Language Modeling (Part 1)

LING83800: METHODS IN COMPUTATIONAL LINGUISTICS II

March 25, 2024

Spencer Caplan

Administrative Updates

- No practicum this Friday
- Back to normal next week
  - Lecture: Monday 4/1
  - Practicum: Friday 4/5
- I'll get HW5 back to you later this week
- HW6 to be released after that – not due until two weeks from today (4/8)

Today

- Question on Probability?
- Language Models
- Unigrams
- Smoothing
- Bigrams
- Evaluation

Overview from last class

- Random events and random variables
- Probability distribution

$$\sum_{\omega \in \Omega} P(\omega) = 1.$$

Overview from last class

- Random events and random variables
- Probability distribution
- MLE
- Joint, conditional, and marginal probabilities

$$P(E_2|E_1) = \frac{P(E_1, E_2)}{P(E_1)} \quad \text{if } P(E_1) > 0$$

Overview from last class

- Random events and random variables
- Probability distribution
- MLE
- Joint, conditional, and marginal probabilities
- Independence
- Expectation

$$E[X|Y=y] = \sum_{x \in \chi} x * P(X = x|Y = y)$$

Overview from last class

- Random events and random variables
- Probability distribution
- MLE
- Joint, conditional, and marginal probabilities
- Independence
- Expectation
- Chain rule
- Markov assumption

$$P(A \wedge B) = P(A \mid B) \cdot P(B) = P(B \mid A) \cdot P(A)$$

Which could it be?

Which is a more **reasonable** English sentence:

a) “I bought a rose”

b) “I bought arose”

Which could it be?

Which is a more **reasonable** English sentence:

a) "What did Peter eat ravioli and?"

b) "What did Peter eat ravioli with?"

Which could it be?

(Knowing that *dog* could be a verb, as in "Accusation of corruption have dogged the former president for years")

a) "Dogs dogs dog dog dogs"
   i.e. "Dogs_N (that other) dogs_N dog_V[bother] also dog_V[trouble] (other) dogs_N"

b) "Cats (that) dogs chase love fish"

The second sentence has the same structure!

"N (that) N V V S"

Which could it be?

a) “I bought a rose”
b) “I bought arose”

A full answer to this problem is **hard**

But we can hack a partial solution using a ***Language Model (LM)***

Language Models and Probability

- Categorical (yes-or-no) vs. gradient (probabilistic) judgements

LMs take as input a sequence of linguistic units and return (an estimate of) the probability of that sequence

The probability of a sequence is a real number between 0 and 1

- High-probability sequences are more likely to occur than low-probability ones
- An LM could rank the sentences at the start of the class and answer our original question – among many other applications (spelling, MT, etc.)

Language *can't* be reduced to probabilities…

a) I went for a walk but I forgot my phone.

b) ?I went for a walk but I forgot my torso.

This point dates all the way back to Noam Chomsky in LSLT (1955)

a) ? Colorless green ideas sleep furiously.

b) * Furiously sleep ideas green colorless.

Science and Engineering

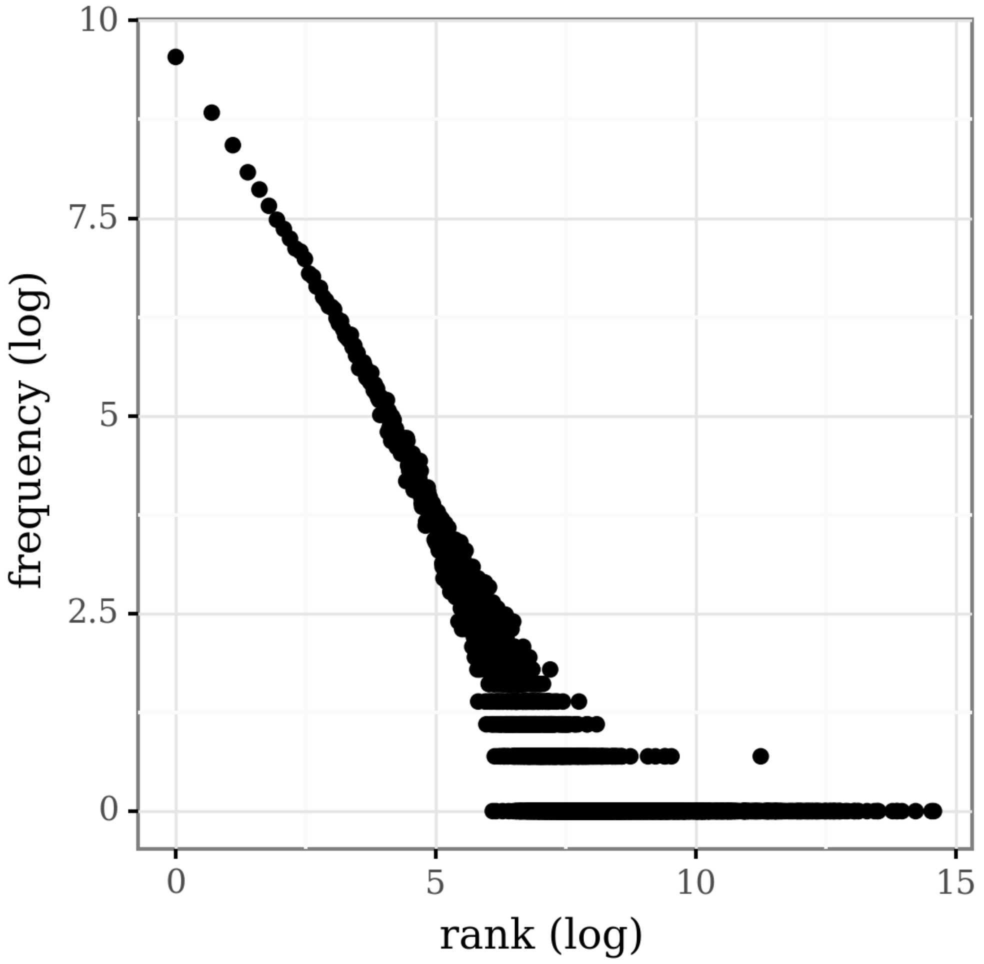
Schism between **cognitive-science** and **engineering** approaches to **modeling of human language**

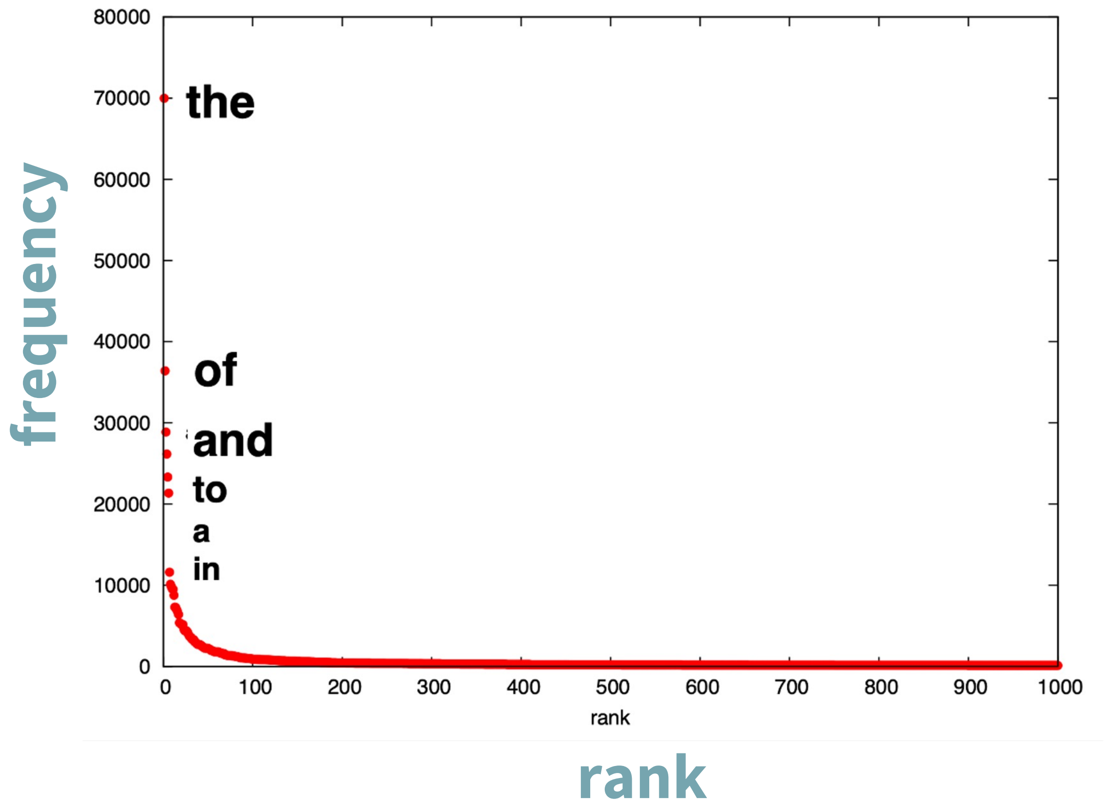
Modern engineering solutions…:

1. are heuristic in nature
2. make few (or weak) affordances for cognitive plausibility, and
3. conflate ill-formed and improbable utterances

Not to mention the **sparse-data problem**

Sparse Data





Few sentences in any given corpus will have ever occurred before

Another way to look at it: paradigm sparsity

A table of Spanish verb forms

- Common in the classroom; absent in the wild
- How many of these will a native speaker actually hear in their lifetime?

	Present Indicative	Preterite Indicative	Imperfect Indicative	Future	Present Subjunctive	Imperfect Subjunctive	Conditional	Imperative	Non-Finite
1sg	hablo	hablé	hablaba	hablaré	hable	hablara	hablaría		hablar
2sg	hablas	hablaste	hablabas	hablarás	hables	hablaras	hablarías	habla	hablando
3sg	habla	habló	hablaba	hablará	hable	hablara	hablaría		hablado
1pl	hablamos	hablamos	hablábamos	hablaremos	hablemos	habláramos	hablaríamos		
2pl	habláis	hablasteis	hablabais	hablaréis	habléis	hablarais	hablaríais	hablad	
3pl	hablan	hablaron	hablaban	hablarán	hablen	hablaran	hablarían		

Another way to look at it: paradigm sparsity

A table of Spanish verb forms

- For *Hablar*, about 30% can be found in a few million words of speech
- The maximum attested (*decir*): around 70%
- Median: about 1 verb form….

	Present Indicative	Preterite Indicative	Imperfect Indicative	Future	Present Subjunctive	Imperfect Subjunctive	Conditional	Imperative	Non-Finite
1sg	hablo	hablé	hablaba	hablaré					
2sg	hablas	hablaste							hablando
3sg	habla	habló	hablaba				hablaría		
1pl									
2pl									
3pl	hablan	hablaron			hablen				

Another way to look at it: paradigm sparsity

A table of Spanish verb forms

- For *Hablar*, about 30% can be found in a few million words of speech
- The maximum attested (*decir*): around 70%
- Median: about 1 verb form….

	Present Indicative	Preterite Indicative	Imperfect Indicative	Future	Present Subjunctive	Imperfect Subjunctive	Conditional	Imperative	Non-Finite
1sg									
2sg									
3sg	suspira								
1pl									
2pl						suspiraseis			
3pl									

A speaker might NEVER hear or produce this form, but they still “know” it!

Another way to look at it: paradigm sparsity

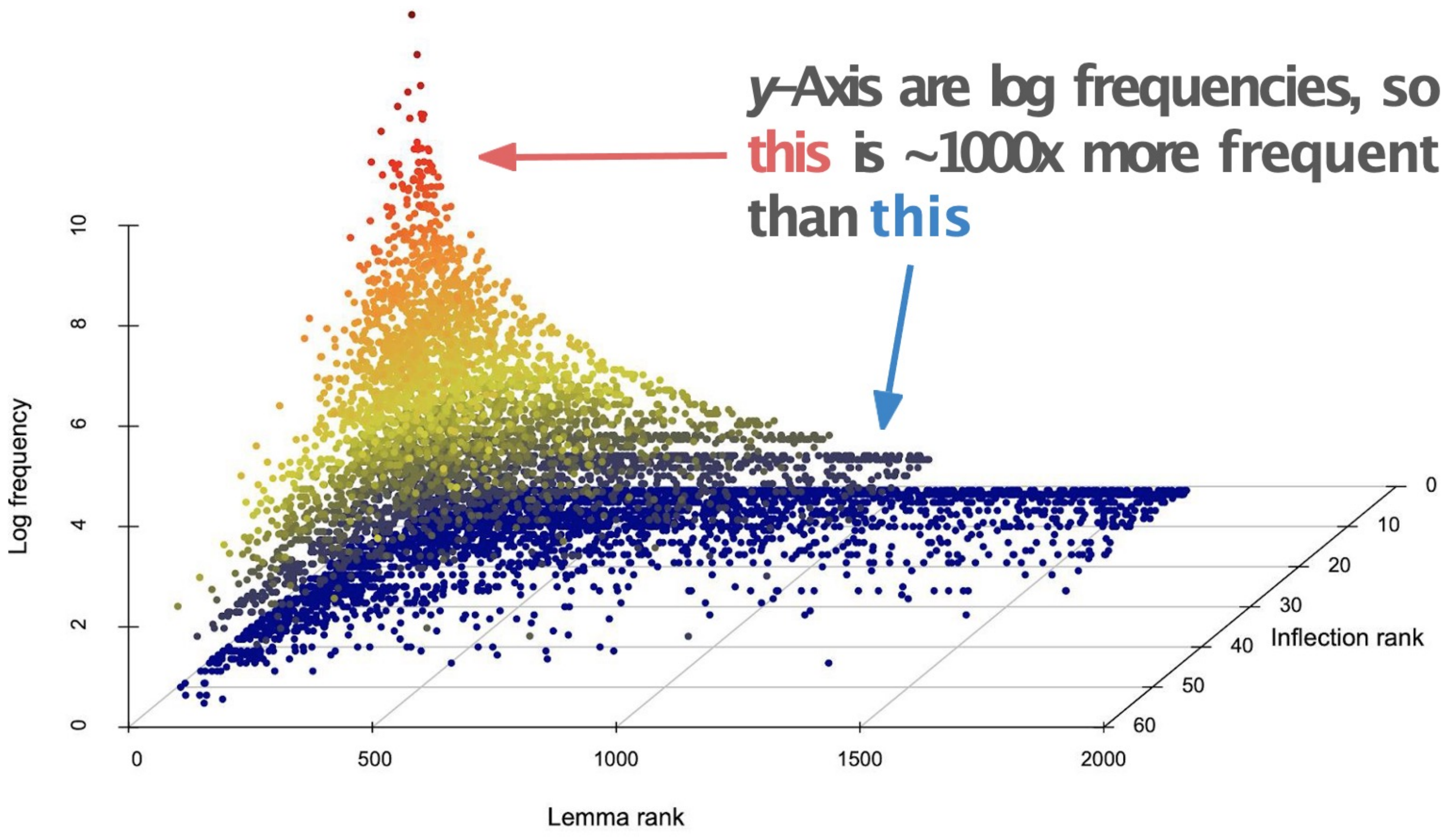


Figure 1: Frequencies of CHILDES Spanish lemmas across inflection categories.

Modeling documents with unigrams

Documents as sequences of words

- Consider *language identification*
  - "Is this document written in French or English?"

- Assume, we have two corpora (sets of documents)
  - One set we know is English, the other set we know is French

- Training data vs. Testing Data

Documents as sequences of words

- Break documents into smaller sequences and compare the pieces

If W = set of possible words, then:

Document of length “n” = $\vec{w} = (w_1, \dots, w_n)$

Documents as sequences of words

How to treat words like a finite set

U = "unknown word"

If W_0 is the set of words appearing in a corpus, then set of possible words is:

$$W = W_0 \cup \text{"*U*"}$$

LMs as models of possible documents

For a document of length “n” = $\vec{w} = (w_1, \dots, w_n)$ then

A language model is just a probability distribution P(W)

But what is the “true” distribution over English documents W (does that even make sense?)

Assumption: training corpus of documents **d** contains a representative sample from P(W) and we can use that to estimate P(W)

Unigram language models

$$P(W) = P(N)\prod_{i=1}^{N} P(W_i)$$

Unigram language models include a strict *independence assumption*:

$$P(W_i = w) = P(W_j = w)$$

A *generative model* of document creation – we'll talk about this more soon

Unigram language models

We need to introduce a parameter to properly model the likelihood of each word:

$$P(W_i = w) = \theta_w$$

$$P(W) = P(N)\prod_{i=1}^{N}\theta_w$$

Maximum likelihood estimates of unigram parameters

How do we estimate the vector of parameters θ of a unigram language model from a corpus of documents $\mathbf{d}$?

Probability jargon:

- A “statistic” is a function of the data
- An “estimator” for a parameter is a function whose value is intended to approximate that parameter

For us, the maximum likelihood estimator (MLE) sets θ_w to be:

$$\hat{\theta}_w = \frac{n_w(d)}{n_0(d)}$$

Maximum likelihood estimates of unigram parameters

- Suppose we have a corpus size n_0(d) = 10^7 . Consider two words, ‘the’ and ‘equilateral’ with counts 2* 10^5 and 2, respectively.

- Then their maximum likelihood estimates are 0.02 and 2* 10^{-7}

Maximum likelihood principle

"to estimate the value of a parameter θ from data *x*, select the value $\hat{\theta}$ of θ that makes *x* as likely as possible"

$$likelihood\ function\ L_x(\theta) = P_\theta(X)$$

Maximum likelihood principle

"to estimate the value of a parameter θ from data *x*, select the value $\hat{\theta}$ of θ that makes *x* as likely as possible"

$$L_d(\theta) = \prod_{w \in W} \theta_w^{n_w(d)}$$

Number of times word "w" appears in the document **d**

Probability of the word type "w"

Maximum likelihood principle

Example 1.6: Consider the "document" (we call it ♡) consisting of the phrase '*I love you*' one hundred times in succession:

$$\begin{aligned} L_{\heartsuit}(\boldsymbol{\theta}) &= (\theta_i)^{n_i(\heartsuit)} \cdot (\theta_{\text{`love'}})^{n_{\text{`love'}}(\heartsuit)} \cdot (\theta_{\text{`you'}})^{n_{\text{`you'}}(\heartsuit)} \\ &= (\theta_i)^{100} \cdot (\theta_{\text{`love'}})^{100} \cdot (\theta_{\text{`you'}})^{100} \end{aligned}$$

The θ_ws in turn are all 100/300=1/3, so

$$\begin{aligned} L_{\heartsuit}(\boldsymbol{\theta}) &= (1/3)^{100} \cdot (1/3)^{100} \cdot (1/3)^{100} \\ &= (1/3)^{300} \end{aligned}$$

Sparse-data Problems

Sparse-data Problems

- Thinking about distinguishing English from French:
  - What would happen if we implemented the current MLE but the test document included a word not in our training documents?

Sparse-data Problems

- Thinking about distinguishing English from French:
  - What would happen if we implemented the current MLE but the test document included a word not in our training documents?

We defined our vocabulary to include *U*, but *U* doesn't appear in our training data, so the maximum likelihood estimate assigns it zero probability)

The document gets assigned zero probability!

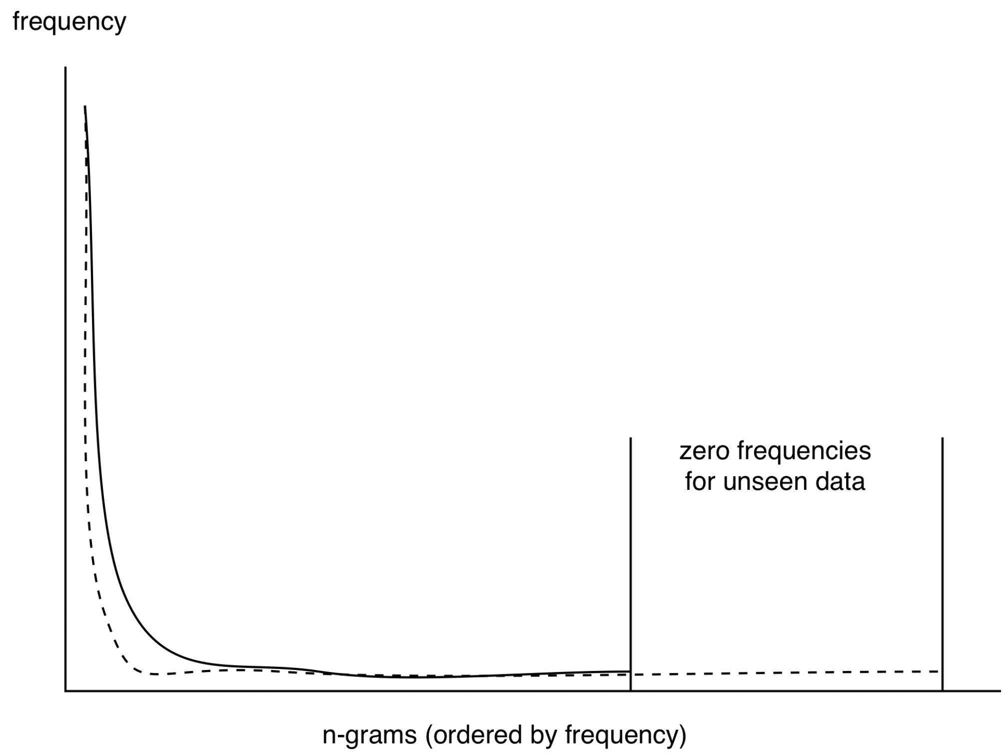
Sparse-data Problems

- Over-fitting
  - Accurately modeling the training data but not generalizing to novel data

Solution: smoothing!

"This dark art is why NLP is taught in the engineering school"
– Jason Eisner (JHU)

Smoothing



Take from the frequent types and give to the infrequent types

Smoothing

There are many kinds of smoothing

We’ll talk about a bunch next week

But for now let’s start with the simplest: add-alpha

Smoothing: add-alpha

Add a positive number α_w to each word w's empirical frequency

- Important that we readjust the denominator so the revised estimates of θ still sum to 1

$$\tilde{\theta}_w = \frac{n_w(d) + \alpha_w}{n_0(d) + \alpha_0}$$

(where $\alpha_0 = \sum_{w \in W} \alpha_w$ is the sum over all words of the pseudo-counts)

Smoothing: add-alpha (Laplace)

We “bin” words into equivalence classes and assign the same pseudo-count to all words in the same group.

- If there’s only a single equivalence class then $\alpha = \alpha_w$ which is used for all words, and we only need to estimate a single parameter for our held-out data.

Smoothing: add-alpha (Laplace)

Example 1.7: Let us assume that all w get the same smoothing constant. In this case Equation 1.4 simplifies to;

$$\tilde{\theta}_w = \frac{n_w(\boldsymbol{d}) + \alpha}{n_\circ(\boldsymbol{d}) + \alpha|\mathcal{W}|}.$$

Suppose we set $\alpha = 1$ and we have $|\mathcal{W}| = 100,000$ and $n_\circ(\boldsymbol{d}) = 10^7$. As in Example 1.5, the two words '*the*' and '*equilateral*' have counts $2{\cdot}10^5$ and 2, respectively.

Smoothing: add-alpha (Laplace)

Their maximum likelihood estimates again are 0.02 and $2{\cdot}10^{-7}$. After smoothing, the estimate for '*the*' hardly changes

$$\widetilde{\theta}_{\text{'}the\text{'}} = \frac{2{\cdot}10^5 + 1}{10^7 + 10^5} \approx 0.02$$

while the estimate for '*equilateral*' goes up by 50%:

$$\widetilde{\theta}_{\text{'}equilateral\text{'}} = \frac{2 + 1}{10^7 + 10^5} \approx 3{\cdot}10^{-7}$$

Why is it called “Laplace” smoothing?

[Pierre-Simon, Marquis de Laplace]

Estimating smoothing parameters

- How would our current MLE apply here to our current training data **d**?

No good!

- The MLE will just set α to zero

Estimating smoothing parameters

Split our data into three sets:

- Primary training corpus **d**
- Secondary held-out training corpus **h** (also called the "development set" or "dev-set")
- Test corpus **t**

(80%, 10%, 10% is a standard train/dev/test split)

Estimating smoothing parameters

Example 1.8: Suppose our training data $\boldsymbol{d}$ is ♡ from Example 1.6 and the held-out data $\boldsymbol{h}$ is ♡′, which consists of eight copies of '*I love you*' plus one copy each of '*I can love you*' and '*I will love you*'. When we preprocess the held-out data both '*can*' and '*will*' become ∗U∗, so $\mathcal{W} = \{$ i love you ∗U∗$\}$. We let $\alpha = 1$.

Now when we compute the likelihood of ♡′ our smoothed θs are as follows:

$$\begin{aligned}
\widetilde{\theta}_{\text{'i'}} &= \frac{100+1}{300+4} \\
\widetilde{\theta}_{\text{'love'}} &= \frac{100+1}{300+4} \\
\widetilde{\theta}_{\text{'you'}} &= \frac{100+1}{300+4} \\
\widetilde{\theta}_{\text{'*U*'}} &= \frac{1}{300+4}
\end{aligned}$$

Estimating smoothing parameters

- We seek the value $\hat{\alpha}$ of α that maximizes the likelihood L_h of the *held-out* corpus **h**

$$\hat{\alpha} = \operatorname*{argmax}_{\alpha} L_{\boldsymbol{h}}(\alpha)$$

$$L_{\boldsymbol{h}}(\alpha) = \prod_{w \in \mathcal{W}} \left(\frac{n_w(\boldsymbol{d}) + \alpha}{n_\circ(\boldsymbol{d}) + \alpha|\mathcal{W}|} \right)^{n_w(\boldsymbol{h})}$$

This just says that the likelihood of the held-out data is the product of the probability of each word token in the data

Estimating smoothing parameters

$$\hat{\alpha} = \underset{\alpha}{\operatorname{argmax}} \, L_{\boldsymbol{h}}(\alpha)$$

$$L_{\boldsymbol{h}}(\alpha) = \prod_{w \in \mathcal{W}} \left(\frac{n_w(\boldsymbol{d}) + \alpha}{n_\circ(\boldsymbol{d}) + \alpha|\mathcal{W}|} \right)^{n_w(\boldsymbol{h})}$$

- The function has a single peak, so a line-search routine can solve this efficiently (e.g. Golden-section search)

A practical note on probabilities

- The probabilities we deal with in NLP are usually extremely small.
  - This leads to underflow errors
- Solution: do everything in log space
  - Avoids underflow
  - (also adding is faster than multiplying)

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

A practical note on probabilities

- The probabilities we deal with in NLP are usually extremely small.
  - This leads to underflow errors

Solution: do everything in log space

Avoids underflow

(also adding is faster than multiplying)

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

Contextual Dependencies

Contextual dependencies

- BUT! Unigram language models assume that words are generated as independent entities.

Thus we still have no way of answering our original motivating question: How to rank (a) as better – more likely – than (b)?

a) “I bought a rose”
b) “I bought arose”

Contextual Dependencies

- Unigrams and the independence assumption

- Cannot capture contextual dependencies among words in the same sentence!

a) "students eat bananas"
b) "bananas eat students"

Contextual Dependencies

- Unigrams and the independence assumption

- Cannot capture contextual dependencies among words in the same sentence!

a) "students eat bananas"
b) "bananas eat students"

Unigram LMs assign equal probability to both

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

Chain Rule

- Recall the definition of conditional probabilities

  $p(B|A) = P(A,B)/P(A)$ Rewriting: **$P(A,B) = P(A)P(B|A)$**

- More variables:

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

- The Chain Rule in General

$$P(x_1,x_2,x_3,\ldots,x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\ldots P(x_n|x_1,\ldots,x_{n-1})$$

The Chain Rule applies to compute joint probability of words in sentence

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

P("its water is so transparent") =

P(its) × P(water|its) × P(is|its water)

× P(so|its water is) × P(transparent|its water is so)

How to estimate these probabilities

- Could we just count and divide?

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

- No! Too many possible sentences!
- We'll never see enough data for estimating these

Markov Assumption

Andrei Markov

- Simplifying assumption:

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

- Or maybe

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

Markov Assumption

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

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

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

Simplest case: Unigram model

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

Some automatically generated sentences from a unigram model

```
fifth, an, of, futures, the, an, incorporated, a,
a, the, inflation, most, dollars, quarter, in, is,
mass

thrift, did, eighty, said, hard, 'm, july, bullish

that, or, limited, the
```

N-grams

- Slide a window of length *n* words over the text
- Overlapping sequences of length n that we see through this window are called **n-grams**

- N=2 (bigrams)
- N=3 (trigrams)
- N=4 (…. Just called 4-grams)

Bigram model

- Condition on the previous word:

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

```
texaco, rose, one, in, this, issue, is, pursuing, growth, in,
a, boiler, house, said, mr., gurria, mexico, 's, motion,
control, proposal, without, permission, from, five, hundred,
fifty, five, yen

outside, new, car, parking, lot, of, the, agreement, reached

this, would, be, a, record, november
```

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(s) which I had just put into the machine room on the fifth floor is (are) crashing."

- But for engineering purposes we can often get away with N-gram models

Estimating N-gram Probabilities

Estimating bigram probabilities

- The Maximum Likelihood Estimate

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

Just different notation for the "n" function on

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

Padding tokens

- Special tokens added to beginning / end of sentence to allow n-gram calculation

Bigram 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>

"Padding" tokens

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

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

Raw bigram counts

- 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 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

Bigram estimates of sentence probabilities

P(<s> I want english food </s>) =

P(I|<s>)

× P(want|I)

× P(english|want)

× P(food|english)

× P(</s>|food)

= .000031

What kinds of knowledge

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

Google Books N-grams

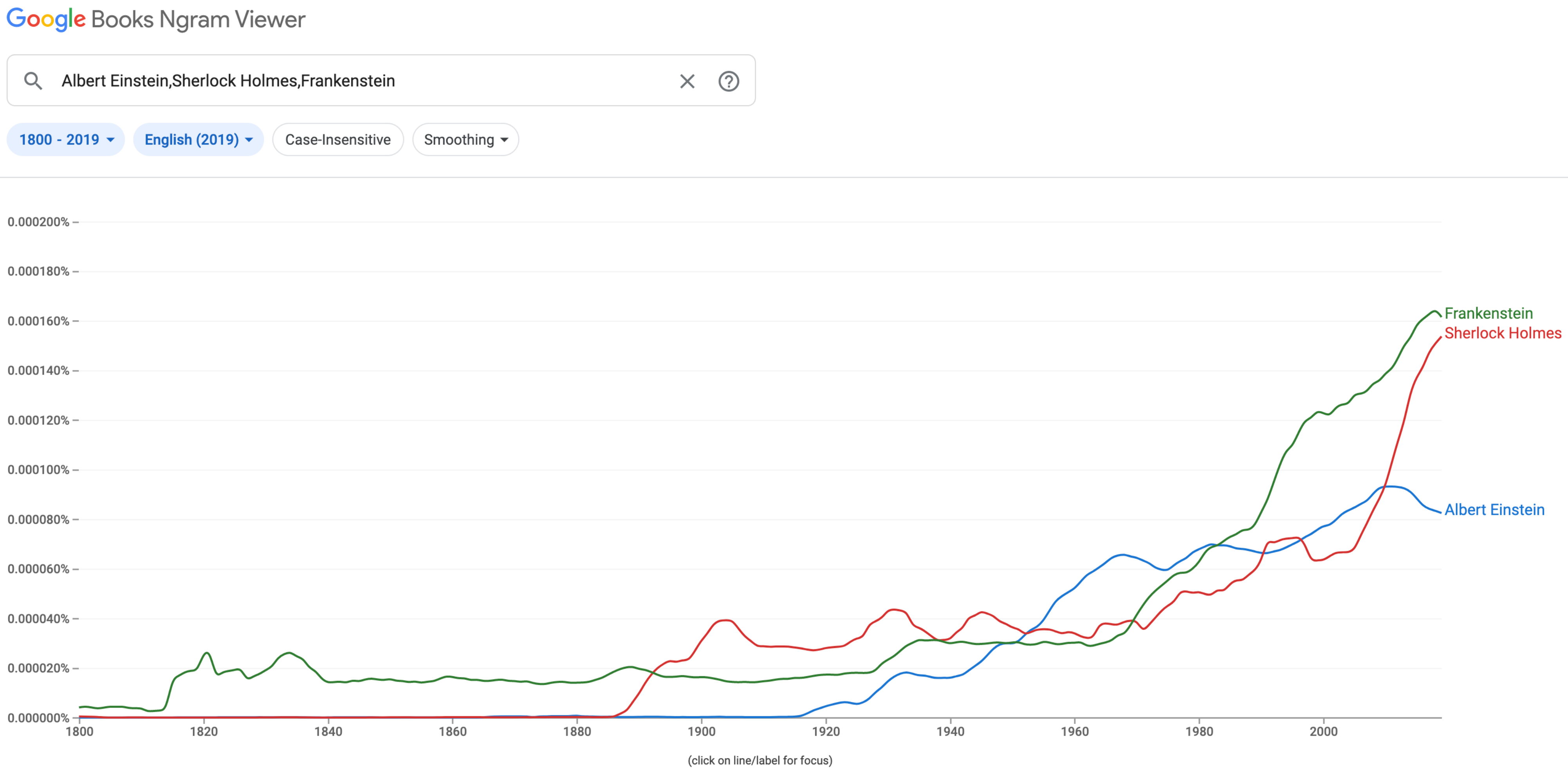
- `serve as the incoming 92`
- `serve as the incubator 99`
- `serve as the independent 794`
- `serve as the index 223`
- `serve as the indication 72`
- `serve as the indicator 120`
- `serve as the indicators 45`
- `serve as the indispensable 111`
- `serve as the indispensible 40`
- `serve as the individual 234`

http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html

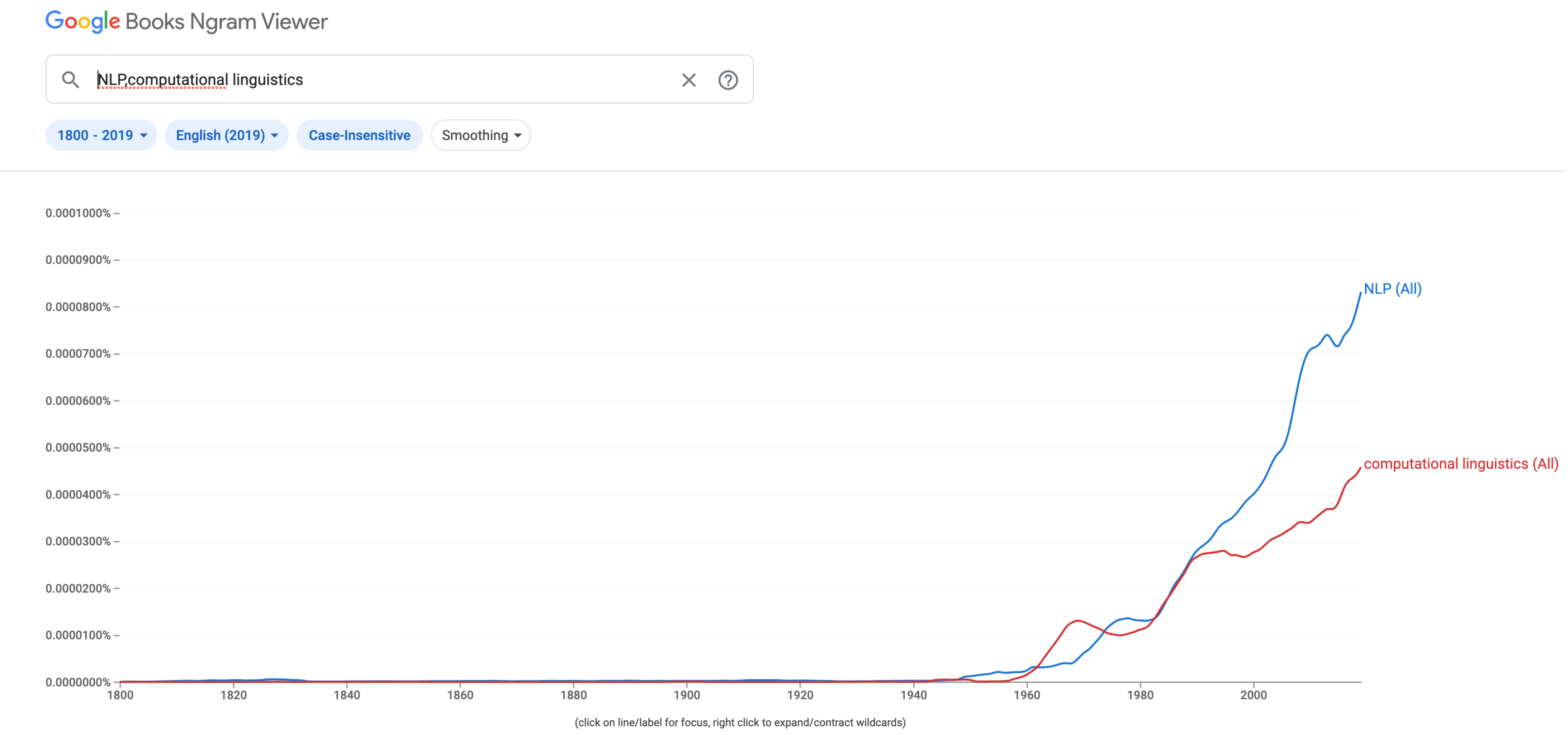
Google Books N-grams

https://books.google.com/ngrams

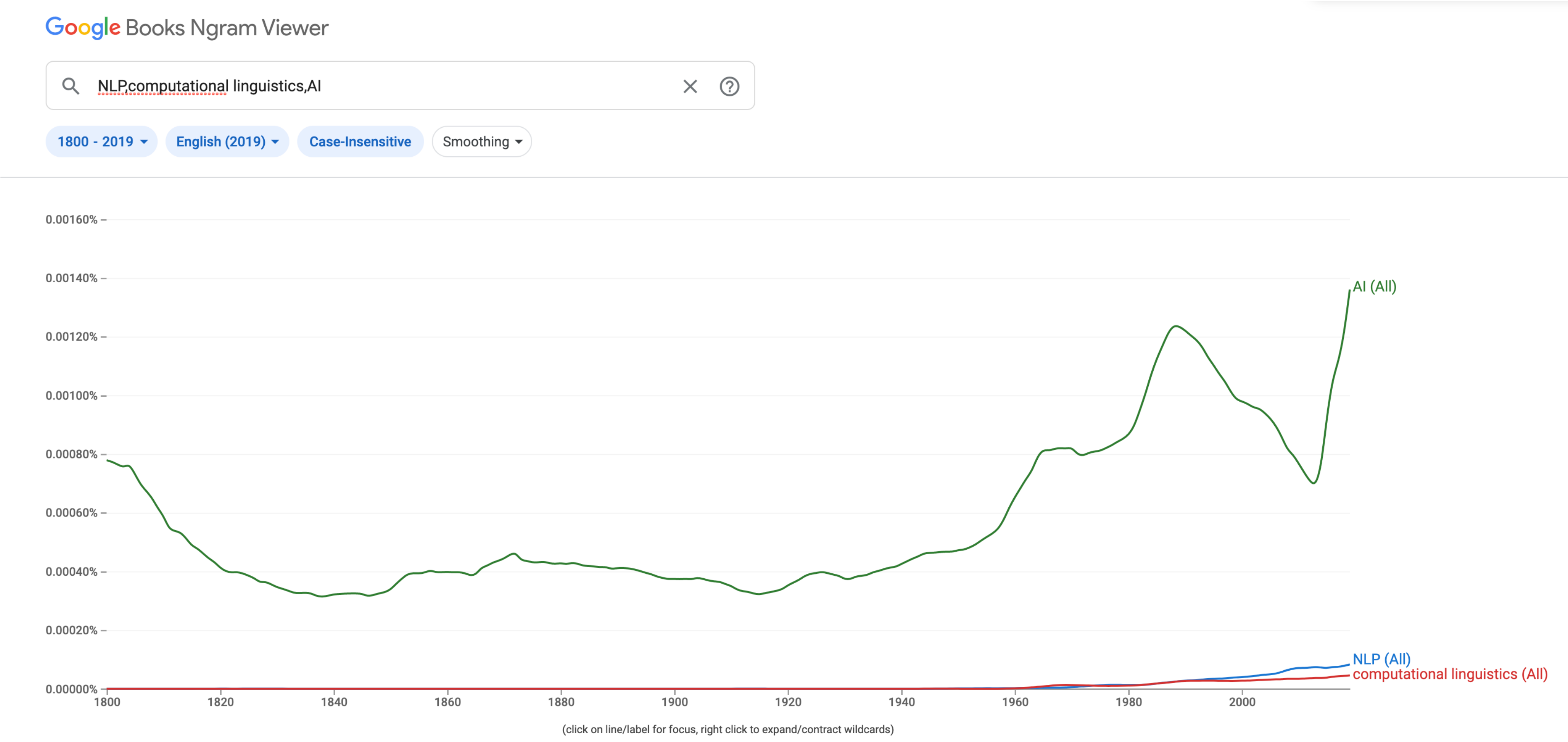
Google N-grams Samples



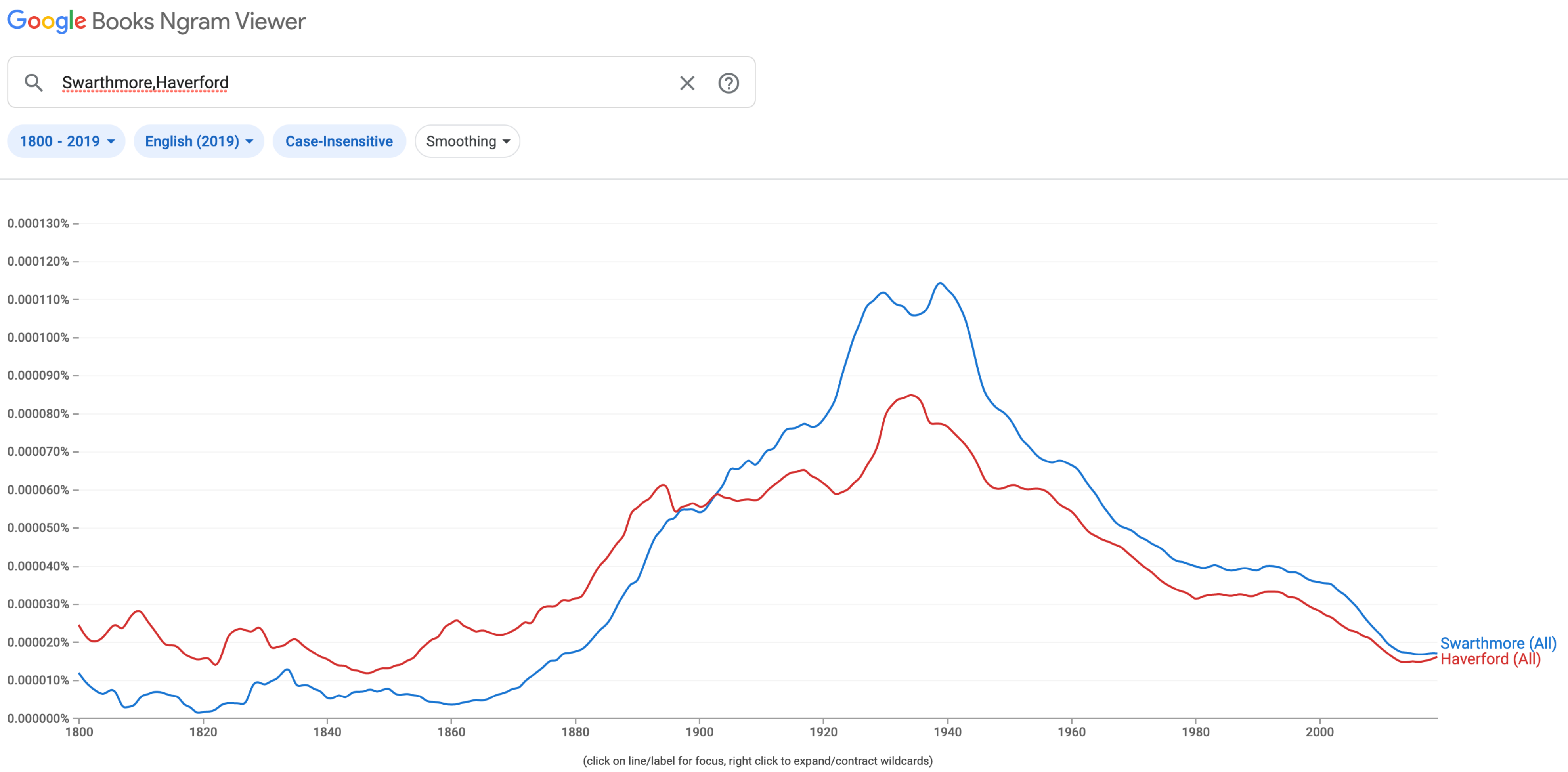
Google N-grams Samples



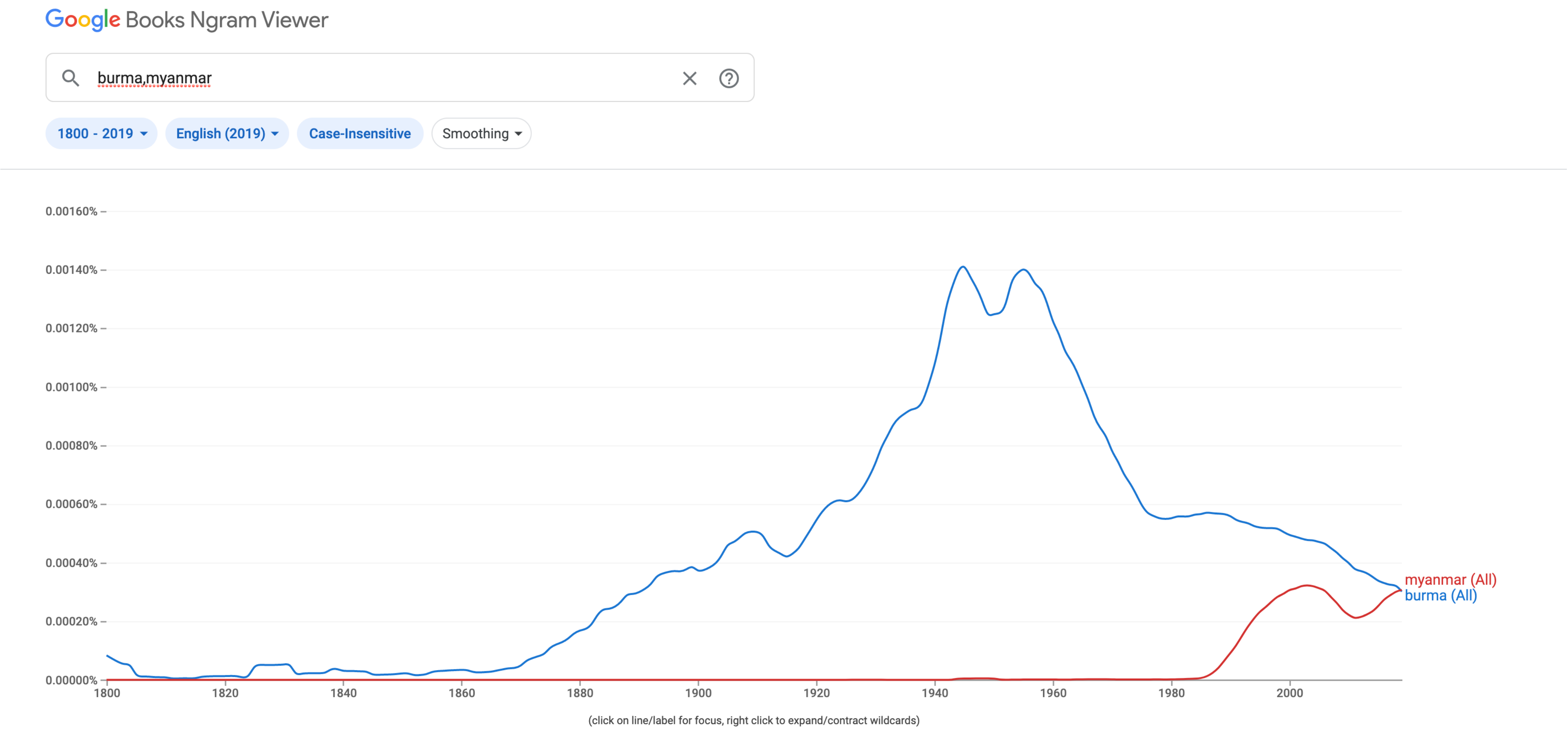
Google N-grams Samples



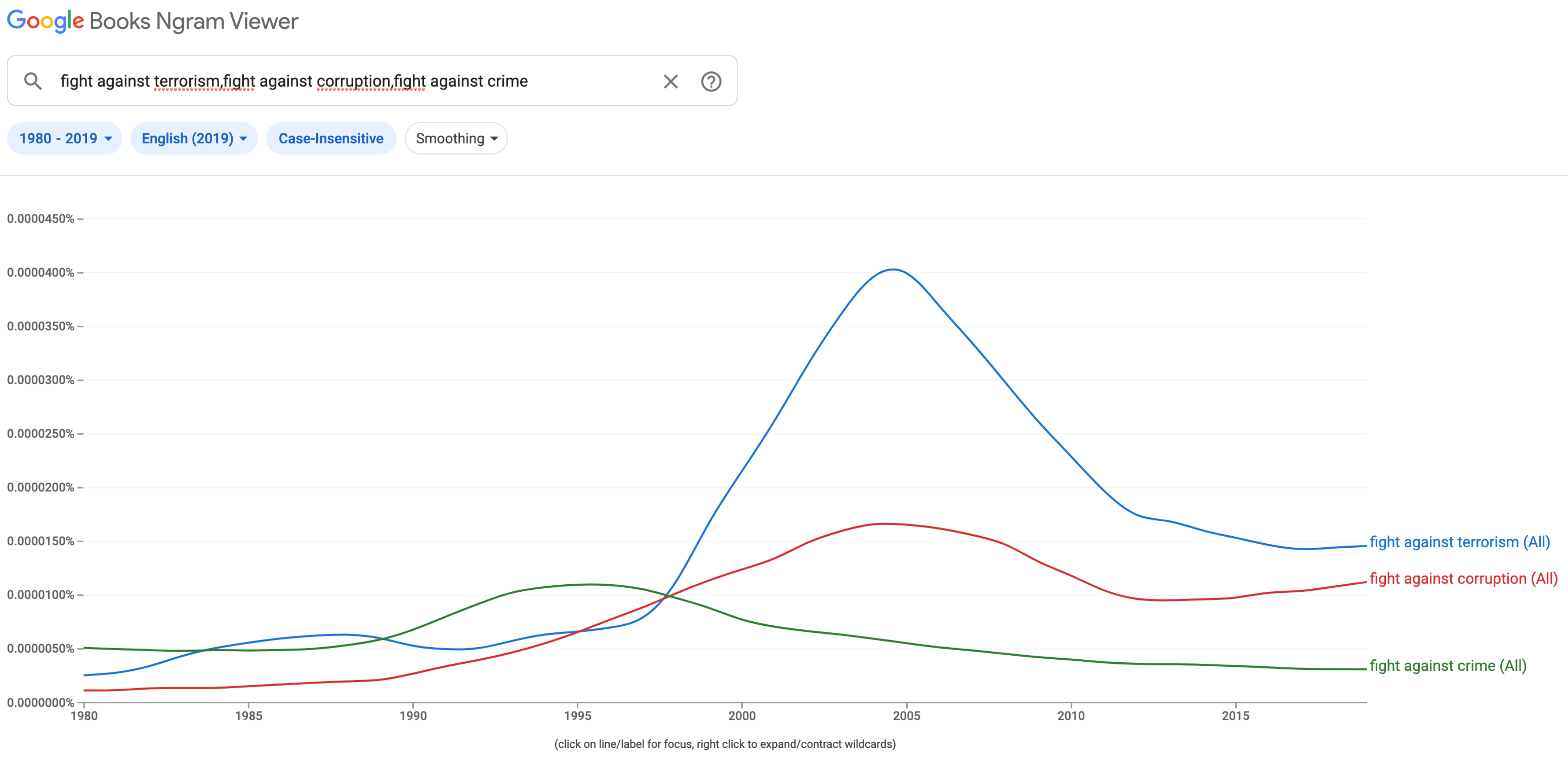
Google N-grams Samples



Google N-grams Samples



Google N-grams Samples



Evaluating Models

Evaluation: How good is our model?

- Does our language model prefer good sentences to bad ones?
  - Assign higher probability to "real" or "frequently observed" sentences
    - Than "ungrammatical" or "rarely observed" sentences?
- We train parameters of our model on a **training set**.
- We test the model's performance on data we haven't seen.
  - A **test set** is an unseen dataset that is different from our training set, totally unused.
  - An **evaluation metric** tells us how well our model does on the test set.

Training on the test set

- We can’t allow test sentences into the training set
- We will assign it an artificially high probability when we set it in the test set

Which is bad science!

Extrinsic evaluation

- Best evaluation for comparing models A and B
  - Put each model in a task
    - spelling corrector, speech recognizer, MT system
  - Run the task, get an accuracy for A and for B
    - How many misspelled words corrected properly
    - How many words translated correctly
  - Compare accuracy for A and B

Difficulty of extrinsic evaluation

- Extrinsic evaluation
  - Time-consuming; can take days or weeks
- So
  - Sometimes use **intrinsic** evaluation: **perplexity**
  - Bad approximation
    - unless the test data looks **just** like the training data
    - So **generally only useful in pilot experiments**
  - But is helpful to think about.

Intuition of Perplexity

- The Shannon Game:
  - How well can we predict the next word?

    I always order pizza with cheese and ____

    The 33rd President of the US was ____

    I saw a ____

  - Unigrams are terrible at this game. (Why?)
- A better model of a text
  - is one which assigns a higher probability to the word that actually occurs

mushrooms 0.1

pepperoni 0.1

anchovies 0.01

….

naan 0.0001

….

and 1e-100

Perplexity

The best language model is one that best predicts an unseen test set

- Gives the highest P(sentence)

$$PP(W) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}}$$

Perplexity is the inverse probability of the test set, normalized by the number of words:

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_N)}}$$

By the chain rule:

$$\text{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \dots w_{i-1})}}$$

For bigrams:

$$\text{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}$$

Minimizing perplexity is the same as maximizing probability

Example for random digits

- Let's suppose a sentence consisting of random digits
- What is the perplexity of this sentence according to a model that assign P=1/10 to each digit?

$$
\begin{aligned}
\text{PP}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\
&= (\frac{1}{10}^N)^{-\frac{1}{N}} \\
&= \frac{1}{10}^{-1} \\
&= 10
\end{aligned}
$$

Lower perplexity = higher probability = better model

- Training 38 million words, test 1.5 million words, WSJ

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109