Self-Supervised Language Modeling

CSCI 601 471/671
NLP: Self-Supervised Models

https://self-supervised.cs.jhu.edu/sp2023/

JOHNS HOPKINS UNIVERSITY

[Slide credit: Mohit Iyyer, Chris Manning, and many others]

Motivation

- Earlier we define **Self-Supervised** models as as predictive models of the world!
- **Word2Vec:** Predictive models given word representations.
- **Now with neural networks:** predictive models of word compositions

Chapter Plan

The

The cat

The cat sat

The cat sat on

The cat sat on __?__

The cat sat on the mat.

The cat sat on the mat.

$$\mathbf{P}(\underbrace{\text{mat}}_{\text{next word}} \mid \underbrace{\text{The cat sat on the}}_{\text{context or prefix}})$$

Probability of Upcoming Word

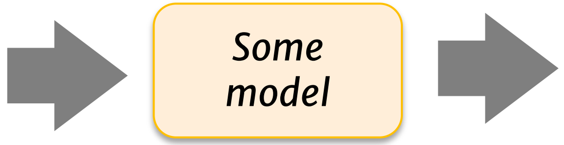
$$\mathbf{P}(\underbrace{X_t}_{\text{next word}} \mid \underbrace{X_1, \ldots, X_{t-1}}_{\text{context or prefix}})$$

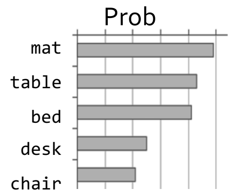
LMs as a Marginal Distribution

- Directly we train models on “marginals”:

$$\mathbf{P}(\underbrace{X_t}_{\text{next word}} | \underbrace{X_1, \ldots, X_{t-1}}_{\text{context}})$$

`“The cat sat on the [MASK]”`





LMs as Implicit Joint Distribution of Language

- Though implicitly we are learning the full distribution over the language:
  - Remember the chain rule: $\mathbf{P}(X_1, \ldots, X_t) = \mathrm{P}(X_1) \prod_{i=1}^{t} \mathrm{P}(X_i \mid X_1, X_2 \ldots, X_i)$

- Language Modeling $\triangleq$ learning prob distribution over language sequence.

Doing Things with Language Model

- What is the probability of ….

"I like Johns Hopkins University"

"like Hopkins I University Johns"

- LMs assign a probability to every sentence (or any string of words).

P("I like Johns Hopkins University EOS") $=10^{-5}$

P("like Hopkins I University Johns EOS") $=10^{-15}$

Doing Things with Language Model (2)

$$\mathbf{P}(\underbrace{X_t}_{\text{next word}} \mid \underbrace{X_1, \ldots, X_{t-1}}_{\text{context}})$$

- We can rank sentences.

- While LMs show "typicality", this may be a proxy indicator to other properties:
    - Grammaticality, fluency, factuality, etc.

P("*I like Johns Hopkins University. EOS*") > **P**("*I like John Hopkins University EOS*")
P("*I like Johns Hopkins University. EOS*") > **P**("*University. I Johns EOS Hopkins like*")
P("*JHU is located in Baltimore. EOS*") > **P**("*JHU is located in Virginia. EOS*")

Doing Things with Language Model (3)

$$\mathbf{P}(\overbrace{X_t}^{\text{next word}} | \overbrace{X_1, \ldots, X_{t-1}}^{\text{context}})$$

- Can also generate strings

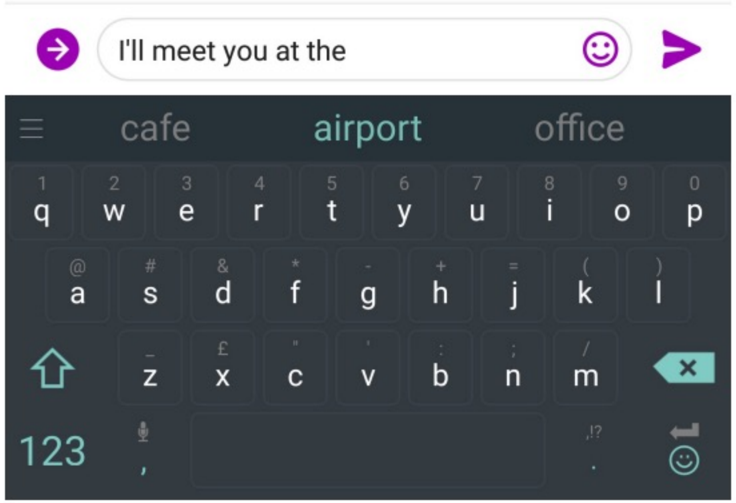
- Let's say we start *"Johns Hopkins is "*
- Using this prompt as initial condition, recursively sample from an LM:

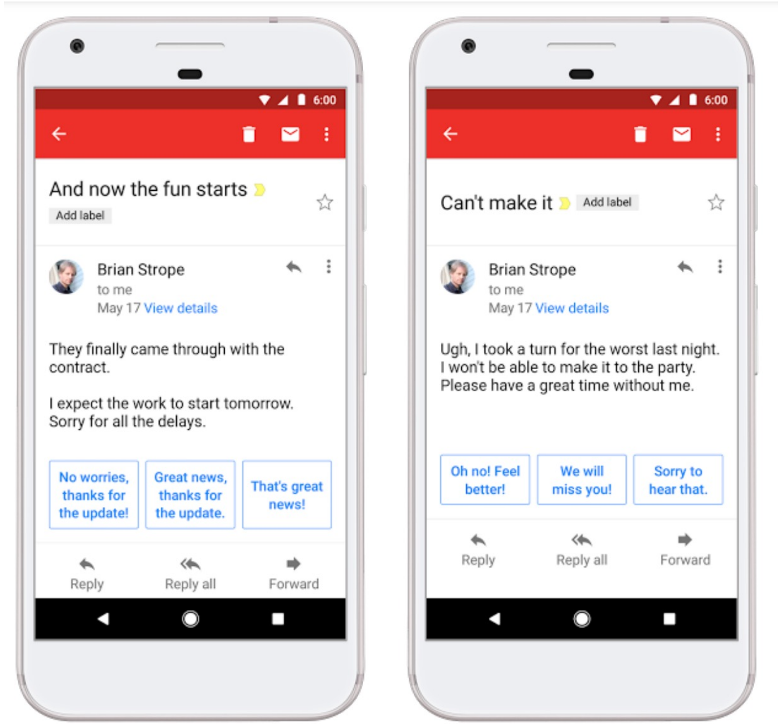
1. Sample from **P**(X| *"Johns Hopkins is "*) ⟶ "located"
2. Sample from **P**(X| *"Johns Hopkins is located"*) ⟶ "at"
3. Sample from **P**(X| *"Johns Hopkins is located at"*) ⟶ "the"
4. Sample from **P**(X| *"Johns Hopkins is located at the"*) ⟶ "state"
5. Sample from **P**(X| *"Johns Hopkins is located at the state"*) ⟶ "of"
6. Sample from **P**(X| *"Johns Hopkins is located at the state of"*) ⟶ "Maryland"
7. Sample from **P**(X| *"Johns Hopkins is located at the state of Maryland"*) ⟶ "EOS"

Why Should We Care About Language Modeling?

- Language Modeling is an effective proxy for language understanding.
    - Effective ability to predict forthcoming words rely on understanding of context/prefix
- Language Modeling is a ~~subcomponent~~ superset of many NLP tasks, especially those involving text generation:
    - Summarization
    - Machine translation
    - Spelling correction
    - Dialogue etc.

You use Language Models every day!


I'll meet you at the


And now the fun starts

Brian Strope
to me
May 17 View details

They finally came through with the contract.

I expect the work to start tomorrow. Sorry for all the delays.

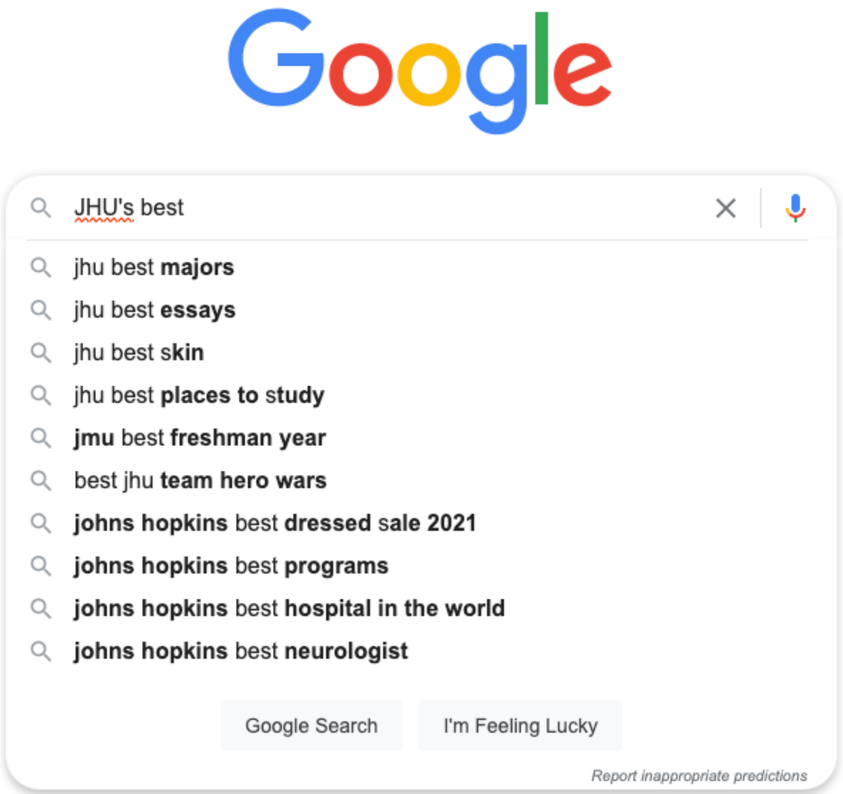
No worries, thanks for the update!
Great news, thanks for the update.
That's great news!
Reply
Reply all
Forward

Can't make it

Brian Strope
to me
May 17 View details

Ugh, I took a turn for the worst last night. I won't be able to make it to the party. Please have a great time without me.

You use Language Models every day!



JHU's best

jhu best **majors**

jhu best **essays**

jhu best **skin**

jhu best **places to study**

jmu best **freshman year**

best jhu **team hero wars**

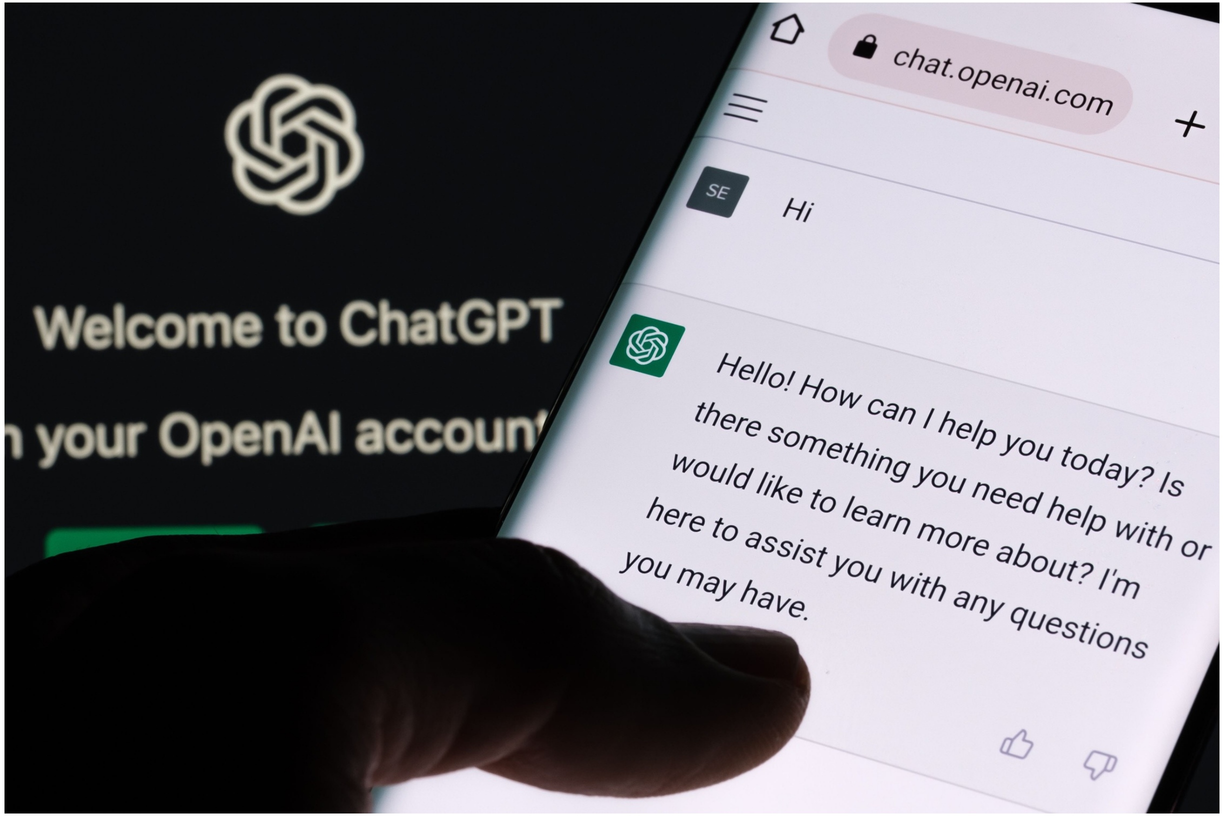
johns hopkins best **dressed sale 2021**

johns hopkins best **programs**

johns hopkins best **hospital in the world**

johns hopkins best **neurologist**

You use Language Models every day!



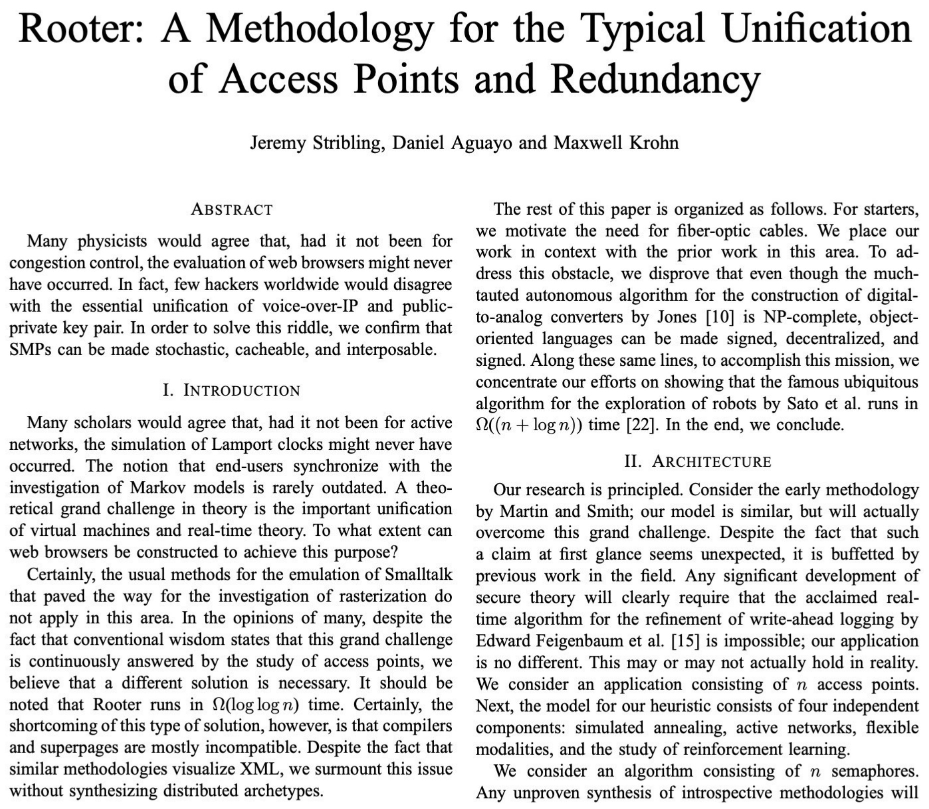
Hi

Hello! How can I help you today? Is there something you need help with or would like to learn more about? I'm here to assist you with any questions you may have.

It Can be Misused Too ...

Is this a real science article?

- A lot more about harms later in the class.



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.

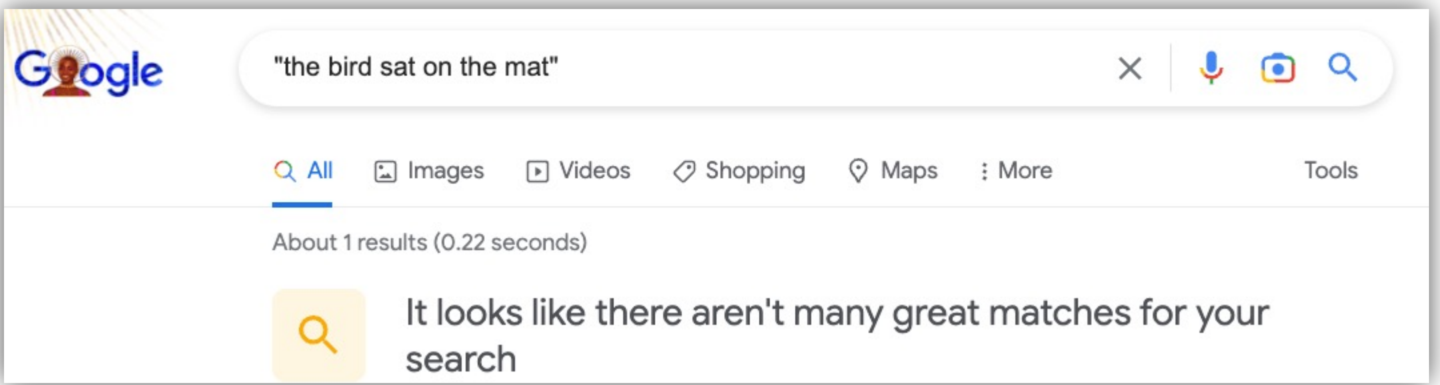
We consider an algorithm consisting of n semaphores. Any unproven synthesis of introspective methodologies will

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

$$\mathrm{P}(X_t \mid X_1, \ldots, X_{t-1})$$

How do we estimate these probabilities?
Let's just count!

$$\text{P(mat | the cat sat on the)} \approx \frac{\text{count(“the cat sat on the mat”)}}{\text{count(“the cat sat on the”)}}$$


"the bird sat on the mat"

All Images Videos Shopping Maps More Tools

About 1 results (0.22 seconds)

It looks like there aren't many great matches for your search

$$\mathbf{P}(X_t | X_1, \ldots, X_{t-1})$$

How do we estimate these probabilities?
Let's just count!

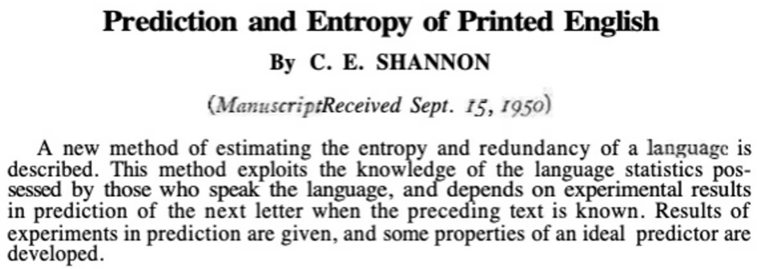
$$\text{P(mat | the cat sat on the)} = \frac{\text{count(“the cat sat on the mat”)}}{\text{count(“the cat sat on the”)}}$$

Challenge: Increasing n makes sparsity problems worse.
Typically, we can't have n bigger than 5.

Some partial solutions (e.g., smoothing and backoffs) though still an open problem.

Language Models: A History

- Shannon (1950): The predictive difficulty (entropy) of English.



Prediction and Entropy of Printed English

By C. E. SHANNON

(ManuscriptReceived Sept. 15, 1950)

A new method of estimating the entropy and redundancy of a language is described. This method exploits the knowledge of the language statistics possessed by those who speak the language, and depends on experimental results in prediction of the next letter when the preceding text is known. Results of experiments in prediction are given, and some properties of an ideal predictor are developed.

[Prediction and Entropy of Printed English, Shanon 1950]

$$\mathbf{P}(X_t \mid X_1, \ldots, X_{t-1})$$

Andrey Markov

Shannon (1950) build an approximate language model with word co-occurrences.

Markov assumptions: every node in a Bayesian network is conditionally independent of its nondescendants, given its parents.

1st order approximation:

$$\mathbf{P}(\text{mat} \mid \text{the cat sat on the}) \approx \mathbf{P}(\text{mat} \mid \overbrace{\text{the}}^{1 \text{ element}})$$

[Prediction and Entropy of Printed English, Shanon 1950]

$$\mathbf{P}(X_t \mid X_1, \ldots, X_{t-1})$$

Shannon (1950) build an approximate language model with word co-occurrences.

Markov assumptions: every node in a Bayesian network is conditionally independent of its nondescendants, given its parents.

2nd order approximation:

$$\mathbf{P}(\text{mat} \mid \text{the cat sat on the}) \approx \mathbf{P}(\text{mat} \mid \overbrace{\text{on the}}^{2 \text{ elements}})$$

[Prediction and Entropy of Printed English, Shanon 1950]

$$\mathbf{P}(X_t | X_1, \dots, X_{t-1})$$

Shannon (1950) build an approximate language model with word co-occurrences.

Markov assumptions: every node in a Bayesian network is conditionally independent of its nondescendants, given its parents.

3rd order approximation:

$$\mathbf{P}(\text{mat} \mid \text{the cat sat on the}) \approx \mathbf{P}(\text{mat} \mid \overbrace{\text{sat on the}}^{3 \text{ elements}})$$

$$\mathbf{P}(X_t | X_1, \ldots, X_{t-1})$$

Shannon (1950) build an approximate language model with word co-occurrences.

Then, we can use counts of approximate conditional probability.
Using the 3rd order approximation, we can:

$$\mathbf{P}(\text{mat} \,|\, \text{the cat sat on the}) \approx \mathbf{P}(\text{mat} \,|\, \text{sat on the}) = \frac{\text{count(“sat on the mat”)}}{\text{count(“on the mat”)}}$$

N-gram Language Models

- **Terminology:** *n*-gram is a chunk of *n* consecutive words:
  - unigrams: "cat", "mat", "sat", ...
  - bigrams: "the cat", "cat sat", "sat on", ...
  - trigrams: "the cat sat", "cat sat on", "sat on the", ...
  - four-grams: "the cat sat on", "cat sat on the", "sat on the mat", ...

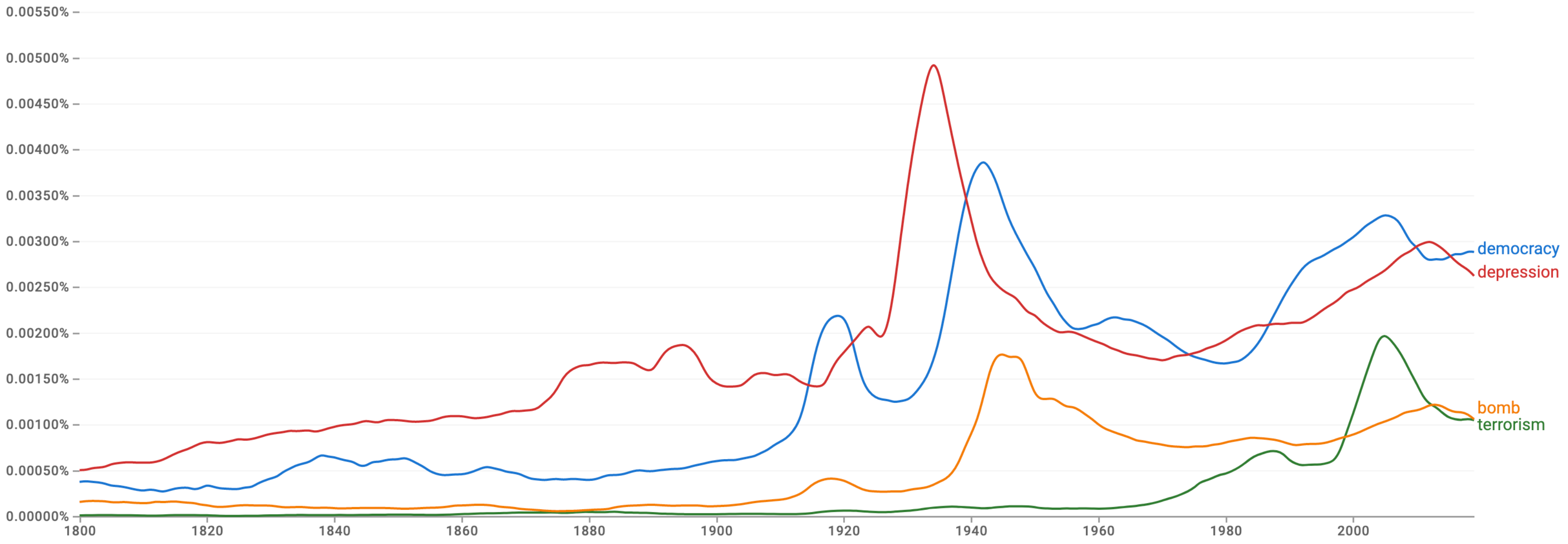
- *n*-gram language model:

$$\mathrm{P}(X_t | X_1, \ldots, X_{t-1}) \approx \mathrm{P}(X_t | \overbrace{X_{t-n+1}, \ldots, X_{t-1}}^{n-1 \text{ elements}})$$

[Prediction and Entropy of Printed English, Shanon 1950]

Pre-Computed N-Grams

Google Books Ngram Viewer

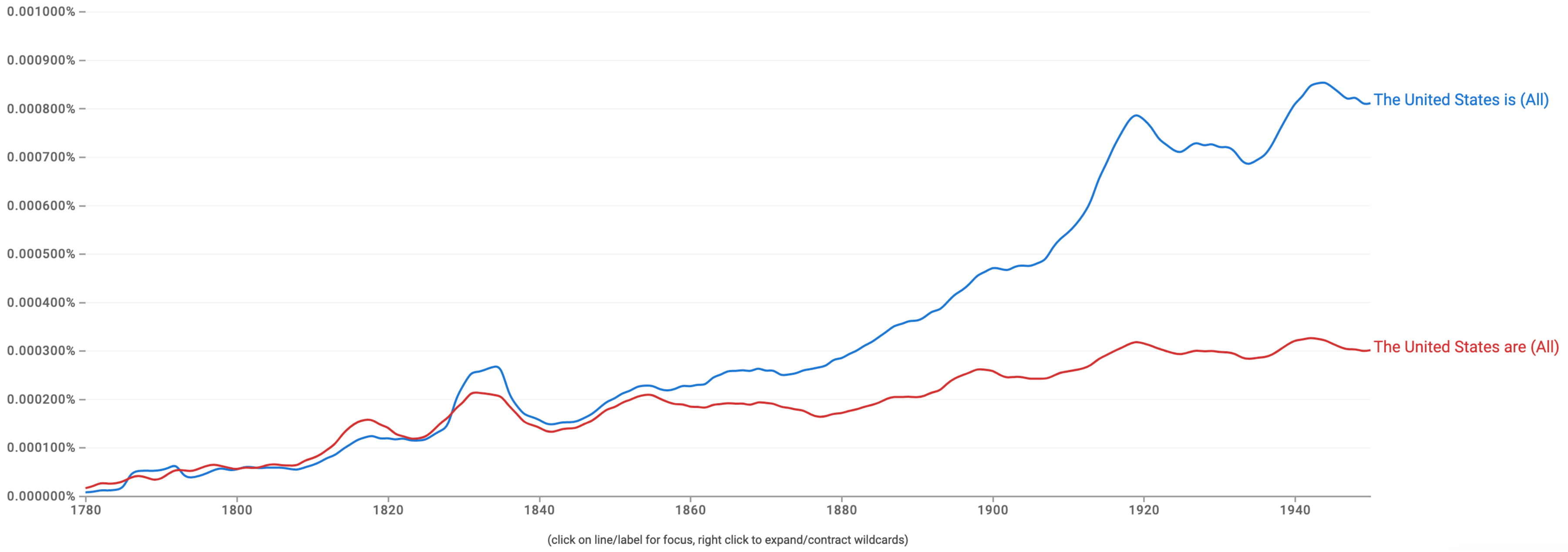


Google n-gram viewer https://books.google.com/ngrams/

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

Pre-Computed N-Grams

Google Books Ngram Viewer

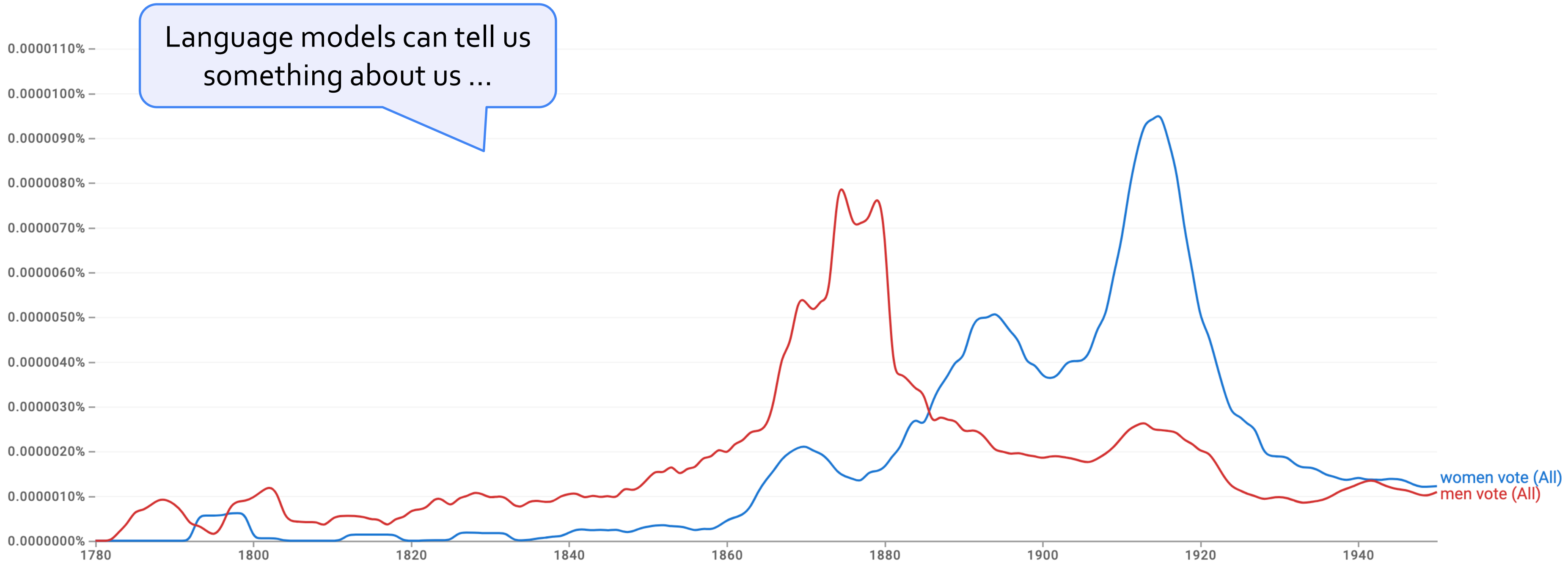


Google n-gram viewer https://books.google.com/ngrams/

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

Pre-Computed N-Grams

Google Books Ngram Viewer



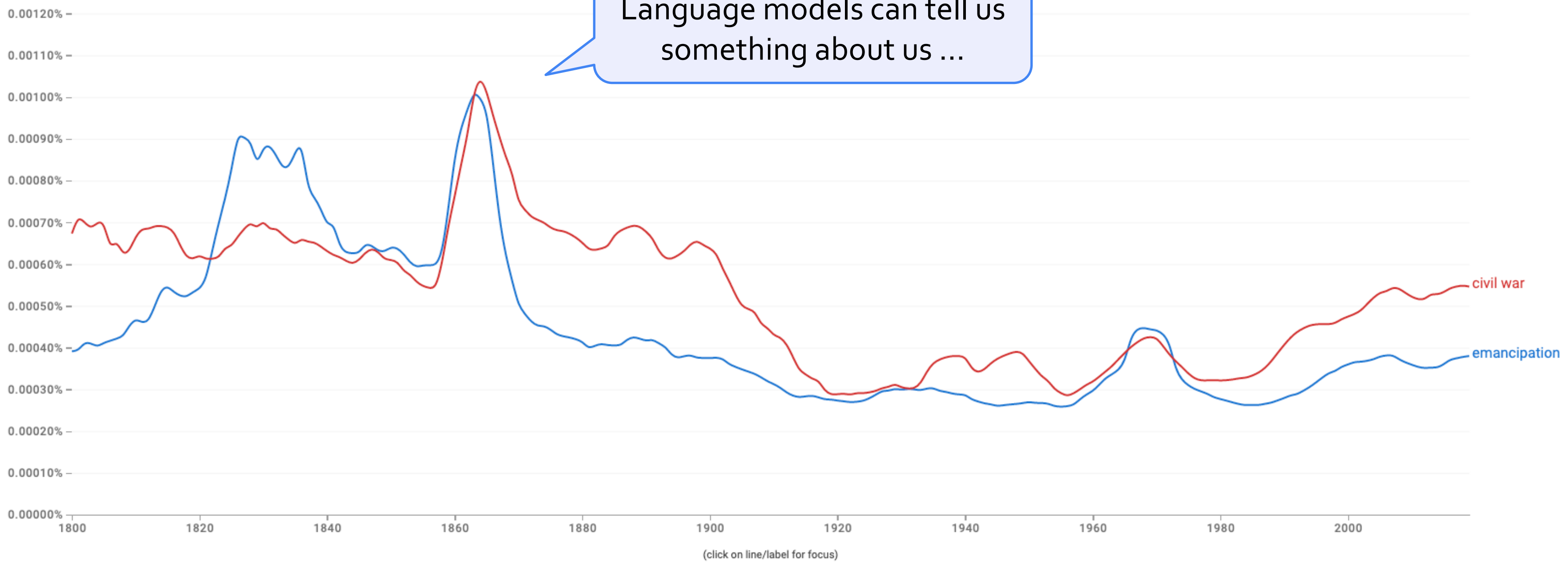
Google n-gram viewer https://books.google.com/ngrams/

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

Pre-Computed N-Grams

Google Books Ngram Viewer

Language models can tell us something about us ...



Google n-gram viewer https://books.google.com/ngrams/
Data: http://storage.googleapis.com/books/ngrams/books/datasetsv2.html