CS474 Natural Language Processing

- Last class
  - Introduction to generative models of language
    - » What are they?
    - » Why they're important
    - » Issues for counting words
    - » Statistics of natural language
    - » Unsmoothed n-gram models
- Today
  - Training n-gram models
  - Smoothing

N-gram approximations

- Bigram model

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-1})$$

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k|w_{k-1})$$

- Trigram model
  - Conditions on the two preceding words
- N-gram approximation

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k|w_{k-N+1}^{k-1})$$

- Markov assumption: probability of some future event (next word) depends only on a limited history of preceding events (previous words)

Bigram grammar fragment

- Berkeley Restaurant Project

eat on	.16	eat Thai	.03
eat some	.06	eat breakfast	.03
eat lunch	.06	eat in	.02
eat dinner	.05	eat Chinese	.02
eat at	.04	eat Mexican	.02
eat a	.04	eat tomorrow	.01
eat Indian	.04	eat dessert	.007
eat today	.03	eat British	.001

- Can compute the probability of a complete string
  - P (I want to eat British food) = P(I|<s>) P(want|I) P(to|want) P(eat|to) P(British|eat) P(food|British)

Training N-gram models

- N-gram models can be trained by counting and normalizing
  - Bigrams

$$P(w_n \mid w_1^{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

  - General case

$$P(w_n \mid w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}w_n)}{C(w_{n-N+1}^{n-1})}$$

  - An example of Maximum Likelihood Estimation (MLE)
    - » Resulting parameter set is one in which the likelihood of the training set T given the model M (i.e. P(T|M)) is maximized.

Bigram counts

	I	want	to	eat	Chinese	food	lunch
I	8	1087	0	13	0	0	0
want	3	0	786	0	6	8	6
to	3	0	10	860	3	0	12
eat	0	0	2	0	19	2	52
Chinese	2	0	0	0	0	120	1
food	19	0	17	0	0	0	0
lunch	4	0	0	0	0	1	0

- Note the number of 0's…

Bigram probabilities

- Problem for the maximum likelihood estimates: sparse data

	I	want	to	eat	Chinese	food	lunch
I	.0023	.32	0	.0038	0	0	0
want	.0025	0	.65	0	.0049	.0066	.0049
to	.00092	0	.0031	.26	.00092	0	.0037
eat	0	0	.0021	0	.020	.0021	.055
Chinese	.0094	0	0	0	0	.56	.0047
food	.013	0	.011	0	0	0	0
lunch	.0087	0	0	0	0	.0022	0

Accuracy of N-gram models

- Accuracy increases as N increases
  - Train various N-gram models and then use each to generate random sentences.
  - Corpus: Complete works of Shakespeare
    - » **Unigram:** *Will rash been and by I the me loves gentle me not slavish page, the and hour; ill let*
    - » **Bigram:** *What means, sir. I confess she? Then all sorts, he is trim, captain.*
    - » **Trigram:** *Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.*
    - » **Quadrigram:** *They say all lovers swear more performance than they are wont to keep obliged faith unforfeited!*

Strong dependency on training data

- Trigram model from WSJ corpus
  - *They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions*

Smoothing

- Need better estimators for rare events
- Approach
  - Somewhat decrease the probability of previously seen events, so that there is a little bit of probability mass left over for previously unseen events
    - » Smoothing
    - » Discounting methods

Add-one smoothing

- Add one to all of the counts before normalizing into probabilities
- Normal unigram probabilities

$$P(w_x) = \frac{C(w_x)}{N}$$

- Smoothed unigram probabilities

$$P(w_x) = \frac{C(w_x)+1}{N+V}$$

- Adjusted counts

$$c_i^* = (c_i + 1)\frac{N}{N+V}$$

Adjusted bigram counts

- Discount d_c

$$d_c = \frac{c^*}{c}$$

- Ratio of the discounted counts to the original counts

	I	want	to	eat	Chinese	food	lunch
I	8	1087	0	13	0	0	0
want	3	0	786	0	6	8	6
to	3	0	10	860	3	0	12
eat	0	0	2	0	19	2	52
Chinese	2	0	0	0	0	120	1
food	19	0	17	0	0	0	0
lunch	4	0	0	0	0	1	0

	I	want	to	eat	Chinese	food	lunch
I	6	740	.68	10	.68	.68	.68
want	2	.42	331	.42	3	4	3
to	3	.69	8	594	3	.69	9
eat	.37	.37	1	.37	7.4	1	20
Chinese	.36	.12	.12	.12	.12	15	.24
food	10	.48	9	.48	.48	.48	.48
lunch	1.1	.22	.22	.22	.22	.44	.22

Too much probability mass is moved

- Estimated bigram frequencies
- AP data, 44million words
- Church and Gale (1991)
- In general, add-one smoothing is a poor method of smoothing
- Much worse than other methods in predicting the actual probability for unseen bigrams
- Variances of the counts are worse than those from the unsmoothed MLE method

$r = f_{MLE}$	f_{emp}	f_{add-1}
0	0.000027	0.000137
1	0.448	0.000274
2	1.25	0.000411
3	2.24	0.000548
4	3.23	0.000685
5	4.21	0.000822
6	5.23	0.000959
7	6.21	0.00109
8	7.21	0.00123
9	8.26	0.00137

Methodology

- Cardinal sin: Testing on the training corpus
- Divide data into training set and test set
  - Train the statistical parameters on the training set; use them to compute probabilities on the test set
  - Test set: 5-10% of the total data, but large enough for reliable results
- Divide training into training and validation/held out set
    - » Obtain counts from training
    - » Tune smoothing parameters on the validation set
- Divide test set into development and final test set
  - Do all algorithm development by testing on the dev set, save the final test set for the very end…