Smoothing

LING 570

Fei Xia

Week 5: 10/26/09

Smoothing

- What? $$P(w_i|w_{i-1}) = \frac{c(w_i, w_{i-1})}{c(w_{i-1})}$$
- Why?
  - To deal with events observed zero times.
  - “event”: a particular ngram
- How?
  - To shave a little bit of probability mass from the higher counts, and pile it instead on the zero counts
  - For the time being, we assume that there are no unknown words; that is, V is a closed vocabulary.

Smoothing methods

- Laplace smoothing (a.k.a. Add-one smoothing)
- Good-Turing Smoothing

- Linear interpolation (a.k.a. Jelinek-Mercer smoothing)
- Katz Backoff

- Class-based smoothing
- Absolute discounting
- Kneser-Ney smoothing

Laplace smoothing

- Add 1 to all frequency counts.
- Let V be the vocabulary size.

unigram: $$P_{Lap}(w_i) = \frac{1+c(w_i)}{V+N}$$

Bigram: $$P_{Lap}(w_i|w_{i-1}) = \frac{1+c(w_{i-1},w_i)}{V+c(w_{i-1})}$$

n-gram: $$P_{Lap}(w_n|w_1, ..., w_{n-1}) = \frac{1+c(w_1,\ldots,w_n)}{V+c(w_1,\ldots,w_{n-1})}$$

Problem with Laplace smoothing

- Example: |V|=100K, a bigram “w1 w2” occurs 10 times, and the bigram ‘w1 w2 w3” occurs 9 times.
  - P_{MLE} (w3 | w1, w2) = 0.9
  - P_{Lap}(w3 | w1, w2) = (9+1)/(10+100K) = 0.0001

- Problem: give too much probability mass to unseen n-grams.

Add-one smoothing does not work well in practice.

Add-δ smoothing

$$P(w_i | w_{i-1}) = \frac{\delta + c(w_i, w_{i-1})}{\delta * V + c(w_{i-1})}$$

Need to choose δ

It works better than add-one,
but still works horribly.

Good-Turing smoothing

Basic ideas

- Re-estimate the frequency of zero-count N-grams with the number of N-grams that occur once.

- Let N_c be the number of n-grams that occurred c times.

- The Good-Turing estimate for any n-gram that occurs c times, we should pretend that it occurs c* times:

$$c^* = (c+1)\frac{N_{c+1}}{N_c}$$

$$P_{GT}(w_1, ..., w_n) = \frac{c^*(w_1, \ldots, w_n)}{N}$$

For unseen n-grams, we assume that each of them occurs c_0^* times.

$$c_0^* = \frac{N_1}{N_0}$$

N_0 is the number of unseen ngrams

Therefore, the prob of EACH unseen ngram is:

$$P_{GT}(w_1, ..., w_n) = \frac{c^*(w_1,...,w_n)}{N} = \frac{c_0^*}{N} = \frac{N_1}{N_0 * N}$$

The total prob mass for unseen ngrams is: $\frac{N_1}{N}$

An example

AP Newswire			Berkeley Restaurant		
c (MLE)	N_c	c^* (GT)	c (MLE)	N_c	c^* (GT)
0	74,671,100,000	0.0000270	0	2,081,496	0.002553
1	2,018,046	0.446	1	5315	0.533960
2	449,721	1.26	2	1419	1.357294
3	188,933	2.24	3	642	2.373832
4	105,668	3.24	4	381	4.081365
5	68,379	4.22	5	311	3.781350
6	48,190	5.19	6	196	4.500000

N-gram counts to conditional probability

$$P_{GT}(w_i | w_1, ..., w_{i-1}) = \frac{c^*(w_i, w_{i-1})}{c^*(w_1, \ldots, w_{i-1})}$$

c^* comes from GT estimate.

Backoff and interpolation

N-gram hierarchy

- $P_3(w3|w1,w2)$, $P_2(w3|w2)$, $P_1(w3)$
- Back off to a lower N-gram
  ➔ backoff estimation
- Mix the probability estimates from all the N-grams ➔ interpolation

Katz Backoff

$P_{katz}(w_i|w_{i-1}) =$

$P_2(w_i \mid w_{i-1})$ if $c(w_{i-1}, w_i) > 0$

$\alpha(w_{i-1})\, P_1(w_i)$ otherwise

$P_{katz}(w_i|w_{i-2}, w_{i-1}) =$

$P_3(w_i \mid w_{i-2}, w_{i-1})$ if $c(w_{i-2}, w_{i-1}, w_i) > 0$

$\alpha(w_{i-2}, w_{i-1})\, P_{katz}(w_i|w_{i-1})$ otherwise

Katz backoff (cont)

- α are used to normalize probability mass so that it still sums to 1, and to “smooth” the lower order probabilities that are used.

- See J&M Sec 4.7.1 for details of how to calculate α (and M&S 6.3.2 for additional discussion)

Jelinek-Mercer smoothing (interpolation)

Bigram: $P(w_i|w_{i-1}) = \lambda_2 P_2(w_i|w_{i-1}) + \lambda_1 P_1(w_i)$

Trigram:

$$P(w_i|w_{i-1}, w_{i-2}) =$$

$$\lambda_3 P_3(w_i|w_{i-1}, w_{i-2}) + \lambda_2 P_2(w_i|w_{i-1}) + \lambda_1 P_1(w_i)$$

Interpolation (cont)

$$P(w_i|w_{i-1}) = \lambda_2(w_{i-1})P_2(w_i|w_{i-1}) + \lambda_1(w_{i-1})P_1(w_i)$$

$$P(w_i|w_{i-2}, w_{i-1}) = \lambda_3(w_{i-2}, w_{i-1})P_3(w_i|w_{i-2}, w_{i-1})$$
$$+\lambda_2(w_{i-2}, w_{i-1})P_2(w_i|w_{i-1}) + \lambda_1(w_{i-2}, w_{i-1})P_1(w_i)$$

How to set the value for λ_i?

How to set λ_i?

- Generally, here's what's done:
  - Split data into training, held-out, and test
  - Train model on training set
  - Use held-out to test different values and pick the ones that works best (i.e., maximize the likelihood of the held-out data)
  - Test the model on the test data

Summary

- Laplace smoothing: δ
- Good-Turing Smoothing: gt_min[i], gt_max[i].
- Linear interpolation: $\lambda_i(w_{i-2}, w_{i-1})$
- Katz Backoff: $\alpha(w_{i-2}, w_{i-1})$

Additional slides

Issues in Good-Turing estimation

- If N_{c+1} is zero, how to estimate c^*?
  - Smooth N_c by some functions:
    Ex: $\log(N_c) = a + b \log(c)$
  - Large counts are assumed to be reliable ➔ gt_max[ ]
    Ex: $c^* = c$ for $c >$ gt_max
- May also want to treat n-grams with low counts (especially 1) as zeroes ➔ gt_min[ ].
- Need to renormalize all the estimate to ensure that the probs add to one.
- Good-Turing is often not used by itself; it is used in combination with the backoff and interpolation algorithms.

One way to implement Good-Turing

- Let N be the number of trigram tokens in the training corpus, and min3 and max3 be the min and max cutoffs for trigrams.

- From the trigram counts

  calculate N_0, N_1, …, N_{max3+1}, and N

  calculate a function f(c) , for c=0, 1, …, max3.

$$f(c) = (c+1)\frac{N_{c+1}}{N_c}$$

- Define $c^* = c$ if c > max3

  $= f(c)$ otherwise

- Do the same for bigram counts and unigram counts.

Good-Turing implementation (cont)

- Estimate trigram conditional prob:

$$P_{GT}(w_3|w_1, w_2) = \frac{c^*(w1, w2, w3)}{c^*(w1, w2)}$$

- For an unseen trigram, the joint prob is:

$$P_{GT}(w1, w2, w3) = \frac{c_0^*}{N} = \frac{N_1}{N_0 * N}$$

Do the same for unigram and bigram models

Another example for Good-Turing

- 10 tuna, 3 unagi, 2 salmon, 1 shrimp, 1 octopus, 1 yellowtail

- How likely is *octopus*? Since c(*octopus*) = 1. The GT estimate is 1*.

- To compute 1*, we need n_1=3 and n_2=1.

$$1^* = 2 * \frac{1}{3} = \frac{2}{3}$$

- What happens when $N_c = 0$?

Absolute discounting

c (MLE)	0	1	2	3	4	5	6	7	8	9
c^* (GT)	0.0000270	0.446	1.26	2.24	3.24	4.22	5.19	6.21	7.24	8.25

$$P_{abs}(y \mid x) = \begin{cases} \frac{c(xy)-D}{c(x)} & \text{if } c(xy) > 0 \\ \alpha(x) P_{abs}(y) & \text{otherwise} \end{cases}$$

What is the value for D?

How to set α(x)?

Intuition for Kneser-Ney smoothing

- I cannot find my reading __

- P(Francisco | reading) > P(glasses | reading)
  - *Francisco* is common, so interpolation gives P(Francisco | reading) a high value

  - But *Francisco* occurs in few contexts (only after *San*), whereas *glasses* occurs in many contexts.

  - Hence weight the interpolation based on number of contexts for the word using discounting

➔ Words that have appeared in more contexts are more likely to appear in some new context as well.

Kneser-Ney smoothing (cont)

$$P_{continuation}(w_i) = \frac{|\{w|c(w,w_i)>0\}|}{\sum_{w'}|\{w|c(w,w')>0\}|}$$

Backoff:

$$P_{KN}(w_i|w_{i-1}) = \begin{cases} \frac{c(w_{i-1},w_i)-D}{c(w_{i-1})} & c(w_{i-1},w_i) > 0 \\ \alpha(w_{i-1})P_{cont}(w_i) & otherwise \end{cases}$$

Interpolation:

$$P_{KN}(w_i|w_{i-1}) = \frac{c(w_{i-1},w_i) - D}{c(w_{i-1})} + \beta(w_{i-1})P_{cont}(w_i)$$

Class-based LM

- Examples:
  - The stock rises to $81.24
  - He will visit Hyderabad next January

- $P(w_i \mid w_{i-1})$
  $\approx P(c_{i-1} \mid w_{i-1}) * P(c_i \mid c_{i-1}) * P(w_i \mid c_i)$

- Hard clustering vs. soft clustering

Summary

- Laplace smoothing (a.k.a. Add-one smoothing)
- Good-Turing Smoothing
- Linear interpolation: $\lambda(w_{i-2},w_{i-1})$, $\lambda(w_{i-1})$
- Katz Backoff: $\alpha(w_{i-2},w_{i-1})$, $\alpha(w_{i-1})$
- Absolute discounting: D, $\alpha(w_{i-2},w_{i-1})$, $\alpha(w_{i-1})$
- Kneser-Ney smoothing: D, $\alpha(w_{i-2},w_{i-1})$, $\alpha(w_{i-1})$
- Class-based smoothing: clusters