Natural Language Processing
SoSe 2014

Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Language Modelling

Dr. Mariana Neves

April 23rd, 2014

(based on the slides of Dr. Saeedeh Momtazi)

Outline

- Motivation
- Estimation
- Evaluation
- Smoothing

Outline

- Motivation
- Estimation
- Evaluation
- Smoothing

Language Modelling

- Finding the probability of a sentence or a sequence of words
  - $P(S) = P(w_1, w_2, w_3, ..., w_n)$

… all of a sudden I notice three guys standing on the sidewalk ...

on guys all I of notice sidewalk three a sudden standing the

Language Modelling

- Applications
  - Word prediction
  - Speech recognition
  - Handwriting recognition
  - Machine translation
  - Spell checking

Applications

- Word prediction
  - „natural language..“
    - processing
    - management

Applications

- Speech recognition
  - „Computers can recognize speech.“
  - „Computers can wreck a nice peach.”

Applications

- Handwriting recognition
  - „Take the money and run", Woody Allen: „I have a gub."
    instead of „I have a gun."

Applications

- Machine translation
  - „The cat eats...“
    - „Die Katze frisst...“
    - „Die Katze isst...“
  - „He briefed to reporters on the chief contents of the statements“
  - „He briefed reporters on the chief contents of the statements“
  - „He briefed to reporters on the main contents of the statements“
  - „He briefed reporters on the main contents of the statements“

Applications

- Spell checking
  - „I want to adver this project"
    - „adverb" (noun)
    - „advert" (verb)
  - „They are leaving in about fifteen minuets to go to her house."
    - „minutes"
  - „The design an construction of the system will take more than a year."
    - „and"

Outline

- Motivation
- Estimation
- Evaluation
- Smoothing

Language Modeling

- Finding the probability of a sentence or a sequence of words
  - $P(S) = P(w_1, w_2, w_3, ..., w_n)$
  - „Computers can recognize speech."
    - P(Computer, can, recognize, speech)

Conditional Probability

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

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

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

http://setosa.io/conditional/

Conditional Probability

$$P(S)=P(w_1)\cdot P(w_2|w_1)\cdot P(w_3|w_1,w_2)\ldots P(w_n|w_1,w_2,,w_3,\ldots,,w_n)$$

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

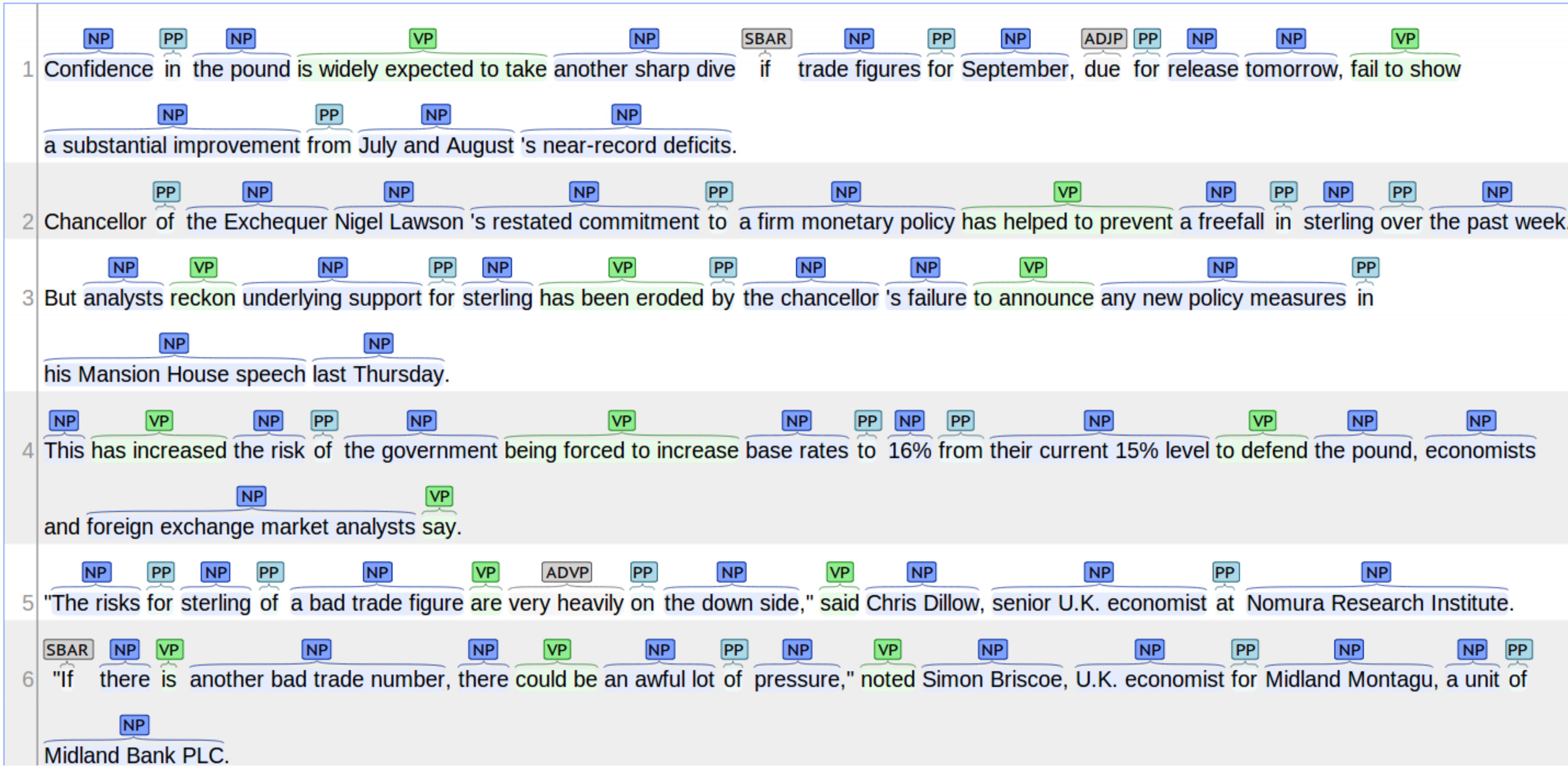
P(Computer,can,recognize,speech) = P(Computer)·
P(can|Computer)·
P(recognize|Computer can)·
P(speech|Computer can recognize)

Hasso Plattner Institut

Corpus

- Probabilities are based on counting things
- Counting of things 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
- Can also be used for evaluation and comparison purposes

Corpus



http://weaver.nlplab.org/~brat/demo/latest/#/not-editable/CoNLL-00-Chunking/train.txt-doc-1

Corpus

- Text Corpora
  - Corpus of Contemporary American English
  - The British National Corpus
  - The International Corpus of English
  - The Google N-gram Corpus (https://books.google.com/ngrams)
  - WBI repository (biomedical domain) (http://corpora.informatik.hu-berlin.de/)

Word occurrence

- A language consists 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

Word occurrence

- Example:
  - „This is a sample text from a book that is read every day.“
    - # Word Tokens: 13
    - # Word Types: 11

Counting

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

- Google N-Gram corpus
  - 1,024,908,267,229 word tokens
  - 13,588,391 word types

- Large English dictionaries have around 500k word types
- Google N-Gram corpus includes
  - Numbers, mispellings, names, acronyms, etc.

Word frequency

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

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,
  - …
- Rank of a word times its frequency is approximately a constant
  - $\text{Rank} \cdot \text{Freq} \approx c$
  - $c \approx 0.1$ for English

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

$$Freq \cdot Rank \approx c$$

Word frequency

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

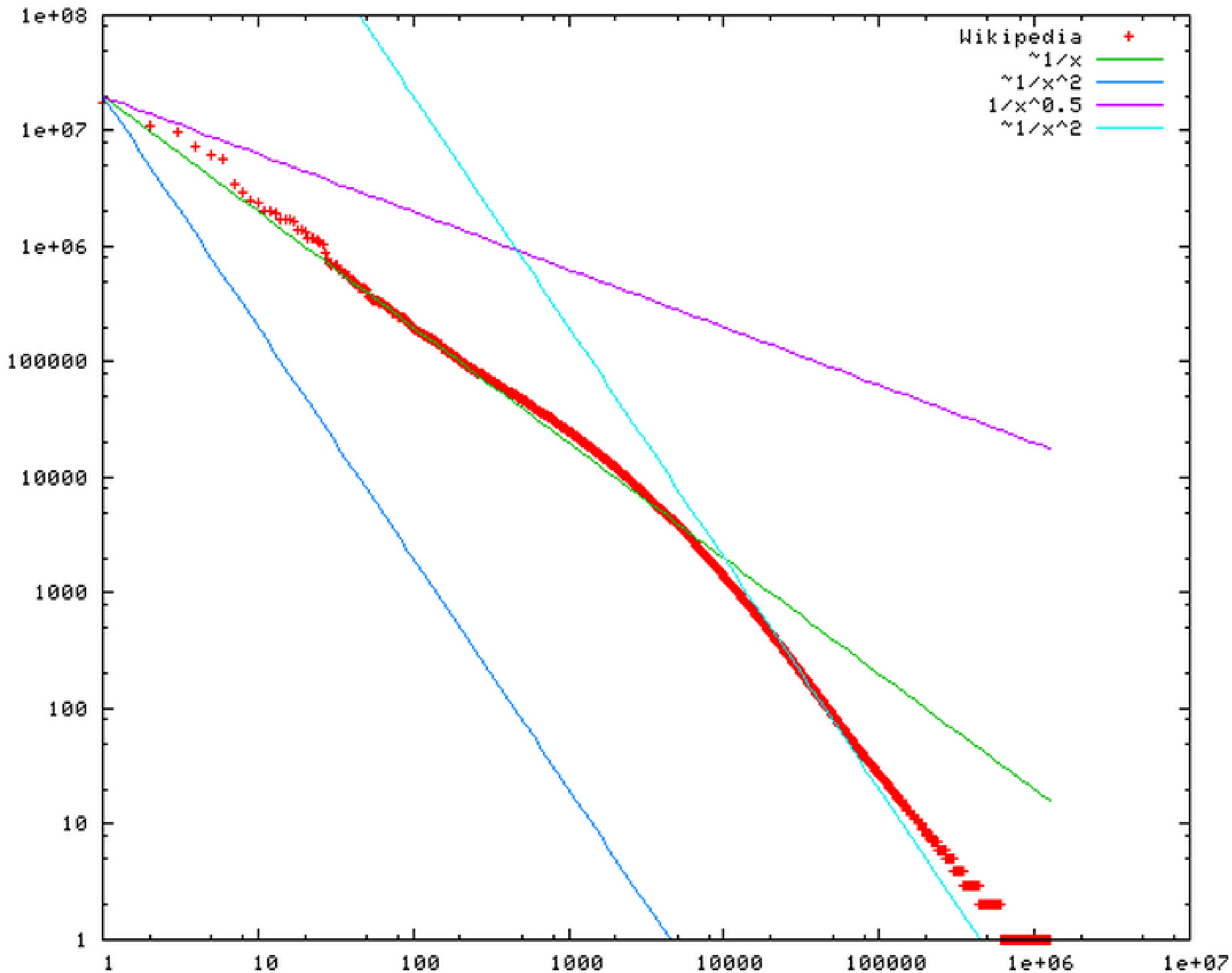
Rank	Word	Count	Freq(%)	Freq x Rank
1	The	69970	6.8872	0.06887
2	of	36410	3.5839	0.07167
3	and	28854	2.8401	0.08520
4	to	26154	2.5744	0.10297
5	a	23363	2.2996	0.11498

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

Zipf's Law



http://en.wikipedia.org/wiki/File:Wikipedia-n-zipf.png

Maximum Likelihood Estimation

- P(speech|Computer can recognize)

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

- Too many phrases
- Limited text for estimating probabilities
- Simplification assumption

Markov assumption

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

$$\Downarrow$$

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

Markov assumption

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

$$\Downarrow$$

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

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

N-gram model

- Unigram $P(S) = {}_{i-1}{}^{n}\prod P(w_i)$
- Bigram $P(S) = {}_{i-1}{}^{n}\prod P(w_i | w_{i-1})$
- Trigram $P(S) = {}_{i-1}{}^{n}\prod P(w_i | w_{i-1}, w_{i-2})$
- N-gram $P(S) = {}_{i-1}{}^{n}\prod P(w_i | w_1, w_2, \ldots, w_{i-1})$

Maximum Likelihood

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

- Vocabulary:
  - V = {I,saw,the,boy,man,is,working,walked,in,street}

  - walked boy working
  - The boy is working
  - street saw the man

Maximum Likelihood

- <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	<s>
1	2	1	1	1	1	1	3	1	1	3

	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
<s>		2						1		

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

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

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

Unkown words

- <s> I saw the woman </s>

- Closed vocabulary: test set can only contain words from this lexicon
- Open vocabulary: test set can contain unknown words
- Out of vocabulary (OOV) words:
  - Choose a vocabulary
  - Convert unknown (OOV) words to <UNK> word token
  - Estimate probabilities for <UNK>

- Alternatively,
  - Replace the first occurrence of every word type by <UNK>

Outline

- Motivation
- Estimation
- Evaluation
- Smoothing

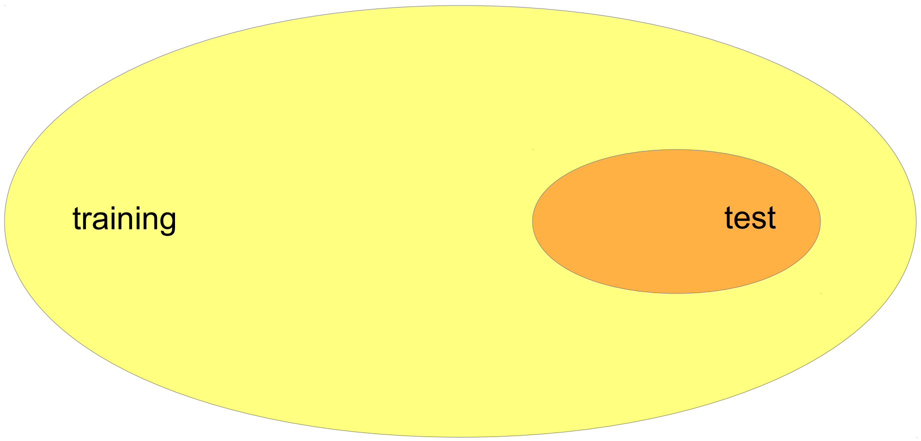
Branching Factor

- 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 …
  - computer, book, apple, banana, umbrella, orange, desk

Perplexity

- Dividing the corpus to two parts
- Building a language model from the training set
  - Word frequencies, etc..
- Estimating the probability of the test set
- Calculate the average branching factor of the test set



Perplexity

- Goal: giving higher probability to frequent texts
  - minimizing the perplexity of the frequent texts

$$P(S) = P(w_1, w_2, \ldots, w_n)$$

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

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

Perplexity

- Maximum branching factor for each language is |V|

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

- Example: predicting next characters instead of next words:
  - |V| = 26, five next characters:

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

Perplexity

- Wall Street Journal (19,979 word vocabulary)
  - Training set: 38 million word tokens
  - Test set: 1.5 million words

- Perplexity:
  - Unigram: 962
  - Bigram: 170
  - Trigram: 109

Outline

- Motivation
- Estimation
- Evaluation
- Smoothing

Maximum Likelihood

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

- <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

- <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

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

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

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

Zero probability

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

- No „man in“ in our corpus

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 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

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

Smoothing

- Giving a small probability to all unseen n-grams
  - 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

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

Smoothing

- Giving a small probability to all as unseen n-grams
  - 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

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

$$\sum\lambda=1$$

Background probability

Parameter tuning

Background probability

- Lower levels of n-gram can be used as background probability
  - Trigram » Bigram
  - Bigram » Unigram
  - Unigram » Zerogram $(\frac{1}{V})$

Back-off

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

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

Background probability

- Lower levels of n-gram can be used as background probability
  - Trigram » Bigram
  - Bigram » Unigram
  - Unigram » Zerogram $\left(\frac{1}{V}\right)$

Interpolation

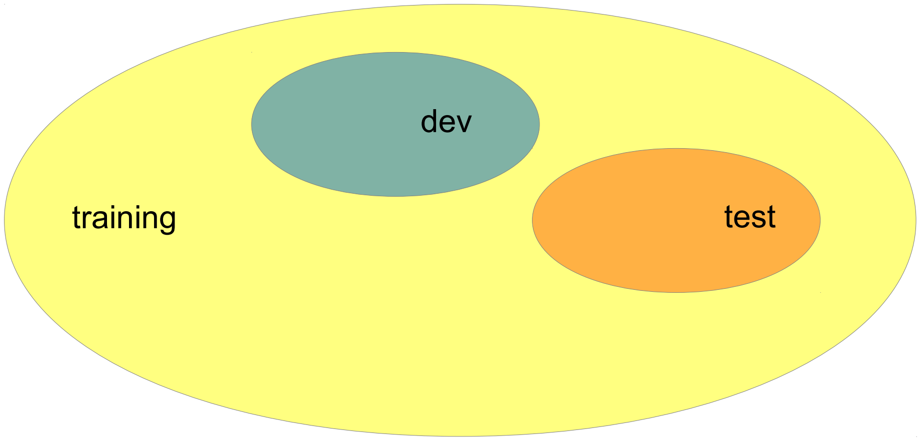
$$P(w_i|w_{i-1}) = \lambda_1 \cdot \frac{\#(w_{i-1}, w_i)}{\#(w_{i-1})} + \lambda_2 \cdot P(w_i)$$

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

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

Parameter tuning

- Held-out dataset (development set)
- 80% (training), 10% (dev-set), 10% (test)
- Minimize the perplexity of the held-out dataset



Advanced smoothing

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

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

(add-k, add-δ smoothing)

Advanced smoothing

- Absolute discounting
  - Good estimates for high counts, discount won't affect them much
  - Lower counts are anyway not trustworthy

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

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

Advanced smoothing

- Estimation based on the lower-order n-gram
  - I cannot see without my reading … („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

- Kneser-Ney discounting

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

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

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

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

Class-based n-grams

- Compute estimation for the bigram „to Shanghai“
- Training data: „to London“, „to Beijing“, „to Denver“
- Classes: CITY_NAME, AIRLINE, DAY_OF_WEEK, MONTH, etc.

$$P(w_i|w_{i-1}) \approx P(c_i|c_{i-1}) \times P(w_i|c_{i-1})$$

Exercise

- Language modelling
- Corpus
- MLE + Perplexity + Laplace (add-one) smoothing

Further reading

- Chapter 4
- http://www.cs.columbia.edu/~mcollins/lm-spring2013.pdf