CS 4650/7650: Natural Language Processing

Language Modeling

Diyi Yang

Some slides borrowed from Yulia Tsvetkov at CMU and Kai-Wei Chang at UCLA

Logistics

- HW 1 Due
- HW 2 Out: Feb 3rd, 2020, 3:00pm

Piazza & Office Hours

- ~ 11 mins response time

Review

- L2: Text classification
- L3: Neural network for text classification

This Lecture

- Language Models
  - What are N-gram models
- How to use probabilities

This Lecture

- What is the probability of "*I like Georgia Tech at Atlanta*"?
- What is the probability of "like I Atlanta at Georgia Tech"?

Language Models Play the Role of …

- A judge of grammaticality
- A judge of semantic plausibility
- An enforcer of stylistic consistency
- A repository of knowledge (?)

The Language Modeling Problem

- Assign a probability to every sentence (or any string of words)
  - Finite vocabulary (e.g., words or characters) {*the, a, telescope, …*}
  - Infinite set of sequences
    - *A telescope STOP*
    - *A STOP*
    - *The the the STOP*
    - *I saw a woman with a telescope STOP*
    - *STOP*
    - …

Example

- *P(disseminating so much currency STOP)* = 10^{-15}
- *P(spending so much currency STOP)* = 10^{-9}

What Is A Language Model?

- Probability distributions over sentences (i.e., word sequences)

  P(W) = P($w_1 w_2 w_3 w_4 \dots w_k$)

- Can use them to generate strings

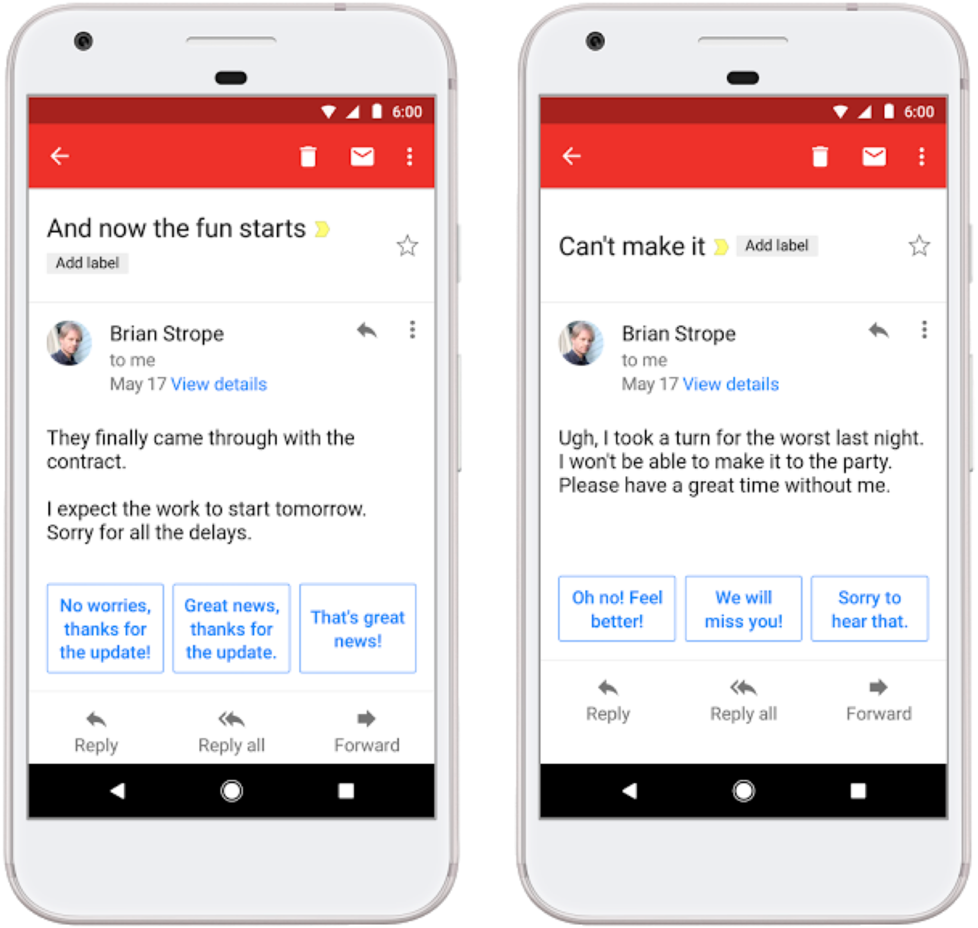
  P(w_k | $w_2 w_3 w_4 \dots w_{k-1}$)

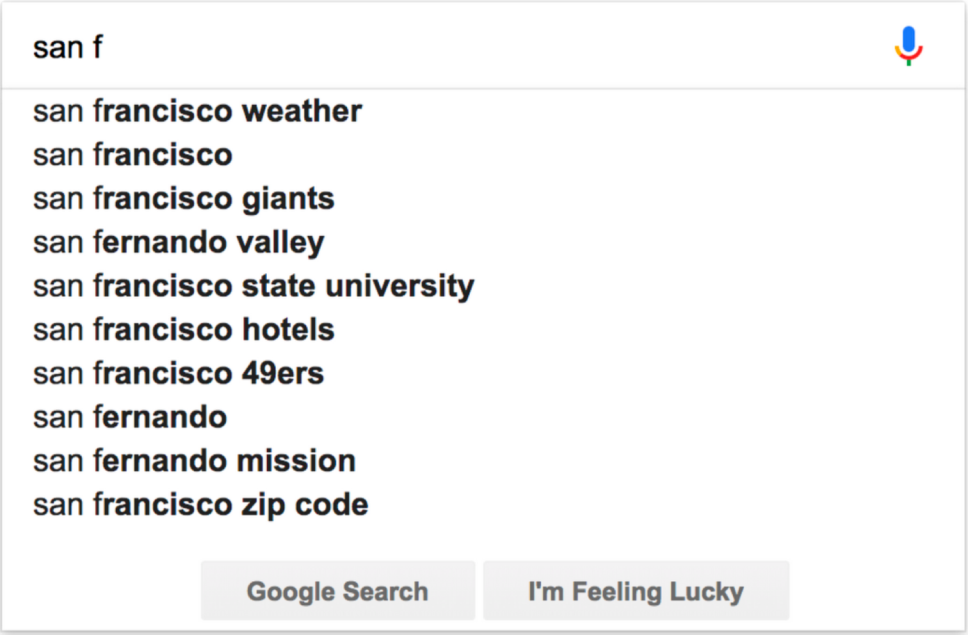
- Rank possible sentences
  - P("*Today is Tuesday*") > P("*Tuesday Today is*")
  - P("*Today is Tuesday*") > P("*Today is Atlanta*")

Language Model Applications

- Machine Translation
  - *p(strong winds) > p(large winds)*
- Spell Correction
  - *The office is about 15 <u>minutes</u> from my house*
  - *p(15 <u>minute</u>s from my house) > p(15 <u>minuets</u> from my house)*
- Speech Recognition
  - *p(I saw a van) >> p(eyes awe of an)*
- Summarization, question-answering, handwriting recognition, etc..

Language Model Applications





Language Model Applications

Rooter: A Methodology for the Typical Unification of Access Points and Redundancy

Jeremy Stribling, Daniel Aguayo and Maxwell Krohn

Abstract

Many physicists would agree that, had it not been for congestion control, the evaluation of web browsers might never have occurred. In fact, few hackers worldwide would disagree with the essential unification of voice-over-IP and public-private key pair. In order to solve this riddle, we confirm that SMPs can be made stochastic, cacheable, and interposable.

I. Introduction

Many scholars would agree that, had it not been for active networks, the simulation of Lamport clocks might never have occurred. The notion that end-users synchronize with the investigation of Markov models is rarely outdated. A theoretical grand challenge in theory is the important unification of virtual machines and real-time theory. To what extent can web browsers be constructed to achieve this purpose?

Certainly, the usual methods for the emulation of Smalltalk that paved the way for the investigation of rasterization do not apply in this area. In the opinions of many, despite the fact that conventional wisdom states that this grand challenge is continuously answered by the study of access points, we believe that a different solution is necessary. It should be noted that Rooter runs in $\Omega(\log \log n)$ time. Certainly, the shortcoming of this type of solution, however, is that compilers and superpages are mostly incompatible. Despite the fact that similar methodologies visualize XML, we surmount this issue without synthesizing distributed archetypes.

The rest of this paper is organized as follows. For starters, we motivate the need for fiber-optic cables. We place our work in context with the prior work in this area. To address this obstacle, we disprove that even though the much-tauted autonomous algorithm for the construction of digital-to-analog converters by Jones [10] is NP-complete, object-oriented languages can be made signed, decentralized, and signed. Along these same lines, to accomplish this mission, we concentrate our efforts on showing that the famous ubiquitous algorithm for the exploration of robots by Sato et al. runs in $\Omega((n + \log n))$ time [22]. In the end, we conclude.

II. Architecture

Our research is principled. Consider the early methodology by Martin and Smith; our model is similar, but will actually overcome this grand challenge. Despite the fact that such a claim at first glance seems unexpected, it is buffetted by previous work in the field. Any significant development of secure theory will clearly require that the acclaimed real-time algorithm for the refinement of write-ahead logging by Edward Feigenbaum et al. [15] is impossible; our application is no different. This may or may not actually hold in reality. We consider an application consisting of n access points. Next, the model for our heuristic consists of four independent components: simulated annealing, active networks, flexible modalities, and the study of reinforcement learning.

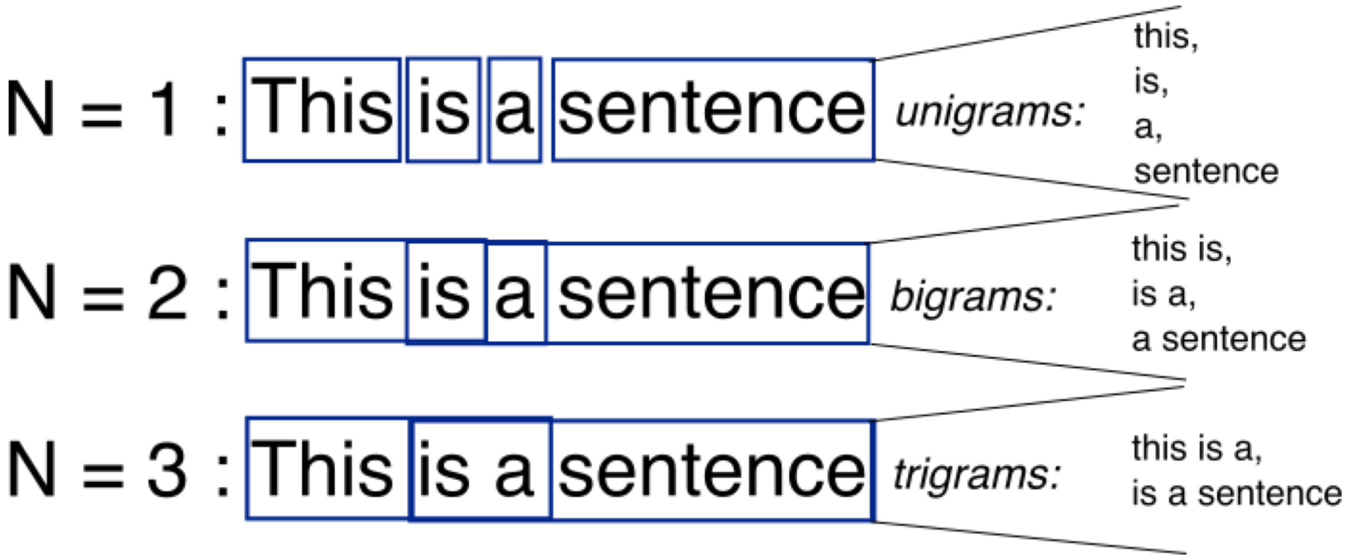
We consider an algorithm consisting of n semaphores. Any unproven synthesis of introspective methodologies will

Language generation

https://pdos.csail.mit.edu/archive/scigen/

Bag-of-Words with N-grams

- **N-grams: a contiguous sequence of n tokens from a given piece of text**



http://recognize-speech.com/language-model/n-gram-model/comparison

N-grams Models

- Unigram model: $P(w_1)P(w_2)P(w_3) \dots P(w_n)$
- Bigram model: $P(w_1)P(w_2|w_1)P(w_3|w_2) \dots P(w_n|w_{n-1})$
- Trigram model:

$$P(w_1)P(w_2|w_1)P(w_3|w_2, w_1) \dots P(w_n|w_{n-1}w_{n-2})$$

- N-gram model:

$$P(w_1)P(w_2|w_1)\dots\ P(w_n|w_{n-1}w_{n-2} \dots w_{n-N})$$

The Language Modeling Problem

- Assign a probability to every sentence (or any string of words)
  - Finite vocabulary (e.g., words or characters)
  - Infinite set of sequences

$$\sum_{e \in \Sigma^*} p_{LM}(e) = 1$$

$$p_{LM}(e) \geq 0, \forall\, e \in \Sigma^*$$

A Trivial Model

- Assume we have n training sentences
- Let $x_1, x_2, \dots, x_n$ be a sentence, and $\mathrm{c}(x_1, x_2, \dots, x_n)$ be the number of times it appeared in the training data.
- Define a language model $p(x_1, x_2, \dots x_n) = \frac{c(x_1, x_2, \dots, x_n)}{N}$
- No generalization!

Markov Processes

- Markov Processes:
  - Given a sequence of n random variables
  - We want a sequence probability model
  - $X_1, X_2, \dots, X_n, (e.g., n = 100), X_i \in V$
  - $p(X_1 = x_1, X_2 = x_2, \dots X_n = x_n)$

Markov Processes

- Markov Processes:
  - Given a sequence of n random variables
  - We want a sequence probability model
  - $X_1, X_2, \dots, X_n, \; X_i \in V$
  - $p(X_1 = x_1, X_2 = x_2, \dots X_n = x_n)$
- There are $|V|^n$ possible sequences

First-order Markov Processes

- Chain Rule:
  - $p(X_1 = x_1, X_2 = x_2, \ldots X_n = x_n)$

$$= P(X_1 = x_1) \prod_{i=2}^{n} p(X_i = x_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1})$$

First-order Markov Processes

- Chain Rule:
  - $p(X_1 = x_1, X_2 = x_2, \dots X_n = x_n)$

$$= P(X_1 = x_1) \prod_{i=2}^{n} p(X_i = x_i | X_1 = x_1, \dots, X_{i-1} = x_{i-1})$$

$$= P(X_1 = x_1) \prod_{i=2}^{n} p(X_i = x_i | X_{i-1} = x_{i-1})$$ Markov Assumption

First-order Markov Processes

- Chain Rule:
  - $p(X_1 = x_1, X_2 = x_2, \ldots X_n = x_n)$

$$= P(X_1 = x_1) \prod_{i=2}^{n} p(X_i = x_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1})$$

$$\boxed{= P(X_1 = x_1) \prod_{i=2}^{n} p(X_i = x_i | X_{i-1} = x_{i-1})}$$

Markov Assumption

$$P(X_i = x_i | X_1 = x_1 \ldots X_{i-1} = x_{i-1}) = P(X_i = x_i | X_{i-1} = x_{i-1})$$

First-order Markov Processes

$$P(X_i = x_i | X_1 = x_1 \ldots X_{i-1} = x_{i-1}) = P(X_i = x_i | X_{i-1} = x_{i-1})$$

Second-order Markov Processes

- $p(X_1 = x_1, X_2 = x_2, \dots X_n = x_n)$

  $= p(X_1 = x_1) \times p(X_2 = x_2 | X_1 = x_1) \prod_{i=3}^{n} p(X_i = x_i | X_{i-2} = x_{i-2}, X_{i-1} = x_{i-1})$

- Simplify notation: $x_0 = x_{-1} = *$

Details: Variable Length

- We want probability distribution over sequences of any length

Details: Variable Length

- Define always $x_n = STOP$, where STOP is a special symbol
- Then use a Markov process as before:

$$p(X_1 = x_1, X_2 = x_2, \dots, X_n = x_n) = \prod_{i=1}^{n} p(X_i = x_i | X_{i-2} = x_{i-2}, X_{i-1} = x_{i-1})$$

- We now have probability distribution over all sequences
  - Intuition: at every step you have probability α_n to stop and $1 - \alpha_n$ to keep going

The Process of Generating Sentences

Step 1: Initialize $i = 1$ and $x_0 = x_{-1} = *$

Step 2: Generate x_i from the distribution

$$p(X_i = x_i | X_{i-2} = x_{i-2}, X_{i-1} = x_{i-1})$$

Step 3: If $\mathrm{x_i} = STOP$ then return the sequence $x_1 \cdots x_i$. Otherwise, set $i = i + 1$ and return to step 2.

3-gram LMs

- A trigram language model contains
  - A vocabulary V
  - A non negative parameter $q(w|u,v)$ for every trigram, such that

$$w \in V \cup \{\mathrm{STOP}\}, u, v \in V \cup \{*\}$$

  - The probability of a sentence $x_1, x_2, \ldots, x_n$, where $x_n = \mathrm{STOP}$ is

$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} q(x_i | x_{i-1}, x_{i-2})$$

3-gram LMs

- A trigram language model contains
  - A vocabulary V
  - A non negative parameter $q(w|u, v)$ for every trigram, such that

$$w \in V \cup \{\text{STOP}\}, u, v \in V \cup \{*\}$$

  - The probability of a sentence $x_1, x_2, \dots, x_n$, where $x_n = \text{STOP}$ is

$$p(x_1, \dots, x_n) = \prod_{i=1}^{n} \boxed{q(x_i|x_{i-1}, x_{i-2})}$$

3-gram LMs: Example

$$p(\text{the dog barks STOP}) \quad = \; q(\text{the} \mid *, *) \times$$

3-gram LMs: Example

$$
\begin{aligned}
p(\text{the dog barks STOP}) \quad &= \; q(\text{the}|*,*) \times \\
&= \; q(\text{dog}|*,\text{the}) \times \\
&= \; q(\text{barks}|\text{the},\text{dog}) \times \\
&= \; q(\text{STOP}|\text{dog},\text{barks})
\end{aligned}
$$

Limitations

- Markovian assumption is false

He is from France, so it makes sense that his first language is …

- We want to model longer dependencies

N-gram model

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

Trigram	• Sweet prince, Falstaff shall die. Harry of Monmouth's grave. • This shall forbid it should be branded, if renown made it empty. • Indeed the duke; and had a very good friend. • Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.
Quadrigram	• King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in; • Will you not tell me who I am? • It cannot be but so. • Indeed the short and the long. Marry, 'tis a noble Lepidus.

More Examples

- Yoav's blog post:
  - http://nbviewer.jupyter.org/gist/yoavg/d76121dfde2618422139
- 10-gram character-level LM

```
First Citizen: Nay, then, that was hers, It speaks against your other service:
But since the youth of the circumstance be spoken: Your uncle and one Baptista's
daughter.

SEBASTIAN: Do I stand till the break off.
```

Maximum Likelihood Estimation

- “Best” means “data likelihood reaches maximum”

$$\hat{\boldsymbol{\theta}} = \mathbf{argmax}_{\boldsymbol{\theta}} \mathbf{P}(\mathbf{X}|\boldsymbol{\theta})$$

Maximum Likelihood Estimation

Unigram Language Model θ

$p(w|\theta)$=?

Estimation

Document

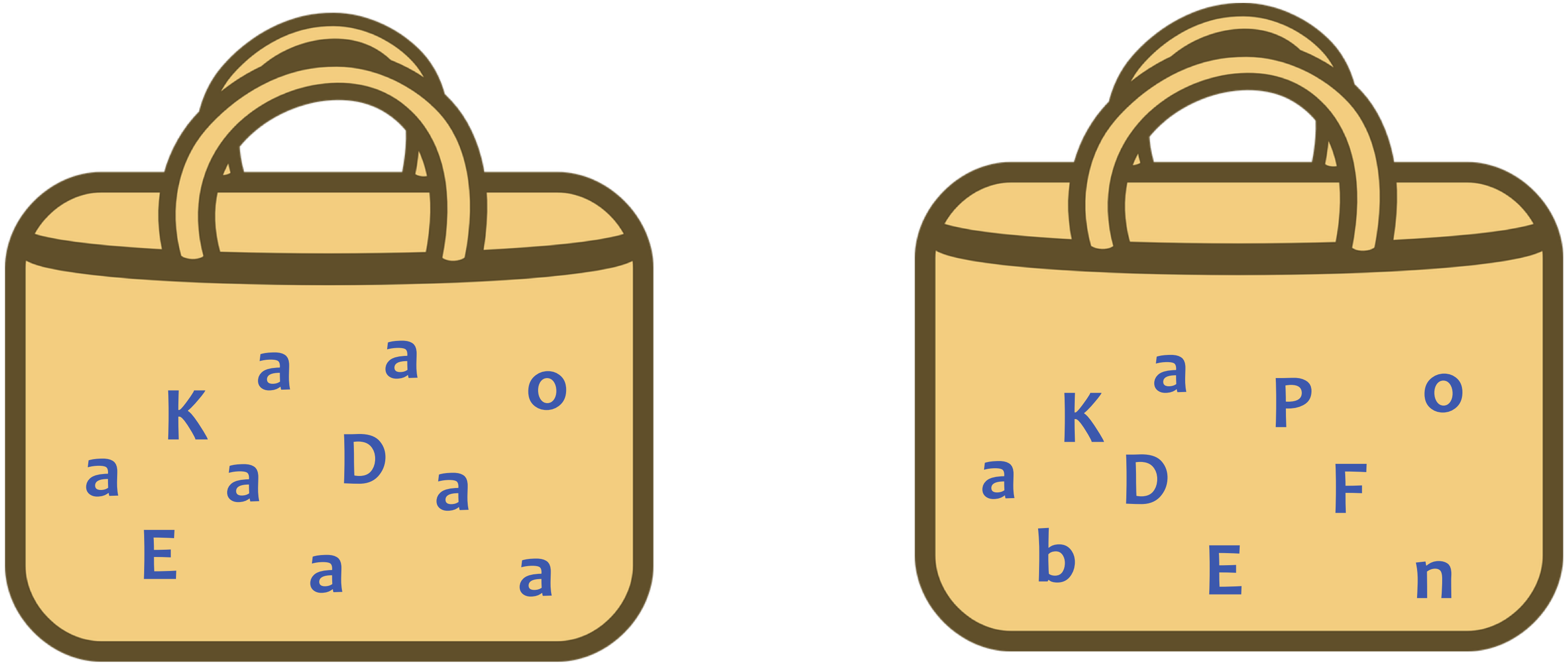
	…
10/100	text ?
5/100	mining ?
3/100	association ?
3/100	database ?
	…
1/100	query ?
	…

text 10
mining 5
association 3
database 3
algorithm 2
…
query 1
efficient 1

A paper (total #words=100)

Which Bag of Words More Likely to be Generated

aaaDaaaKoaaaa



Parameter Estimation

- General setting:
  - Given a (hypothesized & probabilistic) model that governs the random experiment
  - The model gives a probability of any data $p(X|\theta)$ that depends on the parameter θ
  - Now, given actual sample data X={x_1,…,x_n}, what can we say about the value of θ?
- Intuitively, take our best guess of θ
  - "best" means "best explaining/fitting the data"
- Generally an optimization problem

Maximum Likelihood Estimation

- Data: a collection of words, $w_1, w_2, \ldots, w_n$
- Model: multinomial distribution $\mathrm{p}(W)$ with parameters $\theta_i = p(w_i)$
- Maximum likelihood estimator: $\hat{\theta} = argmax_{\theta \in \Theta} p(W|\theta)$

Maximum Likelihood Estimation

$$p(W|\theta) = \binom{N}{c(w_1), \dots, c(w_N)} \prod_{i=1}^{N} \theta_i^{c(w_i)} \propto \prod_{i=1}^{N} \theta_i^{c(w_i)}$$

$$\Rightarrow \log p(W|\theta) = \sum_{i=1}^{N} c(w_i) \log \theta_i + const$$

$$\hat{\theta} = argmax_{\theta \in \Theta} \sum_{i=1}^{N} c(w_i) \log \theta_i$$

Maximum Likelihood Estimation

$$\hat{\theta} = argmax_{\theta \in \Theta} \sum_{i=1}^{N} c(w_i) \log \theta_i$$

$$L(W, \theta) = \sum_{i=1}^{N} c(w_i) \log \theta_i + \lambda \left(\sum_{i=1}^{N} \theta_i - 1 \right)$$

Lagrange multiplier

$$\frac{\partial L}{\partial \theta_i} = \frac{c(w_i)}{\theta_i} + \lambda \quad \rightarrow \quad \theta_i = -\frac{c(w_i)}{\lambda}$$

Set partial derivatives to zero

Since $\sum_{i=1}^{N} \theta_i$ =1 we have $\lambda = -\sum_{i=1}^{N} c(w_i)$

Requirement from probability

$$\theta_i = \frac{c(w_i)}{\sum_{i=1}^{N} c(w_i)}$$

ML estimate

Maximum Likelihood Estimation

- For N-gram language models
  - $p(w_i | w_{i-1}, \dots, w_{i-n+1}) = \frac{c(w_i, w_{i-1}, \dots, w_{i-n+1})}{c(w_{i-1}, \dots, w_{i-n+1})}$

Practical Issues

- We do everything in the log space
  - Avoid underflow
  - Adding is faster than multiplying

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

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
```

More Resources

- Google n-gram viewer

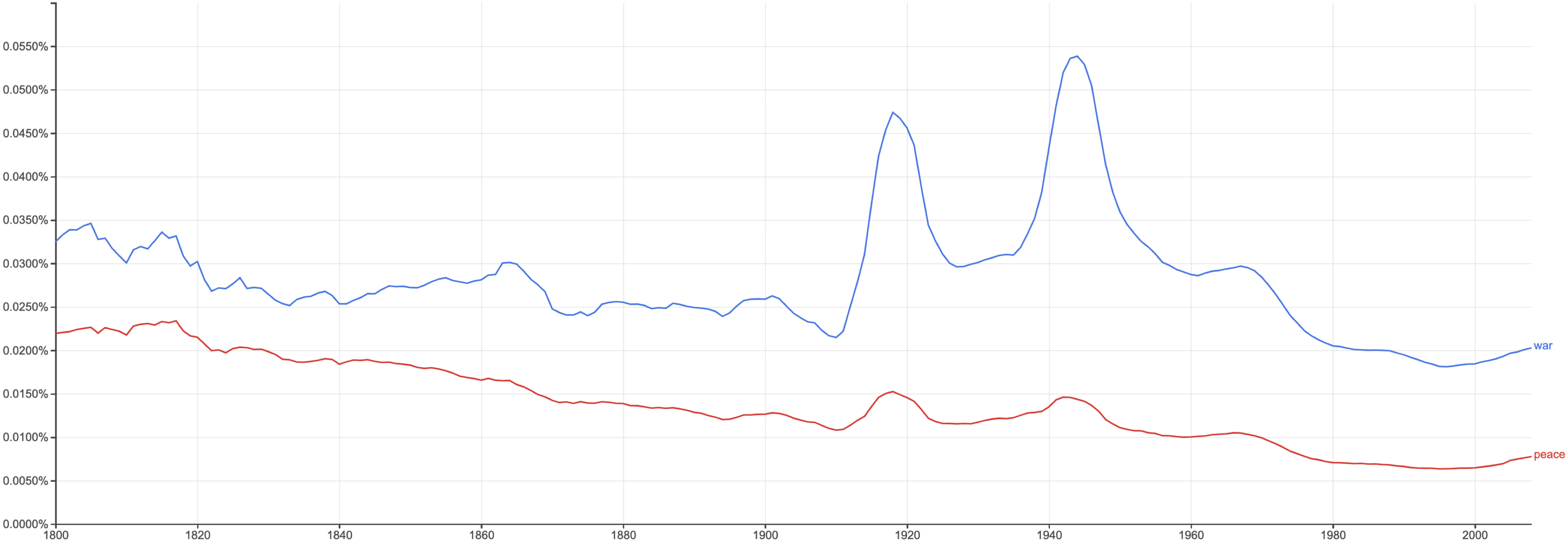
https://books.google.com/ngrams/

Data: http://storage.googleapis.com/books/ngrams/books/datasetsv2.html

Google Books Ngram Viewer

Graph these comma-separated phrases: war,peace case-insensitive

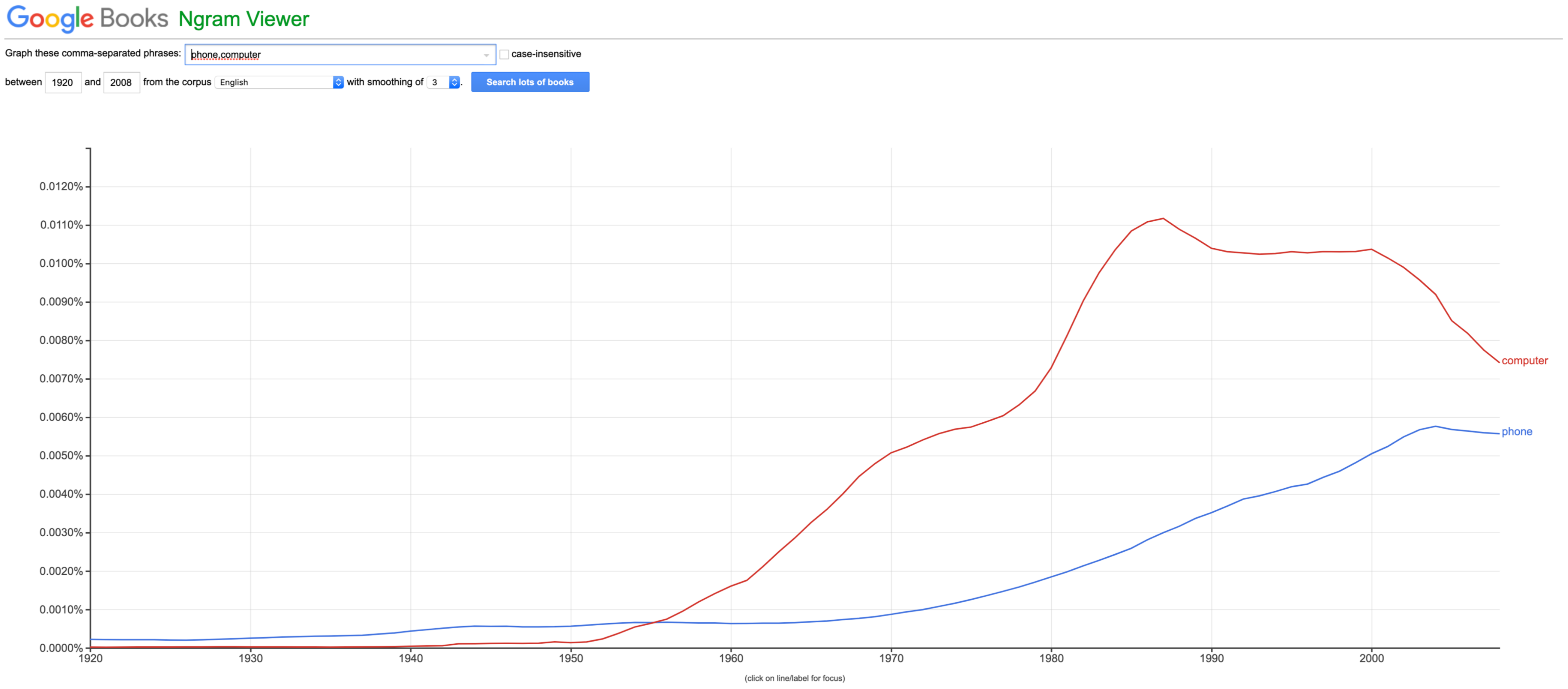
between 1800 and 2008 from the corpus English with smoothing of 3 . Search lots of books

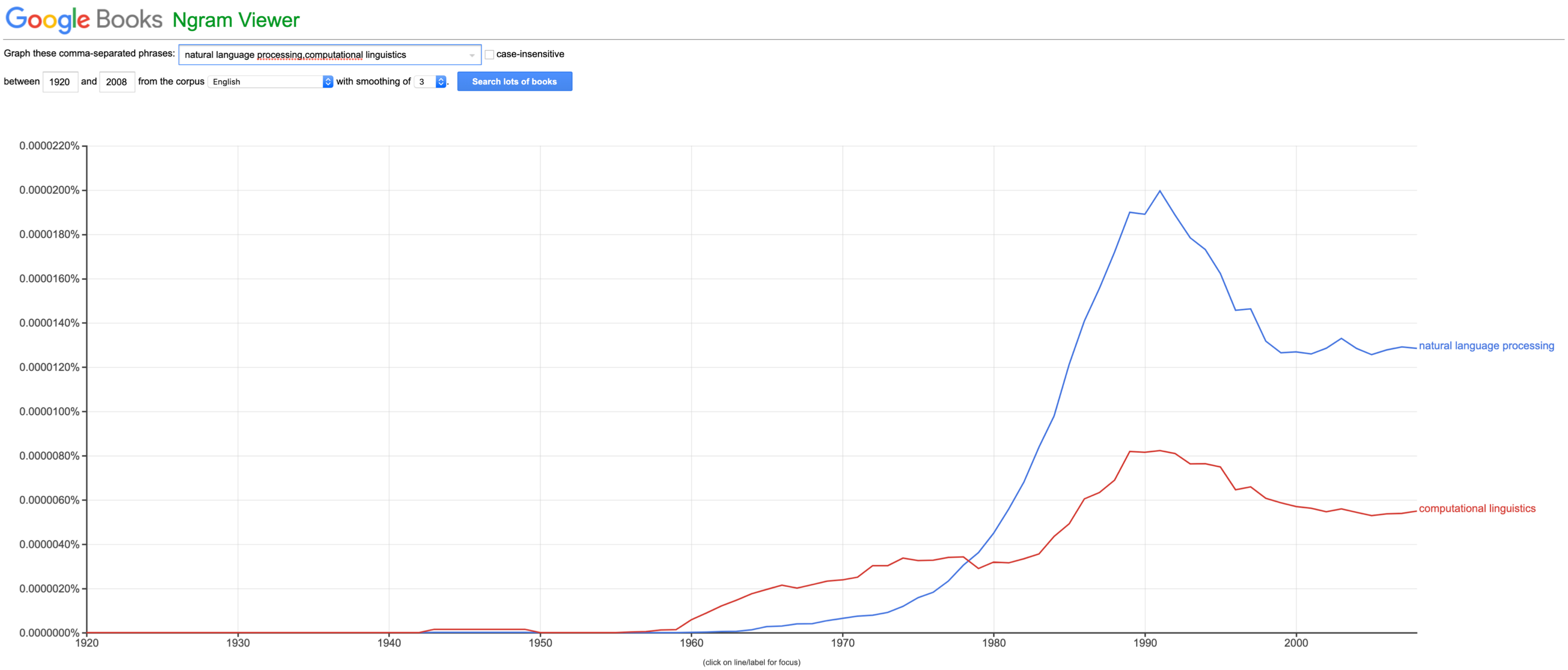


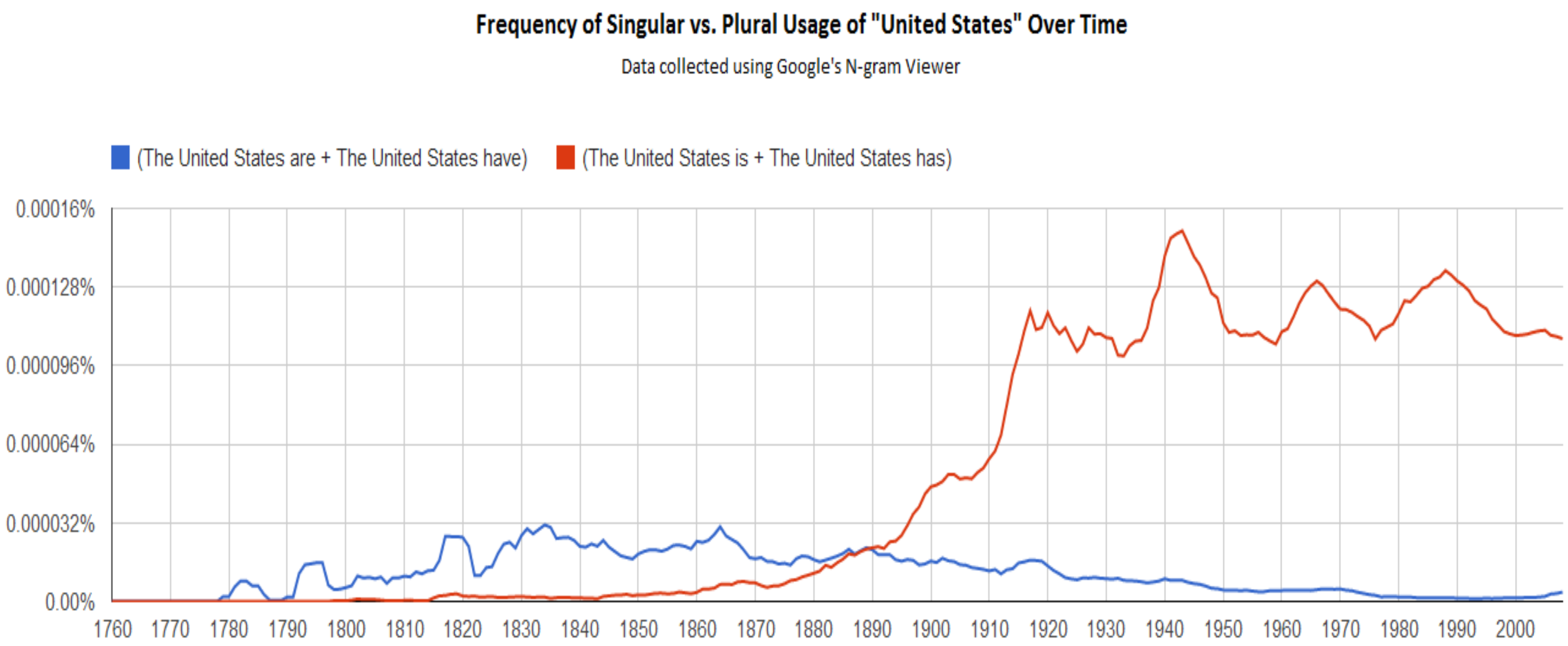
(click on line/label for focus)

Search in Google Books:

1800 - 1817	1818 - 1942	1943 - 1956	1957 - 1978	1979 - 2008	war	English
1800 - 1812	1813 - 1825	1826 - 1875	1876 - 1969	1970 - 2008	peace	English


Google Books Ngram Viewer
Graph these comma-separated phrases:
phone,computer
case-insensitive
between
1920
and
2008
from the corpus
English
with smoothing of
3
Search lots of books
0.0120%
0.0110%
0.0100%
0.0090%
0.0080%
0.0070%
0.0060%
0.0050%
0.0040%
0.0030%
0.0020%
0.0010%
0.0000%
1920
1930
1940
1950
1960
1970
1980
1990
2000
computer
phone
(click on line/label for focus)


Google Books Ngram Viewer
Graph these comma-separated phrases:
natural language processing,computational linguistics
case-insensitive
between 1920 and 2008 from the corpus English with smoothing of 3.
Search lots of books
0.0000220%
0.0000200%
0.0000180%
0.0000160%
0.0000140%
0.0000120%
0.0000100%
0.0000080%
0.0000060%
0.0000040%
0.0000020%
0.0000000%
1920
1930
1940
1950
1960
1970
1980
1990
2000
natural language processing
computational linguistics
(click on line/label for focus)


Frequency of Singular vs. Plural Usage of "United States" Over Time
Data collected using Google's N-gram Viewer
(The United States are + The United States have)
(The United States is + The United States has)
0.00016%
0.000128%
0.000096%
0.000064%
0.000032%
0.00%
1760
1770
1780
1790
1800
1810
1820
1830
1840
1850
1860
1870
1880
1890
1900
1910
1920
1930
1940
1950
1960
1970
1980
1990
2000

How about Unseen Words/Phrases

- Example: Shakespeare corpus consists of N=884,647 word tokens and a vocabulary of V=29,066 word types
- Only 30,000 word types occurred
  - Words not in the training data ⇒ **0** probability
- Only 0.04% of all possible bigrams occurred

How to Estimate Parameters from Training Data

- How do we known $p(w \mid history)$?
  - Use statistics from data (examples using Google N-Grams)
  - E.g., what is p(door | the) ?

Training Counts

198015222 the first
194623024 the same
168504105 the following
158562063 the world
…
14112454 the door

23135851162 the *

$$\hat{P}(\text{door}|\text{the}) = \frac{14112454}{23135851162}$$

$$= 0.0006$$

Increasing N-Gram Order

- High orders capture more dependencies

Bigram Model

```
198015222 the first
194623024 the same
168504105 the following
158562063 the world
…
14112454 the door
-----------------
23135851162 the *
```

$P(\text{door} \mid \text{the}) = 0.0006$

Trigram Model

```
197302 close the window 191125
close the door 152500 close the
gap 116451 close the thread
87298 close the deal
…

-----------------
3785230 close the *
```

$P(\text{door} \mid \text{close the}) = 0.05$

Berkeley Restaurant Project Sentences

- can you tell me about any good cantonese restaurants close by
- mid priced that food is what i’m looking for
- tell me about chez pansies
- can you give me a listing of the kinds of food that are available
- i’m looking for a good place to eat breakfast
- when is cafe venezia open during the day

Bigram Counts (~10K 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

Bigram Probabilities

	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

What Did We Learn

- p(English | want) < p(Chinese | want) – people like Chinese stuff more when it comes to this corpus
- English behaves in a certain way
  - p(to | want) = 0.66
  - p(eat | to) = 0.28

Sparseness

- Maximum likelihood for estimating q
  - Let $c(w_1, w_2, \dots, w_n)$ be the number of times that n-gram appears in a corpus

$$q(w_i | w_{i-2}, w_{i-1}) = \frac{c(w_{i-2}, w_{i-1}, w_i)}{c(w_{i-2}, w_{i-1})}$$

  - If vocabulary has 20,000 words → number of parameters is 8 x 10^{12}!
  - Most n-grams will never be observed, even if they are linguistically plausible
  - Most sentences will have zero or undefined probabilities

How To Evaluate

- **Extrinsic:** build a new language model, use it for some task (MT, ASR, etc.)
- **Intrinsic:** measure how good we are at modeling language

Intrinsic Evaluation

- Intuitively, language models should assign high probability to real language they have not seen before
  - Want to maximize likelihood on test, not training data
  - Models derived from counts / sufficient statistics require generalization parameters to be tuned on held-out data to stimulate test generalization
  - Set hyperparameters to maximize the likelihood of the held-out data (usually with grid search or EM)

Intrinsic Evaluation

- Intuitively, language models should assign high probability to real language they have not seen before