

## Statistical NLP Spring 2007

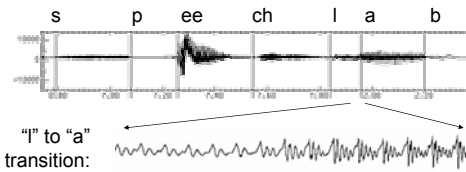


### Lecture 2: Language Models

Dan Klein – UC Berkeley

## Speech in a Slide (or Three)

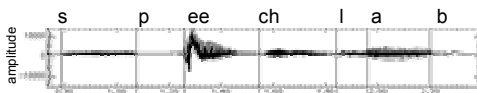
- Speech input is an acoustic wave form



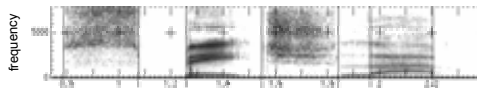
Graphs from Simon Amfield's web tutorial on speech, Sheffield:  
<http://www.psyc.leeds.ac.uk/research/cogn/speech/tutorial/>  
Some later bits from Joshua Goodman's LM tutorial

## Spectral Analysis

- Frequency gives pitch; amplitude gives volume
  - sampling at ~8 kHz phone, ~16 kHz mic (kHz=1000 cycles/sec)

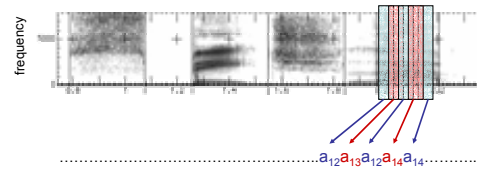


- Fourier transform of wave displayed as a spectrogram
  - darkness indicates energy at each frequency



## Acoustic Feature Sequence

- Time slices are translated into acoustic feature vectors (~15 real numbers per slice, details later in the term)



- Now we have to figure out a mapping from sequences of acoustic observations to words.

## The Speech Recognition Problem

- We want to predict a sentence given an acoustic sequence:

$$s^* = \arg \max_s P(s | a)$$

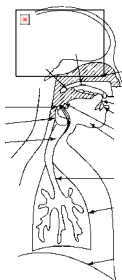
- The noisy channel approach:
  - Build a generative model of production (encoding)

$$P(a, s) = P(s)P(a | s)$$

- To decode, we use Bayes' rule to write

$$\begin{aligned} s^* &= \arg \max_s P(s | a) \\ &= \arg \max_s P(s)P(a | s) / P(a) \\ &= \arg \max_s P(s)P(a | s) \end{aligned}$$

- Now, we have to find a sentence maximizing this product
- Why is this progress?

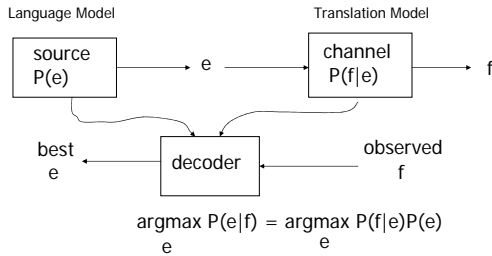


## Just a Code?

- "Also knowing nothing official about, but having guessed and inferred considerable about, the powerful new mechanized methods in cryptography—methods which I believe succeed even when one does not know what language has been coded—one naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. When I look at an article in Russian, I say: 'This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode.'"

- Warren Weaver (1955:18, quoting a letter he wrote in 1947)

## MT System Components



## Other Noisy-Channel Processes

- Handwriting recognition

$$P(\text{text} | \text{strokes}) \propto P(\text{text})P(\text{strokes} | \text{text})$$

- OCR

$$P(\text{text} | \text{pixels}) \propto P(\text{text})P(\text{pixels} | \text{text})$$

- Spelling Correction

$$P(\text{text} | \text{typos}) \propto P(\text{text})P(\text{typos} | \text{text})$$

- Translation

$$P(\text{english} | \text{french}) \propto P(\text{english})P(\text{french} | \text{english})$$

## Probabilistic Language Models

- Want to build models which assign scores to sentences.
  - $P(\text{I saw a van}) \gg P(\text{eyes awe of an})$
  - Not really grammaticality:  $P(\text{artichokes intimidate zippers}) \approx 0$
- One option: empirical distribution over sentences?
  - Problem: doesn't generalize (at all)
- Two ways of generalizing
  - Decomposition: sentences generated in small steps which can be recombined in other ways
  - Smoothing: allow for the possibility of unseen events

## N-Gram Language Models

- No loss of generality to break sentence probability down with the chain rule

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

- Too many histories!

- $P(??? | \text{No loss of generality to break sentence}) ?$
- $P(??? | \text{the water is so transparent that}) ?$

- N-gram solution: assume each word depends only on a short linear history

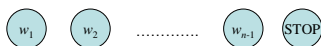
$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i | w_{i-k} \dots w_{i-1})$$

## Unigram Models

- Simplest case: unigrams

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i)$$

- Generative process: pick a word, pick a word, ...
- As a graphical model:



- To make this a proper distribution over sentences, we have to generate a special STOP symbol last. (Why?)

- Examples:

- [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]
- []
- [after, any, on, consistently, hospital, lake, of, of, other, and, factors, raised, analyst, too, allowed, mexico, never, consider, fall, bungled, davison, that, obtain, price, lines, the, to, sass, the, the, further, board, a, details, machinists, the, companies, which, rivals, an, because, longer, oakes, percent, a, they, three, edward, it, currier, an, within, in, three, wrote, is, you, s., longer, institute, dentistry, pay, however, said, possible, to, rooms, hiding, eggs, approximate, financial, canada, the, so, workers, advancers, half, between, nasdaq]

## Bigram Models

- Big problem with unigrams:  $P(\text{the the the the}) \gg P(\text{I like ice cream})!$
- Condition on previous word:

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



- Any better?

- [texaco, rose, one, in, this, issue, is, pursuing, growth, in, a, boiler, house, said, mr., gurria, mexico, 's, motion, control, proposal, without, permission, from, five, hundred, fifty, five, yen]
- [outside, new, car, parking, lot, of, the, agreement, reached]
- [although, common, shares, rose, forty, six, point, four, hundred, dollars, from, thirty, seconds, at, the, greatest, play, disingenuous, to, be, reset, annually, the, buy, out, of, american, brands, vying, for, mr., womack, currently, sharedata, incorporated, believe, chemical, prices, undoubtedly, will, be, as, much, is, scheduled, to, conscientious, teaching]
- [this, would, be, a, record, november]

## More N-Gram Examples

Unigram	<ul style="list-style-type: none"> <li>To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have</li> <li>Every enter now severally so, let</li> <li>Hill he late speaks; or! a more to leg less first you enter</li> <li>Are where exeunt and sighs have rise excellency took of.. Sleep knave we. near; vile like</li> </ul>
Bigram	<ul style="list-style-type: none"> <li>What means, sir. I confess she? then all sorts, he is trim, captain.</li> <li>Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.</li> <li>What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman?</li> <li>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</li> </ul>
Trigram	<ul style="list-style-type: none"> <li>Sweet prince, Falstaff shall die. Harry of Monmouth's grave.</li> <li>This shall forbid it should be branded, if renown made it empty.</li> <li>Indeed the duke, and had a very good friend.</li> <li>Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.</li> </ul>
Quadrigram	<ul style="list-style-type: none"> <li>King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in;</li> <li>Will you not tell me who I am?</li> <li>It cannot be but so.</li> <li>Indeed the short and the long. Marry, 'tis a noble Lepidus.</li> </ul>

## Regular Languages?

- N-gram models are (weighted) regular processes
  - Why can't we model language like this?
    - Linguists have many arguments why language can't be merely regular.
    - Long-distance effects:
      - "The computer which I had just put into the machine room on the fifth floor crashed."
  - Why CAN we often get away with n-gram models?
- PCFG LM (later):
  - [This, quarter, 's, surprisingly, independent, attack, paid, off, the, risk, involving, IRS, leaders, and, transportation, prices, .]
  - [It, could, be, announced, sometime, .]
  - [Mr., Toseland, believes, the, average, defense, economy, is, drafted, from, slightly, more, than, 12, stocks, .]

## Is This Working?

- The game isn't to pound out fake sentences!
- What we really want to know is:
  - Will our model prefer good sentences to bad ones?
  - Bad ≠ ungrammatical!
  - Bad ≈ unlikely
  - Bad = sentences that our acoustic model really likes but aren't the correct answer

## Measuring Model Quality

- Word Error Rate (WER)  $\frac{\text{insertions} + \text{deletions} + \text{substitutions}}{\text{true sentence size}}$ 
  - Correct answer: Andy saw a part of the movie
  - Recognizer output: And he saw apart of the movie
  - WER:  $\frac{4}{7} = 57\%$
- The "right" measure:
  - Task error driven
  - For speech recognition
  - For a specific recognizer!
- For general evaluation, we want a measure which references only good text, not mistake text

## Measuring Model Quality

- The Shannon Game:
  - How well can we predict the next word?
    - When I order pizza, I wipe off the \_\_\_
    - Many children are allergic to \_\_\_
    - I saw a \_\_\_
  - Unigrams are terrible at this game. (Why?)
    - grease 0.5
    - sauce 0.4
    - dust 0.05
    - ....
    - mice 0.0001
    - ....
    - the 1e-100
- The "Entropy" Measure
  - Really: average cross-entropy of a text according to a model
$$H(S|M) = \frac{\log_2 P_M(S)}{|S|} = \frac{\sum_i \log_2 P_M(s_i)}{\sum_i |s_i|} \quad \sum_i \log_2 P_M(w_j | w_{j-1})$$

## Measuring Model Quality

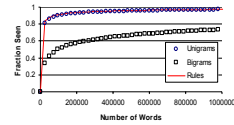
- Problem with entropy:
  - 0.1 bits of improvement doesn't sound so good
  - Solution: perplexity
- Note that even though our models require a stop step, people typically don't count it as a symbol when taking these averages.

$$P(S|M) = 2^{H(S|M)} = \frac{1}{\prod_{i=1}^n P_M(w_i | h)}$$

## Sparsity

### Problems with n-gram models:

- New words appear all the time:
  - Synaptitude
  - 132,701.03
  - fuzzification
- New bigrams: even more often
- Trigrams or more – still worse!



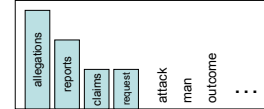
### Zipf's Law

- Types (words) vs. tokens (word occurrences)
- Broadly: most word types are rare ones
- Specifically:
  - Rank word types by token frequency
  - Frequency inversely proportional to rank
- Not special to language: randomly generated character strings have this property (try it!)

## Smoothing

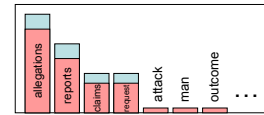
- We often want to make estimates from sparse statistics:

$P(w | \text{denied the})$   
 3 allegations  
 2 reports  
 1 claims  
 1 request  
 7 total



- Smoothing flattens spiky distributions so they generalize better

$P(w | \text{denied the})$   
 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!
- We'll illustrate with bigrams today (h = previous word, could be anything).

## Smoothing: Add-One, Etc.

$c$	number of word tokens in training data
$c(w)$	count of word $w$ in training data
$c(w, w_{-1})$	count of word $w$ following word $w_{-1}$
$V$	total vocabulary size (assumed known)
$N_k$	number of word types with count $k$

### One class of smoothing functions:

- Add-one / delta: assumes a uniform prior

$$P_{\text{ADD-}\delta}(w | w_{-1}) = \frac{c(w, w_{-1}) + \delta(1/V)}{c(w_{-1}) + \delta}$$

- Better to assume a unigram prior

$$P_{\text{UNI-PRIOR}}(w | w_{-1}) = \frac{c(w, w_{-1}) + \delta \hat{P}(w)}{c(w_{-1}) + \delta}$$

## Linear Interpolation

- One way to ease the sparsity problem for  $n$  grams is to use less sparse  $n-1$  gram estimates
- General linear interpolation:

$$P(w | w_{-1}) = [1 - \lambda(w, w_{-1})] \hat{P}(w | w_{-1}) + [\lambda(w, w_{-1})] P(w)$$

- Having a single global mixing constant is generally not ideal:

$$P(w | w_{-1}) = [1 - \lambda] \hat{P}(w | w_{-1}) + [\lambda] P(w)$$

- Solution: have different constant buckets
  - Buckets by count
  - Buckets by average count (better)

## Held-Out Data

- Important tool for getting models to generalize:



- When we have a small number of parameters that control the degree of smoothing, we set them to maximize the (log-)likelihood of held-out data

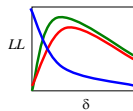
$$LL(w_1 \dots w_n | M(\lambda_1 \dots \lambda_k)) = \sum_i \log P_{M(\lambda_1 \dots \lambda_k)}(w_i | w_{i-1})$$

- Can use any optimization technique (line search or EM usually easiest)

- Examples:

$$P_{\text{LIN}(\lambda_1, \lambda_2)}(w | w_{-1}) = \lambda_1 \hat{P}(w | w_{-1}) + \lambda_2 \hat{P}(w)$$

$$P_{\text{UNI-PRIOR}(\delta)}(w | w_{-1}) = \frac{c(w, w_{-1}) + \delta \hat{P}(w)}{c(w_{-1}) + \delta}$$



## Held-Out Reweighting

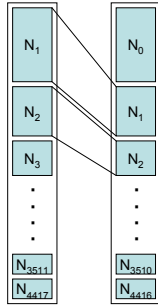
- What's wrong with unigram-prior smoothing?
- Let's look at some real bigram counts [Church and Gale 91]:

Count in 22M Words	Actual $c^*$ (Next 22M)	Add one's $c^*$	Add 0.0000027's $c^*$
1	0.448	2/7e 0	-1
2	1.25	3/7e 0	-2
3	2.24	4/7e 0	-3
4	3.23	5/7e 0	-4
5	4.21	6/7e 0	-5
Mass on New	9.2%	~100%	9.2%
Ratio of 2/1	2.8	1.5	~2

- Big things to notice:
  - Add-one vastly overestimates the fraction of new bigrams
  - Add-0.0000027 still underestimates the ratio 2/1\*
- One solution: use held-out data to predict the map of  $c$  to  $c^*$

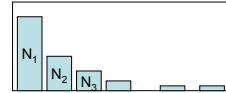
## Good-Turing Reweighting I

- We'd like to not need held-out data (why?)
- Idea: leave-one-out validation
  - Take each of the  $c$  training words out in turn
  - $c$  training sets of size  $c-1$ , held-out of size 1
  - What fraction of held-out words are unseen in training?
    - $N_i/c$
  - What fraction of held-out words are seen  $k$  times in training?
    - $(k+1)N_{k+1}/c$
  - So in the future we expect  $(k+1)N_{k+1}/c$  of the words to be those with training count  $k$
  - There are  $N_k$  words with training count  $k$
  - Each should occur with probability:
    - $(k+1)N_{k+1}/cN_k$
  - ...or expected count  $(k+1)N_{k+1}/N_k$

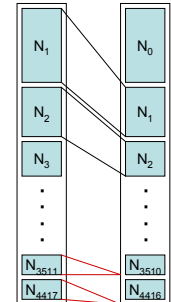
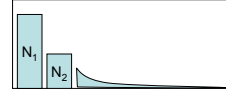


## Good-Turing Reweighting II

- Problem: what about "the"? (say  $c=4417$ )
  - For small  $k$ ,  $N_k > N_{k+1}$
  - For large  $k$ , too jumpy, zeros wreck estimates



- Simple Good-Turing [Gale and Sampson]: replace empirical  $N_k$  with a best-fit power law once count counts get unreliable



## Good-Turing Reweighting III

- Hypothesis: counts of  $k$  should be  $k^* = (k+1)N_{k+1}/N_k$

Count in 22M Words	Actual $c^*$ (Next 22M)	GT's $c^*$
1	0.448	0.446
2	1.25	1.26
3	2.24	2.24
4	3.23	3.24
Mass on New	9.2%	9.2%

- Katz Smoothing
  - Use GT discounted *bigram* counts (roughly - Katz left large counts alone)
  - Whatever mass is left goes to empirical unigram

$$P_{KATZ}(w | w_{-1}) = \frac{c^*(w, w_{-1})}{\sum_w c(w, w_{-1})} + \alpha(w_{-1})\hat{P}(w)$$

## Kneser-Ney Smoothing I

- Something's been very broken all this time
  - Shannon game: There was an unexpected \_\_\_\_?
    - delay?
    - Francisco?
  - "Francisco" is more common than "delay"
  - ... but "Francisco" always follows "San"
- Solution: Kneser-Ney smoothing
  - In the back-off model, we don't want the unigram probability of  $w$
  - Instead, probability given that we are observing a novel continuation
  - Every bigram type was a novel continuation the first time it was seen

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

## Kneser-Ney Smoothing II

- One more aspect to Kneser-Ney:
  - Look at the GT counts:

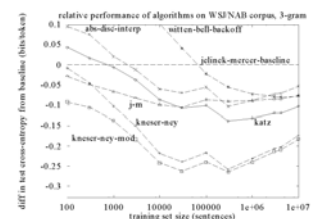
Count in 22M Words	Actual $c^*$ (Next 22M)	GT's $c^*$
1	0.448	0.446
2	1.25	1.26
3	2.24	2.24
4	3.23	3.24

- Absolute Discounting
  - Save ourselves some time and just subtract 0.75 (or some  $d$ )
  - Maybe have a separate value of  $d$  for very low counts

$$P_{KN}(w | w_{-1}) = \frac{c(w, w_{-1}) - D}{\sum_w c(w, w_{-1})} + \alpha(w_{-1})P_{CONTINUATION}(w)$$

## What Actually Works?

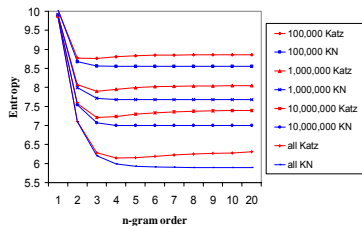
- Trigrams:
  - Unigrams, bigrams too little context
  - Trigrams much better (when there's enough data)
  - 4-, 5-grams usually not worth the cost (which is more than it seems, due to how speech recognizers are constructed)
- Good-Turing-like methods for count adjustment
  - Absolute discounting, Good-Turing, held-out estimation, Witten-Bell
- Kneser-Ney equalization for lower-order models
- See [Chen+Goodman] reading for tons of graphs!



[Graphs from Joshua Goodman]

## Data >> Method?

- Having more data is always good...



- ... but so is picking a better smoothing mechanism!
- $N > 3$  often not worth the cost (greater than you'd think)

## Beyond N-Gram LMs

### Caching Models

- Recent words more likely to appear again

$$P_{CACHE}(w | history) = \lambda P(w | w_{-1}w_{-2}) + (1 - \lambda) \frac{c(w \in history)}{|history|}$$

- Can be disastrous in practice for speech (why?)

### Skipping Models

$$P_{SKIP}(w | w_{-1}w_{-2}) = \lambda_1 \hat{P}(w | w_{-1}w_{-2}) + \lambda_2 P(w | w_{-1} \_ ) + \lambda_3 P(w | \_ w_{-2})$$

- Clustering Models: condition on word classes when words are too sparse
- Trigger Models: condition on bag of history words (e.g., maxent)
- Structured Models: use parse structure (we'll see these later)

## For Next Time

- Readings: J+M 2<sup>nd</sup> Ed Ch 4, Chen & Goodman (on web page)
- Assignment 1 out soon