/08 7



Very Easy Estimate

- According to Google those counts are 5/9.
  - Unfortunately... 2 of those are to these slides... So its really
  - 3/7

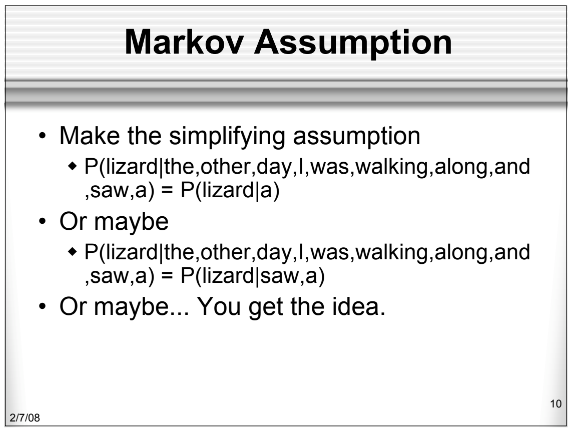
2/7/08 8



Unfortunately

- There are a lot of possible sentences
- In general, we'll never be able to get enough data to compute the statistics for those long prefixes
- P(lizard|the,other,day,I,was,walking,along,a nd, saw,a)

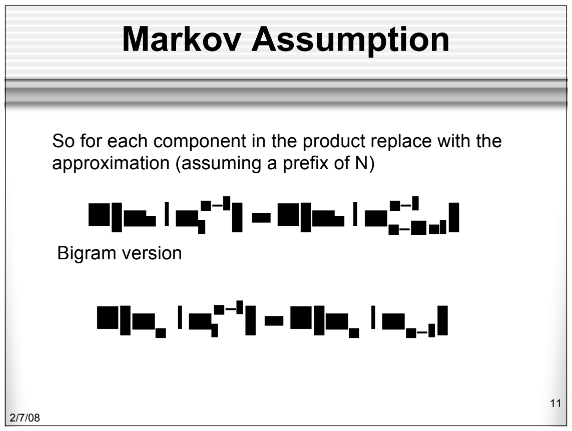
2/7/08 9



Markov Assumption

- Make the simplifying assumption
  - P(lizard|the,other,day,I,was,walking,along,and ,saw,a) = P(lizard|a)
- Or maybe
  - P(lizard|the,other,day,I,was,walking,along,and ,saw,a) = P(lizard|saw,a)
- Or maybe... You get the idea.

2/7/08 10



Markov Assumption

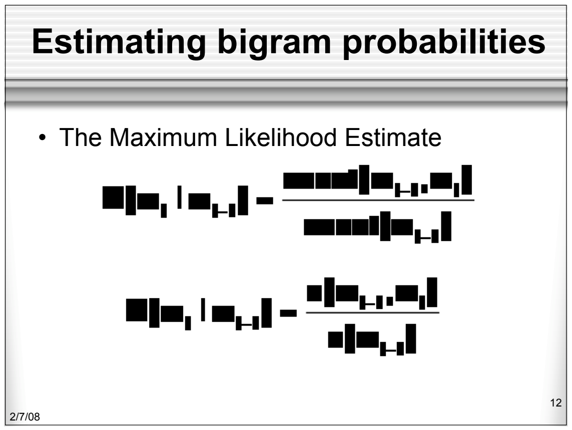
So for each component in the product replace with the approximation (assuming a prefix of N)

[illegible]

Bigram version

[illegible]

2/7/08 11



Estimating bigram probabilities

- The Maximum Likelihood Estimate

[illegible]

[illegible]

2/7/08 12

An example

- <s> I am Sam </s>
- <s> Sam I am </s>
- <s> I do not like green eggs and ham </s>

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

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

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

2/7/08 13

Maximum Likelihood Estimates

- The maximum likelihood estimate of some parameter of a model M from a training set T
  - Is the estimate that maximizes the likelihood of the training set T given the model M
- Suppose the word Chinese occurs 400 times in a corpus of a million words (Brown corpus)
- What is the probability that a random word from some other text from the same distribution will be “Chinese”
- MLE estimate is 400/1000000 = .004
  - This may be a bad estimate for some other corpus
- But it is the **estimate** that makes it **most likely** that “Chinese” will occur 400 times in a million word corpus.

2/7/08 14

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*

2/7/08 15

Raw Bigram Counts

- Out of 9222 sentences: Count(col | row)

	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

2/7/08 16

Raw Bigram Probabilities

- Normalize by unigrams:

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

	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

2/7/08 17

Bigram Estimates of Sentence Probabilities

- P(<s> I want english food </s>) =
  p(i|<s>) x p(want|I) x p(english|want)
  x p(food|english) x p(</s>|food)
  =.000031

2/7/08 18

Kinds of knowledge?

- P(english|want) = .0011
- P(chinese|want) = .0065
- P(to|want) = .66
- P(eat | to) = .28
- P(food | to) = 0
- P(want | spend) = 0
- P (i | <s>) = .25

• World knowledge

•Syntax

•Discourse

2/7/08 19

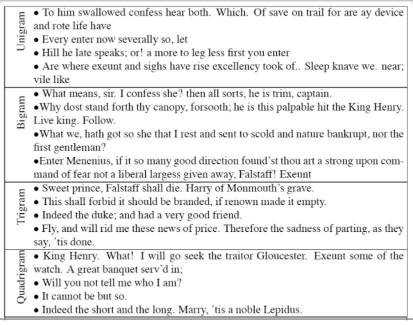
The Shannon Visualization Method

- Generate random sentences:
- 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>
```

2/7/08 20

Shakespeare



Unigram	• To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have • Every enter now severally so, let • Hill he late speaks; or! a more to leg less first you enter • Are where exeunt and sighs have rise excellency took of.. Sleep knave we. near; vile like
Bigram	• What means, sir. I confess she? then all sorts, he is trim, captain. •Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. •What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman? •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
Trigram	• Sweet prince, Falstaff shall die. Harry of Monmouth's grave. • This shall forbid it should be branded, if renown made it empty. • Indeed the duke; and had a very good friend. • Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.
Quadrigram	• 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.

2/7/08 21

Shakespeare as 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

2/7/08 22

The Wall Street Journal is Not Shakespeare

unigram: Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives

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

trigram: They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

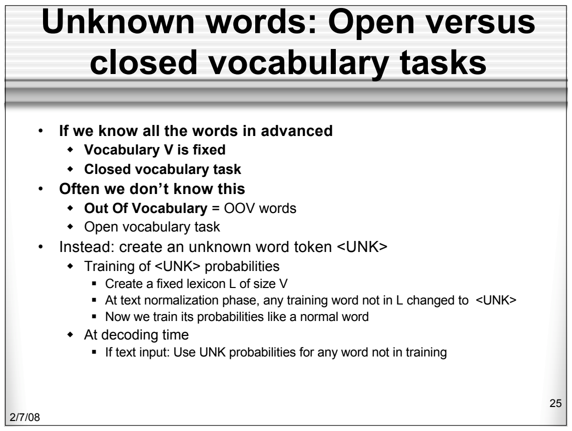
2/7/08 23

Why?

- Why would anyone want the probability of a sequence of words?
- Typically because of

[illegible]

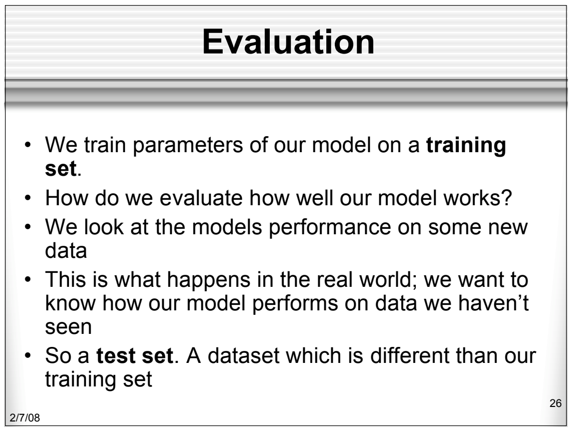
2/7/08 24



Unknown words: Open versus closed vocabulary tasks

- **If we know all the words in advanced**
  - **Vocabulary V is fixed**
  - **Closed vocabulary task**
- **Often we don't know this**
  - **Out Of Vocabulary** = OOV words
  - Open vocabulary task
- Instead: create an unknown word token <UNK>
  - Training of <UNK> probabilities
    - Create a fixed lexicon L of size V
    - At text normalization phase, any training word not in L changed to <UNK>
    - Now we train its probabilities like a normal word
  - At decoding time
    - If text input: Use UNK probabilities for any word not in training

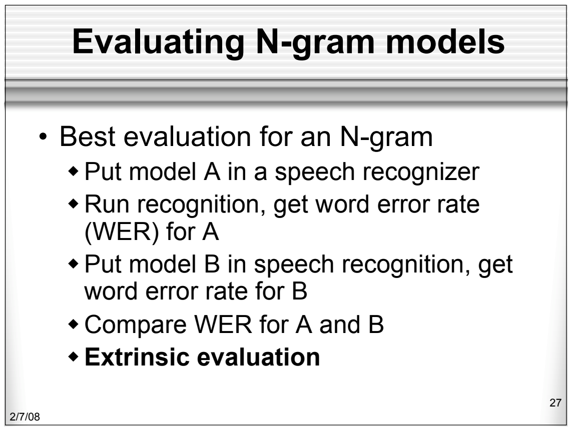
2/7/08 25



Evaluation

- We train parameters of our model on a **training set**.
- How do we evaluate how well our model works?
- We look at the models performance on some new data
- This is what happens in the real world; we want to know how our model performs on data we haven't seen
- So a **test set**. A dataset which is different than our training set

2/7/08 26



Evaluating N-gram models

- Best evaluation for an N-gram
  - Put model A in a speech recognizer
  - Run recognition, get word error rate (WER) for A
  - Put model B in speech recognition, get word error rate for B
  - Compare WER for A and B
  - **Extrinsic evaluation**

2/7/08 27

Difficulty of extrinsic (in-vivo) evaluation of N-gram models

- Extrinsic evaluation
  - This is really time-consuming
  - Can take days to run an experiment
- So
  - As a temporary solution, in order to run experiments
  - To evaluate N-grams we often use an **intrinsic** evaluation, an approximation called **perplexity**
  - But perplexity is a poor approximation unless the test data looks **just** like the training data
  - So is **generally only useful in pilot experiments (generally is not sufficient to publish)**
  - But is helpful to think about.

2/7/08 28

Perplexity

- Perplexity is the probability of the test set (assigned by the language model), normalized by the number of words:

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

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

2/7/08 29

A Different Perplexity Intuition

- How hard is the task of recognizing digits '0,1,2,3,4,5,6,7,8,9': pretty easy
- How hard is recognizing (30,000) names at Microsoft. Hard: perplexity = 30,000
- Perplexity is the weighted equivalent branching factor provided by your model

Slide from Josh Goodman

2/7/08 30

Lower perplexity = better model

- Training 38 million words, test 1.5 million wo

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

2/7/08 31

Lesson 1: the perils of overfitting

- N-grams only work well for word prediction if the test corpus looks like the training corpus
  - In real life, it often doesn't
  - We need to train robust models, adapt to test set, etc

2/7/08 32

Lesson 2: zeros or not?

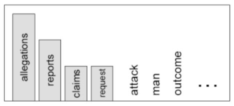
- Zipf's Law:
  - A small number of events occur with high frequency
  - A large number of events occur with low frequency
  - You can quickly collect statistics on the high frequency events
  - You might have to wait an arbitrarily long time to get valid statistics on low frequency events
- Result:
  - Our estimates are sparse! no counts at all for the vast bulk of things we want to estimate!
  - Some of the zeroes in the table are really zeros But others are simply low frequency events you haven't seen yet. After all, ANYTHING CAN HAPPEN!
  - How to address?
- Answer:
  - Estimate the likelihood of unseen N-grams!

2/7/08 33

Smoothing is like Robin Hood: Steal from the rich and give to the poor (in probability mass)

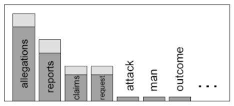
- We often want to make predictions from sparse statistics:

P(w | denied the)
3 allegations
2 reports
1 claims
1 request
7 total



- Smoothing flattens spiky distributions so they generalize better

P(w | denied the)
2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other
7 total



- Very important all over NLP, but easy to do badly!

Laplace smoothing

- Also called add-one smoothing
- Just add one to all the counts!
- Very simple

$$P(w_i) = \frac{c_i}{N}$$

- MLE estimate:

$$P_{\text{Laplace}}(w_i) = \frac{c_i + 1}{N + V}$$

- Laplace estimate:

$$c_i^* = (c_i + 1)\frac{N}{N + V}$$

- Reconstructed counts:

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

2/7/08 37

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

2/7/08 38

Big Changes to Counts

- C(count to) went from 608 to 238!
- P(to|want) from .66 to .26!
- Discount d= c*/c
  - d for "chinese food" =.10!!! A 10x reduction
  - So in general, Laplace is a blunt instrument
  - Could use more fine-grained method (add-k)
- Despite its flaws Laplace (add-k) is however still used to smooth other probabilistic models in NLP, especially
  - For pilot studies
  - in domains where the number of zeros isn't so huge.

2/7/08 39

Better Discounting Methods

- Intuition used by many smoothing algorithms
  - Good-Turing
  - Kneser-Ney
  - Witten-Bell
- Is to use the count of things we've seen once to help estimate the count of things we've never seen

2/7/08 40

Good-Turing

- Imagine you are fishing
  - There are 8 species: carp, perch, whitefish, trout, salmon, eel, catfish, bass
- You have caught
  - 10 carp, 3 perch, 2 whitefish, 1 trout, 1 salmon, 1 eel = 18 fish (tokens)
    = 6 species (types)
- How likely is it that you'll next see another trout?

2/7/08 41

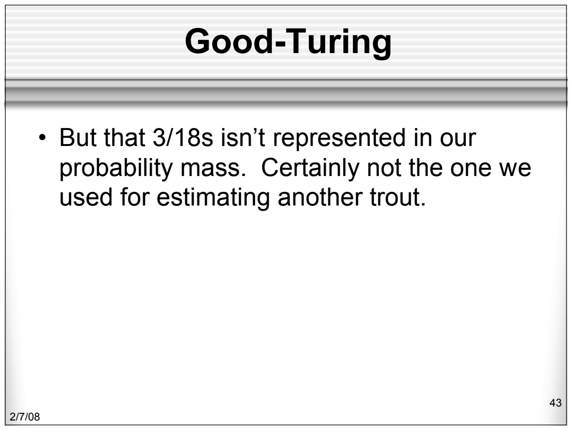
Good-Turing

- Now how likely is it that next species is new (i.e. catfish or bass)

There were 18 distinct events... 3 of those represent singleton species

3/18

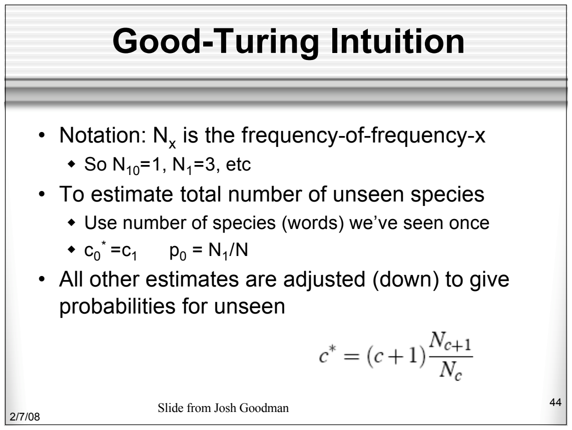
2/7/08 42



Good-Turing

- But that 3/18s isn't represented in our probability mass. Certainly not the one we used for estimating another trout.

2/7/08 43



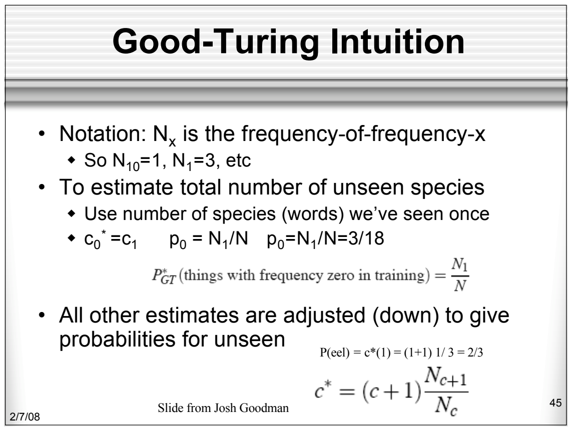
Good-Turing Intuition

- Notation: N_x is the frequency-of-frequency-x
  - So N_{10}=1, N_1=3, etc
- To estimate total number of unseen species
  - Use number of species (words) we've seen once
  - $c_0^* = c_1$ $p_0 = N_1/N$
- All other estimates are adjusted (down) to give probabilities for unseen

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

Slide from Josh Goodman

2/7/08 44



Good-Turing Intuition

- Notation: N_x is the frequency-of-frequency-x
  - So N_{10}=1, N_1=3, etc
- To estimate total number of unseen species
  - Use number of species (words) we've seen once
  - $c_0^* = c_1$ $p_0 = N_1/N$ $p_0 = N_1/N = 3/18$

$$P^*_{GT}(\text{things with frequency zero in training}) = \frac{N_1}{N}$$

- All other estimates are adjusted (down) to give probabilities for unseen

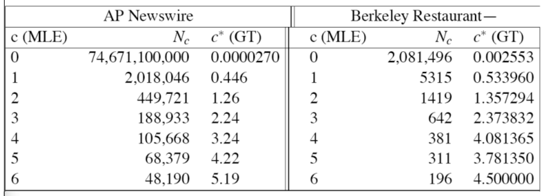
P(eel) = c*(1) = (1+1) 1/ 3 = 2/3

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

Slide from Josh Goodman

2/7/08 45

Bigram frequencies of frequencies and GT re-estimates



AP Newswire			Berkeley Restaurant—		
c (MLE)	N_c	c^* (GT)	c (MLE)	N_c	c^* (GT)
0	74,671,100,000	0.0000270	0	2,081,496	0.002553
1	2,018,046	0.446	1	5315	0.533960
2	449,721	1.26	2	1419	1.357294
3	188,933	2.24	3	642	2.373832
4	105,668	3.24	4	381	4.081365
5	68,379	4.22	5	311	3.781350
6	48,190	5.19	6	196	4.500000

2/7/08 46

Backoff and Interpolation

- Another really useful source of knowledge
- If we are estimating:
  - trigram p(z|xy)
  - but c(xyz) is zero
- Use info from:
  - Bigram p(z|y)
- Or even:
  - Unigram p(z)
- How to combine the trigram/bigram/unigram info?

2/7/08 47

Backoff versus interpolation

- **Backoff**: use trigram if you have it, otherwise bigram, otherwise unigram
- **Interpolation**: mix all three

2/7/08 48

Interpolation

- Simple interpolation

$$\sum_i \lambda_i = 1$$

$$\begin{aligned}\hat{P}(w_n|w_{n-1}w_{n-2}) &= \lambda_1 P(w_n|w_{n-1}w_{n-2}) \\ &+\lambda_2 P(w_n|w_{n-1}) \\ &+\lambda_3 P(w_n)\end{aligned}$$

- Lambdas conditional on context:

$$\begin{aligned}\hat{P}(w_n|w_{n-2}w_{n-1}) &= \lambda_1(w_{n-2}^{n-1})P(w_n|w_{n-2}w_{n-1}) \\ &+\lambda_2(w_{n-2}^{n-1})P(w_n|w_{n-1}) \\ &+\lambda_3(w_{n-2}^{n-1})P(w_n)\end{aligned}$$

2/7/08 49

How to set the lambdas?

- Use a **held-out** corpus
- Choose lambdas which maximize the probability of some held-out data
  - I.e. fix the N-gram probabilities
  - Then search for lambda values
  - That when plugged into previous equation
  - Give largest probability for held-out set
  - Can use EM to do this search

2/7/08 50

GT smoothed bigram probs

	i	want	to	eat	chinese	food	lunch	spend
i	0.0014	0.326	0.00248	0.00355	0.000205	0.0017	0.00073	0.000489
want	0.00134	0.00152	0.656	0.000483	0.00455	0.00455	0.00384	0.000483
to	0.000512	0.00152	0.00165	0.284	0.000512	0.0017	0.00175	0.0873
eat	0.00101	0.00152	0.00166	0.00189	0.0214	0.00166	0.0563	0.000585
chinese	0.00283	0.00152	0.00248	0.00189	0.000205	0.519	0.00283	0.000585
food	0.0137	0.00152	0.0137	0.00189	0.000409	0.00366	0.00073	0.000585
lunch	0.00363	0.00152	0.00248	0.00189	0.000205	0.00131	0.00073	0.000585
spend	0.00161	0.00152	0.00161	0.00189	0.000205	0.0017	0.00073	0.000585

2/7/08 51

OOV words: <UNK> word

- **Out Of Vocabulary** = OOV words
- We don't use GT smoothing for these
  - Because GT assumes we know the number of unseen events
- Instead: create an unknown word token <UNK>
  - Training of <UNK> probabilities
    - Create a fixed lexicon L of size V
    - At text normalization phase, any training word not in L changed to <UNK>
    - Now we train its probabilities like a normal word
  - At decoding time
    - If text input: Use UNK probabilities for any word not in training

2/7/08 52

Practical Issues

- We do everything in log space
  - Avoid underflow

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

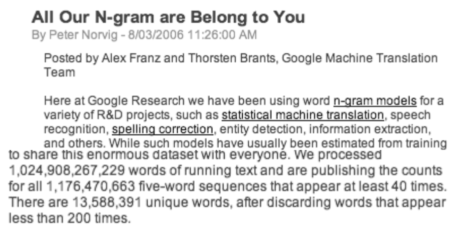
2/7/08 53

Language Modeling Toolkits

- SRILM
- CMU-Cambridge LM Toolkit

2/7/08 54

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

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.

2/7/08 55

Google N-Gram Release

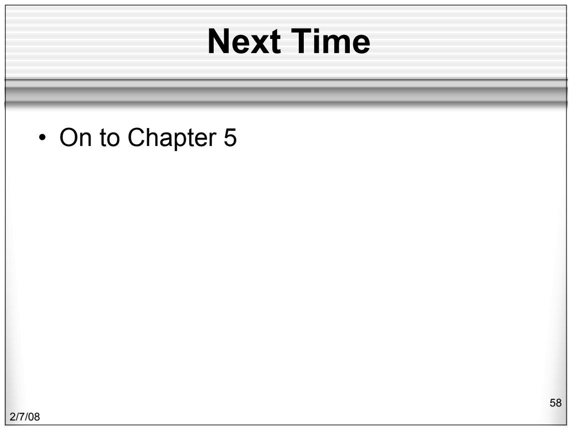
- `serve as the incoming 92`
- `serve as the incubator 99`
- `serve as the independent 794`
- `serve as the index 223`
- `serve as the indication 72`
- `serve as the indicator 120`
- `serve as the indicators 45`
- `serve as the indispensable 111`
- `serve as the indispensible 40`
- `serve as the individual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

2/7/08 56

Summary

- Probability
  - Basic probability
  - Conditional probability
  - Bayes Rule
- Language Modeling (N-grams)
  - N-gram Intro
  - The Chain Rule
    - Perplexity
  - Smoothing:
    - Add-1
    - Good-Turing

2/7/08 57



Next Time

- On to Chapter 5

2/7/08

58