

Statistical NLP

Spring 2008

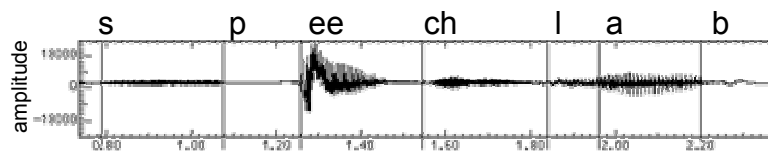


Lecture 2: Language Models

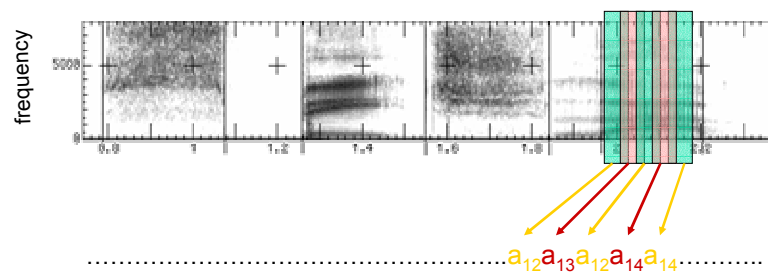
Dan Klein – UC Berkeley

Speech in a Slide

- Frequency gives pitch; amplitude gives volume



- Frequencies at each time slice processed into observation vectors



The Noisy-Channel Model

- We want to predict a sentence given acoustics:

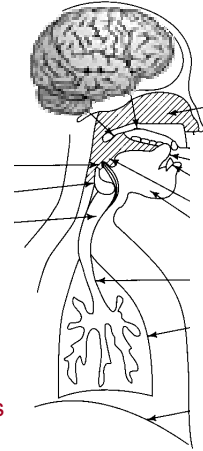
$$w^* = \arg \max_w P(w|a)$$

- The noisy channel approach:

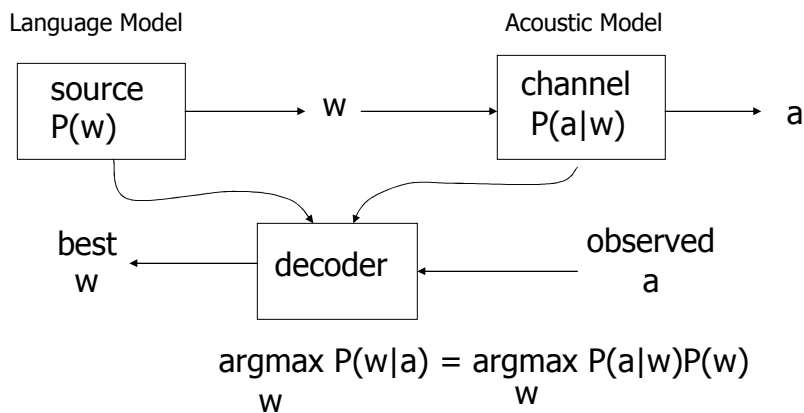
$$\begin{aligned} w^* &= \arg \max_w P(w|a) \\ &= \arg \max_w P(a|w)P(w)/P(a) \\ &\propto \arg \max_w P(a|w)P(w) \end{aligned}$$

Acoustic model: HMMs over word positions with mixtures of Gaussians as emissions

Language model: Distributions over sequences of words (sentences)



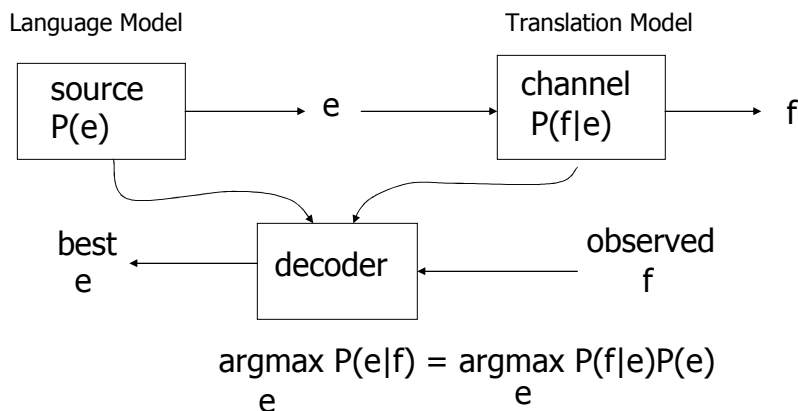
ASR System Components



Translation: Codebreaking?

- “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

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

- More...

Probabilistic Language Models

- Input: many observations of sentences x
- Output: model which assigns probabilities $P(x)$ to any sentence*
- Probabilities should broadly indicate likelihood of sentences
 - $P(\text{I saw a van}) \gg P(\text{eyes awe of an})$
 - *Not 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 incrementally using small steps which can be recombined in new ways
 - Smoothing: allow for the possibility of unseen events

N-Gram Language Models

- No loss of generality: break sentence probability down

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

- Too many histories!
 - P(??? | No loss of generality to break sentence) ?
 - P(??? | the water is so transparent that) ?
- N-gram models: assume each word depends only on a short linear history

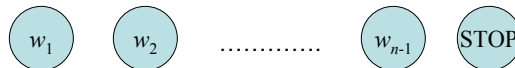
$$P(w_1 \dots w_n) = \prod_i P(w_i | w_{i-k} \dots w_{i-1})$$

Unigram Models

- Simplest case: unigrams

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

- 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?)

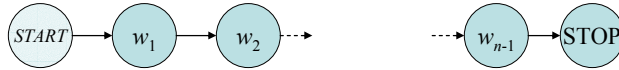
- Examples:

- [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:

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



- Obvious that this should help – in probabilistic terms, we're using weaker conditional independence assumptions (what's the cost?)
- 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
- 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

Regular Languages?

- N-gram models are (weighted) regular languages
 - Many linguistic arguments that language isn't regular.
 - Long-distance effects: "The computer which I had just put into the machine room on the fifth floor crashed."
 - Recursive structure
 - 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, .]

Model Parameters

- The parameters of an n-gram model:
 - The conditional probability estimates, we'll call them θ
 - Obvious estimate is the *relative frequency estimate* (aka the *maximum likelihood estimate*)

$$\hat{P}(w|w_{-1}) = \frac{c(w_{-1}, w)}{\sum_{w'} c(w_{-1}, w')}$$

- General method
 - Assume a training set X and a test set X'
 - Compute an estimate θ from X
 - Use it to assign probabilities to other sentences, X'

- Some quantities of interest

- Training likelihood

$$L(X|\theta) = \prod_{x \in X} P(x|\theta)$$

- Test likelihood

$$L(X'|\theta) = \prod_{x' \in X'} P(x'|\theta)$$

Is This Working?

- The game isn't to pound out fake sentences!
 - Obviously, generated sentences get "better" as we increase the model order
 - More precisely: using ML estimators, higher order is always better on train, but not test
- What we really want to know is:
 - Will our model prefer good sentences to bad ones?
 - Bad \neq ungrammatical!
 - Bad \approx 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

- 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 (why?)

WER: $\frac{4}{7}$
= 57%

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

- “Entropy”: really per-word test likelihood (misnamed)

$$H(X|\theta) = -\frac{1}{|X|} \sum_{x \in X} \log_2 P(x|\theta)$$

$$\sum_{x \in X} |x| \quad \leftarrow \quad \sum_i \log P(x_i|x_{i-1}, \theta)$$

Measuring Model Quality

- Problem with “entropy”:

- 0.1 bits of improvement doesn't sound so good
- Solution: perplexity

$$\text{perp}(X, \theta) = 2^{H(X|\theta)}$$

- Interpretation: average branching factor in model

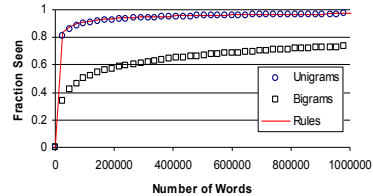
- Big notes:

- It's easy to get bogus perplexities by having bogus probabilities that sum to more than one over their event spaces. 30% of you will do this on HW1.
- Even though our models require a stop step, averages are per word, not per derivation step.

Sparsity

- Problems with n-gram models:

- New words appear all the time:
 - Synaptitude
 - 132,701.03
 - multidisciplinarization
- 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!)

Parameter Estimation

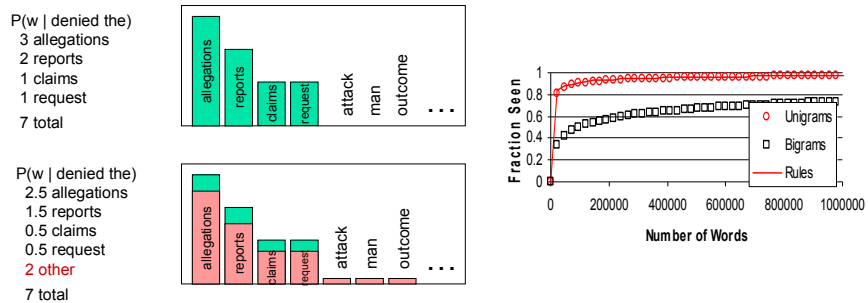
- Maximum likelihood estimates won't get us very far

$$\hat{P}(w|w_{-1}) = \frac{c(w_{-1}, w)}{\sum_{w'} c(w_{-1}, w')}$$

- Need to *smooth* these estimates
- General method (procedurally)
 - Take your empirical counts
 - Modify them in various ways to improve estimates
- General method (mathematically)
 - Often can give estimators a formal statistical interpretation
 - ... but not always
 - Stuff that works not always the same as stuff we can explain (yet!)

Smoothing

- Dealing with sparsity well: smoothing / shrinkage
 - For most histories $P(w | h)$, relatively few observations
 - Very intricately explored for the speech n-gram case
 - Easy to do badly



Priors on Parameters

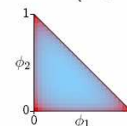
- Most obvious formal solution: use MAP estimate instead of ML estimate for a multinomial $P(X)$
- Maximum likelihood estimate: $\max P(X|\theta)$

$$\theta_{ML} = \frac{c(x)}{\sum_{x'} c(x')}$$

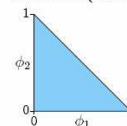
- MAP estimate: $\max P(\theta|X)$
 - Dirichlet priors are a convenient choice
 - Specified by a center θ' and strength k , $\text{Dir}(\theta', k)$ or $\text{Dir}(k\theta')$
 - Mean is center, higher strength means lower variance
 - MAP estimate is then

$$\theta_{MAP} = \frac{c(x) + k\theta_x - 1}{\sum_{x'} (c(x') + k\theta_x - 1)}$$

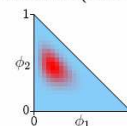
Dirichlet(.5,.5,.5)



Dirichlet(1,1,1)



Dirichlet(5,10,8)



Smoothing: Add-One, Etc.

- With a uniform prior, get estimates of the form

$$P_{\text{add-}\delta}(x) = \frac{c(x) + \delta}{\sum_{x'} (c(x') + \delta)}$$

- Add-one smoothing especially often talked about
- For a bigram distribution, can use a prior centered on the empirical unigram:

$$P_{\text{dir}}(w|w_{-1}) = \frac{c(w_{-1}, w) + k\hat{P}(w)}{(\sum_{w'} c(w_{-1}, w')) + k}$$

- Can consider hierarchical formulations in which trigram is centered on smoothed bigram estimate, etc [Mackay and Peto, 94]
- Basic idea of conjugacy is powerful: prior shape shows up a *pseudo-counts*
- Problem: works quite poorly!

Linear Interpolation

- Problem: $\hat{P}(w|w_{-1}, w_{-2})$ is supported by few counts
- Classic solution: mixtures of related, denser histories, e.g.:

$$\lambda\hat{P}(w|w_{-1}, w_{-2}) + \lambda'\hat{P}(w|w_{-1}) + \lambda''\hat{P}(w)$$

- The mixture approach tends to work better than the Dirichlet prior approach for several reasons
 - Can flexibly include multiple back-off contexts, not just a chain
 - Good ways of learning the mixture weights with EM (later)
 - Not entirely clear why it works so much better
- All the details you could ever want: [Chen and Goodman, 98]

Held-Out Data

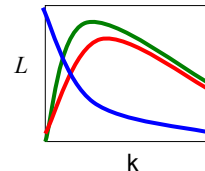
- Important tool for getting models to generalize:



- Set a small number of hyperparameters that control the degree of smoothing by maximizing the (log-)likelihood of held-out data
- Can use any optimization technique (line search or EM usually easiest)

- Examples:

$$P_{dir}(w|w_{-1}, k) = \frac{c(w_{-1}, w) + k\hat{P}(w)}{(\sum_{w'} c(w_{-1}, w')) + k}$$



$$P_{lin}(w|w_{-1}, \lambda, \lambda', \lambda'') = \lambda\hat{P}(w|w_{-1}, w_{-2}) + \lambda'\hat{P}(w|w_{-1}) + \lambda''\hat{P}(w)$$

Held-Out Reweighting

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

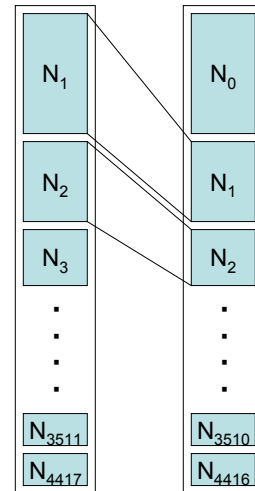
Count in 22M Words	Actual c* (Next 22M)	Add-one's c*	Add-0.0000027's c*
1	0.448	2/7e-10	~1
2	1.25	3/7e-10	~2
3	2.24	4/7e-10	~3
4	3.23	5/7e-10	~4
5	4.21	6/7e-10	~5

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*

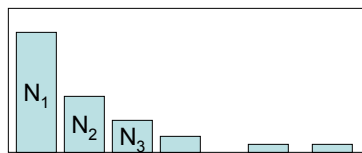
Good-Turing Reweighting I

- We'd like to not need held-out data (why?)
- Idea: leave-one-out validation
 - N_k : number types which occur k times in the entire corpus
 - Take each of the c tokens out of corpus in turn
 - c "training" sets of size $c-1$, "held-out" of size 1
 - What fraction of held-out tokens are unseen in training?
 - N_1/c
 - What fraction of held-out words are seen k times in training?
 - $(k+1)N_{k+1}/c$
 - Words with training count k seem to take up $(k+1)N_{k+1}/c$ fraction of future text
 - There are N_k words with training count k
 - Each should occur with probability:
 - $(k+1)N_{k+1}/cN_k$
 - ...or expected count $(k+1)N_{k+1}/N_k$

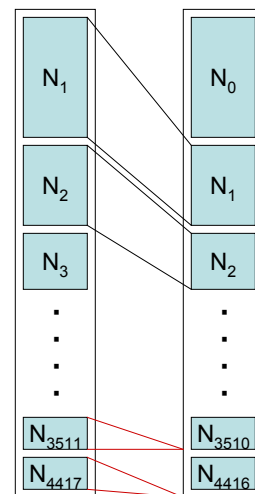
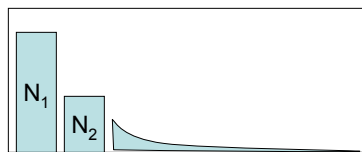


Good-Turing Reweighting II

- 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

$$P_{\text{katz}}(w|w') = \frac{c^*(w, w')}{c(w')} + \alpha \hat{P}(w)$$

Kneser-Ney: Discounting

- Kneser-Ney smoothing: very successful but slightly ad hoc estimator
- Idea: observed n -grams occur more in training than they will later:

Count in 22M Words	Avg in Next 22M	Good-Turing 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

$$P_{\text{ad}}(w|w') = \frac{c(w, w') - d}{c(w')} + \alpha \hat{P}(w)$$

Kneser-Ney: Continuation

- 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:** $P^l(w) \propto |w' : c(w, w') > 0|$
 - In the back-off model, we don't want the probability of w as a unigram
 - Instead, want the probability that w is allowed

Kneser-Ney

- Kneser-Ney smoothing combines these two ideas
 - Absolute discounting

$$P(w|w') = \frac{c(w, w') - d}{c(w')} + \alpha P^l(w)$$

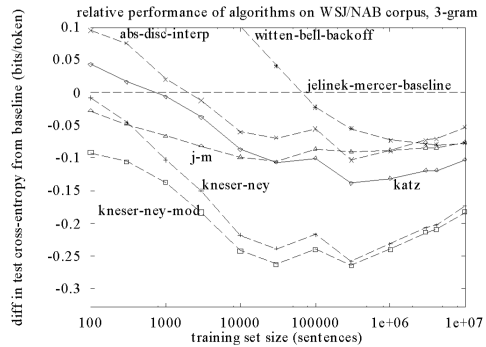
- Lower order models take a special form

$$P^l(w) \propto |w' : c(w, w') > 0|$$

- KN smoothing repeatedly proven effective
 - But we've never been quite sure why
 - And therefore never known how to make it better
- [Teh, 2006] shows KN smoothing is a kind of approximate inference in a hierarchical Pitman-Yor process (and better approximations are superior to basic KN)

What Actually Works?

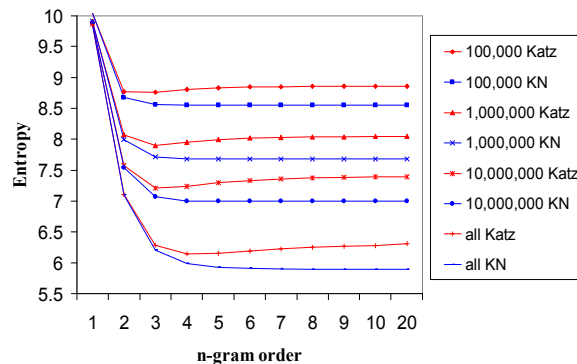
- Trigrams:
 - Unigrams, bigrams too little context
 - Trigrams much better (when there's enough data)
 - 4-, 5-grams often not worth the cost (which is more than it seems, due to how speech recognizers are constructed)
- Good-Turing-like methods for count adjustment
 - Absolute discounting, Good-Turing, 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 better...



- ... but so is using a better model
- Another issue: $N > 3$ has huge costs in speech recognizers

Beyond N-Gram LMs

- Lots of ideas we won't have time to discuss:
 - Caching models: recent words more likely to appear again
 - Trigger models: recent words trigger other words
 - Topic models
- A few recent ideas
 - Syntactic models: use tree models to capture long-distance syntactic effects [Chelba and Jelinek, 98]
 - Discriminative models: set n-gram weights to improve final task accuracy rather than fit training set density [Roark, 05, for ASR; Liang et. al., 06, for MT]
 - Structural zeros: some n-grams are syntactically forbidden, keep estimates at zero [Mohri and Roark, 06]
 - Bayesian document and IR models [Daume 06]