# CSEP 517
# Natural Language Processing
# Autumn 2013

## Language Models

Luke Zettlemoyer

Many slides from Dan Klein and Michael Collins

# Overview

- The language modeling problem
- N-gram language models
- Evaluation: perplexity
- Smoothing
  - Add-N
  - Linear Interpolation
  - Discounting Methods

# The Language Modeling Problem

- Setup: Assume a (finite) vocabulary of words

$$\mathcal{V} = \{\text{the}, \text{a}, \text{man}, \text{telescope}, \text{Beckham}, \text{two}, \text{Madrid}, \ldots\}$$

- We can construct an (infinite) set of strings

$$\mathcal{V}^\dagger = \{\text{the}, \text{a}, \text{the a}, \text{the fan}, \text{the man}, \text{the man with the telescope}, \ldots\}$$

- Data: given a *training set* of example sentences $x \in \mathcal{V}^\dagger$
- Problem: estimate a probability distribution

$$\sum_{x \in \mathcal{V}^\dagger} p(x) = 1$$

and $p(x) \geq 0$ for all $x \in \mathcal{V}^\dagger$

$$p(\text{the}) = 10^{-12}$$
$$p(\text{a}) = 10^{-13}$$
$$p(\text{the fan}) = 10^{-12}$$
$$p(\text{the fan saw Beckham}) = 2 \times 10^{-8}$$
$$p(\text{the fan saw saw}) = 10^{-15}$$
$$\ldots$$

- Question: why would we ever want to do this?

# The Noisy-Channel Model

- We want to predict a sentence given acoustics:

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

- The noisy channel approach:

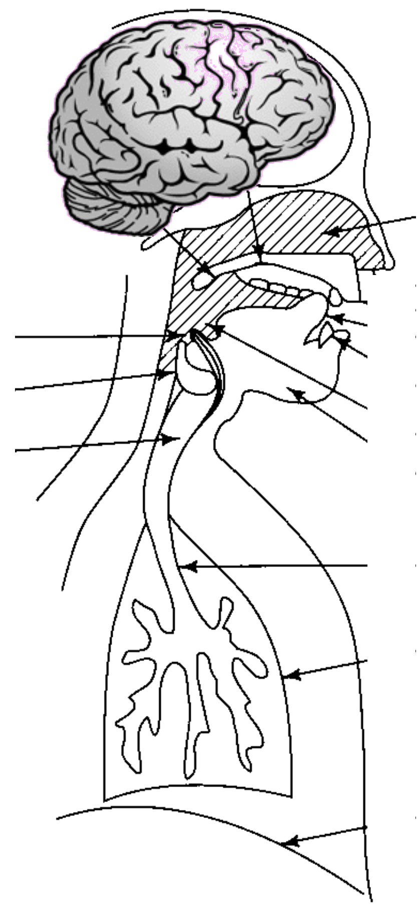
$$w^* = \arg\max_w P(w|a)$$

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

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

Acoustic model: Distributions over acoustic waves given a sentence

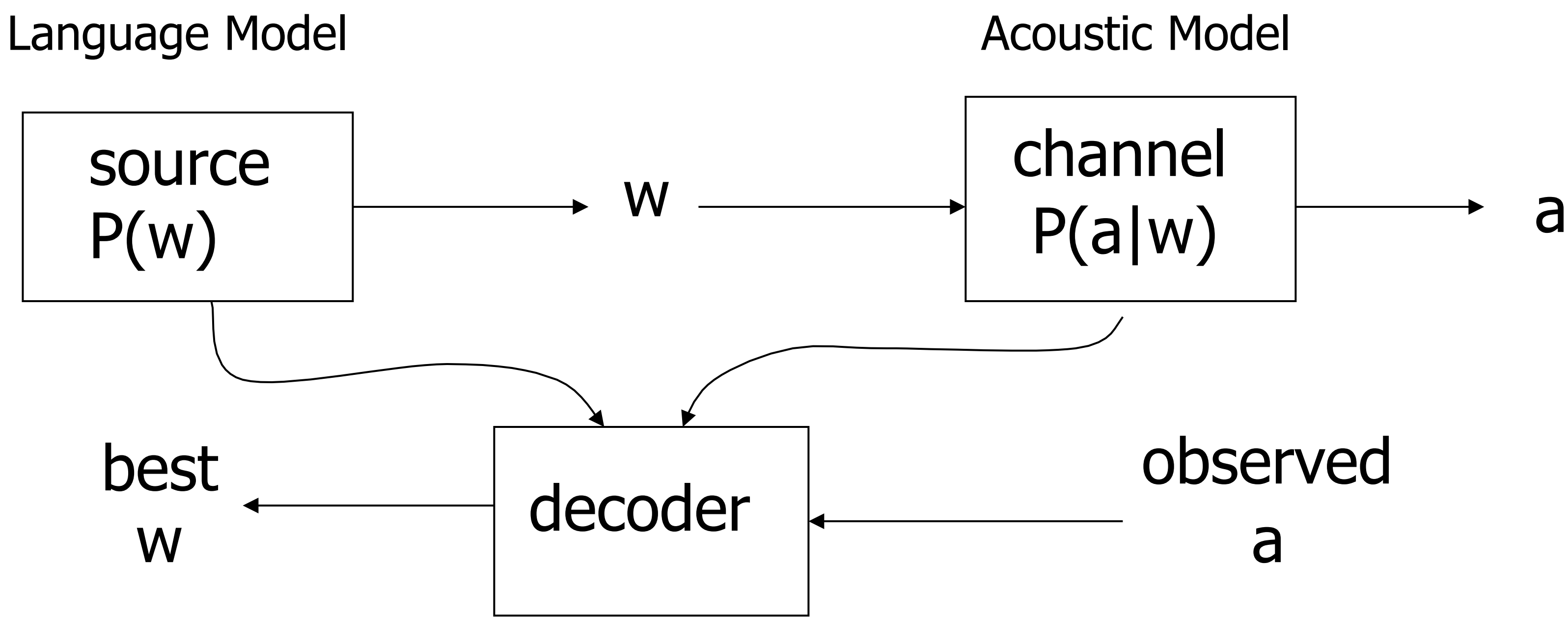
Language model: Distributions over sequences of words (sentences)



# 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

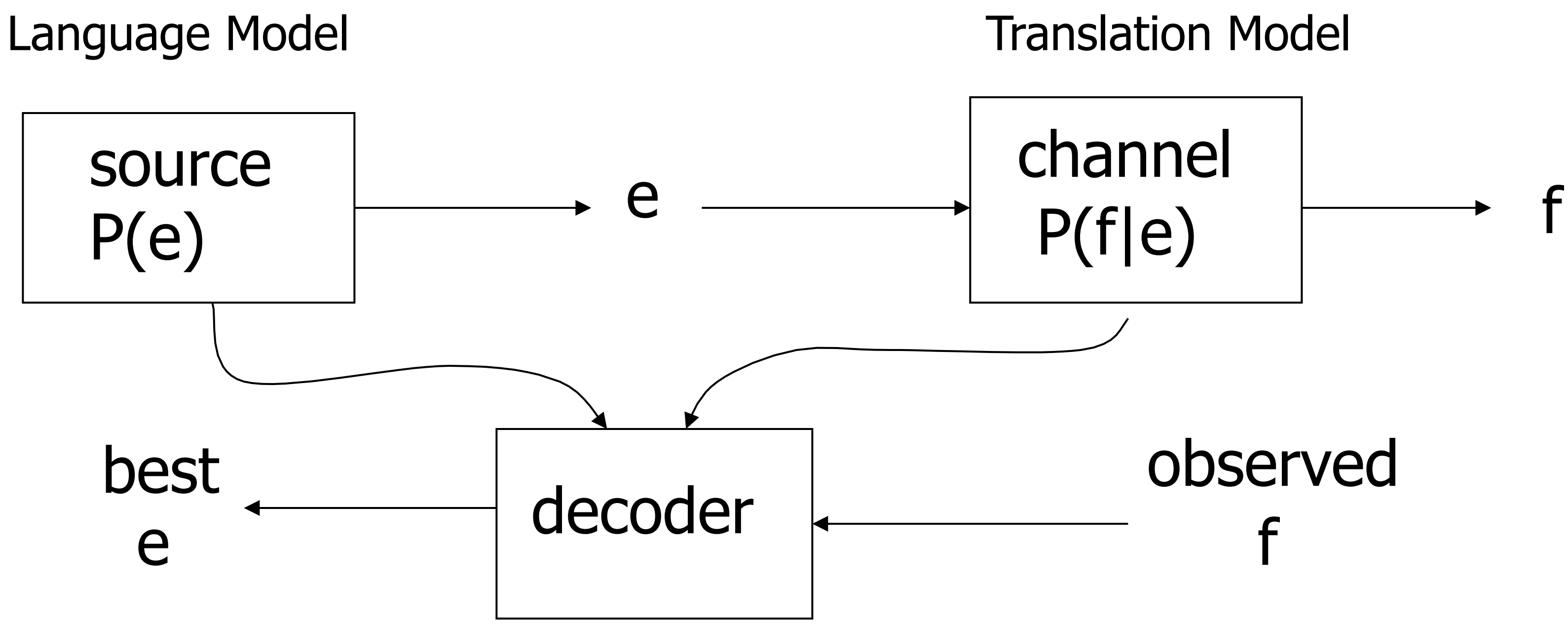



$$\underset{w}{\operatorname{argmax}}\, P(w|a) = \underset{w}{\operatorname{argmax}}\, P(a|w)P(w)$$

# 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 System Components




$$\operatorname*{argmax}_{e} P(e|f) = \operatorname*{argmax}_{e} P(f|e)P(e)$$

# Learning 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…

$$p(x_1 \dots x_n) = \frac{c(x_1 \dots x_n)}{N} \text{ for sentence } x = x_1 \dots x_n$$

  - Problem: does not generalize (at all)
  - Need to assign non-zero probability to previously unseen sentences!

# Unigram Models

- Simplest case: unigrams

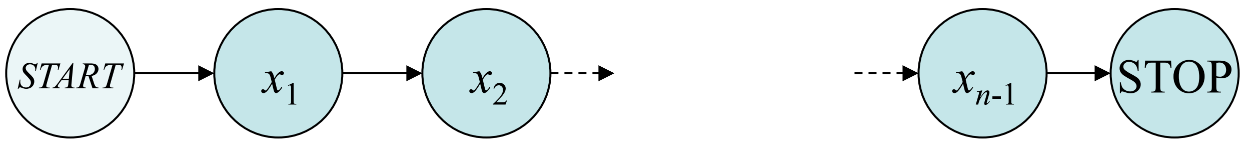
$$p(x_1 \ldots x_n) = \prod_{i=1}^{n} p(x_i)$$

- 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]
- Big problem with unigrams: P(the the the the) >> P(I like ice cream)!

# Bigram Models

- Condition on previous single word: $$p(x_1 \dots x_n) = \prod_{i=1}^{n} p(x_i | x_{i-1})$$
- Generative process: pick START, pick a word conditioned on previous one, repeat until to pick STOP
- Graphical Model:




- 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]
- But, what is the cost?

# Higher Order N-grams?

Please close the door

Please close the first window on the left

198015222 the first
194623024 the same
168504105 the following
158562063 the world
…
14112454 the door
-----------------
23135851162 the *

197302 close the window
191125 close the door
152500 close the gap
116451 close the thread
87298 close the deal
-----------------
3785230 close the *

3380 please close the door
1601 please close the window
1164 please close the new
1159 please close the gate
…
0 please close the first
-----------------
13951 please close the *

# N-Gram Model Decomposition

- Exact decomposition: law of conditional probability

$$p(x_1 \dots x_n) = p(x_1) \prod_{i=2}^{n} p(x_i | x_1 \dots x_{i-1})$$

- Impractical to condition on everything before
  - P(??? | Turn to page 134 and look at the picture of the) ?

- k-gram models (k>1): condition on k-1 previous words

$$p(x_1 \dots x_n) = \prod_{i=1}^{n} q(x_i | x_{i-(k-1)} \dots x_{i-1})$$

where $x_i \in \mathcal{V} \cup \{STOP\}$ and $x_1 \dots x_{k-1} = START$

- Learning: estimate the distributions $q(x_i | x_{i-(k-1)} \dots x_{i-1})$

# Unigram LMs are a Well Defined Dist'ns*

- Simplest case: unigrams

$$p(x_1 \ldots x_n) = \prod_{i=1}^{n} p(x_i)$$

- Generative process: pick a word, pick a word, … until you pick STOP
- For all strings x (of any length): $p(x) \geq 0$
- Claim: the sum over string of all lengths is 1 : $\Sigma_x p(x) = 1$
  - Step 1: decompose sum over length (p(n) is prob. of sent. with n words)

$$\sum_x p(x) = \sum_{n=0}^{\infty} p(n) \sum_{x_1 \ldots x_n} p(x_1 \ldots x_n)$$

  - Step 2: For each length, inner sum is 1

$$\sum_{x_1 \ldots x_n} p(x_1 \ldots x_n) = \sum_{x_1 \ldots x_n} \prod_{i=1}^{n} p(x_i) = \sum_{x_1} \ldots \sum_{x_n} p(x_1) \times \ldots \times p(x_n) = \sum_{x_1} p(x_1) \times \ldots \times \sum_{x_n} p(x_n) = 1$$

  - Step 3: For stopping prob. $p_s = P(STOP)$, we get a geometric series

$$\sum_{n=0}^{\infty} p(n) = \sum_{n=0}^{\infty} p_s (1-p_s)^n = p_s \sum_{n=0}^{\infty} (1-p_s)^n = p_s \frac{1}{1-(1-p_s)} = p_s \frac{1}{p_s} = 1$$

- Question: What about the n-gram case?

# N-Gram Model Parameters

- The parameters of an n-gram model:
  - *Maximum likelihood estimate*: relative frequency

$$q_{ML}(w) = \frac{c(w)}{c()}, \quad q_{ML}(w|v) = \frac{c(w,v)}{c(v)}, \quad q_{ML}(w|u,v) = \frac{c(u,v,w)}{c(u,v)}, \quad \ldots$$

where c is the empirical counts on a training set

- General approach
  - Take a training set X and a test set X'
  - Compute an estimate of the qs 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 *
```

$$q(\text{door}|\text{the}) = \frac{14112454}{2313581162}$$
$$= 0.0006$$

# 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 ___."
    - 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, .]

# Measuring Model Quality

- The goal 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?)

$$\left\{\begin{array}{ll} \text{grease} & 0.5 \\ \text{sauce} & 0.4 \\ \text{dust} & 0.05 \\ \text{....} & \\ \text{mice} & 0.0001 \\ \text{....} & \\ \text{the} & 1\text{e-}100 \end{array}\right.$$

- How good are we doing?

Compute per word log likelihood (M words, m test sentences $s_i$):

$$l = \frac{1}{M} \sum_{i=1}^{m} \log p(s_i)$$

# Measuring Model Quality

- But, we usually report perplexity

$$2^{-l} \text{ where } l = \frac{1}{M} \sum_{i=1}^{m} \log p(s_i)$$

  - Lower is better!
  - Example: $|\mathcal{V}| = N$ and $q(w| \ldots) = \frac{1}{N}$
    - uniform model → perplexity is N
  - Interpretation: effective vocabulary size (accounting for statistical regularities)
  - Typical values for newspaper text:
    - Uniform: 20,000; Unigram: 1000s, Bigram: 700-1000, Trigram: 100-200
- Important note:
  - It's easy to get bogus perplexities by having bogus probabilities that sum to more than one over their event spaces. Be careful in homeworks!

# Measuring Model Quality (Speech)

- 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

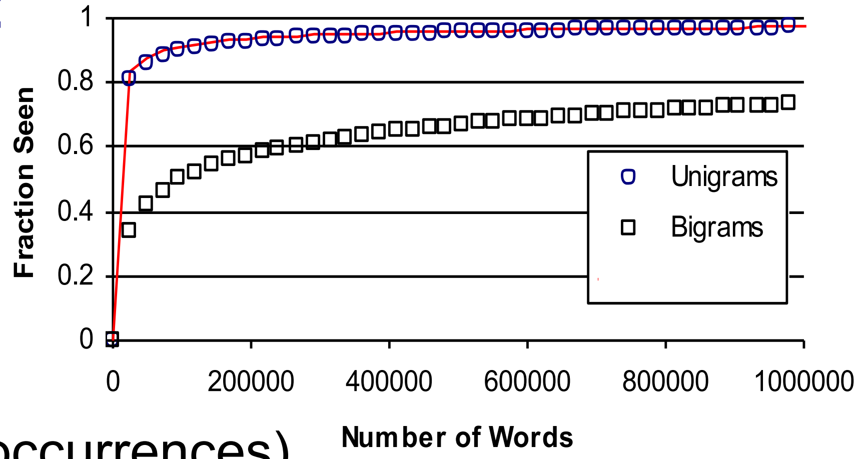
*WER: 4/7*
*= 57%*

- 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 n-grams: even more often




- 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!)
- This is particularly problematic when…
  - Training set is small (does this happen for language modeling?)
  - Transferring domains: e.g., newswire, scientific literature, Twitter

# Parameter Estimation

- Maximum likelihood estimates won't get us very far

$$q_{ML}(w) = \frac{c(w)}{c()}, \quad q_{ML}(w|v) = \frac{c(w,v)}{c(v)}, \quad q_{ML}(w|u,v) = \frac{c(u,v,w)}{c(u,v)}, \quad \ldots$$

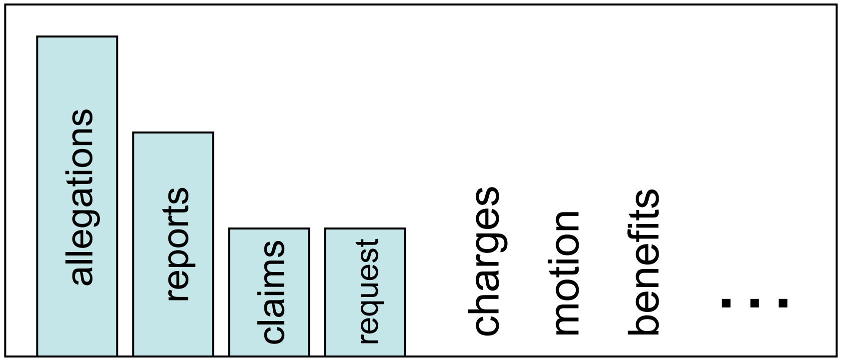
- Need to *smooth* these estimates
- General method (procedurally)
  - Take your empirical counts
  - Modify them in various ways to improve estimates
- General method (mathematically)
  - Often can give estimators a formal statistical interpretation … but not always
  - Approaches that are mathematically obvious aren't always what works

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 *

# Smoothing

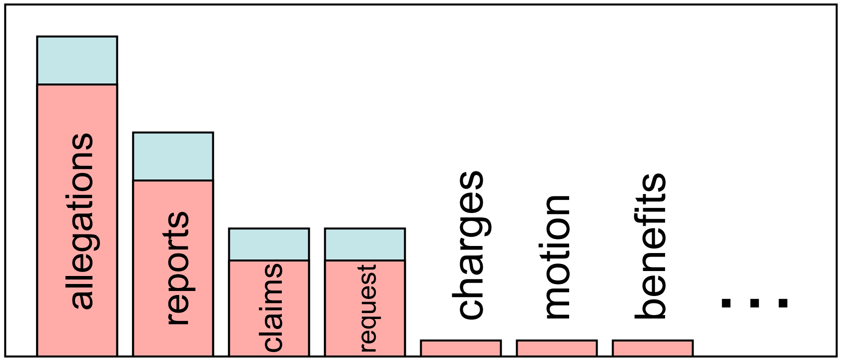
- We often want to make estimates from sparse statistics:

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 (and ML more generally), but easy to do badly!
- Question: what is the best way to do it?

# Smoothing: Add-One, Etc.

- Classic solution: add counts (Laplace smoothing)

$$q_{add-\delta}(w) = \frac{c(w) + \delta}{\sum_{w'} (c(w') + \delta)}$$

  - Add-one smoothing especially often talked about

- For a bigram distribution, can add counts shaped like the unigram:

$$q_{uni-\delta}(w|v) = \frac{c(v, w) + \delta q_{ML}(w)}{\left(\sum_{w'} c(v, w')\right) + \delta}$$

- Can consider hierarchical formulations: trigram is recursively centered on smoothed bigram estimate, etc. [MacKay and Peto, 94]

- Bayesian: Can be derived from Dirichlet / multinomial conjugacy - prior shape shows up as *pseudo-counts*

- Problem: works quite poorly!

# Linear Interpolation

- Problem: $q_{ML}(w|u,v)$ is supported by few counts
- Classic solution: mixtures of related, denser histories:

$$q(w|u,v) = \lambda_3 q_{ML}(w|u,v) + \lambda_2 q_{ML}(w|v) + \lambda_1 q_{ML}(w)$$

- Is this a well defined distribution?
  - Yes, if all $\lambda_i \geq 0$ and they sum to 1
- The mixture approach tends to work better than add-δ approach for several reasons
  - Can flexibly include multiple back-off contexts
  - 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

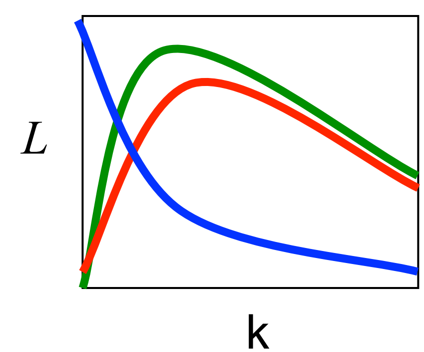
- Important tool for optimizing how models generalize:

| 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:

$$q_{uni-\delta}(w|v) = \frac{c(v,w) + \delta q_{ML}(w)}{\left(\sum_{w'} c(v,w')\right) + \delta}$$




$$q(w|u,v) = \lambda_3 q_{ML}(w|u,v) + \lambda_2 q_{ML}(w|v) + \lambda_1 q_{ML}(w)$$

# 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*

# Absolute Discounting

- 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 (Bigram case)
  - No need to actually have held-out data; just subtract 0.75 (or some d)

$$c^*(v,w) = c(v,w) - 0.75 \text{ and } q(w|v) = \frac{c^*(v,w)}{c(v)}$$

  - But, then we have “extra” probability mass

$$\alpha(v) = 1 - \sum_w \frac{c^*(v,w)}{c(v)}$$

  - Question: How to distribute α between the unseen words?

# Katz Backoff

- Absolute discounting, with backoff to unigram estimates

$$c^*(v,w) = c(v,w) - d \qquad \alpha(v) = 1 - \sum_{w} \frac{c^*(v,w)}{c(v)}$$

- Define the words into seen and unseen

$$\mathcal{A}(v) = \{w : c(u,w) > 0\} \quad \mathcal{B}(v) = \{w : c(u,w) = 0\}$$

- Now, backoff to maximum likelihood unigram estimates for unseen words

$$q_{BO}(w|v) = \begin{cases} \frac{c^*(v,w)}{c(v)} & \text{If } w \in \mathcal{A}(v) \\ \alpha(v) \times \frac{q_{ML}(w)}{\sum_{w' \in \mathcal{B}(v)} q_{ML}(w')} & \text{If } w \in \mathcal{B}(v) \end{cases}$$

- Can consider hierarchical formulations: trigram is recursively backed off to Katz bigram estimate, etc
- Can also have multiple count thresholds (instead of just 0 and >0)

# Good-Turing Discounting*

- Question: why the same d for all n-grams?
- Good-Turing Discounting invented during WWII by Alan Turing and later published by Good. Frequency estimates were needed for Enigma code-breaking effort.
- Let $n_r$ be the number of n-grams x for which c(x) = r
- Now, use the modified counts

$$c^*(x) = (r+1)\frac{n_{r+1}}{n_r} \text{ iff } c(x) = r,\ r > 0$$

- Then, our estimate of the missing mass is:

$$\alpha(v) = \frac{n_1}{N}$$

- Where N is the number of tokens in the training set

# Kneser-Ney Backoff*

- Idea: Type-based fertility
  - Shannon game: There was an unexpected ____?
    - delay?
    - Francisco?
  - "Francisco" is more common than "delay"
  - … but "Francisco" (almost) always follows "San"
  - … so it's less "fertile"

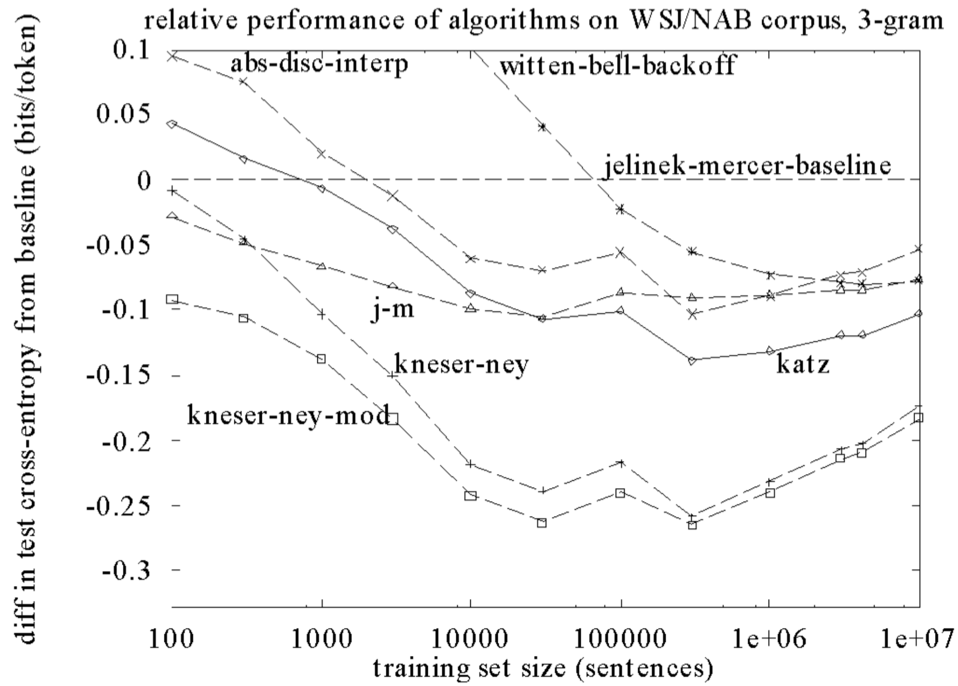
- Solution: type-continuation probabilities
  - In the back-off model, we don't want the unigram estimate $p_{ML}$
  - Instead, want the probability that w is *allowed in a novel context*
  - For each word, count the number of bigram types it completes

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

  - KN smoothing repeatedly proven effective
  - [Teh, 2006] shows it is a kind of approximate inference in a hierarchical Pitman-Yor process (and other, better approximations are possible)

# What Actually Works?

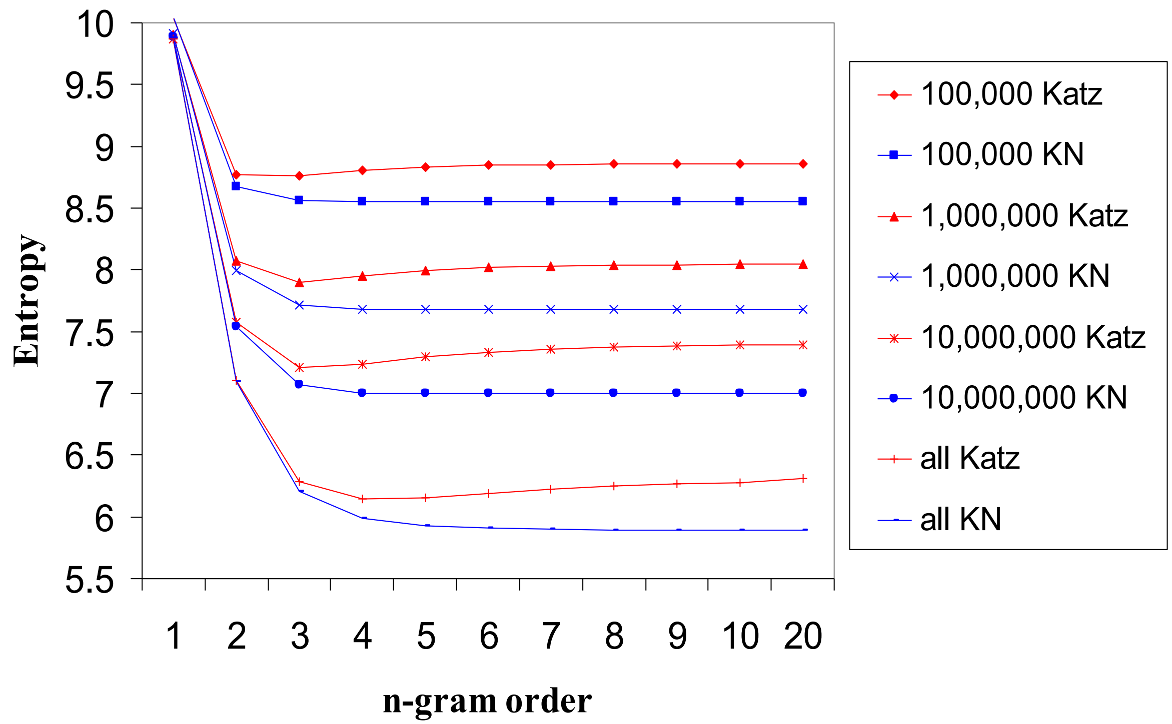
- Trigrams and beyond:
  - Unigrams, bigrams generally useless
  - Trigrams much better (when there's enough data)
  - 4-, 5-grams really useful in MT, but not so much for speech
- Discounting
  - Absolute discounting, Good-Turing, held-out estimation, Witten-Bell, etc…
- See [Chen+Goodman] reading for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




[Graphs from Joshua Goodman]

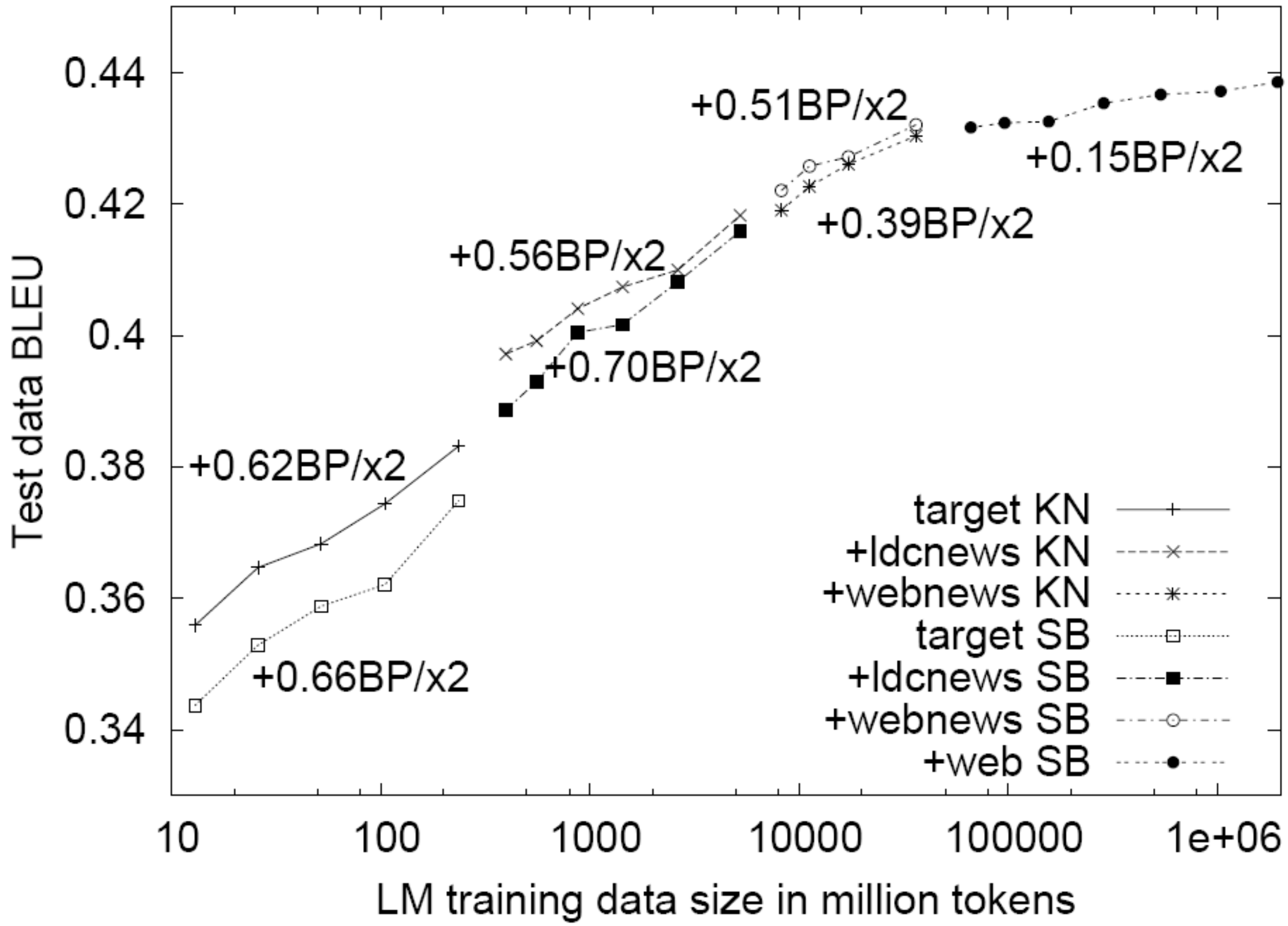
# Data vs. Method?

- 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?




- Tons of data closes gap, for extrinsic MT evaluation

# 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]