Smoothing

Manish Shrivastava

Language Modeling - Recap

- Prediction based on History
- Markov Assumption
- NGrams
- MLE

Example

- Data
  - The boy ate a chocolate
  - The girl bought a chocolate
  - The girl then ate a chocolate
  - The horse bought a boy

- The boy bought a chocolate
  - Unigram Probabilities
    - (4/16)*(2/16)*(2/16)*(4/16)*(3/16)
    - (4*2*2*4*3)/21^5 = 0.000047
  - Bi-gram Probabilities
    - <The boy> <boy bought> <bought a> <a chocolate>
    - (2/4)*(0/4)*(2/2)*(3/4) = **<u>0</u>**

Smoothing

- Kinds of Zeros.
- Higher order n-grams perform well but suffer from data sparsity
- Lower order n-grams are not reliable
- Standard MLE does not work for unseen data

Smoothing

- Laplace smoothing (add-one)
- Lidstone's law
- Held-out Estimation
- Good Turing Estimation

Laplace Smoothing

- The oldest smoothing method available
- Each unseen n-gram is given a very low estimate
- $P_{Lap}(w_1 \dots w_{n-1} w_n) = \frac{C(w_1 \dots w_n)+1}{N+B}$
- Probability 1/(N+B) or Frequency (estimated) N/(N+B)
- N is the number of seen n-grams and B is the number of possible n-grams

Laplace Smoothing

* Estimating Recounts
* What if V is infinite ?

Lidstone's Law

- The probability mass assigned to unseen events is too high with Laplace smoothing
- Solution : assign smaller addition
  - $P_{Lid}(w_1 \dots w_{n-1} w_n) = \frac{C(w_1 \dots w_n)+\lambda}{N+B\lambda}$
- Jeffereys-Perks law
  - Set λ as ½
  - Expectation of λ that maximizes above equation

Held Out Estimation

- So far, we have not considered how the learned estimates behave in real life
- It is possible that the learnt values are just a property of the training data
  - Imagine training from Chemical domain and applying in Physics domain
- Also, adding a random value is not very sound
  - At higher total count the probability of something that occurred once and something that never occurred would be very close
  - Adding small value also increases the total denominator.
    - $B\lambda$ might become a very large part of $(N + B\lambda)$
    - In other words, a large probability mass might be assigned to unseen events

Held-out estimates

- Treat part of the data as “real life” data
- Intuition :
  - Estimates of bigrams with similar frequencies, would be similar
  - Example: if <a boy> occurs 3 times and <a girl> occurs 3 times, instead of treating them as different, we can consider them to belong to one class or bin ;
    - Bigrams with frequency 3
    - Now we can calculate the average probability for all bigrams with frequency 3

Held-Out Estimation

- Method:
  - Partition data into two parts
    - Training and Held-Out
  - Nr = Number of n-grams with frequency ‘r’ in Training data
  - Tr = sum of number of times all the n-grams with frequency r, as identified in the training data, appear in the Held-Out data

$$T_r = \sum_{\{w_1 \ldots w_n : C_{tr}(w_1 \ldots w_n) = r\}} C_{ho}(w_1 \ldots w_n)$$

- Then

$$P_{ho}(w_1 \ldots w_n) = \frac{T_r}{N_r N}$$

Held Out Estimation

Freq	Training Data (number of words with given frequency)	Held Out Data (Sum of Frequencies of words with r freq in Training)
1	N1	T1 = (Nr*r*)
2	N2	T2
3	N3	T3
4	N4	T4
.		
.		
r	Nr (average frequency of these words = Nr*r/Nr = r)	Tr (average frequency of these words = Tr/Nr = r*)
.		
.		

Held-out Estimation

- Probability is calculated keeping both training and held-out data in mind.
  - Is it the best estimate?
  - In this scenario?
  - If the role of training and held-out data is reversed ?
    - $P_{Ho}(w_1 \ldots w_n) = \frac{T_r^{01}}{N_r^0 N}$
    - $P_{Ho}(w_1 \ldots w_n) = \frac{T_r^{01}}{N_r^0 N}$

Deleted Estimation (Cross-Estimation)

- Jelinek and Mercer 1985
  - $P_{Del}(w_1 \ldots w_n) = \frac{T_r^{01} + T_r^{10}}{(N_r^0 + N_r^1)N}$
  - Performs really well
  - Still a way off for low frequency events

Good Turing Smoothing

- Word classes with similar frequency counts can be treated similarly
  - Or can help each other
- A word which is at frequency 1 is a rare word, it is just by chance that you have been able to see it (data property)
- We should be able to compute (approximately) probabilities of lower frequency words with the help of higher frequency words

$$r^* = (r+1)\frac{E(N_r+1)}{E(N_r)}$$

- Here r* is the adjusted frequency
- The probability then is r*/N

Good-Turing Smoothing

- Use Number of times n-grams of frequency r have occurred
  - To fit for both high frequency and low frequency terms, use a curve fitting function
    - Where
  - For unseen events,
- Simple Good Turing
  - S() = Power Curve for high frequency terms

- Use Number of times n-grams of frequency r have occurred
  - To fit for both high frequency and low frequency terms, use a c
  - $P_{GT}(w_1 \dots w_n) = \frac{r^*}{N}$
    - Where $r^* = \frac{(r+1)S(r+1)}{S(r)}$
  - For unseen events,
    - $P_{GT}(w_1 \dots w_n) = \frac{N_1}{N * N_0}$
- Simple Good Turing
  - S() = Power Curve for high frequency terms
  - $N_r = ab^r$

Example

- Consider the data
  - I like strong coffee
  - I bought some coffee
  - I drank black coffee
  - Good coffee is strong coffee
- Then the test sentence
  - I like black strong coffee
- The trigram <black strong coffee> is never seen before and will be assigned some small probability
  - But bigram <strong coffee> is seen and is very informative

Back Off

- Back off is “using lower order N-grams for estimating higher orders”

Back off

- Back-Off
  - Linear Interpolation
  - Katz Backoff
  - General linear Interpolation

Back off

- So far, all we have done is get estimates for various n-grams
- Problems (Same old),
  - For n-grams with low frequency, estimates are not really good
- Solution,
  - If n-gram does not work, fall back to (n-1)-gram
  - Better still, combine estimators for n-gram, (n-1)-gram, (n-2)-gram … unigram

Katz Back-off

- if Counts are greater than a certain k
- $P_{li}(w_n|w_1 \ldots w_{n-1}) = \alpha * \frac{C(w_1 \ldots w_n)}{C(w_1 \ldots w_{n-1})}$ if Counts are greater than a certain k
- Else
- Else
- $P_{li}(w_n|w_1 \ldots w_{n-1}) = (1-\alpha) * P_{li}(w_n|w_2 \ldots w_{n-1})$

Linear Interpolation

- $P_{li}(w_n|w_1 \dots w_{n-1}) = \lambda_1 P(w_n) + \lambda_2 P_{li}(w_n|w_{n-1}) + \lambda_3 P_{li}(w_n|w_{n-1}, w_{n-2})$
- Works really well
- Reserve Probability mass for unseen items
- λ_is need to be learned separately

- Works really well
- Reserve Probability mass for unseen items
- s need to be learned separately

General Linear Interpolation

- *Why just back off to immediate lower order?*
- This method allows random back off schemes moderated by the weights λ_is
- One might back off to any of the lower orders or to any other estimator as chosen by the designer
- $P_{li}(w_n|h) = \sum_{i=1}^{k} \lambda_i P_i(w_n|h)$

Witten Bell Smoothing

$$P_{wb}(w_i \mid w_{i-n+1} \ldots w_{i-1}) = \lambda_{w_{i-n+1} \ldots w_{i-1}} P_{wb}(w_i \mid w_{i-n+1} \ldots w_{i-1}) + (1 - \lambda_{w_{i-n+1} \ldots w_{i-1}}) P_{wb}(w_i \mid w_{i-n+2} \ldots w_{i-1})$$

Where,

$$(1 - \lambda_{w_{i-n+1} \ldots w_{i-1}}) = \frac{|\{w_i \mid C(w_{i-n+1} \ldots w_i) > 0\}|}{|\{w_i \mid C(w_{i-n+1} \ldots w_i) > 0\}| + \sum_{w_i} C(w_{i-n+1} \ldots w_i)}$$

Sample Data

- The boy ate a chocolate
- The girl bought a chocolate
- The girl then ate a chocolate
- The boy bought a horse
- The boy has a horse
- The boy stole a chocolate

- Compute likelihood of
  - The girl stole a horse

Questions?