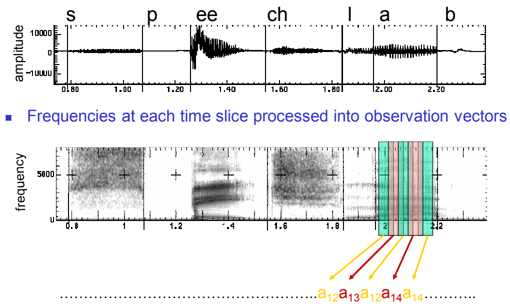
Statistical NLP
Spring 2010

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:

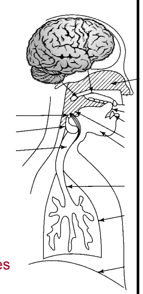
$$w^* = \arg\max_w P(w|a)$$

$$= \arg\max_w P(a|w)P(w)/P(a)$$

$$\propto \arg\max_w P(a|w)P(w)$$

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

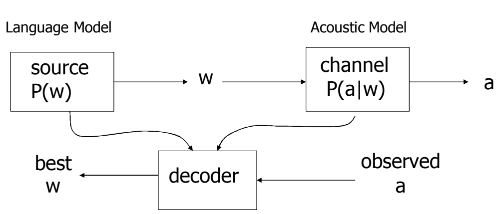
Language model: Distributions over sequences of words (sentences)



Acoustically Scored Hypotheses

the station signs are in deep in english	-14732
the stations signs are in deep in english	-14735
the station signs are in deep into english	-14739
the station 's signs are in deep in english	-14740
the station signs are in deep in the english	-14741
the station signs are indeed in english	-14757
the station 's signs are indeed in english	-14760
the station signs are indians in english	-14790
the station signs are indian in english	-14799
the stations signs are indians in english	-14807
the stations signs are indians and english	-14815

ASR System Components

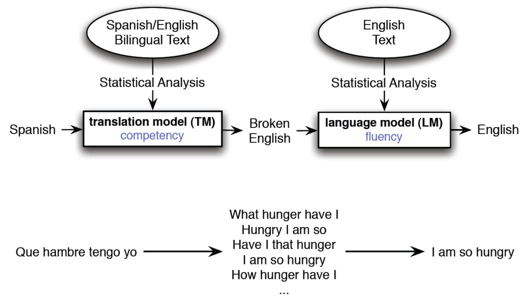


argmax P(w|a) = argmax P(a|w)P(w)
w w

Translation: Codebreaking?

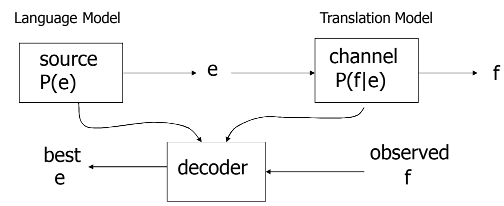
- "Also knowing nothing official about, but having guessed and inferred considerable about, the powerful new mechanized methods in cryptography—methods which I believe succeed even when one does not know what language has been coded—one naturally wonders if the problem of translation could conceivably 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 Overview



7

MT System Components



$$\underset{e}{\text{argmax}}\ P(e|f) = \underset{e}{\text{argmax}}\ P(f|e)P(e)$$

Other Noisy-Channel Processes

- Spelling Correction

$$P(words \mid characters) \propto P(words)P(characters \mid words)$$

- Handwriting recognition

$$P(words \mid strokes) \propto P(words)P(strokes \mid words)$$

- OCR

$$P(words \mid pixels) \propto P(words)P(pixels \mid words)$$

- More…

Probabilistic Language Models

- Goal: Assign useful probabilities P(x) to sentences x
  - Input: many observations of training sentences x
  - Output: system capable of computing P(x)
- Probabilities should broadly indicate plausibility of sentences
  - P(I saw a van) >> P(eyes awe of an)
  - *Not grammaticality*: P(artichokes intimidate zippers) ≈ 0
  - In principle, "plausible" depends on the domain, context, speaker…
- One option: empirical distribution over training sentences?
  - Problem: doesn't generalize (at all)
- Two aspects of generalization
  - Decomposition: break sentences into small pieces which can be recombined in new ways (conditional independence)
  - Smoothing: allow for the possibility of unseen pieces

N-Gram Model Decomposition

- Chain rule: break sentence probability down

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

- Impractical to condition on everything before
  - P(??? | Turn to page 134 and look at the picture of the) ?
- N-gram models: assume each word depends only on a short linear history

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

- Example: $P(\text{please close the door}) =$

$$P(\text{please}|\text{START})P(\text{close}|\text{please}) \ldots P(\text{STOP}|door)$$

N-Gram Model Parameters

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

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

- General approach
  - Take a training set X and a test set X'
  - Compute an estimate θ from X
  - Use it to assign probabilities to other sentences, such as those in X'

Training Counts

198015222 the first
194623024 the same
168504105 the following
158562063 the world
…
14112454 the door

23135851162 the *

$$\hat{P}(\text{door}|\text{the}) = \frac{14112454}{23135851162}$$

$$= 0.0006$$

Higher Order N-grams?

Please close the door

Please close the first window on the left

198015222 the first	197302 close the window	3380 please close the door
194623024 the same	191125 close the door	1601 please close the window
168504105 the following	152500 close the gap	1164 please close the new
158562063 the world	116451 close the thread	1159 please close the gate
…	87298 close the deal	900 please close the browser
14112454 the door	-----------------	-----------------
-----------------	3785230 close the *	13951 please close the *
23135851162 the *		

Unigram Models

- Simplest case: unigrams

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

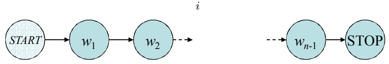
- Generative process: pick a word, pick a word, … until you pick STOP
- As a graphical model:

- 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(the the the the) >> P(I like ice cream)!
- Condition on previous single word:

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



- 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]

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, .]

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

Measuring Model Quality

- 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 likelihood on train, but not test
- 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

- The Shannon Game:
  - How well can we predict the next word?

    When I eat 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": per-word test log likelihood (misnamed)

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

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

3516 wipe off the excess
1034 wipe off the dust
547 wipe off the sweat
518 wipe off the mouthpiece
...
120 wipe off the grease
0 wipe off the sauce
0 wipe off the mice

28048 wipe off the *

Measuring Model Quality

- Problem with "entropy":
  - 0.1 bits of improvement doesn't sound so good
  - "Solution": perplexity

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

  - Interpretation: average branching factor in model
- Important 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 actual word, not per derivation step.

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

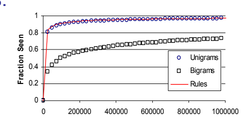
Recognizer output: And he saw apart of the movie

WER: 4/7
= 57%

- The "right" measure:
  - Task error driven
  - For speech recognition
  - For a specific recognizer!
- Common issue: intrinsic measures like perplexity are easier to use, but extrinsic ones are more credible

Sparsity

- Problems with n-gram models:
  - New words appear all the time:
    - Synaptitute
    - 132,701.03
    - multidisciplinarization
  - New bigrams: even more often
  - Trigrams or more – still worse!



- Zipf's Law
  - Types (words) vs. tokens (word occurences)
  - 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

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

3516 wipe off the excess
1034 wipe off the dust
547 wipe off the sweat
518 wipe off the mouthpiece
...
120 wipe off the grease
0 wipe off the sauce
0 wipe off the mice

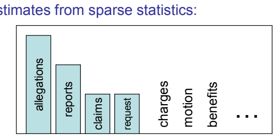
28048 wipe off the *

- 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
  - Approaches that are mathematically obvious aren't always what works

Smoothing

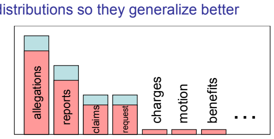
- We often want to make estimates from sparse statistics:

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!
- We'll illustrate with bigrams today (h = previous word, could be anything).

Puzzle: Unknown Words

- Imagine we look at 1M words of text
  - We'll see many thousands of word types
  - Some will be frequent, others rare
  - Could turn into an empirical P(w)
- Questions:
  - What fraction of the next 1M will be new words?
  - How many total word types exist?

Language Models

- In general, we want to place a distribution over sentences
- Basic / classic solution: n-gram models

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

- Question: how to estimate conditional probabilities?

$$P(w|w') =$$

- Problems:
  - Known words in unseen contexts
  - Entirely unknown words
    - Many systems ignore this – why?
    - Often just lump all new words into a single UNK type

Smoothing: Add-One, Etc.

- Classic solution: add counts (Laplace smoothing / Dirichlet prior)

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

  - Add-one smoothing especially often talked about
- For a bigram distribution, can add counts shaped like the unigram:

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

- Can consider hierarchical formulations: trigram is recursively centered on smoothed bigram estimate, etc [MacKay and Peto, 94]
- Can be derived from Dirichlet / multinomial conjugacy: prior shape shows up as *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
  - Often multiple weights, depending on bucketed counts
  - 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]

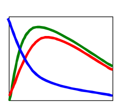
Held-Out Data

- Important tool for optimizing how models generalize:

Training Data	Held-Out Data	Test Data

  - 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:

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



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

Held-Out Reweighting

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

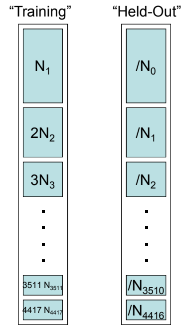
Count in 22M Words	Actual c* (Next 22M)	Add-one's c*	Add-0.0000027's c*
1	0.448	2/7e-10	~1
2	1.25	3/7e-10	~2
3	2.24	4/7e-10	~3
4	3.23	5/7e-10	~4
5	4.21	6/7e-10	~5

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 vastly underestimates the ratio 2*/1*
- One solution: use held-out data to predict the map of c to c*

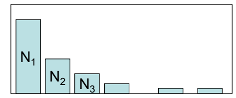
Good-Turing Reweighting I

- We'd like to not need held-out data (why?)
- Idea: leave-one-out validation
  - N_k: number of 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
  - How many held-out tokens are unseen in training?
    - N_1
  - How many held-out tokens are seen k times in training?
    - $(k+1)N_{k+1}$
  - There are N_k words with training count k
  - Each should occur with expected count
    - $(k+1)N_{k+1}/N_k$
  - Each should occur with probability:
    - $(k+1)N_{k+1}/(cN_k)$

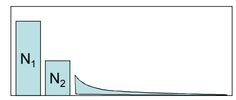


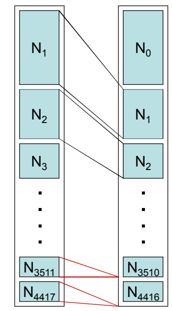
Good-Turing Reweighting II

- Problem: what about "the"? (say k=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

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

Kneser-Ney: Discounting

- Kneser-Ney smoothing: very successful estimator using two ideas
- Idea 1: observed n-grams occur more in training than they will later:

Count in 22M Words	Future c* (Next 22M)
1	0.448
2	1.25
3	2.24
4	3.23

- Absolute Discounting
  - No need to actually have held-out data; just subtract 0.75 (or some d)
  - Maybe have a separate value of d for very low counts

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

Kneser-Ney: Continuation

- Idea 2: Type-based fertility
  - Shannon game: There was an unexpected ____?
    - delay?
    - Francisco?
  - "Francisco" is more common than "delay"
  - … but "Francisco" always follows "San"
  - … so it's less "fertile"
- Solution: type-continuation probabilities
  - In the back-off model, we don't want the probability of w as a unigram
  - Instead, want the probability that w is *allowed in a novel context*
  - For each word, count the number of bigram types it completes

$$P_{\text{c}}(w) \propto |w' : c(w',w) > 0|$$

Kneser-Ney

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

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

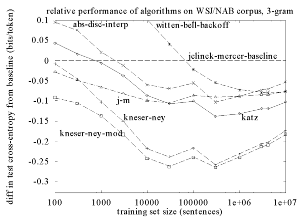
  - Lower order continuation probabilities

$$P_c(w) \propto |w' : c(w',w) > 0|$$

- KN smoothing repeatedly proven effective
- [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?

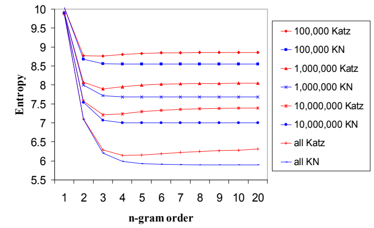
- Trigrams and beyond:
  - Unigrams, bigrams generally useless
  - Trigrams much better (when there's enough data)
  - 4-, 5-grams really useful in MT, but not so much for speech
- Discounting
  - Absolute discounting, Good-Turing, held-out estimation, Witten-Bell
- Context counting
  - Kneser-Ney construction oflower-order models
- See [Chen+Goodman] reading for tons of graphs!



[Graphs from Joshua Goodman]

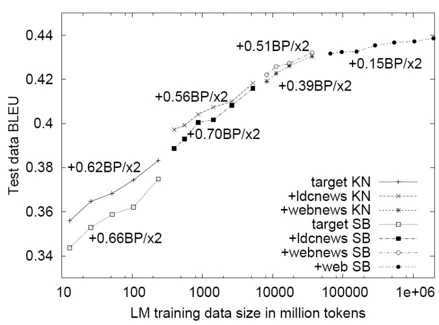
Data >> Method?

- Having more data is better…



- … but so is using a better estimator
- Another issue: N > 3 has huge costs in speech recognizers

Tons of Data?



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]

Overview

- So far: language models give P(s)
  - Help model fluency for various noisy-channel processes (MT, ASR, etc.)
  - N-gram models don't represent any deep variables involved in language structure or meaning
  - Usually we want to know something about the input other than how likely it is (syntax, semantics, topic, etc)
- Next: Naïve-Bayes models
  - We introduce a single new global variable
  - Still a very simplistic model family
  - Lets us model hidden properties of text, but only very non-local ones…
  - In particular, we can only model properties which are largely invariant to word order (like topic)