Statistical NLP

Spring 2007

Lecture 3: Language Models II

Dan Klein – UC Berkeley

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

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: 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

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

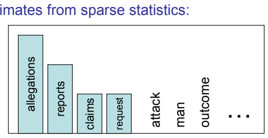
  - Better to assume a unigram prior

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

Smoothing

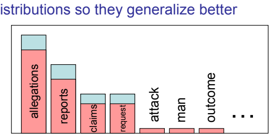
- We often want to make estimates from sparse statistics:

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).

Linear Interpolation

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

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

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

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

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

Held-Out Data

- Important tool for getting models to generalize:

Training Data	Held-Out Data	Test Data

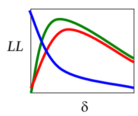
  - When we have a small number of parameters that control the degree of smoothing, we set them to maximize the (log-)likelihood of held-out data

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

  - 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]:

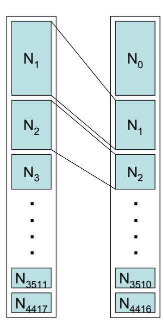
Count in 22M Words	Actual c* (Next 22M)	Add one's c*	Add 00000027's c*
1	0.448	2/7e 10	~1
2	1.25	3/7e 10	~2
3	2.24	4/7e 10	~3
4	3.23	5/7e 10	~4
5	4.21	6/7e 10	~5

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*
- One solution: use held-out data to predict the map of c to c*

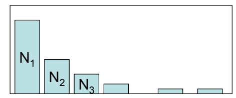
Good-Turing Reweighting I

- 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 $(k+1)N_{k+1}/N_k$

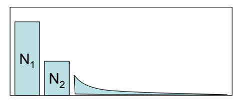


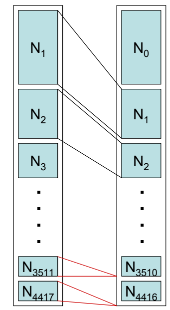
Good-Turing Reweighting II

- 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

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

Beyond N-Gram LMs

- Caching Models
  - Recent words more likely to appear again

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

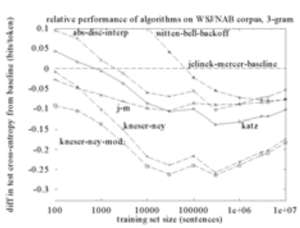
  - Can be disastrous in practice for speech (why?)
- Skipping Models

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

- 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)

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)

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…

Text Categorization

- Want to classify documents into broad semantic topics (e.g. politics, sports, etc.)

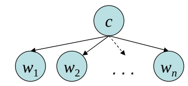
Democratic vice presidential candidate John Edwards on Sunday accused President Bush and Vice President Dick Cheney of misleading Americans by implying a link between deposed Iraqi President Saddam Hussein and the Sept. 11, 2001 terrorist attacks.

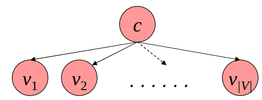
While No. 1 Southern California and No. 2 Oklahoma had no problems holding on to the top two spots with lopsided wins, four teams fell out of the rankings — Kansas State and Missouri from the Big 12 and Clemson from the Atlantic Coast Conference and Oregon from the Pac-10.

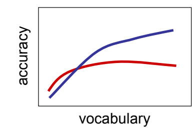
- Which one is the politics document? (And how much deep processing did that decision take?)
- One approach: bag-of-words and Naïve-Bayes models
- Another approach next lecture…

Two NB Formulations

- Two NB models for text categorization
  - The class-conditional unigram model, a.k.a. multinomial model
    - One node per word in the document
    - Driven by words which are present
    - Multiple occurrences, multiple evidence
    - Better overall – plus, know how to smooth
  - The binary model
    - One node for each word in the vocabulary







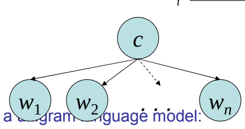
- Incorporates explicit negative correlations
- Know how to do feature selection (e.g. keep words with high mutual information with the class variable)

Naïve-Bayes Models

- Idea: pick a topic, then generate a document using a language model for that topic.
- Naïve-Bayes assumption: all words are independent given the topic.

$$P(c, w_1, w_2, \ldots w_n) = P(c)\prod_i \boxed{P(w_i \mid c)}$$

We have to smooth these!



w_n = STOP

- Compare to a unigram language model:

$$P(w_1, w_2, \ldots w_n) = \prod_i P(w_i)$$

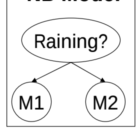
Example: Barometers

Reality

Raining | Sunny

P(+,+,r) = 3/8 P(-,-,r) = 1/8 P(+,+,s) = 1/8 P(-,-,s) = 3/8

NB Model



NB FACTORS:
- P(s) = 1/2
- P(+|s) = 1/4
- P(+|r) = 3/4

PREDICTIONS:
- P(r,+,+) = (½)(¾)(¾)
- P(s,+,+) = (½)(¼)(¼)
- P(r|+,+) = 9/10
- P(s|+,+) = 1/10

Overconfidence!

Using NB for Classification

- We have a joint model of topics and documents

$$P(c, w_1, w_2, \ldots w_n) = P(c)\prod_i P(w_i \mid c)$$

- Gives posterior likelihood of topic given a document

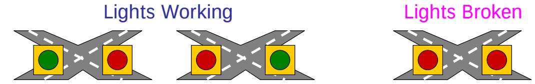
$$P(c \mid w_1, w_2, \ldots w_n) = \frac{P(c)\prod_i P(w_i \mid c)}{\sum_{c'}\left[P(c')\prod_i P(w_i \mid c')\right]}$$

- What about totally unknown words?
- Can work shockingly well for textcat (especially in the wild)
- How can unigram models be so terrible for language modeling, but class-conditional unigram models work for textcat?
- Numerical / speed issues
- How about NB for spam detection?

Example: Stoplights

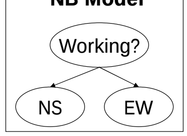
Reality

Lights Working | Lights Broken



P(g,r,w) = 3/7 P(r,g,w) = 3/7 P(r,r,b) = 1/7

NB Model

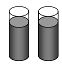


NB FACTORS:
- P(w) = 6/7
- P(r|w) = 1/2
- P(g|w) = 1/2
- P(b) = 1/7
- P(r|b) = 1
- P(g|b) = 0

P(b|r,r) = 4/10 (what happened?)

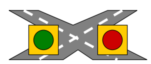
(Non-)Independence Issues

- Mild Non-Independence
  - Evidence all points in the right direction
  - Observations just not entirely independent
  - Results
    - Inflated Confidence
    - Deflated Priors
  - What to do? Boost priors or attenuate evidence



$$P(c, w_1, w_2, \ldots w_n) \text{ "=" } P(c)^{boost>1} \prod_i P(w_i \mid c)^{boost<1}$$

- Severe Non-Independence
  - Words viewed independently are misleading
  - Interactions have to be modeled
  - What to do?
    - Change your model!



Language Identification

- How can we tell what language a document is in?

The 38th Parliament will meet on Monday, October 4, 2004, at 11:00 a.m. The first item of business will be the election of the Speaker of the House of Commons. Her Excellency the Governor General will open the First Session of the 38th Parliament on October 5, 2004, with a Speech from the Throne.

La 38e législature se réunira à 11 heures le lundi 4 octobre 2004, et la première affaire à l'ordre du jour sera l'élection du président de la Chambre des communes. Son Excellence la Gouverneure générale ouvrira la première session de la 38e législature avec un discours du Trône le mardi 5 octobre 2004.

- How to tell the French from the English?
  - Treat it as word-level textcat?
    - Overkill, and requires a lot of training data
    - You don't actually need to know about words!

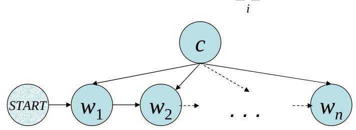
      Σύμφωνο σταθερότητας και ανάπτυξης
      Patto di stabilità e di crescita

  - Option: build a character-level language model

Class-Conditional LMs

- Can have a topic variable for other language models

$$P(c, w_1, w_2, \ldots w_n) = P(c) \prod_i P(w_i \mid w_{i-1}, c)$$



- Could be characters instead of words, used for language ID (HW2)
- Could sum out the topic variable and use as a language model
- How might a class-conditional n-gram language model behave differently from a standard n-gram model?