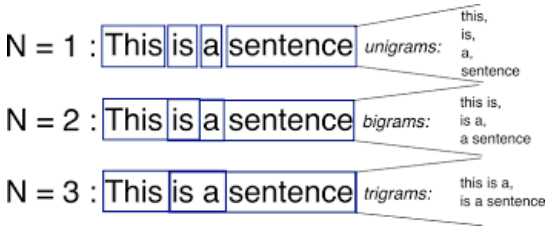
Mineração de Textos



Aula 03:

Language Models

Prof. Rinaldo Lima
rinaldo.ufrpe@gmail.com

UFRPE Universidade Federal Rural de Pernambuco
DEINFO Departamento de Estatística e Informática

12-abr-18

Table of Contents

Counting Corpora

UFRPE Universidade Federal Rural de Pernambuco | DEINFO Departamento de Estatística e Informática

12-abr-18

Corpora in Computational Linguistics

- increasing focus on language use and empirical evidence in recent years
- based on **corpora** = (usually large) machine-readable samples of naturally ocurring language
- some applications of corpus data
  - test hypotheses about formal system of language
  - validation of linguists' introspective judgements
  - observable result of human language production
  - model for linguistic experience of human speaker
  - training data for statistical NLP applications
- corpus = sample ➜ need for **statistical analysis**
  - standard methodologies are being established
  - random sample assumption is controversial for most corpora ➜ statistical inference may be unreliable
  - ☞ ongoing research into appropriate statistical models

Statistical Inference from Corpus Data

- only observable data are **corpus frequencies**
- commonly used terminology: **types** vs. **tokens**
  - categorization into fixed or open-ended set of **types**: distinct word forms or lemmas, parts of speech, etc.
  - of central interest are **type frequencies** $f(\omega)$
  - corpus is interpreted as a **random sample** of tokens
    ➜ inferences about type probabilities π_ω from $f(\omega)$

5

Corpus Terminology

- **Word Form**: the inflected form of the word that appears in the corpus
- **Lemma:** an abstract form, shared by word forms having the same stem, part of speech, and word sense
- **# Types:** number of distinct words in a corpus (**vocabulary size**)
- **# Tokens**: total number of words

6

Counting: Types and Tokens

- *He stepped out into the hall, was delighted to encounter a water brother.*
  - 13 tokens, 15 if we include "," and "." as separate tokens.
- *They picnicked by the pool, then lay back on the grass and looked at the stars.*
  - 18 tokens (again counting punctuation)
  - only 16 types

7

Word Counts in Corpora

- ***Word Counts to find out***:
  - What are the most common words in the text.
  - How many words are in the text (word tokens and word types).
  - What the average frequency of each word in the text is.
- ***Limitation of word counts:*** Most words appear very infrequently and it is hard to predict much about the behavior of words that do not occur often in a corpus. ==> Zipf's Law.

8

Zipf´s Law

UFRPE Universidade Federal Rural de Pernambuco | DEINFO Departamento de Estatística e Informática

12-abr-18

Zipf´s Law

For a word type in large corpus the

- frequency of the word is the number of times it occurs,
- rank of the word is its position in a list words ordered from highest to lowest frequency.

Zipf's law says that for each word

$$\text{frequency} \times \text{rank} = \text{constant}.$$

The constant is fixed throughout a given corpus, but depends on the corpus.

Zipf´s Law

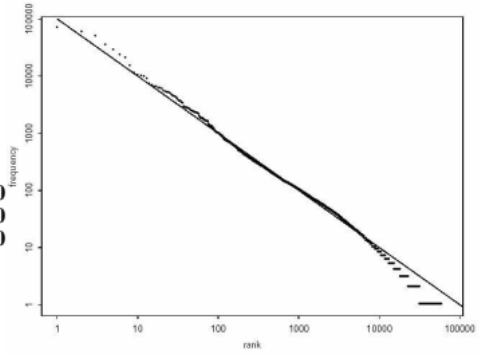
- If we count up how often each word type of a language occurs in a large corpus and then list the words in order of their frequency of occurrence, we can explore the relationship between the frequency of a word, ***f***, and its position in the list, known as its rank, ***r***.
- Zipf's Law says that: $f \propto 1/r$
- Significance of Zipf's Law: For most words, our data about their use will be exceedingly sparse. Only for a few words will we have a lot of examples.

11

Zipf's Law – Empirical Evaluation (English)

- **Tom Sawyer Book**

Word	Freq.	Rank	f * r
the	3332	1	3332
and	2972	2	5944
a	1775	3	5235
he	877	10	8770
but	410	20	8400
be	294	30	8820
there	222	40	8880
one	172	50	8600
about	158	60	9480
more	138	70	9660
never	124	80	9920
Oh	116	90	10440
two	104	100	10400
turned	51	200	10200
you'll	30	300	9000
name	21	400	8400
comes	16	500	8000
group	13	600	7800
lead	11	700	7700
friends	10	800	8000
begin	9	900	8100
family	8	1000	8000

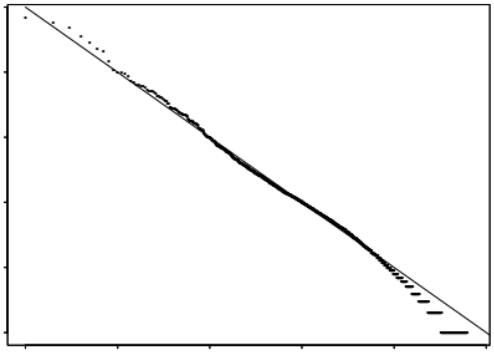


12

Zipf's Law – Empirical Evaluation

- **Brown Corpus**

The obligatory Zipf's law slide:
Zipf's law for the Brown corpus

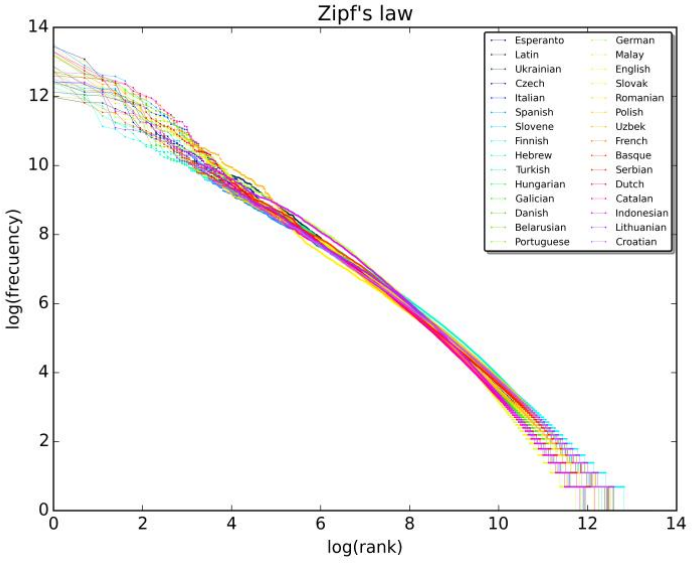


$$f \propto \frac{1}{r} \quad \text{or, there is a } k \text{ such that } \quad f \cdot r = k$$

13

Zipf´s Law – Other Languages

A plot of the frequency vs. rank for the first 10 million words in 30 Wikipedias (dumps from October 2015) in a log-log scale.



14

Zipf´s Law – Other Languages

- The principle of least effort is a possible explanation:
- Zipf himself proposed that
  - neither speakers nor hearers using a given language want to work any harder than necessary to reach understanding, and
  - the process that results in approximately equal distribution of effort leads to the observed Zipf distribution

George Kingsley Zipf
Linguist, Harvard University

15

N-Gram Language Model

UFRPE Universidade Federal Rural de Pernambuco | DEINFO Departamento de Estatística e Informática

12-abr-18

Probabilistic Language Models

- Goal: compute the probability of a sentence or sequence of words:

  $P(W) = P(w_1,w_2,w_3,w_4,w_5...w_n)$

- Related task: probability of an upcoming word:

  $P(w_5|w_1,w_2,w_3,w_4)$

- A model that computes either of these:

  $P(W)$ or $P(w_n|w_1,w_2...w_{n-1})$ is called a **language model**.

- Better: **the grammar** But **language model** or **LM** is standard

17

How to Compute P(W)

- How to compute this joint probability:
  - P(its, water, is, so, transparent, that)
- Intuition: let's rely on the Chain Rule of Probability

18

The Chain Rule of Probability

Reminder: The Chain Rule

Recall the definition of conditional probabilities:

$$P(A|B) = \frac{P(A,B)}{P(B)} \quad \text{Rewriting} \Rightarrow \quad P(A|B) \times P(B) = P(A,B)$$

or $P(A,B) = P(A|B) \times P(B)$

More variables:

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

The Chain Rule in general is:

$$P(w_1,w_2,w_3,\ldots,w_n) = P(w_1) \times P(w_2|w_1) \times P(w_3|w_1,w_2) \times \ldots \times P(w_n|w_1,\ldots,w_{n-1})$$

19

The Chain Rule Applied to Language

$$P(w_1^n) = P(w_1)P(w_2|w_1)P(w_3|w_1^2)\ldots P(w_n|w_1^{n-1})$$
$$= \prod_{k=1}^{n} P(w_k|w_1^{k-1})$$

Example:

P(its water was so transparent)=

P(its) *

P(water|its) *

P(was|its water) *

P(so|its water was) *

P(transparent|its water was so)

20

Estimating Probabilities from Corpora

How to Estimate Probabilities

Given a large corpus of English (that represents the language), should we just divide all words and count all probabilities?

$$P(\text{the} \mid \text{its water is so transparent that}) = \frac{Count(\text{its water is so transparent that the})}{Count(\text{its water is so transparent that})}$$

We'll never have enough data (the counts of all possible sentences) for estimating these.

21

Independence Assumption

- **Make the simplifying assumption**
  - P(lizard|the,other,day,I,was,walking,along,and,saw,a) = P(lizard|a)
- **Or maybe**
  - P(lizard|the,other,day,I,was,walking,along,and,saw,a) = P(lizard|saw, a)
- This kind of **independence assumption** is called a *Markov assumption* after the Russian mathematician **Andrei Markov.**

22

Markov Models

UFRPE Universidade Federal Rural de Pernambuco | DEINFO Departamento de Estatística e Informática

12-abr-18

N-Gram Model

- N-gram language model assumes each word depends only on the last n-1 words (Markov assumption)

Andrei Markov

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i \mid w_{i-k} \ldots w_{i-1})$$

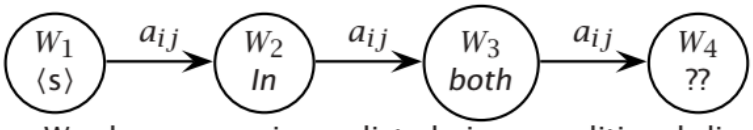
- In other words, we approximate each component in the product

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-k} \ldots w_{i-1})$$

24

Simple linear models of language

- Markov models a.k.a. n-gram models:



  - Word sequence is predicted via a conditional distribution
  - Conditional Probability Table (CPT): e.g., $P(X|both)$
    - $P(of|both) = 0.066$
    - $P(to|both) = 0.041$
    - $P(in|both) = 0.038$
  - Amazingly successful as a simple engineering model

20

25

The Simplest Case: Unigram Model

Simplest case: Unigram model

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i)$$

- The maximum likelihood estimate of the **unigram probability** of the word w_i is its count c_i normalized by the total number of word tokens N:

$$P(w_i) = \frac{c_i}{N}$$

$$P(w_k|w_1, \ldots, w_{k-1}) \approx P(w_k)$$

$$\Rightarrow P(w_1, w_2, \ldots, w_T) \approx P(w_1)P(w_2)\ldots P(w_T)$$

26

Bigram Model

- Condition on the previous word:

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-1})$$

(5) $P(over|The, quick, brown, fox, jumped) \approx P(over|jumped)$

The probability of a sentence:

(6) $P(w_1, ..., w_n) = P(w_1)P(w_2|w_1)P(w_3|w_2)...P(w_n|w_{n-1})$

27

Trigram Model

- Condition on the two previous words

Trigrams ($n = 3$) encode more context

- Wider context: $P(know|did, he)$ vs. $P(know|he)$
- Generally, trigrams are still short enough that we will have enough data to gather accurate probabilities

❖ Example: trigram (3-gram)

$P(w_n \mid w_1, \ldots w_{n-1}) = P(w_n \mid w_{n-2}, w_{n-1})$

$P(w_1, \ldots w_n) =$

$P(w_1)P(w_2|w_1) \ldots P(w_n \mid w_{n-2}, w_{n-1})$

$P("Today\ is\ a\ sunny\ day")$

=P("Today")P("is"|"Today")P("a"|"is", "Today")…

P("day"|"sunny", "a")

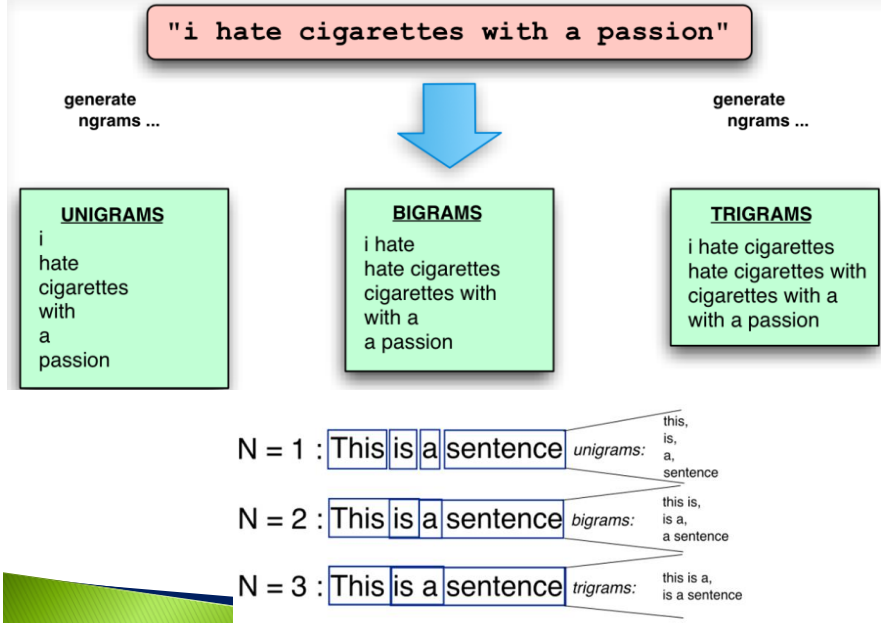
28

N-Gram Models

- We can extend to trigrams, 4-grams, 5-grams
- In general this is an insufficient model of language
  - because language has **long-distance dependencies**:

  "The computer which I had just put into the machine room on the fifth floor crashed."
- But we can often get away with N-gram models

N-Gram models: Example



Reviewing: N-Gram Models

- Unigram model: $P(w_1)P(w_2)P(w_3) \ldots P(w_n)$
- Bigram model:
  $P(w_1)P(w_2|w_1)P(w_3|w_2) \ldots P(w_n|w_{n-1})$
- Trigram model:
  $P(w_1)P(w_2|w_1)P(w_3|w_2,w_1) \ldots P(w_n|w_{n-1}w_{n-2})$
- N-gram model:
  $P(w_1)P(w_2|w_1) \ldots P(w_n|w_{n-1}w_{n-2} \ldots w_{n-N})$

31

Building N-Gram LM

- **Use existing sentences to compute n-gram probability estimates (training)**

- **Terminology:**
  - N = total number of words in training data (tokens)
  - V = vocabulary size or number of unique words (types)
  - $C(w_1,\ldots,w_k)$ = frequency of n-gram $w_1, \ldots, w_k$ in training data
  - $P(w_1, \ldots, w_k)$ = probability estimate for n-gram $w_1 \ldots w_k$
  - $P(w_k|w_1, \ldots, w_{k-1})$ = conditional probability of producing w_k given the history $w_1, \ldots w_{k-1}$

32

Estimating Unigram Probabilities

Assume a language has ***V* word types** in its lexicon

how likely is word "x" to follow word "y"

- In other words:
  - estimate likelihood of x occurring in new text based on its general frequency of occurrence estimated from a corpus (unigram probability)

33

Building N-Gram Models

- Start with what's easiest!
- Compute maximum likelihood estimates for individual n-gram probabilities
  - Unigram: $P(w_i) = \frac{C(w_i)}{N}$
  - Bigram: $P(w_i, w_j) = \frac{C(w_i, w_j)}{N}$

Why not just substitute $P(w_i)$?

$$P(w_j|w_i) = \frac{P(w_i, w_j)}{P(w_i)} = \frac{C(w_i, w_j)}{\sum_w C(w_i, w)} = \frac{C(w_i, w_j)}{C(w_i)}$$

- Uses relative frequencies as estimates
- Maximizes the likelihood of the data given the model P(D|M)

34

Estimating Bigram Probabilities

- The Maximum Likelihood Estimate

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

35

Estimating Bigram Probabilities: Example

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

$P(\texttt{I} \mid \texttt{<s>}) = \frac{2}{3} = .67$ $P(\texttt{Sam} \mid \texttt{<s>}) = \frac{1}{3} = .33$ $P(\texttt{am} \mid \texttt{I}) = \frac{2}{3} = .67$
$P(\texttt{</s>} \mid \texttt{Sam}) = \frac{1}{2} = 0.5$ $P(\texttt{Sam} \mid \texttt{am}) = \frac{1}{2} = .5$ $P(\texttt{do} \mid \texttt{I}) = \frac{1}{3} = .33$

36

Estimating Bigram Probabilities: Berkley Restaurant Corpus

- First step: calculating the bigram frequencies

- Out of 9222 sentences

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Raw bigram counts

Eg. "I want" occurred 827 times

37

Estimating Bigram Probabilities: Example

- 2nd Step: Calcuting the probabilities

Raw bigram probabilities → **Divide bigram counts by prefix unigram counts to get probabilities.**

- Normalize by unigrams:

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

- Result:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

38

Estimating Bigram Probabilities: Example

- **Given**

$$P(\texttt{i}\,|\,\texttt{<s>}) = 0.25 \qquad P(\texttt{english}\,|\,\texttt{want}) = 0.0011$$
$$P(\texttt{food}\,|\,\texttt{english}) = 0.5 \qquad P(\texttt{</s>}\,|\,\texttt{food}) = 0.68$$

- **Ex.:** Calcuting the probability of a whole sentence S using bigrams estimates

$$\begin{aligned} &P(\texttt{<s> i want english food </s>}) \\ &\quad = P(\texttt{i}\,|\,\texttt{<s>})P(\texttt{want}\,|\,\texttt{i})P(\texttt{english}\,|\,\texttt{want}) \\ &\qquad\qquad P(\texttt{food}\,|\,\texttt{english})P(\texttt{</s>}\,|\,\texttt{food}) \\ &\quad = .25 \times .33 \times .0011 \times 0.5 \times 0.68 \\ &\quad = = .000031 \end{aligned}$$

Kinds of Knowledge

- As crude as they are, *N*-gram probabilities capture a range of interesting facts about language.

- **P(english|want) = .0011**
- **P(chinese|want) = .0065**
- **P(to|want) = .66**
- **P(eat | to) = .28**
- **P(food | to) = .003**
- **P(want | spend) = 0**
- **P (i | <s>) = .25**

World knowledge

Syntax

Discourse

Data Sparsity – Zipf Law

P(I | <s>) = 2/3 = 0.67
P(am | I) = 2/3 = 0.67
P(</s> | Sam)= 1/2 = 0.50
...

P(Sam | <s>) = 1/3 = 0.33
P(do | I) = 1/3 = 0.33
P(Sam | am) = 1/2 = 0.50

Bigram Probability Estimates

P(I like ham)

= P(I | <s>) P(like | I) P(ham | like) P(</s> | ham)

= 0

Why?
Why is this bad?

Data Sparsity

- Serious problem in language modeling!
- Becomes more severe as N increases
  - What's the tradeoff?
- Solution 1: Use larger training corpora
  - Can't always work... Blame Zipf's Law (Looong tail)
- Solution 2: Assign non-zero probability to unseen n-grams
  - Known as smoothing

Smoothing Technique

- The simplest way to do smoothing is to add 1 (+1) to all the bigram counts, before we normalize them into probabilities
- This algorithm is called **Laplace smoothing**
- It is also a practical smoothing algorithm for other tasks like **text classification**

43

Laplace Smoothing (Add-one): Unigrams

- Recall that the unsmoothed maximum likelihood estimate of the **unigram probability** of the word ***wi*** is its count ***ci*** normalized by the total number of word tokens *N*:

$$P(w_i) = \frac{c_i}{N}$$

Laplace smoothing merely adds one to each count (hence its alternate name **add one** smoothing).

Since there are *V* words in the vocabulary and each one was incremented, we also need to adjust the denominator to take into account the extra *V*

$$P_{\text{Laplace}}(w_i) = \frac{c_i + 1}{N + V}$$

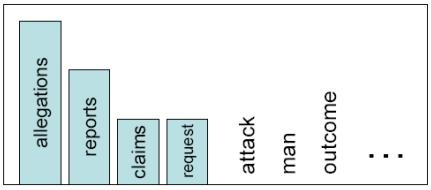
44

Smoothing Technique

Smoothing is like Robin Hood: Steal from the rich and give to the poor (in probability mass)

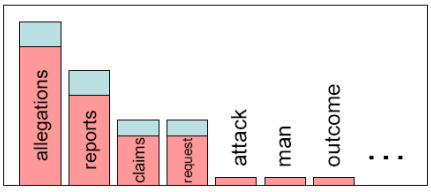
- We often want to make predictions from sparse statistics:

P(w | denied the)
3 allegations
2 reports
1 claims
1 request
7 total



- Smoothing flattens spiky distributions so they generalize better

P(w | denied the)
2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other
7 total



- Very important all over NLP, but easy to do badly!

45

Laplace Smoothing (Add-one): Bigrams

Recall that normal bigram probabilities are computed by normalizing each row of counts by the unigram count:

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

For add-one smoothed bigram counts, we need to augment the unigram count by the number of total word types in the vocabulary V

$$P^*_{\text{Laplace}}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

46

Laplace Smoothing (Add-one) : Bigrams

- Out of 9222 sentences

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Add-one smoothing

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

.7

Laplace Smoothing (Add-one) : Bigrams

	i	**want**	**to**	**eat**	**chinese**	**food**	**lunch**	**spend**
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Figure 4.2 Bigram probabilities for eight words in the Berkeley Restaurant Project corpus of 9332 sentences. Zero probabilities are in gray.

	i	**want**	**to**	**eat**	**chinese**	**food**	**lunch**	**spend**
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

Figure 4.6 Add-one smoothed bigram probabilities for eight of the words (out of $V = 1446$) in the BeRP corpus of 9332 sentences. Previously-zero probabilities are in gray.

Smoothing Example

So, if *treasure trove* never occurred in the data, but *treasure* occurred twice, we have:

(11) $P^*(trove|treasure) = \frac{0+1}{2+V}$

The probability won't be very high, but it will be better than 0

- If the surrounding probabilities are high, *treasure trove* could be the best pick
- If the probability were zero, there would be no chance of appearing

49

A Last Problem: Zero Probabilities

Recall that

$$\Rightarrow P(w_1, w_2, \ldots, w_T) \approx P(w_1)P(w_2)\ldots P(w_T)$$

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i \mid w_{i-k} \ldots w_{i-1})$$

- We do everything in log space
  - Avoid underflow
  - (also adding is faster than multiplying)

$$\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

50

N-Grams – Other Granularities

- You can use different granularities of "tokens" in the N-Gram model
  - Character based n-grams
    - Language Identification
  - Word-based n-grams
    - Spam classification
  - POS-based n-grams
    - Information Extraction

51

Language Identification: Character level N-grams

A COMPARATIVE ASSESSMENT OF LANGUAGE IDENTIFICATION APPROACHES IN TEXTUAL DOCUMENTS

Luciano de Souza Cabral[1,2], Rafael Dueire Lins[1], Rinaldo Lima[1] and Steven J. Simske[3]
[1] *Federal University of Pernambuco, Recife, Brazil*
[2] *Federal Institute of Pernambuco, Caruaru, Brazil*
[3] *Hewlett-Packard Labs., Fort Collins, CO 80528, USA*

ABSTRACT

This paper presents several experiments conducted for assessing distinct methods for language identification of written texts. After introducing a new method for the language identification problem, we conducted some standard experiments aiming at evaluating the proposed approaches against three other ones. In order to perform fair comparisons, we used the same corpus (EuroParl Corpus), which contains 21,000 sentences evenly distributed in 21 languages. We discuss the experimental results as well as the strengths and limitations of the compared algorithms. In addition, the accuracy results achieved by the proposed method introduced in this research work showed that it is very competitive with other state of the art methods.

52

Applications of N-Gram

- **Word Prediction**: being able to predict the next word (or any linguistic unit) in a sequence is very useful

- It lies at the core of the following applications
  - Automatic speech recognition
  - Handwriting and character recognition
  - **Spelling correction**
  - Machine translation
  - **Augmentative communication**
  - Word similarity, generation, POS tagging, etc.
  - **Language Identification**
  - **Spam Detection and Classificaction Tasks**
  - Author Identification (stylometry)
  - Criptoanalysis (code breaking)

N-Gram Corpora On the Web

More data available

Scaling up

because more data are better data for statistical NLP (Church and Mercer 1993)

- 1964: 1 million words (Brown Corpus)
- 1995: 100 million words (British National Corpus)
- 2003: 1,000+ million words (English Gigaword, WaCky)
- 2006: 1,000,000 million words (Google Web 1T 5-Grams)

The Google Web 1T 5-Gram Database

Available Resources

There are many available Language models that you can try

Google N-Gram Release, August 2006

AUG 3

All Our N-gram are Belong to You

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

Here at Google Research we have been using word n-gram models for a variety of R&D projects,

…

That's why we decided to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.

The Google Web 1T 5-Gram Database

The Google Web 1T 5-Gram database

Brants and Franz (2006)

- Not the full 1 trillion words of English Web text, but . . .
- Frequency counts for bigrams, trigrams, 4-grams and 5-grams extracted from this corpus
  - thresholds: $f \geq 200$ for terms, $f \geq 40$ for n-grams
- Multiple compressed text files with total size of 24.4 GiB
- No linguistic pre-processing (case-folding, lemmatization, POS tagging, parsing, word sense disambiguation, . . .)
- Little boilerplate cleanup ("*from collectibles to cars*")

57

Google N-Gram Web

More resources

- Google n-gram:

https://research.googleblog.com/2006/08/all-our-n-gram-are-belong-to-you.html

```
File sizes: approx. 24 GB compressed (gzip'ed) text files

Number of tokens: 1,024,908,267,229
Number of sentences: 95,119,665,584
Number of unigrams: 13,588,391
Number of bigrams: 314,843,401
Number of trigrams: 977,069,902
Number of fourgrams: 1,313,818,354
Number of fivegrams: 1,176,470,663
```

58

The Google Web 1T 5-Gram Database

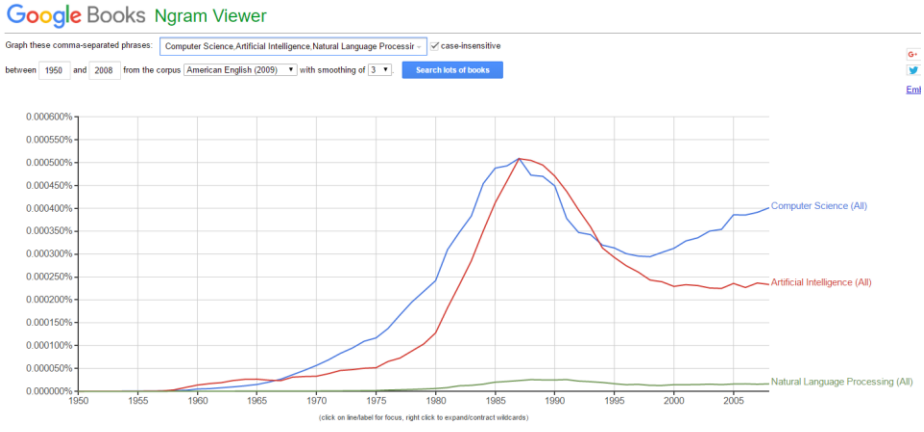
The Google Web 1T 5-Gram database

Brants and Franz (2006)

word 1	word 2	word 3	f
supplement	depend	on	193
supplement	depending	on	174
supplement	depends	entirely	94
supplement	depends	on	338
supplement	derived	from	2668
supplement	des	coups	77
supplement	described	in	200

excerpt from file 3gm-0088.gz

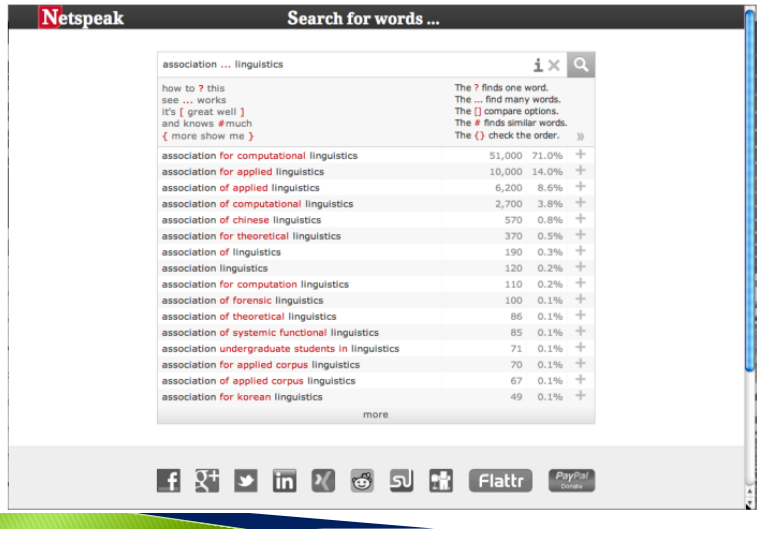
Google N-Gram Viewer: books



N-Gram on the Web

Application example: Netspeak

http://www.netspeak.org/



61

N-GRAM Corpora and Statistics for Downloading

Google Books - Ngram Viewer

- http://storage.googleapis.com/books/ngrams/books/datasetsv2.html
- https://aws.amazon.com/datasets/google-books-ngrams/

google-ngram-downloader 4.0.1 in Python

- https://pypi.python.org/pypi/google-ngram-downloader/

Corpus of Contemporaty American English

- http://www.ngrams.info/download_coca.asp

Counting N-Gram in Portuguese (Tutorial)

- http://www.nltk.org/howto/portuguese_en.html

62

Langage Model Softwares

Language modeling software

Tools and Software for English - Language modeling

For languages other than English, see List of resources by language.

Contents [hide]
1 Language modeling software
1.1 Free software
1.2 Non-Free software
2 Models

Language modeling software

Free software

- IRSTLM - Free software for language modeling
- KenLM - Fast, Free software for language modeling
- MITLM - MIT Language Modeling (MITLM) toolkit
- OpenGrm NGram library - Free software for language modeling. Built on OpenFst.
- Positional Language Model
- RandLM - Free software for randomised language modeling
- VariKN - Free software for creating, growing and pruning Kneser-Ney smoothed n-gram models.

Non-Free software

- cmuclmtk - CMU-Cambridge Statistical Language Modeling toolkit. Older versions are available here.
- SRILM - The SRI Language Modeling Toolkit. Nitin Madnani has created Perl and Python wrappers, which are available from his homepage.
- WOPR - Memory-based word prediction and language modeling - uses TiMBL. GPL.

https://www.aclweb.org/aclwiki/index.php?title=Language_modeling_software

Language Modeling Toolkits

AntConc - Toolkit for Concordancy and Text analysis

- http://www.laurenceanthony.net/software.html

KENML

- https://www.sri.com/engage/products-solutions/sri-language-modeling-toolkit
- http://www.kheafield.com/code/kenlm/
- https://github.com/kpu/kenlm

SRILIM

- http://www.speech.sri.com/projects/srilm/