Natural Language Processing

SoSe 2017

Language Model

Dr. Mariana Neves

May 15th, 2017

Language model

- 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 model

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

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

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

Motivation: Speech recognition

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

- „Give peace a chance.“
- „Give peas a chance.“

- Ambiguity in speech:
  - „Friday“ vs. „fry day“
  - „ice cream“ vs. „I scream“

(http://worldsgreatestsmile.com/html/phonological_ambiguity.html)

Motivation: Handwriting recognition

- „Take the money and run“, Woody Allen:
  - „Abt naturally.“ vs. „Act naturally.“
  - „I have a gub.” vs. „I have a gun.“

(https://www.youtube.com/watch?v=I674fBVanAA)

Motivation: Machine Translation

- „The cat eats...“
  - „Die Katze frisst...“
  - „Die Katze isst...“

- Chinese to English:
  - „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“

Motivation: 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“

Language model

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

- „Computers can recognize speech.“
  - P(Computer, can, recognize, speech)

Conditional Probability

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

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

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

Conditional Probability

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

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

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

Corpus

- Probabilities are based on counting things
- A corpus is a computer-readable collection of text or speech
  - Corpus of Contemporary American English
  - The British National Corpus
  - The International Corpus of English
  - The Google N-gram Corpus (https://books.google.com/ngrams)

Word occurrence

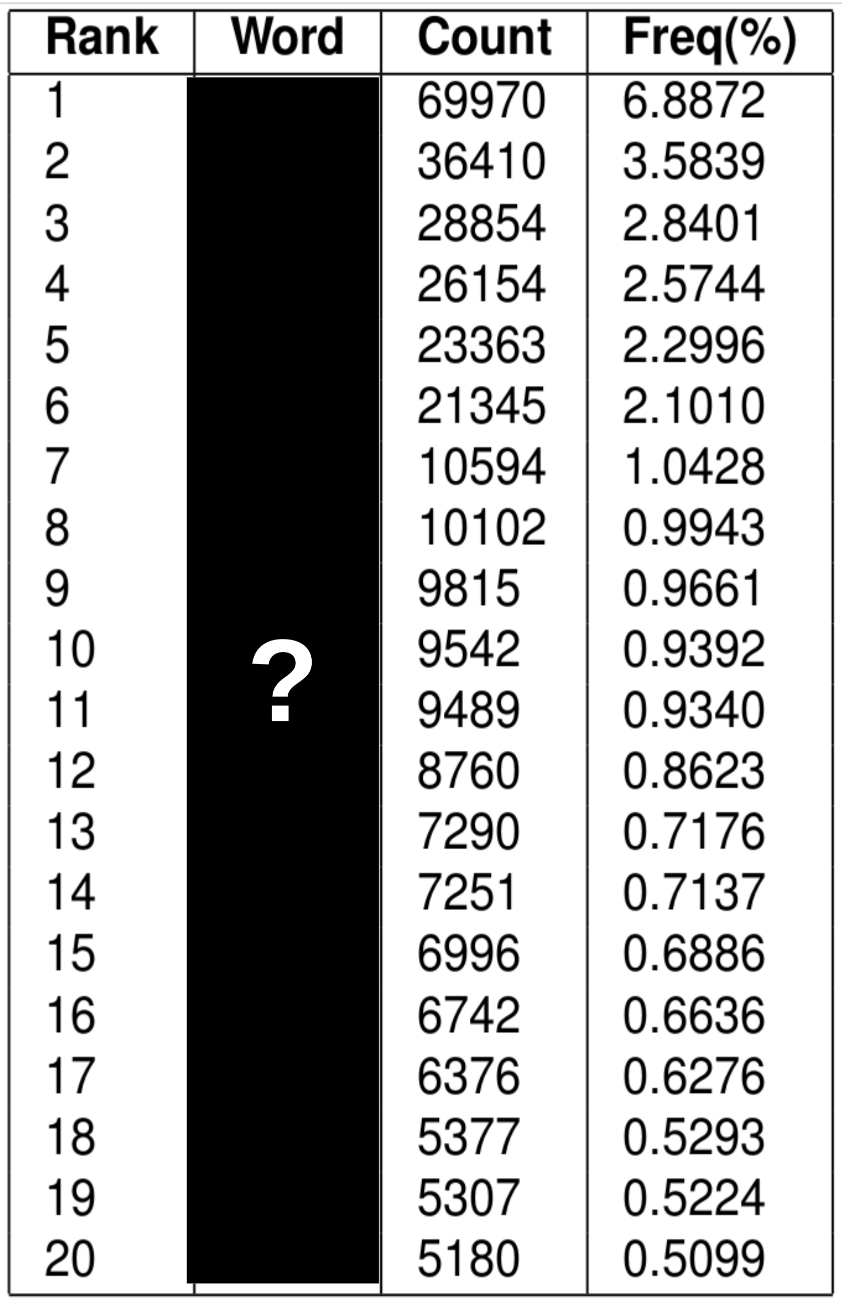
- A language consists of a set of „V“ words (Vocabulary)
- A word can occur several times in a text
  - Word Token: each occurrence of words in text
  - Word Type: each unique occurrence of words in the text

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

Word occurrence

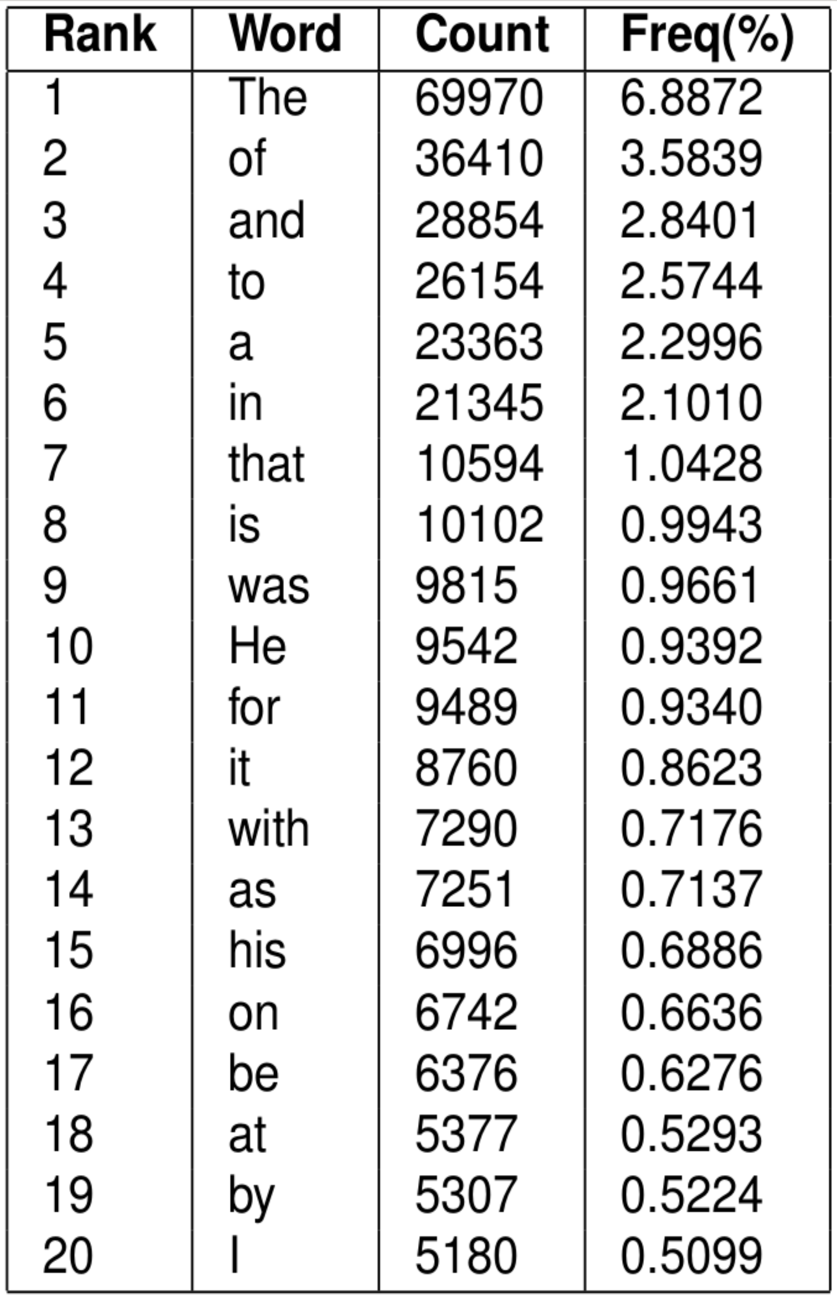
- Google N-Gram corpus
  - 1,024,908,267,229 word tokens
  - 13,588,391 word types
- Why so many word types?
  - Large English dictionaries have around 500k word types

Word frequency



Rank	Word	Count	Freq(%)
1	[redacted]	69970	6.8872
2	[redacted]	36410	3.5839
3	[redacted]	28854	2.8401
4	[redacted]	26154	2.5744
5	[redacted]	23363	2.2996
6	[redacted]	21345	2.1010
7	[redacted]	10594	1.0428
8	[redacted]	10102	0.9943
9	[redacted]	9815	0.9661
10	[redacted] ?	9542	0.9392
11	[redacted]	9489	0.9340
12	[redacted]	8760	0.8623
13	[redacted]	7290	0.7176
14	[redacted]	7251	0.7137
15	[redacted]	6996	0.6886
16	[redacted]	6742	0.6636
17	[redacted]	6376	0.6276
18	[redacted]	5377	0.5293
19	[redacted]	5307	0.5224
20	[redacted]	5180	0.5099

Word frequency

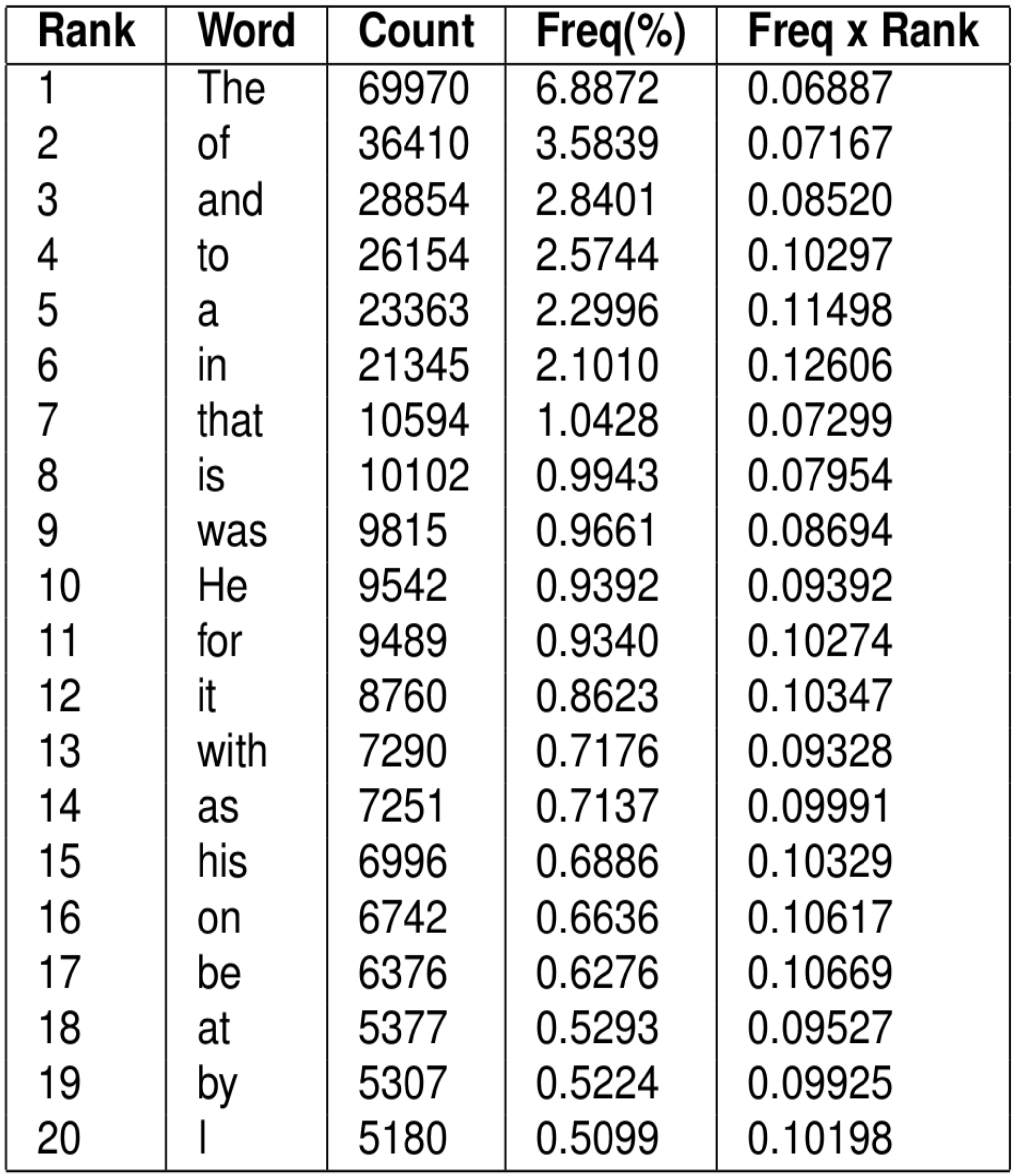


Rank	Word	Count	Freq(%)
1	The	69970	6.8872
2	of	36410	3.5839
3	and	28854	2.8401
4	to	26154	2.5744
5	a	23363	2.2996
6	in	21345	2.1010
7	that	10594	1.0428
8	is	10102	0.9943
9	was	9815	0.9661
10	He	9542	0.9392
11	for	9489	0.9340
12	it	8760	0.8623
13	with	7290	0.7176
14	as	7251	0.7137
15	his	6996	0.6886
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

Zipf's Law

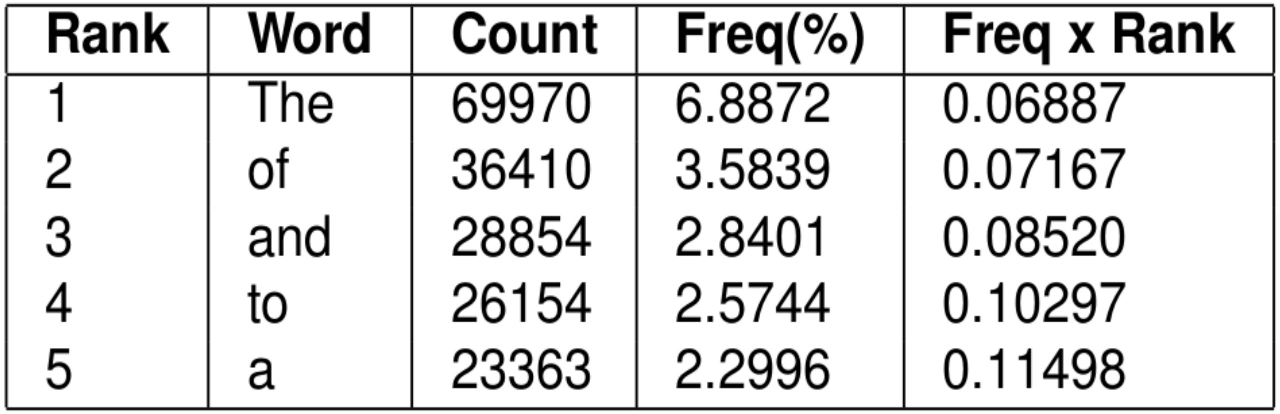


Rank	Word	Count	Freq(%)	Freq x Rank
1	The	69970	6.8872	0.06887
2	of	36410	3.5839	0.07167
3	and	28854	2.8401	0.08520
4	to	26154	2.5744	0.10297
5	a	23363	2.2996	0.11498
6	in	21345	2.1010	0.12606
7	that	10594	1.0428	0.07299
8	is	10102	0.9943	0.07954
9	was	9815	0.9661	0.08694
10	He	9542	0.9392	0.09392
11	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

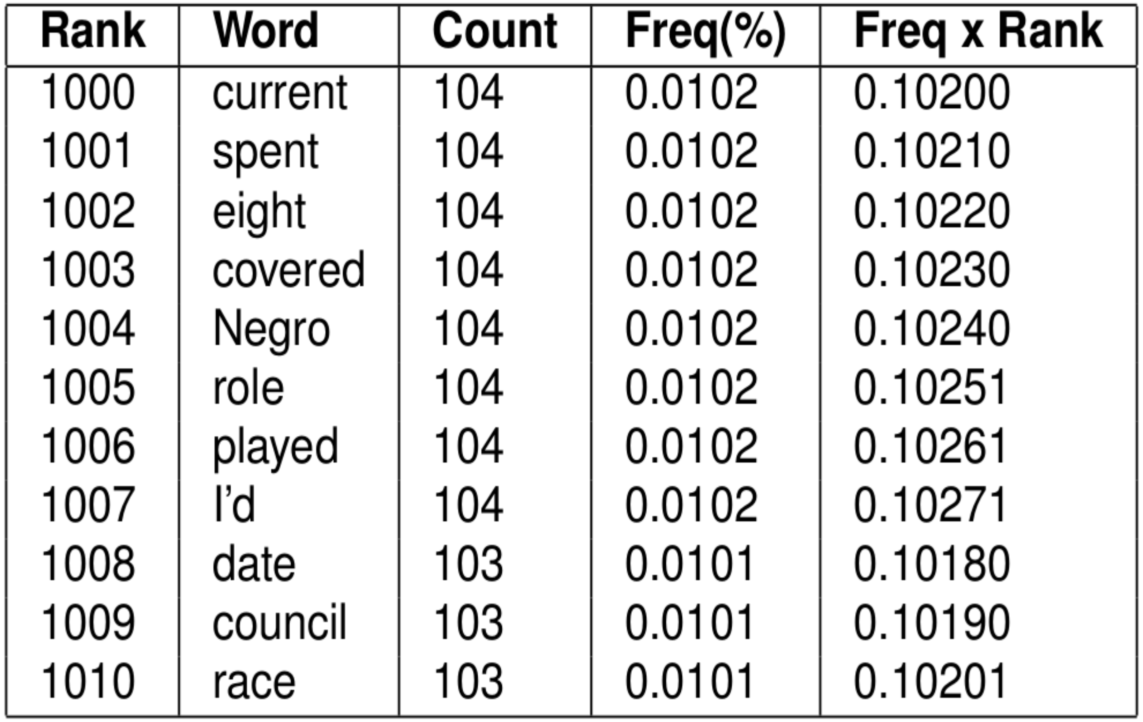
- Zipf’s Law is not very accurate for very frequent and very infrequent words



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

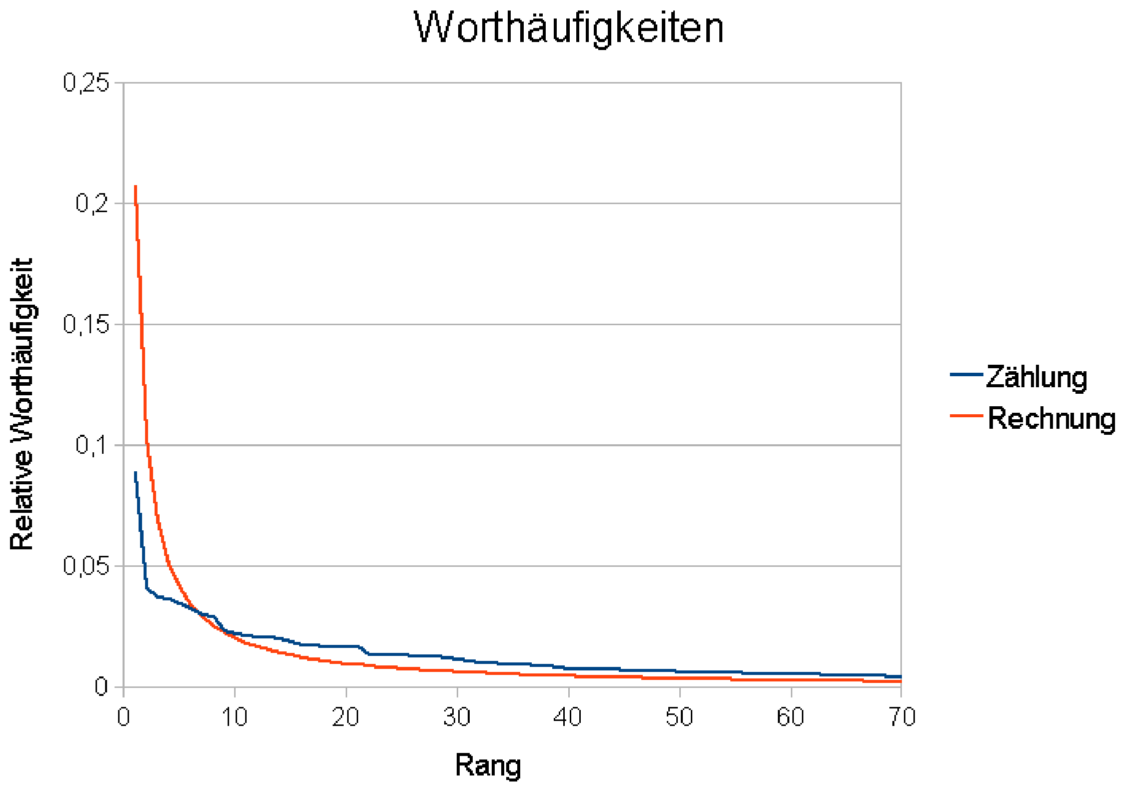
Zipf's Law

- But very precise for intermediate ranks



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 for German



(https://de.wikipedia.org/wiki/Zipfsches_Gesetz#/media/File:Zipf-Verteilungn.png)

Back to Conditional Probability

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

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

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

Maximum Likelihood Estimation

- P(speech|Computer can recognize)

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

- What is the problem of this approach?

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 probabilities
- Simplification assumption → Markov 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(Computer,can,recognize,speech) = P(Computer)·
P(can|Computer)·
P(recognize|Computer can)·
P(speech|Computer can recognize)

⬇

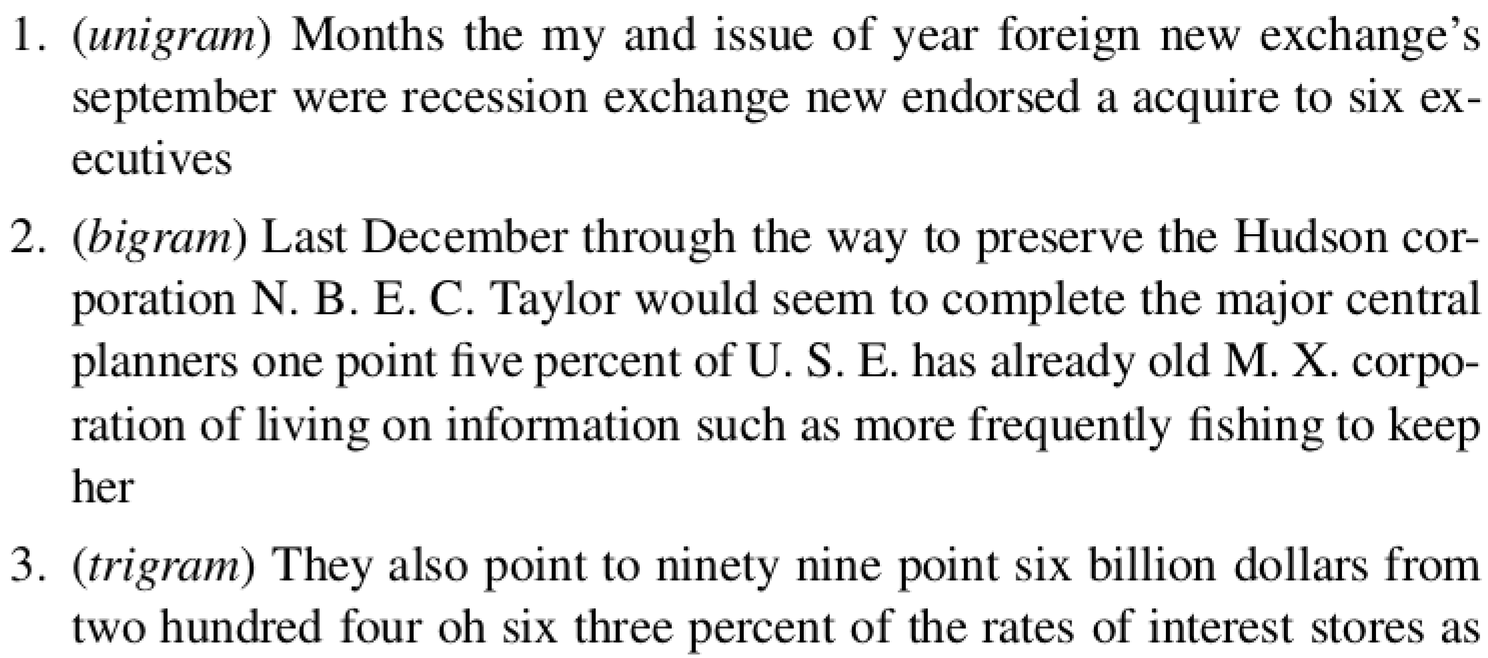
P(Computer,can,recognize,speech) = P(Computer)·
P(can|Computer)·
P(recognize|can)·
P(speech|recognize)

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

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: $P(S)=\prod_{i-1}^{n} P(w_i|w_{i-1},w_{i-2})$
- N-gram: $P(S)=\prod_{i-1}^{n} P(w_i|w_1,w_2,\dots,w_{i-1})$

N-gram model



1. (*unigram*) Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
2. (*bigram*) Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her
3. (*trigram*) They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as

Maximum Likelihood Estimation

- <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}

Maximum Likelihood Estimation

- <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 Estimation

- Estimation of maximum likelihood for a new sentence
  - <s> I saw the man </s>

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

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

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

Unknown words

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

- Possible Solutions?

Unknown words

- Possible Solutions:
  - 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>
  - Replace the first occurrence of every word type in the training data by <UNK>

Evaluation

- Divide the corpus to two parts: training and test
- Build a language model from the training set
- Estimate the probability of the test set
- Calculate the average branching factor of the test set

Branching factor

- 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, i.e., give a higher probability to the words that occur in real texts

Perplexity

- Goals: give higher probability to frequent texts
  - minimize the perplexity of the frequent texts

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

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

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

Example of perplexity

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

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

Unknown n-grams

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

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

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

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

Smoothing – Laplace (Add-one)

- Small probability to all unseen n-grams
    - Add one to all counts

	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 – Back-off

- Use a background probability (P_{BG})
  - For instance „Scottish beer drinkers“ but no „Scottish beer eaters“
  - Back-off to bigram „beer drinker“ and „beer eaters“

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

Backgroung probability

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

Background probability – Back-off

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

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

Smoothing – Interpolation

- Higher and lower order n-gram models have different strengths and weaknesses
  - high-order n-grams are sensitive to more context, but have sparse counts
  - low-order n-grams consider only very limited context, but have robust counts

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

Background probability – Interpolation

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

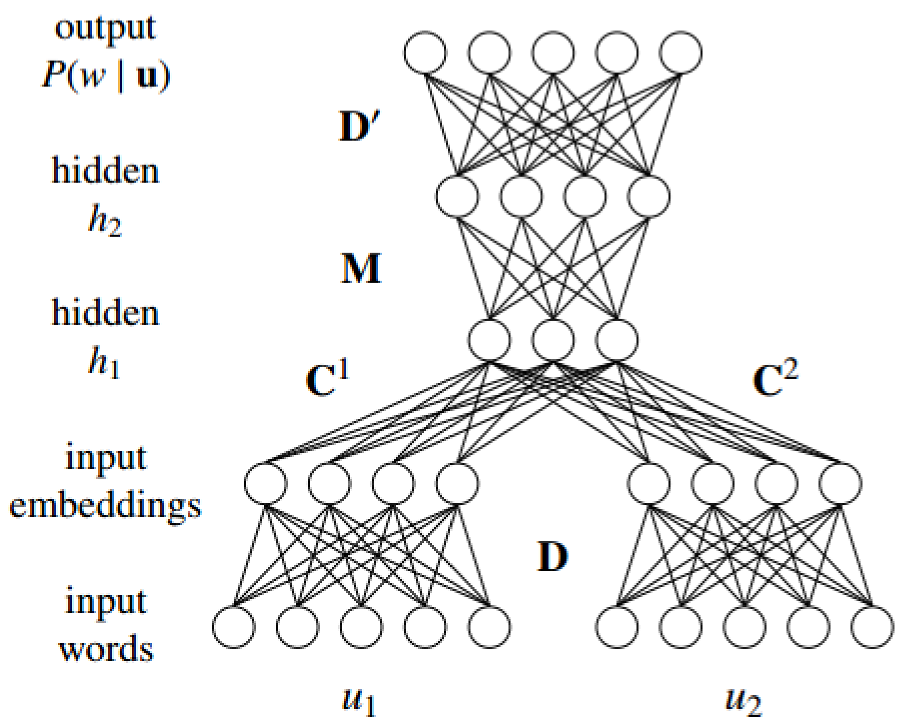
$$P(w_i) = \lambda_1 \cdot \frac{\#(w_i)}{N} + \lambda_2 \cdot \frac{1}{V}$$

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

Parameter Tuning

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

N-gram Neural LM with feed-forward NN



Input as **one-hot representations** of the words in context u (n-1),

where n is the order of the language model

(https://www3.nd.edu/~dchiang/papers/vaswani-emnlp13.pdf)

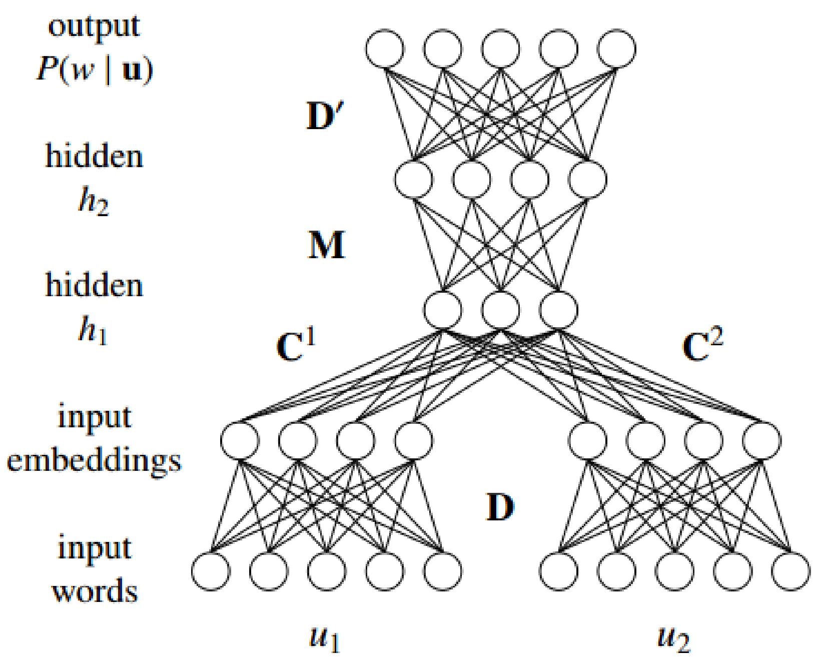
One-hot representation

- Corpus: „the man runs.“
- Vocabulary = {man,runs,the,.}
- Input/output for p(runs|the man)

$$x_0 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \qquad x_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad y_{true} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

N-gram Neural LM with feed-forward NN

- Input: context of n-1 previous words
- Output: probability distribution for next word
- Size of input/output: vocabulary size
- One or many hidden layers
- Embedding layer is lower dimensional and dense
  - Smaller weight matrices
  - Learns to map similar words to similar points in the vector space



(https://www3.nd.edu/~dchiang/papers/vaswani-emnlp13.pdf)

Summary

- Words
  - Tokenization, Segmentation
- Language Model
  - Word occurrence (word type and word token)
  - Zipf's Law
  - Maximum Likelihood Estimation
    - Markov assumption and n-grams
  - Evaluation: Perplexity
  - Smoothing methods
  - Neural networks

Further reading

- Book Jurafski & Martin
  - Chapter 4
  - https://web.stanford.edu/~jurafsky/slp3/4.pdf
- „Classical“ neural network language model (Bengio et al 2003):
  - http://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf