Natural Language Processing

Paper Code: CS-821/6

UG 8th Sem

Dr. Samit Biswas, *Assistant Professor*

Department of Computer Sc. and Technology

Indian Institute of Engineering Science and Technology, Shibpur

Email: samit@cs.iiests.ac.in

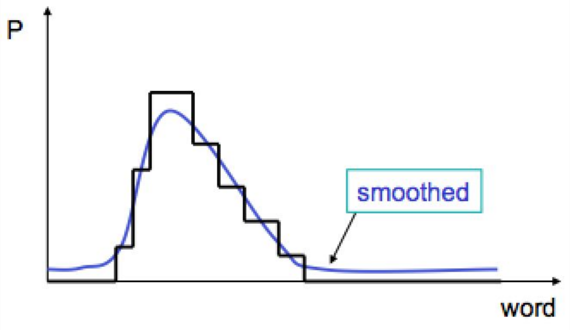
Plan for Today

- Language Model Smoothing

Language Model Smoothing

Language Model Smoothing

- Since N-gram tables are too sparse, there will be a lot of entries with zero probability (or with very low probability).
- The reason for this, our corpus is finite and it is not big enough to get that much information.
- **The task of re-evaluating some of zero-probability and low-probability N-Grams is called Smoothing.**



Zero Probability

- **Probability Function:**

Training Set	Test Set
… denied the allegations	… denied the offer
... denied the reports	... denied the loan
… denied the claims	
… denied the request	

P("offer" | denied the) = 0

Smoothing Techniques

Smoothing Techniques

- **Add one smoothing :** add one to all counts
- **Witten Bell Discounting:** use the count of things you have seen once to help estimate the count of things you have never seen.
- **Good-Turing Discounting** – a slightly more complex form of *Witten-Bell Discounting*.
- **Backoff** – using lower level N-Gram probabilities when N-gram probability is zero.

Add-one Smoothing

Add-one Smoothing (Laplace Correction)

- Assume each bigram having zero occurrence has a count of 1.
- Increase the count of all non-zero occurrence words by one. This increases the total number of words **N** in the corpus by the vocabulary **V**.
- Probability of each word after **add-one smoothing**:
    - **Unigram:** $$P_L(w_i) = \frac{C(w_i)+1}{N+V} = \frac{C_i+1}{N+V}$$
    - **Bigram** $$P_L(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

Example: Add-one Smoothing

Example: Add-one Smoothing

xya	100	100/300	101	101 / 326
xyb	0	0/300	1	1 / 326
xyc	0	0/300	1	1 / 326
xyd	200	200/300	201	201 / 326
xye	0	0/300	1	1 / 326
⋮				
xyz	0	0/300	1	1 / 326
Total xy	300	300/300	326	326 / 326

Add-one Smoothing

Problem with Add-one Smoothing

- each individual **unseen n-gram** is given a low probability.
- but there is a huge number of **unseen n-grams**:
    - Adding a little of probability over a huge number of unseen events gives too much probability mass to all unseen events
- Instead of giving small portion of probability to **unseen events**, most of the probability space is given to **unseen events.**

Concept of “Discounting”

Concept of “Discounting”

- The concept is the central idea in all smoothing algorithms.
- To assign some probability mass to unseen event, we need to take away some probability mass from seen events.
- Discounting is the lowering each non-zero count c to c^* according the smoothing algorithm.
- It is convenient to describe a smoothing algorithm as a corrective constant that affects the numerator by defining an adjusted count c^* as follows:

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

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

Discounting

- A related way to view smoothing is as discounting (lowering) some non-zero counts in order to get the correct probability mass that will be assigned to the zero counts.
- Thus instead of referring to the discounted counts c, we might describe a smoothing algorithm in terms of a relative discount d_c, the ratio of the discounted counts to the original counts:

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

Witten-Bell Discounting

Witten-Bell Discounting

- Use the count of things that you‘ve seen only once to estimate the count of things you have never seen.
- Total probability mass assigned to zero-frequency unigrams (T #observed types; N # word instances/tokens):

$$\sum_{i:c_i=0} p_i^* = \frac{T}{N+T}$$

- So each zero **N-gram** gets the probability:

$$Z = \sum_{i:c_i=0} 1$$

$$p_i^* = \frac{T}{Z(N+T)}$$

Witten-Bell: why Discounting

- Now of course we have to take away something ('discount') from the probability of the events seen more than once:

$$\text{If } c_i > 0 \;\Big|\; p_i^* = \frac{c_i}{N+T}$$

Witten-Bell: for bigrams

- We 'relativize' the types to the previous word:

$$\sum_{i:c(w_x w_i)=0} p^*(w_i|w_x) = \frac{T(w_x)}{N(w_x) + T(w_x)}$$

- this probability mass, must be distributed in equal parts over all unseen bigrams
  - $Z(w_1)$: number of unseen n-grams starting with w_1

for each unseen event	$P(w_2|w_1) = \frac{1}{Z(w_1)} \frac{T(w_1)}{N(w_1)+T(w_1)}$

Example: **Witten-Bell discounting**

	a	b	c	d	Total = $N(w_1)$ seen tokens	$T(w_1)$ seen types	$z(w_1)$ unseen types
a	10	10	10	0	30	3	1
b	0	0	30	0	30	1	3
c	0	0	300	0	300	1	3
d							
...							

- all unseen bigrams starting with a will share a probability mass of

$$\frac{T(a)}{N(a)+T(a)} = \frac{3}{30+3} = 0.091 \qquad (10)$$

- each unseen bigram starting with a will have an equal part of this

$$P(d|a) = \frac{1}{Z(a)} \frac{T(a)}{N(a)+T(a)} = \frac{1}{1} \frac{3}{30+3} = 0.091 \qquad (11)$$

Example: **Witten-Bell discounting**

	a	b	c	d	Total = $N(w_1)$ seen tokens	$T(w_1)$ seen types	$z(w_1)$ unseen types
a	10	10	10	0	30	3	1
b	0	0	30	0	30	1	3
c	0	0	300	0	300	1	3
d							
...							

- all unseen bigrams starting with b will share a probability mass of

$$\frac{T(b)}{N(b)+T(b)} = \frac{1}{30+1} = 0.032 \qquad (12)$$

- each unseen bigram starting with b will have an equal part of this

$$P(a|b) = P(b|b) = P(d|b) = \frac{1}{Z(b)}\frac{T(b)}{N(b)+T(b)} = \frac{1}{3}\times 0.032 = 0.011$$

Example: **Witten-Bell discounting ...**

	a	b	c	d	Total = $N(w_1)$ seen tokens	$T(w_1)$ seen types	$z(w_1)$ unseen types
a	10	10	10	0	30	3	1
b	0	0	30	0	30	1	3
c	0	0	300	0	300	1	3
d							
...							

- all unseen bigrams starting with c will share a probability mass of

$$\frac{T(c)}{N(c)+T(c)} = \frac{1}{300+1} = 0.0033 \qquad (14)$$

- each unseen bigram starting with c will have an equal part of this

$$P(a|c) = P(b|c) = P(d|c) = \frac{1}{Z(c)} \frac{T(c)}{N(c)+T(c)} = \frac{1}{3} \times 0.0033 = 0.0011$$

Back to counts: **Witten-Bell discounting**

- To get from probabilities back to the counts, we know that:

$$P(w_2|w_1) = \frac{C(w_2|w_1)}{N(w_1)}$$

- so, we get:

$$\begin{aligned} C(w_2|w_1) &= P(w_2|w_1) \times N(w_1) \\ &= \frac{1}{Z(w_1)} \frac{T(w_1)}{N(w_1) + T(w_1)} \times N(w_1) \\ &= \frac{T(w_1)}{Z(w_1)} \times \frac{N(w_1)}{N(w_1) + T(w_1)} \end{aligned}$$

References

References

- Daniel Jurafsky & James H. Martin, “*Speech and Language Processing*”, Prentice Hall
- Diana Maynard, Kalina Bontcheva, Isabelle Augenstein, “*Natural Language Processing for the Semantic Web*”, A Publication in the Morgan & Claypool Publishers series.
- Steven Bird, Ewan Klein and Edward Loper, “*Natural Language Processing with Python*”, By O‘Reilly.
- Christopher Manning and Hinrich Schtze, “*Foundations of Statistical Natural Language Processing*”, The MIT Press.

Thank You

Contacts:

Dr. Samit Biswas
Department of CST,
IIEST, Shibpur

samit@cs.iiests.ac.in
https://www.iiests.ac.in/IIEST/Faculty/cs-samit