CS 294-5: Statistical Natural Language Processing

Smoothing Methods

Lecture 3: 9/6/05

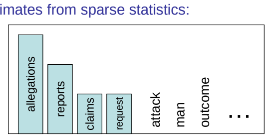
Recap: Language Models

- Why are language models useful?
- Why did I show samples of generated text?
- What are the main challenges in building n-gram language models?

Smoothing

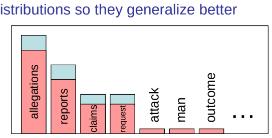
- We often want to make estimates from sparse statistics:

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).

Vocabulary Size

- Key issue for language models: open or closed vocabulary?
  - When would you want an open vocabulary?
  - When would you want a closed vocabulary?
- How to set the vocabulary size V?
  - By external factors (e.g. speech recognizers)
  - Using statistical estimates?
  - Difference between estimating unknown token rate and probability of a given unknown word
- For the homework:
  - OK to assume there is only one unknown word type UNK
  - UNK be quite common in new text!
  - UNK stands for all unknown word type

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
N_k	number of word types with count k

- One class of smoothing functions:
  - Add-one / delta: assumes a uniform prior

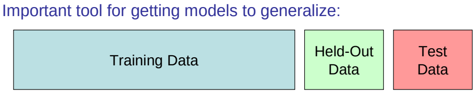
$$P_{ADD-\delta}(w \mid w_{-1}) = \frac{c(w,w_{-1}) + \delta(1/V)}{c(w_{-1}) + \delta}$$

  - Better to assume a unigram prior

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

Held-Out Data

- Important tool for getting models to generalize:



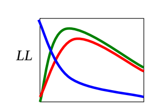
  - When we have a small number of parameters that control the degree of smoothing, we set them to maximize the (log-)likelihood of held-out data

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

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

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

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



Held-Out Reweighting

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

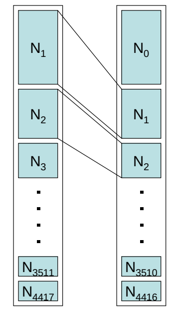
Count in 22M Words	Actual c* (Next 22M)	Add one's c*	Add 00000027's c*
1	0.448	2/7e+10	~1
2	1.25	3/7e+10	~2
3	2.24	4/7e+10	~3
4	3.23	5/7e+10	~4
5	4.21	6/7e+10	~5

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*
  - Issue: which distribution are we smoothing?
- One solution: use held-out data to predict the map of c to c*

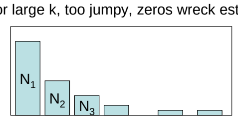
Good-Turing Reweighting I

- 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_1/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:
    - $(k+1)N_{k+1}/c/N_k$
  - …or expected count $(k+1)N_{k+1}/N_k$

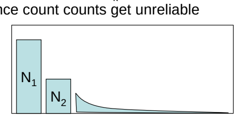


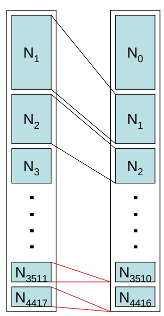
Good-Turing Reweighting II

- 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

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

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

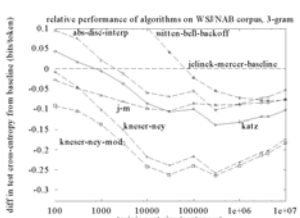
$$P_{KN}(w \mid w_{-1}) = \frac{c(w, w_{-1}) - D}{\sum_{w'} c(w', w_{-1})} + \alpha(w_{-1})P_{CONTINUATION}(w)$$

Higher-Order Models

Expectation-Maximization

What Actually Works?

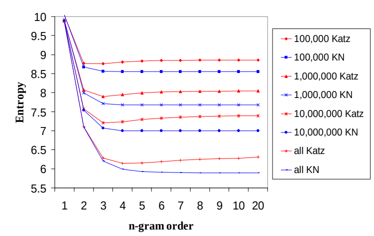
- Trigrams:
  - Unigrams, bigrams too little context
  - Trigrams much better (when there's enough data)
  - 4-, 5-grams usually not worth the cost (which is more than it seems, due to how speech recognizers are constructed)
- Good-Turing-like methods for count adjustment
  - Absolute discounting, Good-Turing,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

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

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

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

- 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: M+S 6, J+M 6, Chen & Goodman (on web page)
- Next up: Text Categorization, Naïve-Bayes