HPI Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Natural Language Processing

Language Modeling

Potsdam, 19 April 2012

Saeedeh Momtazi
Information Systems Group

based on the slides of the course book

Outline

1. Motivation
2. Estimation
3. Evaluation
4. Smoothing

Outline

1. Motivation
2. Estimation
3. Evaluation
4. Smoothing

Language Modeling

Hasso Plattner Institut

- Finding the probability of a sentence or a sequence of words

$$P(S) = P(w_1, w_2, w_3, ..., w_n)$$

- Applications:
  - Word prediction
  - Speech recognition
  - Machine translation
  - Spell checker

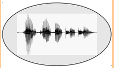
Applications

- Word Prediction

"natural language ..." $\Rightarrow$ *"processing"*
"management"

Applications

- Speech recognition



⇒ *"Computers can recognize speech."*
"Computers can wreck a nice peach."

Applications

- Machine translation

"The cat eats ..."	⇒	*"Die Katze frisst ..."*
		"Die Katze isst ..."

Applications

Hasso Plattner Institut

- Spell checker

"I want to adver this project." ⇒ *"advert"*
"adverb"

Outline

Language Modeling

- Finding the probability of a sentence or a sequence of words

$$P(S) = P(w_1, w_2, w_3, ..., w_n)$$

P(Computer, can, recognize, speech)

Conditional Probability

$$P(B|A) = \frac{P(A,B)}{P(A)}$$

$$P(A,B) = P(A) \cdot P(B|A)$$

$$P(A,B,C,D) = P(A) \cdot P(B|A) \cdot P(C|A,B) \cdot P(D|A,B,C)$$

$$P(S) = P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_1,w_2) \cdots P(w_n|w_1,w_2,w_3,...,w_{n-1})$$

$$P(S) = \prod_{i=1}^{n} P(w_i|w_1,w_2,...,w_{i-1})$$

Conditional Probability

$$P(S) = \prod_{i=1}^{n} P(w_i | w_1, w_2, ..., w_{i-1})$$

$P(Computer, can, recognize, speech) =$

$P(Computer) \cdot P(can|Computer) \cdot P(recognize|Computer\ can) \cdot P(speech|Computer\ can\ recognize)$

Corpus

- Probabilities are based on counting things
- Counting of thing in natural language is based on a corpus
  (plural: corpora)
- A computer-readable collection of text or speech
  - The Brown Corpus
    - A million-word collection of samples
    - 500 written texts from different genres
      (newspaper, fiction, non-fiction, academic, ...)
    - Assembled at Brown University in 1963-1964
  - The Switchboard Corpus
    - A collection of 240 hours of telephony conversations
    - 3 million words in 2430 conversations averaging 6 minutes each
    - Collected in early 1990s

Corpus

- Text Corpora
  - The Brown Corpus
  - Corpus of Contemporary American English
  - The British National Corpus
  - The International Corpus of English
  - The Google *N*-gram Corpus

Word Occurrence

HPI Hasso Plattner Institut

- A language consist of a set of V words (Vocabulary)
- A text is a sequence of the words from the vocabulary

- A word can occur several times in a text
  - Word Token: each occurrence of words in text
  - Word Type: each unique occurrence of words in the text

Example:

This is a sample text from a book that is read every day

\# Word Tokens: 13
\# Word Types: 11

Counting

- Brown
  - 1,015,945 word tokens
  - 47,218 word types

- Google *N*-gram
  - 1,024,908,267,229 word tokens
  - 13,588,391 word types

That seems like a lot of types...
Even large dictionaries of English have only around 500k types.
Why so many here?
Numbers
Misspellings
Names
Acronyms

Word Frequency

Hasso Plattner Institut

Rank	Word	Count	Freq(%)
1	The	69970	6.8872
2	of	36410	3.5839
3	and	28854	2.8401
4	to	26154	2.5744
5	a	23363	2.2996
6	in	21345	2.1010
7	that	10594	1.0428
8	is	10102	0.9943
9	was	9815	0.9661
10	He	9542	0.9392
11	for	9489	0.9340
12	it	8760	0.8623
13	with	7290	0.7176
14	as	7251	0.7137
15	his	6996	0.6886
16	on	6742	0.6636
17	be	6376	0.6276
18	at	5377	0.5293
19	by	5307	0.5224
20	I	5180	0.5099

Word Frequency

Rank	Word	Count	Freq(%)	Freq x Rank
1	The	69970	6.8872	0.06887
2	of	36410	3.5839	0.07167
3	and	28854	2.8401	0.08520
4	to	26154	2.5744	0.10297
5	a	23363	2.2996	0.11498
6	in	21345	2.1010	0.12606
7	that	10594	1.0428	0.07299
8	is	10102	0.9943	0.07954
9	was	9815	0.9661	0.08694
10	He	9542	0.9392	0.09392
11	for	9489	0.9340	0.10274
12	it	8760	0.8623	0.10347
13	with	7290	0.7176	0.09328
14	as	7251	0.7137	0.09991
15	his	6996	0.6886	0.10329
16	on	6742	0.6636	0.10617
17	be	6376	0.6276	0.10669
18	at	5377	0.5293	0.09527
19	by	5307	0.5224	0.09925
20	I	5180	0.5099	0.10198

$$Freq \cdot Rank \approx c$$

Zipf's Law

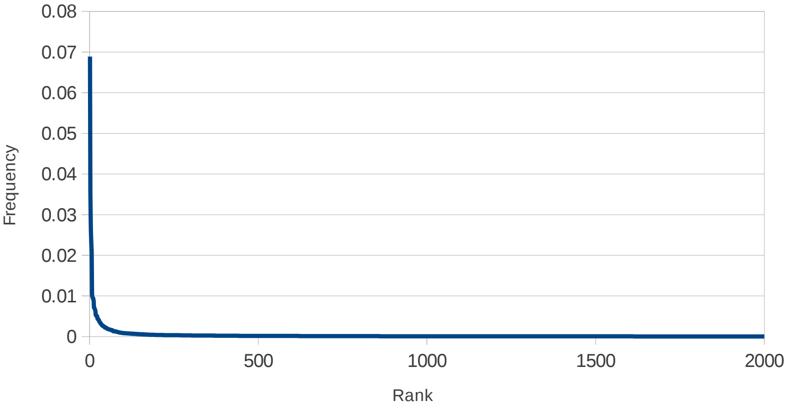
- The frequency of any word is inversely proportional to its rank in the frequency table
- Given a corpus of natural language utterances, the most frequent word will occur approximately
  - twice as often as the second most frequent word,
  - three times as often as the third most frequent word,
  - ...

$\Rightarrow$ Rank of a word times its frequency is approximately a constant

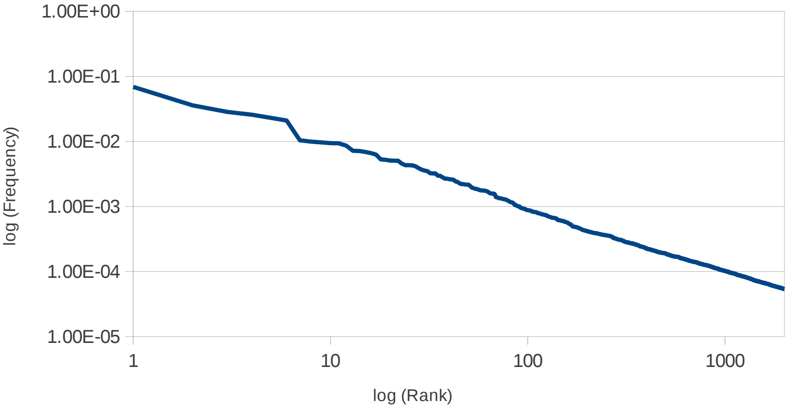
$$Rank \cdot Freq \approx c$$

$$c \approx 0.1 \text{ for English}$$

Zipf's Law


0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
Frequency
0
500
1000
1500
2000
Rank

Zipf's Law


1.00E+00
1.00E-01
1.00E-02
1.00E-03
1.00E-04
1.00E-05
log (Frequency)
1
10
100
1000
log (Rank)

Word Frequency

Hasso Plattner Institut

- Zipf's Law is not very accurate for very frequent and very infrequent words

Rank	Word	Count	Freq(%)	Freq x Rank
1	The	69970	6.8872	0.06887
2	of	36410	3.5839	0.07167
3	and	28854	2.8401	0.08520
4	to	26154	2.5744	0.10297
5	a	23363	2.2996	0.11498

Word Frequency

- Zipf's Law is not very accurate for very frequent and very infrequent words

Rank	Word	Count	Freq(%)	Freq x Rank
1000	current	104	0.0102	0.10200
1001	spent	104	0.0102	0.10210
1002	eight	104	0.0102	0.10220
1003	covered	104	0.0102	0.10230
1004	Negro	104	0.0102	0.10240
1005	role	104	0.0102	0.10251
1006	played	104	0.0102	0.10261
1007	I'd	104	0.0102	0.10271
1008	date	103	0.0101	0.10180
1009	council	103	0.0101	0.10190
1010	race	103	0.0101	0.10201

Maximum Likelihood Estimation

$$P(speech|Computer\ can\ recognize)$$

$$P(speech|Computer\ can\ recognize) = \frac{\#(Computer\ can\ recognize\ speech)}{\#(Computer\ can\ recognize)}$$

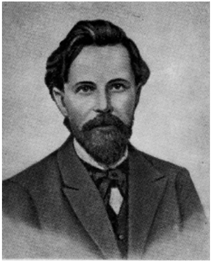
- Too many phrases
- Limited text for estimating the probability

$\Rightarrow$ Making a simplification assumption

Markov Assumption

$$P(S) = \prod_{i=1}^{n} P(w_i | w_1, w_2, ..., w_{i-1})$$

$$P(S) = \prod_{i=1}^{n} P(w_i | w_{i-1})$$



$P(Computer, can, recognize, speech) =$
$P(Computer) \cdot P(can|Computer) \cdot P(recognize|can) \cdot P(speech|recognize)$

$$P(speech|recognize) = \frac{\#(recognize\ speech)}{\#(recognize)}$$

N-gram Model

Unigram $P(S) = \prod_{i=1}^{n} P(w_i)$

Bigram $P(S) = \prod_{i=1}^{n} P(w_i | w_{i-1})$

Trigram $P(S) = \prod_{i=1}^{n} P(w_i | w_{i-2}, w_{i-1})$

N-gram $P(S) = \prod_{i=1}^{n} P(w_i | w_1, w_2, ..., w_{i-1})$

Maximum Likelihood

Hasso Plattner Institut

<s> I saw the boy </s>
<s> the man is working </s>
<s> I walked in the street </s>

Vocab:
I saw the boy man is working walked in street

boy I in is man saw street the walked working

Maximum Likelihood

Hasso Plattner Institut

<s> I saw the boy </s>
<s> the man is working </s>
<s> I walked in the street </s>

boy	I	in	is	man	saw	street	the	walked	working
1	2	1	1	1	1	1	3	1	1

	boy	I	in	is	man	saw	street	the	walked	working
boy	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	1	0	0	1	0
in	0	0	0	0	0	0	0	1	0	0
is	0	0	0	0	0	0	0	0	0	1
man	0	0	0	1	0	0	0	0	0	0
saw	0	0	0	0	0	0	0	1	0	0
street	0	0	0	0	0	0	0	0	0	0
the	1	0	0	0	1	0	1	0	0	0
walked	0	0	1	0	0	0	0	0	0	0
working	0	0	0	0	0	0	0	0	0	0

Maximum Likelihood

<s> I saw the man </s>

boy	I	in	is	man	saw	street	the	walked	working
1	2	1	1	1	1	1	3	1	1

	boy	I	in	is	man	saw	street	the	walked	working
boy	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	1	0	0	1	0
in	0	0	0	0	0	0	0	1	0	0
is	0	0	0	0	0	0	0	0	0	1
man	0	0	0	1	0	0	0	0	0	0
saw	0	0	0	0	0	0	0	1	0	0
street	0	0	0	0	0	0	0	0	0	0
the	1	0	0	0	1	0	1	0	0	0
walked	0	0	1	0	0	0	0	0	0	0
working	0	0	0	0	0	0	0	0	0	0

$$P(S) = P(I) \cdot P(saw|I) \cdot P(the|saw) \cdot P(man|the)$$

$$P(S) = \frac{\#(I)}{\#(<s>)} \cdot \frac{\#(I\ saw)}{\#(I)} \cdot \frac{\#(saw\ the)}{\#(saw)} \cdot \frac{\#(the\ man)}{\#(the)}$$

$$P(S) = \frac{2}{3} \cdot \frac{1}{2} \cdot \frac{1}{1} \cdot \frac{1}{3}$$

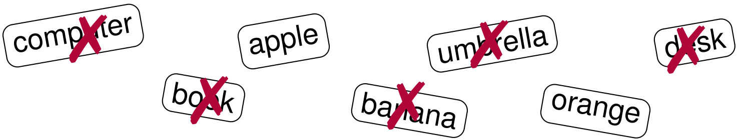
Outline

1. Motivation
2. Estimation
3. Evaluation
4. Smoothing

Branching Factor

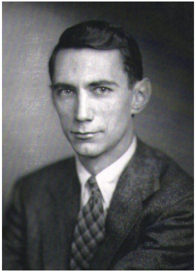
- Branching factor is the number of possible words that can be used in each position of a text
  - Maximum branching factor for each language is V
  - A good language model should be able to
    - minimize this number
    - give a higher probability to the words that occur in real texts

John eats an ...



Shannon Game

Shannon's Experiment to Calculate the Entropy of English



`http://www.math.ucsd.edu/~crypto/java/ENTROPY/`

Can we give the same knowledge to a computer to predict the next character?

Perplexity

- Dividing the corpus to two parts

- Building a language model from the training set
- Estimating the probability of the test set
- Calculate the average branching factor of the test set

Perplexity

$$P(S) = P(w_1, w_2, ..., w_n)$$

$$Perplexity(S) = P(w_1, w_2, ..., w_n)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1, w_2, ..., w_n)}}$$

$$Perplexity(S) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1, w_2, ..., w_{i-1})}}$$

Goal: giving higher probability to frequent texts
$\Rightarrow$ minimizing the perplexity of the frequent texts

Perplexity

- Maximum branching factor for each language is $|V|$

$$Perplexity(S) = (\prod_{i=1}^{N} P(w_i|w_1, w_2, ..., w_{i-1}))^{-\frac{1}{N}}$$

- Example: predicting next characters instead of next words $(|V| = 26)$

_	_	_	_	_
26	26	26	26	26

$$Perplexity(S) = ((\frac{1}{26})^5)^{-\frac{1}{5}} = 26$$

Perplexity

- Wall Street Journal
  - Training set: 38 million word tokens
  - Test set: 1.5 million words

	Unigram	Bigram	Trigram
Perplexity	962	170	109

Outline

Hasso Plattner Institut

1. Motivation
2. Estimation
3. Evaluation
4. Smoothing

Maximum Likelihood

<s> I saw the man </s>

$$P(S) = P(I) \cdot P(saw|I) \cdot P(the|saw) \cdot P(man|the)$$

$$P(S) = \frac{\#(I)}{\#(<s>)} \cdot \frac{\#(I\ saw)}{\#(I)} \cdot \frac{\#(saw\ the)}{\#(saw)} \cdot \frac{\#(the\ man)}{\#(the)}$$

$$P(S) = \frac{2}{3} \cdot \frac{1}{2} \cdot \frac{1}{1} \cdot \frac{1}{3}$$

Zero Probability

Hasso Plattner Institut

<s> I saw the man in the street </s>

boy	I	in	is	man	saw	street	the	walked	working
1	2	1	1	1	1	1	3	1	1

	boy	I	in	is	man	saw	street	the	walked	working
boy	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	1	0	0	1	0
in	0	0	0	0	0	0	0	1	0	0
is	0	0	0	0	0	0	0	0	0	1
man	0	0	0	1	0	0	0	0	0	0
saw	0	0	0	0	0	0	0	1	0	0
street	0	0	0	0	0	0	0	0	0	0
the	1	0	0	0	1	0	1	0	0	0
walked	0	0	1	0	0	0	0	0	0	0
working	0	0	0	0	0	0	0	0	0	0

$$P(S) = P(I) \cdot P(saw|I) \cdot P(the|saw) \cdot P(man|the) \cdot P(in|man) \cdot P(the|in) \cdot P(street|the)$$

$$P(S) = \frac{\#(I)}{\#(<s>)} \cdot \frac{\#(I\ saw)}{\#(I)} \cdot \frac{\#(saw\ the)}{\#(saw)} \cdot \frac{\#(the\ man)}{\#(the)} \cdot \frac{\#(man\ in)}{\#(man)} \cdot \frac{\#(in\ the)}{\#(in)} \cdot \frac{\#(the\ street)}{\#(the)}$$

$$P(S) = \frac{2}{3} \cdot \frac{1}{2} \cdot \frac{1}{1} \cdot \frac{1}{3} \cdot \frac{0}{1} \cdot \frac{1}{1} \cdot \frac{1}{3}$$

Smoothing

- Giving a small probability to all as unseen *n*-grams
  - Laplace Smoothing
    - Add one to all counts (Add-one)

	boy	I	in	is	man	saw	street	the	walked	working
boy	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	1	0	0	1	0
in	0	0	0	0	0	0	0	1	0	0
is	0	0	0	0	0	0	0	0	0	1
man	0	0	0	1	0	0	0	0	0	0
saw	0	0	0	0	0	0	0	1	0	0
street	0	0	0	0	0	0	0	0	0	0
the	1	0	0	0	1	0	1	0	0	0
walked	0	0	1	0	0	0	0	0	0	0
working	0	0	0	0	0	0	0	0	0	0

Smoothing

- Giving a small probability to all as unseen n-grams
  - Laplace Smoothing
    - Add one to all counts (Add-one)

	boy	I	in	is	man	saw	street	the	walked	working
boy	1	1	1	1	1	1	1	1	1	1
I	1	1	1	1	1	2	1	1	2	1
in	1	1	1	1	1	1	1	2	1	1
is	1	1	1	1	1	1	1	1	1	2
man	1	1	1	2	1	1	1	1	1	1
saw	1	1	1	1	1	1	1	2	1	1
street	1	1	1	1	1	1	1	1	1	1
the	2	1	1	1	2	1	2	1	1	1
walked	1	1	2	1	1	1	1	1	1	1
working	1	1	1	1	1	1	1	1	1	1

$$P(w_i|w_{i-1}) = \frac{\#(w_{i-1},w_i)}{\#(w_{i-1})} \quad \Rightarrow \quad P(w_i|w_{i-1}) = \frac{\#(w_{i-1},w_i)+1}{\#(w_{i-1})+V}$$

Smoothing

- Giving a small probability to all as unseen n-grams
  - Laplace Smoothing
    - Add one to all counts (Add-one)
  - Interpolation and Back-off Smoothing
    - Use a background probability

$$P(w_i|w_{i-1}) = \frac{\#(w_{i-1}, w_i)}{\#(w_{i-1})}$$

Back-off

$$P(w_i|w_{i-1}) = \begin{cases} \frac{\#(w_{i-1}, w_i)}{\#(w_{i-1})} & \text{if } \#(w_{i-1}, w_i) > 0 \\ P_{BG} & \text{otherwise} \end{cases}$$

Smoothing

- Giving a small probability to all as unseen n-grams
  - Laplace Smoothing
    - Add one to all counts (Add-one)
  - Interpolation and Back-off Smoothing
    - Use a background probability

$$P(w_i|w_{i-1}) = \frac{\#(w_{i-1}, w_i)}{\#(w_{i-1})}$$

Interpolation

$$P(w_i|w_{i-1}) = \lambda_1 \frac{\#(w_{i-1}, w_i)}{\#(w_{i-1})} + \lambda_2 \, P_{BG} \qquad \sum \lambda = 1$$

Parameter Tuning

Background Probability

Background Probability

- Lower levels of *n*-gram can be used as background probability
  - trigram $\rightarrow$ bigram
  - bigram $\rightarrow$ unigram
  - unigram $\rightarrow$ zerogram ($\frac{1}{V}$)

Back-off

$$P(w_i|w_{i-1}) = \begin{cases} \frac{\#(w_{i-1},w_i)}{\#(w_{i-1})} & \text{if } \#(w_{i-1},w_i) > 0 \\ P(w_i) & \text{otherwise} \end{cases}$$

$$P(w_i) = \begin{cases} \frac{\#(w_i)}{N} & \text{if } \#(w_i) > 0 \\ \frac{1}{V} & \text{otherwise} \end{cases}$$

Background Probability

- Lower levels of *n*-gram can be used as background probability
  - trigram $\rightarrow$ bigram
  - bigram $\rightarrow$ unigram
  - unigram $\rightarrow$ zerogram ($\frac{1}{V}$)

Interpolation

$$P(w_i|w_{i-1}) = \lambda_1 \frac{\#(w_{i-1}, w_i)}{\#(w_{i-1})} + \lambda_2 P(w_i)$$

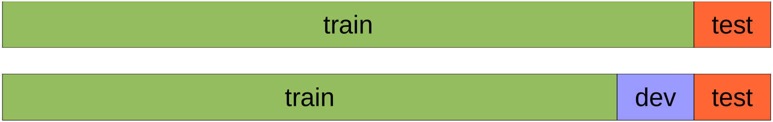
$$P(w_i) = \lambda_1 \frac{\#(w_i)}{N} + \lambda_2 \frac{1}{V}$$

$$P(w_i|w_{i-1}) = \lambda_1 \frac{\#(w_{i-1}, w_i)}{\#(w_{i-1})} + \lambda_2 \frac{\#(w_i)}{N} + \lambda_3 \frac{1}{V}$$

Parameter Tuning

Hasso Plattner Institut

- Dataset



Held-out Set (Development Set)

Using different values for parameters and select the best value which minimize the perplexity of the held-out data.

Advanced Smoothing

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

$$P(w_i|w_{i-1}) = \frac{\#(w_{i-1}, w_i) + k}{\#(w_{i-1}) + kV}$$

$$P(w_i|w_{i-1}) = \frac{\#(w_{i-1}, w_i) + \mu(\frac{1}{V})}{\#(w_{i-1}) + \mu}$$

$$\mu = kV$$

$$P(w_i|w_{i-1}) = \frac{\#(w_{i-1}, w_i) + \mu\ P_{BG}}{\#(w_{i-1}) + \mu}$$

Bayesian Smoothing with Dirichlet Prior

Advanced Smoothing

$$P(w_i|w_{i-1}) = \begin{cases} \frac{\#(w_{i-1},w_i)}{\#(w_{i-1})} & \text{if } \#(w_{i-1},w_i) > 0 \\ P_{BG} & \text{otherwise} \end{cases}$$

$$P(w_i|w_{i-1}) = \begin{cases} \frac{\#(w_{i-1},w_i)-\delta}{\#(w_{i-1})} & \text{if } \#(w_{i-1},w_i) > 0 \\ \alpha P_{BG} & \text{otherwise} \end{cases}$$

Absolute Discounting

Advanced Smoothing

Hasso Plattner Institut

$$P(w_i|w_{i-1}) = \frac{\#(w_{i-1}, w_i) - \delta}{\#(w_{i-1})} + \alpha P_{BG}$$

$$\alpha = \frac{\delta}{\#(w_{i-1})} \cdot B$$

B : the number of times $\#(w_{i-1}, w_i) > 0$

(the number of times that we applied discounting)

Absolute Discounting

$$P(w_i|w_{i-1}) = \frac{\max(\#(w_{i-1}, w_i) - \delta, 0)}{\#(w_{i-1})} + \alpha P_{BG}$$

Advanced Smoothing

- Estimation base on the lower-order *n*-gram

I cannot see without my reading ... $\Rightarrow$ *"Francisco"* *"glasses"*

- Observations:
  - *"Francisco"* is more common than *"glasses"*
  - But *"Francisco"* always follows *"San"*
  - *"Francisco"* is not a novel continuation for a text
- Solution:
  - Instead of $P(w)$: "How likely is w to appear in a text"
  - $P_{continuation}(w)$: "How likely is w to appear as a novel continuation"
    - Count the number of words types that w appears after them

$$P_{continuation}(w) \propto |w_{i-1} : \#(w_{i-1}, w_i) > 0|$$

Advanced Smoothing

- How many times does *w* appear as a novel continuation

$$P_{continuation}(w) \propto |w_{i-1} : \#(w_{i-1}, w_i) > 0|$$

- Normalized by the total number of bigram types

$$P_{continuation}(w) = \frac{|w_{i-1} : \#(w_{i-1}, w_i) > 0|}{|(w_{j-1}, w_j) : \#(w_{j-1}, w_j) > 0|}$$

- Alternatively: normalized by the number of words preceding all words

$$P_{continuation}(w) = \frac{|w_{i-1} : \#(w_{i-1}, w_i) > 0|}{\sum_{w'} |w'_{i-1} : \#(w'_{i-1}, w'_i) > 0|}$$

Advanced Smoothing

$$P(w_i|w_{i-1}) = \frac{\max(\#(w_{i-1}, w_i) - \delta, 0)}{\#(w_{i-1})} + \alpha P_{BG}$$

$$P(w_i|w_{i-1}) = \frac{\max(\#(w_{i-1}, w_i) - \delta, 0)}{\#(w_{i-1})} + \alpha P_{continuation}$$

$$\alpha = \frac{\delta}{\#(w_{i-1})} \cdot B$$

B : the number of times $\#(w_{i-1}, w_i) > 0$

Kneser-Ney Discounting

Further Reading

- Speech and Language Processing
  - Chapter 4