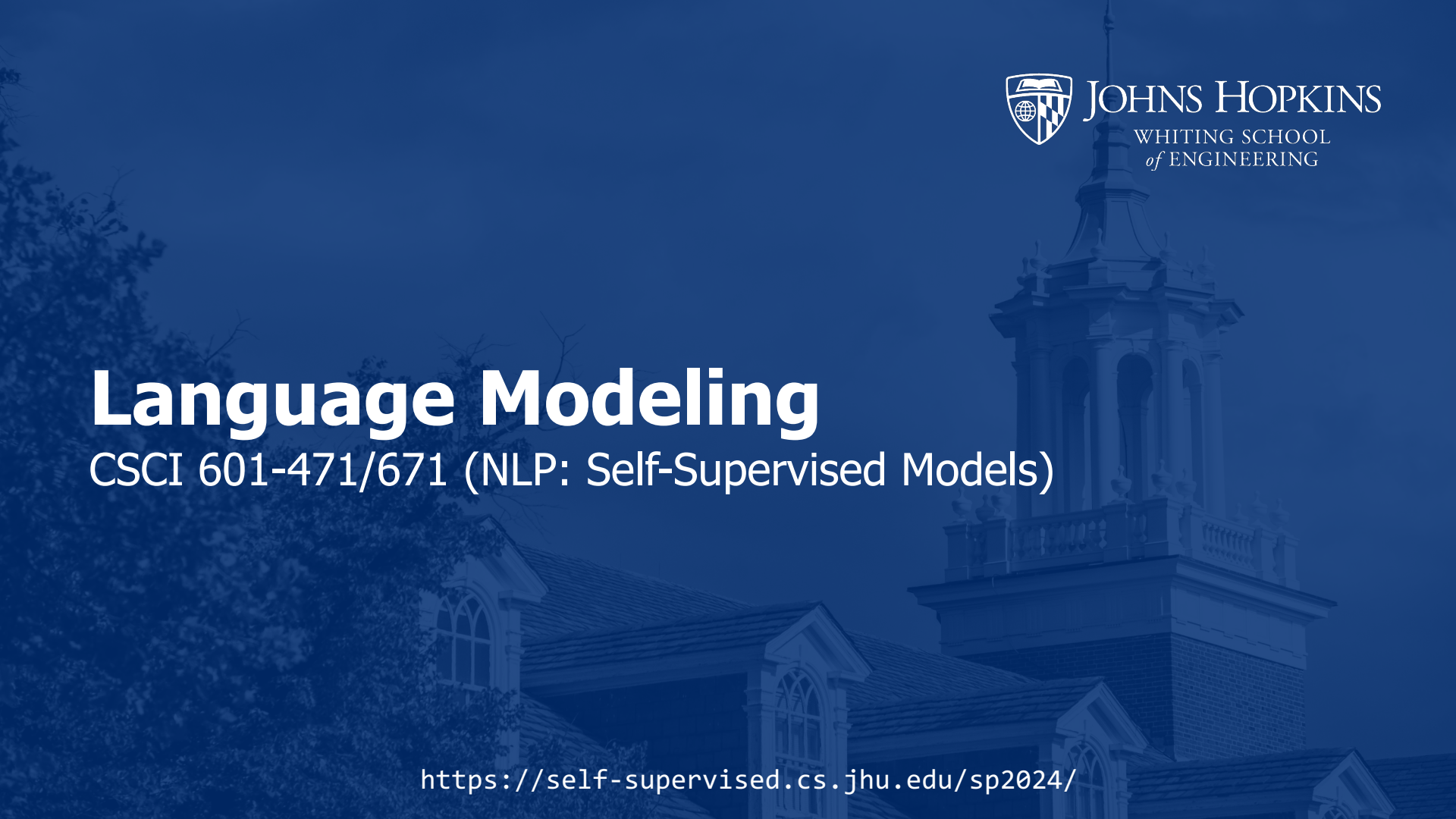


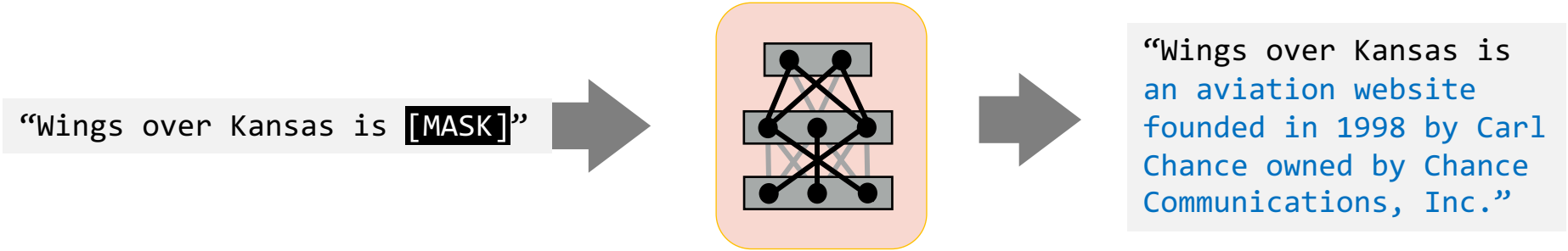
Language Modeling

CSCI 601-471/671 (NLP: Self-Supervised Models)

https://self-supervised.cs.jhu.edu/sp2024/

Recap: Self-Supervised Models

- Earlier we define **Self-Supervised** models as as predictive models of the world!



Language Modeling: Motivation

- Earlier we define **Self-Supervised** models as as predictive models of the world!
- **Language models** are self-supervised, or predictive models of language.
- How do you formulate? How do you build them?

Language Modeling: Chapter Plan

1. Language modeling: definitions and history
2. Language modeling with counting
3. Measuring language modeling quality
4. Language Modeling as a Machine Learning problem

Chapter goal — getting comfortable with the concept of "language modeling."

Language Modeling: Definitions and History

The

The cat

The cat sat

The cat sat on

The cat sat on __?__

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

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

16

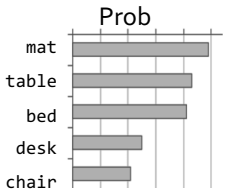
LMs as a Marginal Distribution

- Directly we train models on "marginals":

next word / context

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

"The cat sat on the [MASK]" → *Language Model* →



LMs as Implicit Joint Distribution over Language

- While language modeling involves learning the marginals, we are implicitly learning the full/joint distribution of 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"

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)

- We can rank sentences.

next word context

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

- 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)

- Can also generate strings!

next word context

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

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

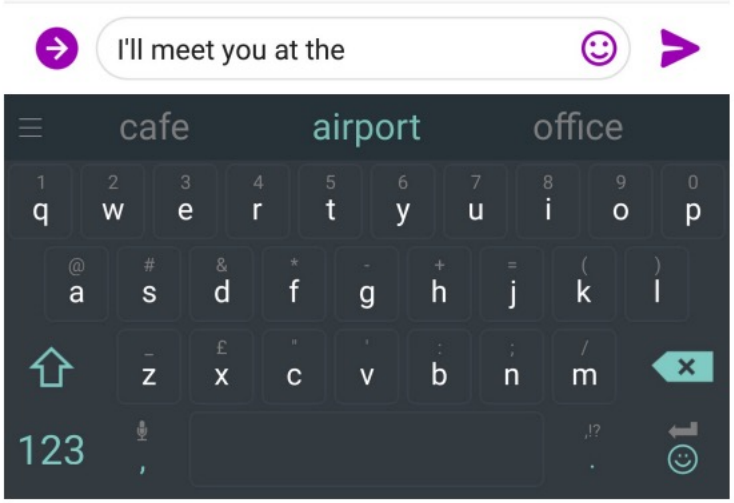
1. Sample from **P**(X|*"Johns Hopkins is "*) →"located"
2. Sample from **P**(X|*"Johns Hopkins is located"*) →"at"
3. Sample from **P**(X|*"Johns Hopkins is located at"*) →"the"
4. Sample from **P**(X|*"Johns Hopkins is located at the"*) →"state"
5. Sample from **P**(X|*"Johns Hopkins is located at 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"

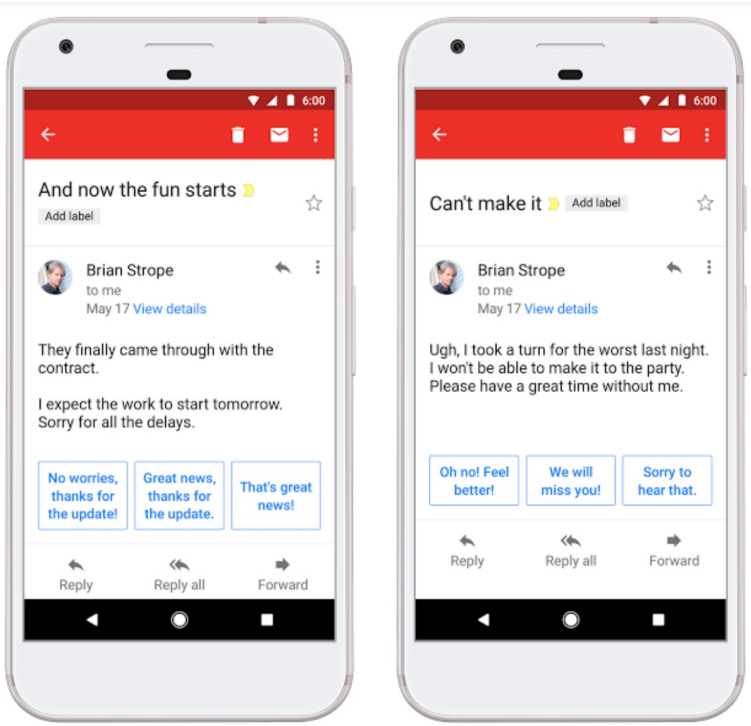
Why Care About Language Modeling?

- Language Modeling is a ~~subcomponent~~ superset of many tasks:
  - Summarization
  - Machine translation
  - Spelling correction
  - Dialogue etc.

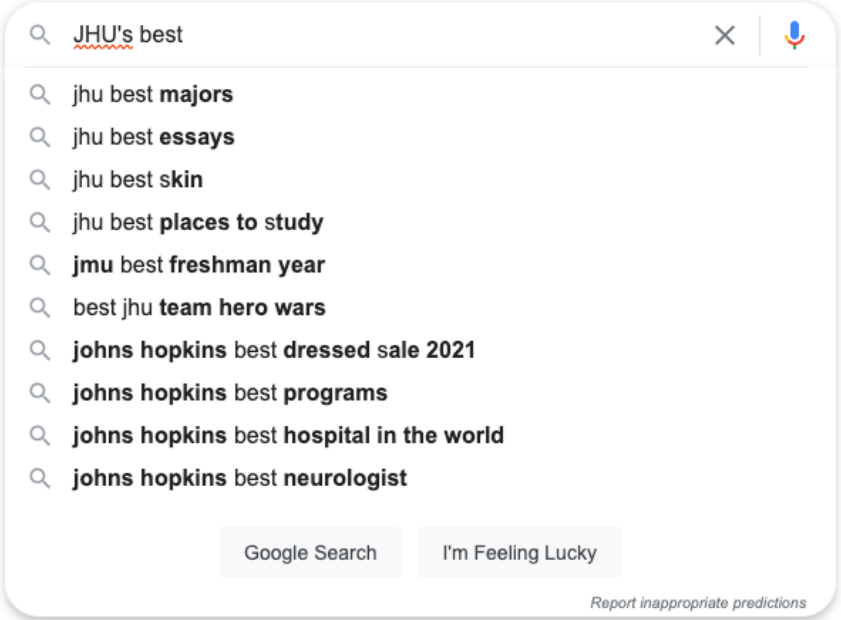
- Language Modeling is an effective proxy for language understanding.
  - Effective ability to predict forthcoming words requires on understanding of context/prefix.

You use Language Models every day!

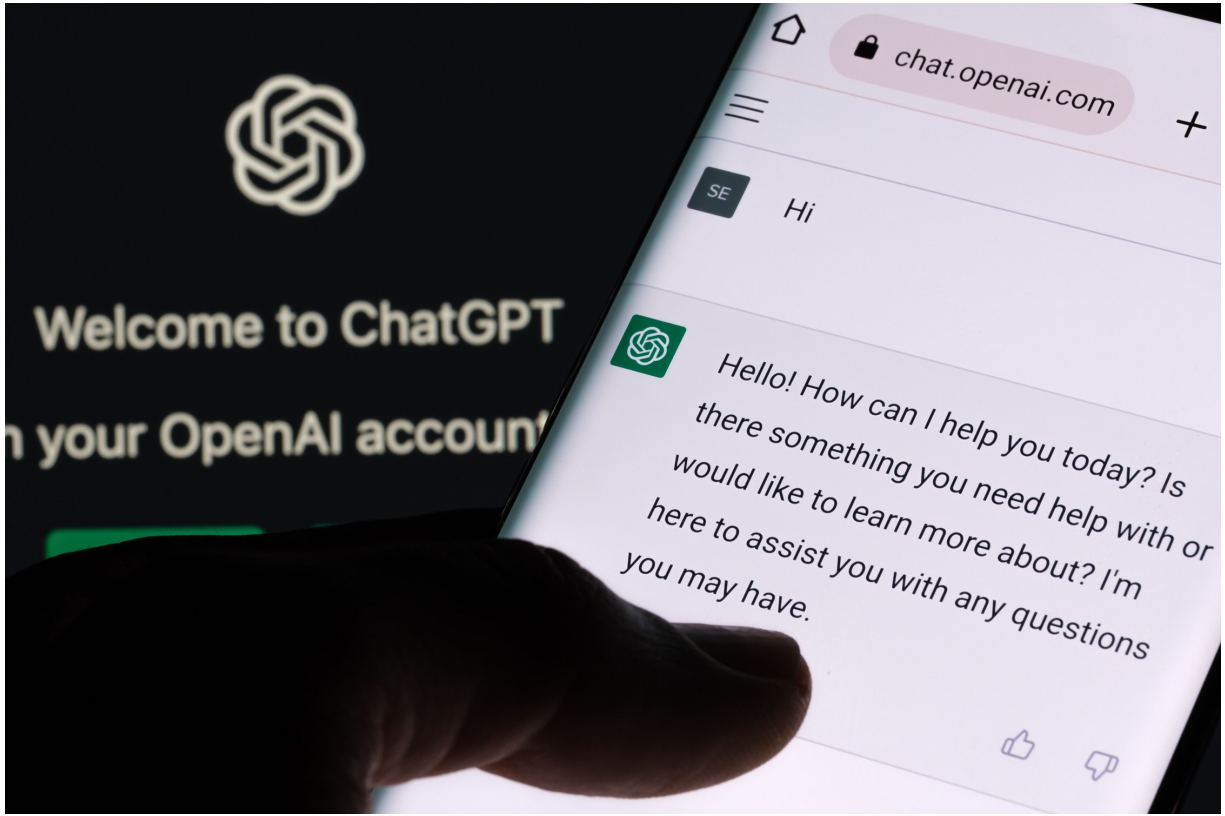




You use Language Models every day!



You use Language Models every day!



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/

Summary

- **Language modeling:** building probabilistic distribution over language.

- An accurate distribution of language enables us to solve many important tasks that involve language communication.

- **The remaining question**: how do you actually estimate this distribution?

Language Modeling with Counting

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

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

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

Count how often
"the cat sat on the mat"
has appeared in the world (internet)!

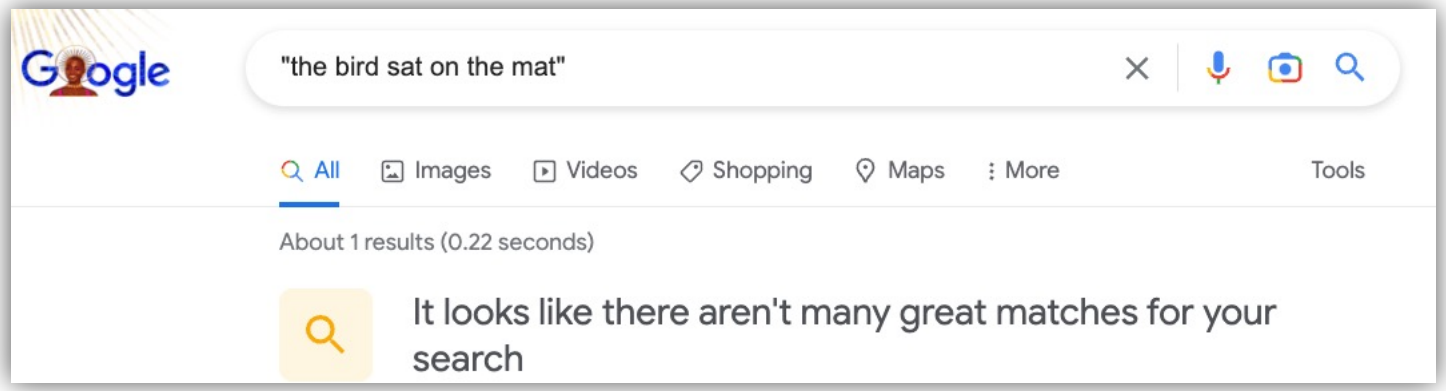
Divide that by, the count of
"the cat sat on the"
in the world (internet)!

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

How do we estimate these probabilities?

Let's just count!

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



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

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

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

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 redundancy and predictability (entropy) of English.

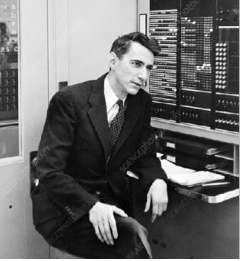
Prediction and Entropy of Printed English

By C. E. SHANNON

(*Manuscript Received 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) built an approximate language model with word co-occurrences.

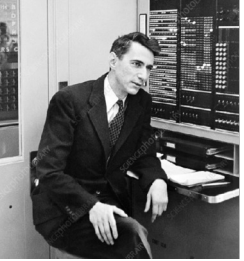
Markov assumptions: every node in a Bayesian network is conditionally independent of its non-descendants, given its parents.

1st order approximation:

1 element

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

[Prediction and Entropy of Printed English, Shanon 1950]



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

Andrey Markov

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

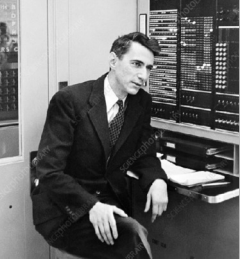
Markov assumptions: every node in a Bayesian network is conditionally independent of its non-descendants, given its parents.

2nd order approximation:

2 elements

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

[Prediction and Entropy of Printed English, Shanon 1950]



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

Andrey Markov

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

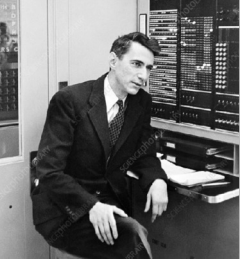
Markov assumptions: every node in a Bayesian network is conditionally independent of its non-descendants, given its parents.

3rd order approximation:

3 elements

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

[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.

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

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

[Prediction and Entropy of Printed English, Shanon 1950]

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", …

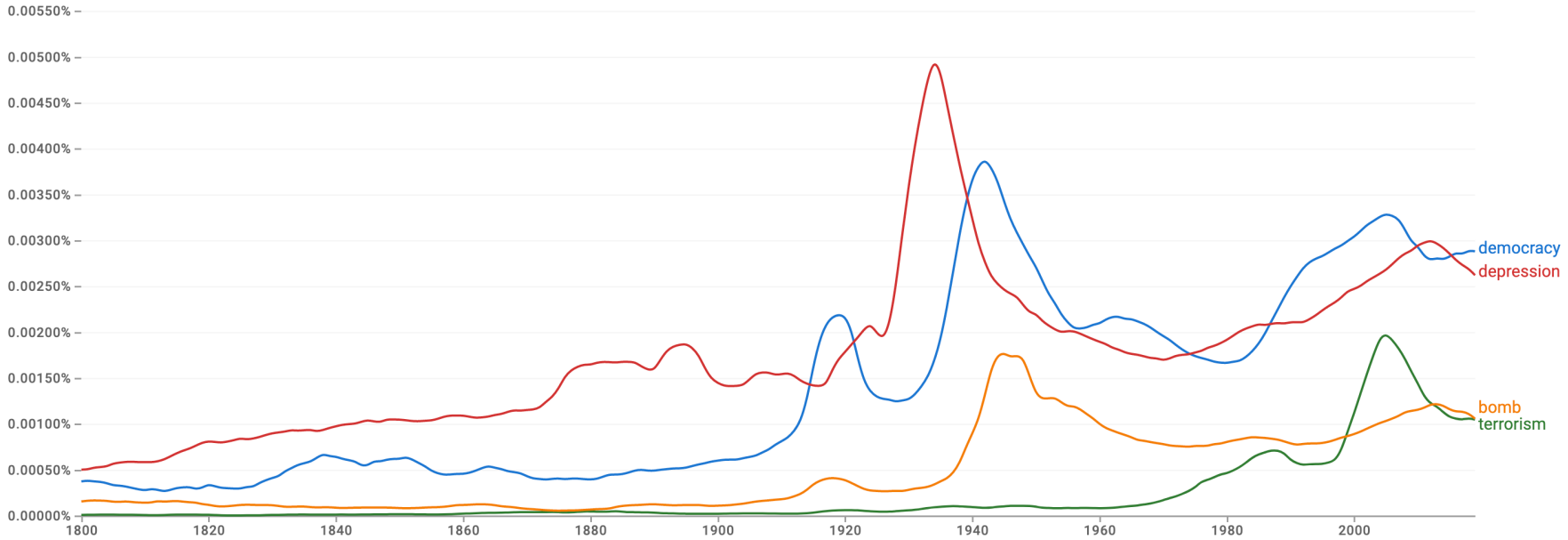
- *n*-gram language model:

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

[Prediction and Entropy of Printed English, Shanon 1950]

Pre-Computed N-Grams

Google Books Ngram Viewer

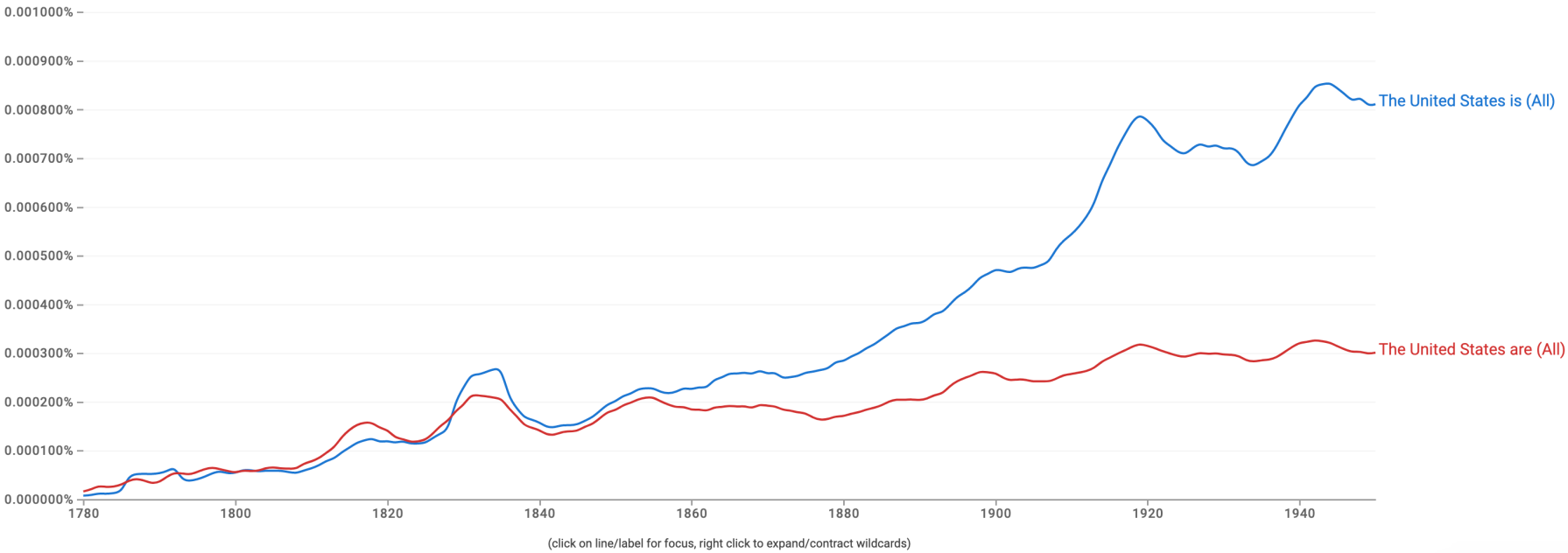


Google n-gram viewer https://books.google.com/ngrams/

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

Pre-Computed N-Grams

Google Books Ngram Viewer

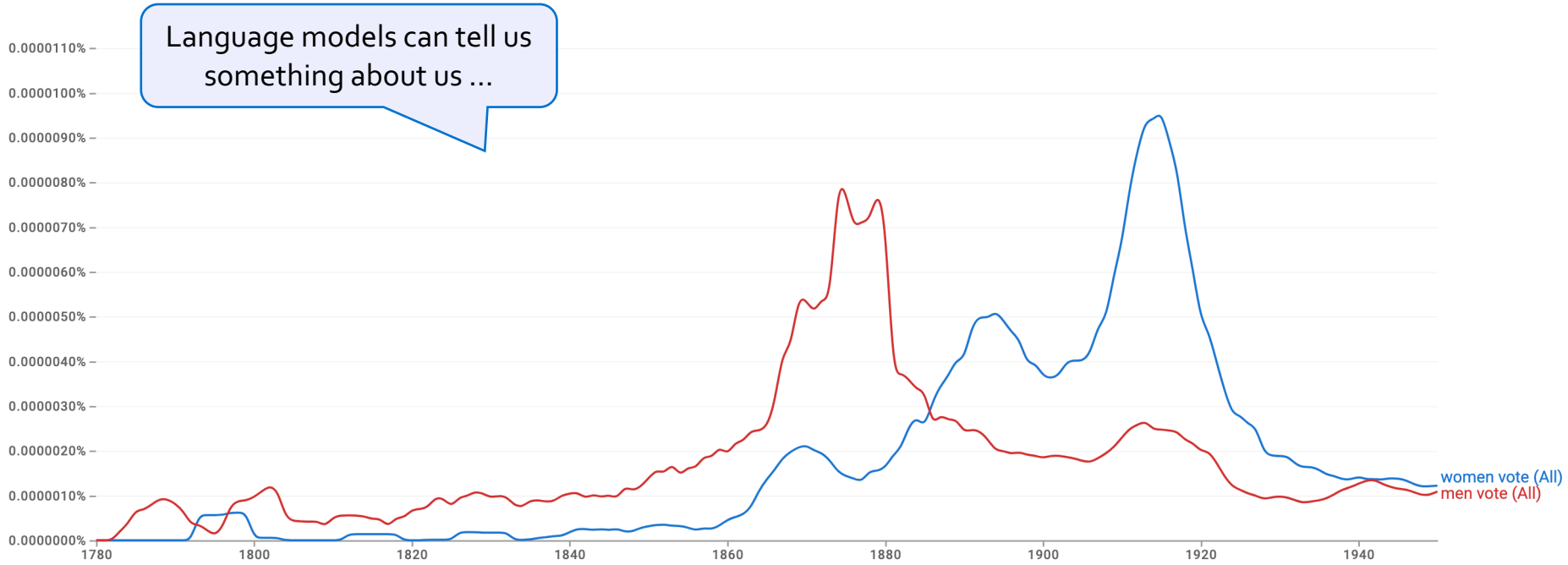


Google n-gram viewer https://books.google.com/ngrams/

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

Pre-Computed N-Grams

Google Books Ngram Viewer

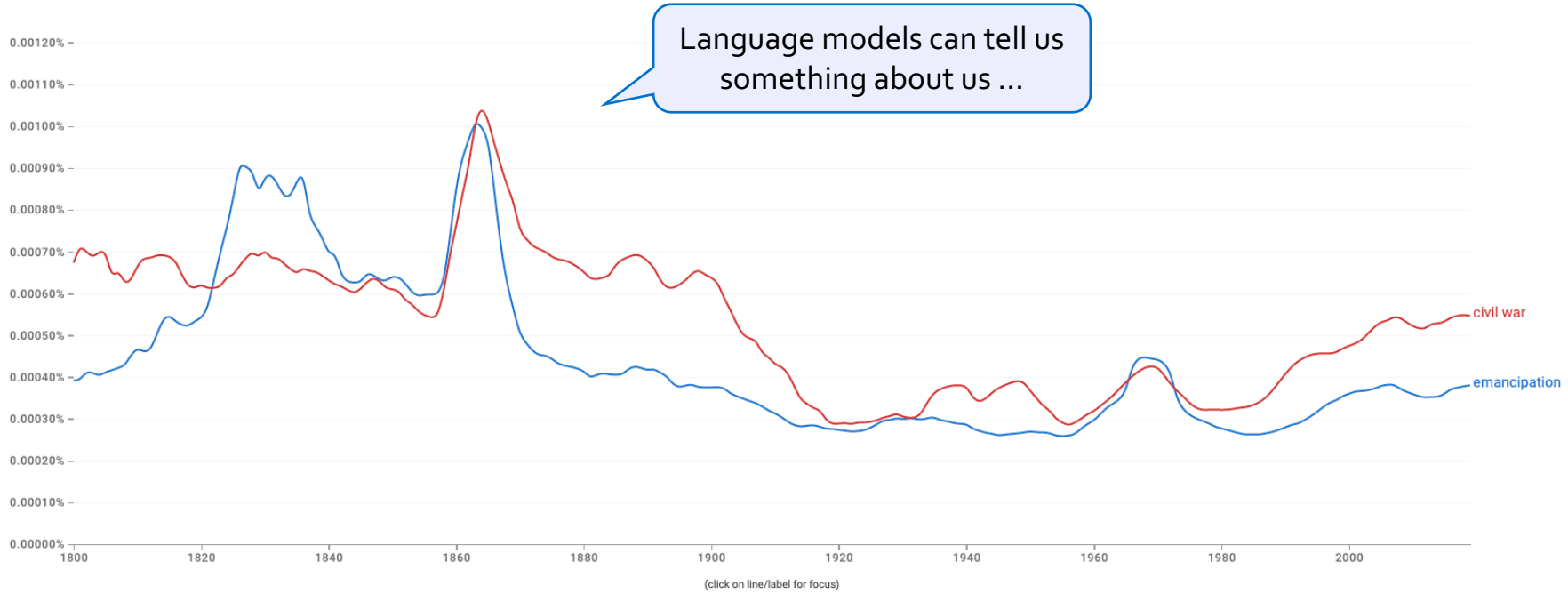


Google n-gram viewer https://books.google.com/ngrams/

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

Pre-Computed N-Grams

Google Books Ngram Viewer



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

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

Generation from N-Gram Models

- You can build a simple **tri**gram Language Model over a 1.7 million words corpus in a few seconds on your laptop*

`today the ____`

get probability distribution

```
company   0.153
bank      0.153
price     0.077
italian   0.039
emirate   0.039
...
```

Sparsity problem: not much granularity in the probability distribution

Otherwise, seems reasonable!

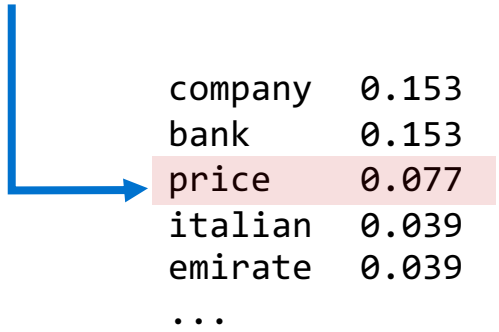
* Try for yourself: https://nlpforhackers.io/language-models/

Generation from N-Gram Models

- Now we can sample from this mode:

`today the ____`

get probability distribution



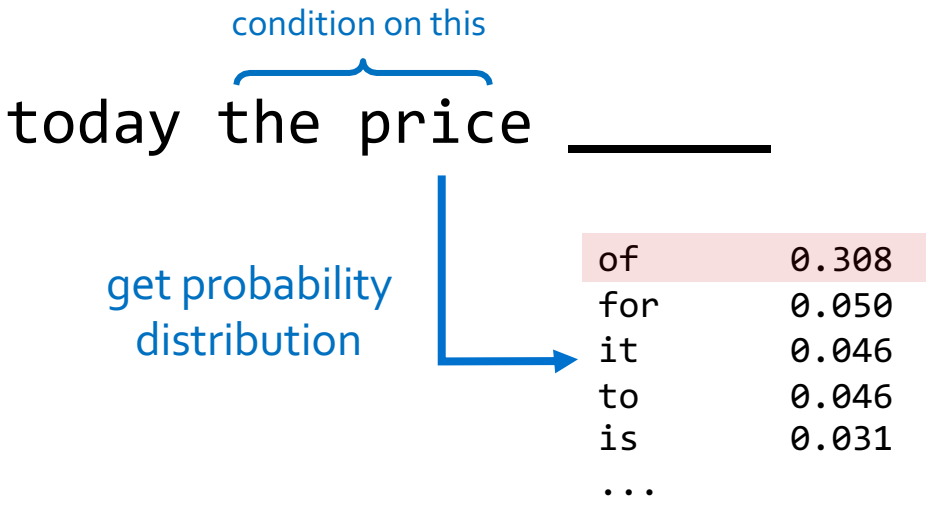
Sparsity problem: not much granularity in the probability distribution

Otherwise, seems reasonable!

* Try for yourself: https://nlpforhackers.io/language-models/

Generation from N-Gram Models

- Now we can sample from this mode:



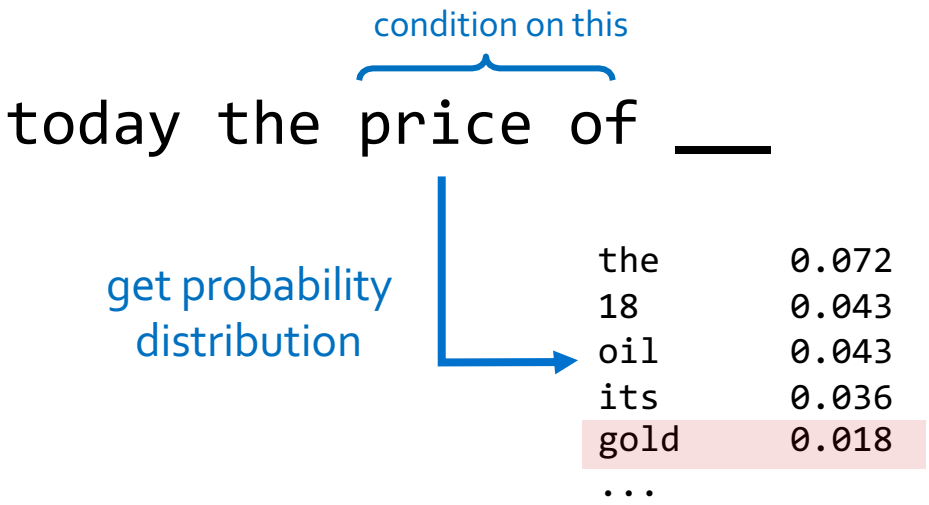
Sparsity problem: not much granularity in the probability distribution

Otherwise, seems reasonable!

* Try for yourself: https://nlpforhackers.io/language-models/

Generation from N-Gram Models

- Now we can sample from this mode:



Sparsity problem: not much granularity in the probability distribution

Otherwise, seems reasonable!

* Try for yourself: https://nlpforhackers.io/language-models/

N-Gram Models in Practice

- Now we can sample from this mode:

```
today the price of gold per ton , while production of shoe
lasts and shoe industry , the bank intervened just after it
considered and rejected an imf demand to rebuild depleted
european stocks , sept 30 end primary 76 cts a share .
```

Surprisingly grammatical!

But quite incoherent! To improve coherence, one may consider increasing larger than 3-grams, but that would worsen the sparsity problem!

* Try for yourself: https://nlpforhackers.io/language-models/

N-Gram Language Models, A Historical Highlight

"Every time I fire a linguist, the performance of the speech recognizer goes up"!!

Fred Jelinek
(1932-2010)

- Probabilistic n-gram models of text generation [Jelinek+ 1980's, …]
  - Applications: Speech Recognition, Machine Translation

Continuous Speech Recognition by Statistical Methods

FREDERICK JELINEK, FELLOW, IEEE

Abstract—Statistical methods useful in automatic recognition of continuous speech are described. They concern modeling of a speaker and of an acoustic processor, extraction of the models' statistical parameters, and hypothesis search procedures and likelihood computations of linguistic decoding. Experimental results are presented that indicate the power of the methods.

utterance models used will incorporate more grammatical features, and statistics will have been grafted onto grammatical models. Most methods presented here concern modeling of the speaker's and acoustic processor's performance and should, therefore, be universally useful.

Automatic recognition of continuous (English) speech is an

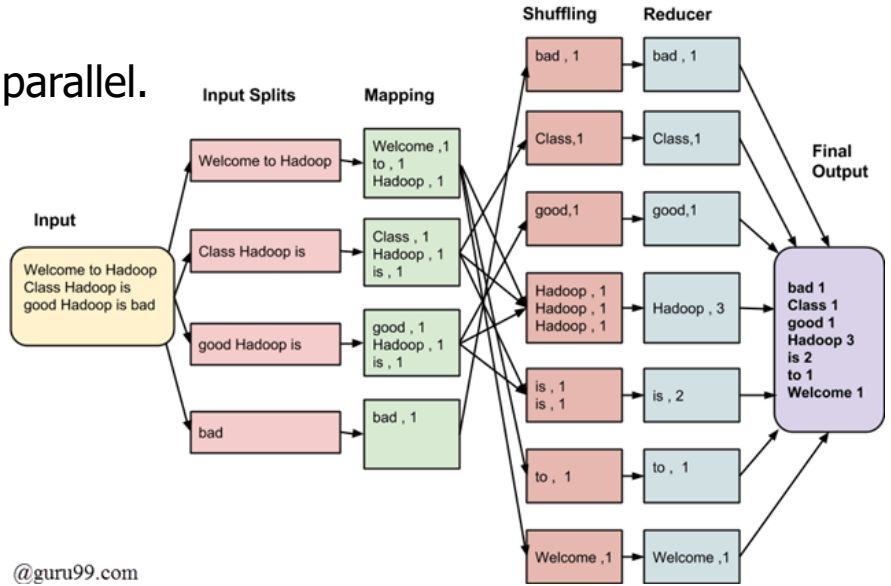
Scaling N-Grams

- Counting n-grams on the web is a computationally non-trivial task.
  - What is the cost a naïve implementation? 🤔
    - Size of the internet: k words
    - Size of your string: n words

Scaling N-Grams

- A more effective approach is distributed processing.
  - Imagine you have m CPUs (say, m =10,000)
  - Split the internet data into m shards
  - Run your n-gram search on all shards in parallel.
  - Then merge their individual results.

- A fancier version of this is using Map-Reduce framework.



Large Language Models in Machine Translation, 2007

https://github.com/stepthom/CountNGrams/blob/master/src/CountNGrams.java

Scaling N-Grams

- We can extend to trigrams, 4-grams, 5-grams, but soon we will hit is the sparsity limitations.
  - Many of these long n-grams will be zeros.

Understanding Sparsity: A Thought Experiment

- How common are zero-probabilities? 🤔
- **Example:** Shakespeare as a text corpus
  - n=884,647 tokens (the length of Shakespeare's writing),
  - |V|=29,066 (the size vocab used by Shakespeare)
  - Shakespeare produced ~300,000 bigrams
  - Out of |V|^2= 844 million possible bigrams (some of them don't make sense, but ok!)
- So, 99.96% of the possible bigrams are never seen (have zero entries in the table)

[Slide credit: Mohit Iyyer]

Scaling N-Grams

- In general, count-based LMS are insufficient models of language because language has long-distance dependencies:

"The computer which I had just put into the machine room on the fifth floor crashed."

Summary

- Learning a language model ~ learning conditional probabilities over language.
- One approach to estimating these probabilities: counting word co-occurrences.

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

- Word co-occurrences become rare for long sequences. (the sparsity issue)
- But language understanding requires long-range dependencies.
  - We need a better alternative! 😤

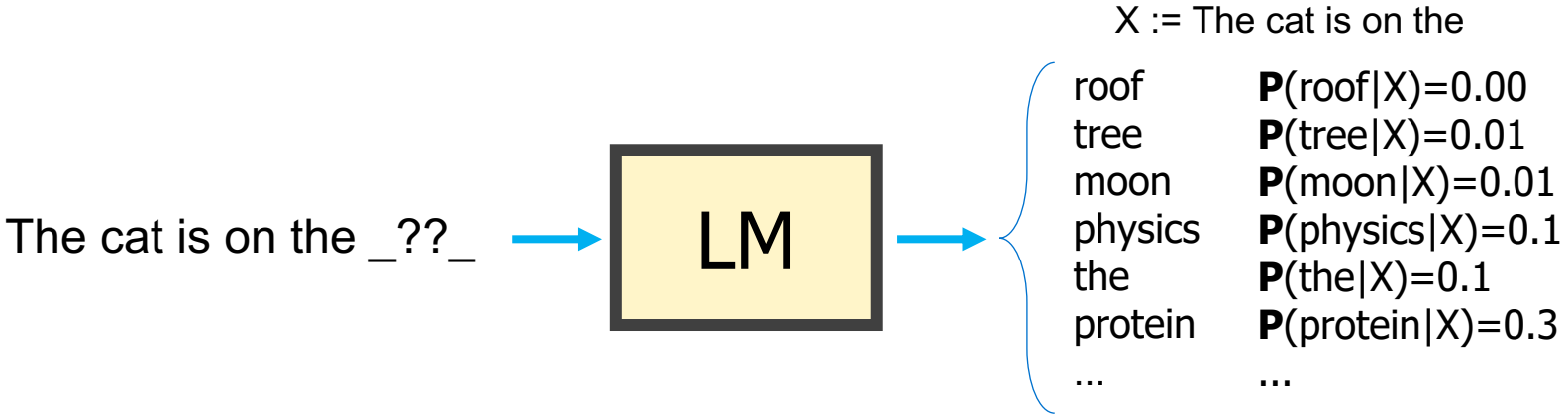
- **Next:** Measuring quality of language models.

How Good are Language Models?

[slide credit: Arman Cohan]

Large Language Models

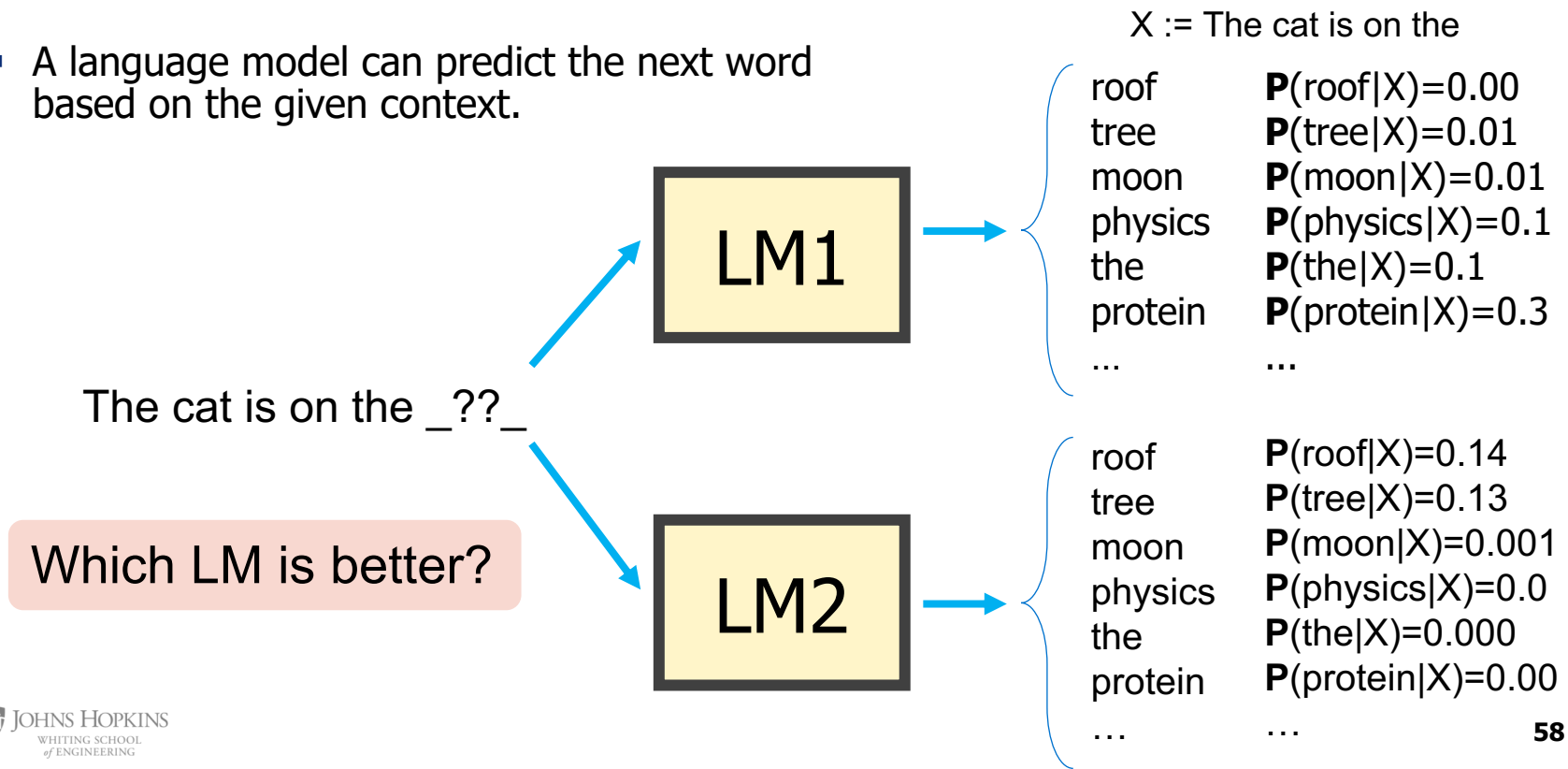
- A language model can predict the next word based on the given context.



[slide credit: Arman Cohan]

Large Language Models

- A language model can predict the next word based on the given context.



[slide inspiration: Arman Cohan]

Rank the LMs!

A

The most challenging part of the NLP: Self-supervised learning class will be able to wrap our minds around the complex concepts presented in the course. NLP is a broad and constantly evolving field, and self-supervised learning is a relatively new and advanced technique within it.

`gpt-3.5-turbo-instruct`

B

The most challenging part of the NLP: Self-supervised learning class will be reading on Japanese philosophy, which is not something that's usually possible with a Fully Funded Program like this. But, regardless of the course, we are certain it'll change you and the way you approach the world in significant ways.

`davinci-002`

C

The most challenging part of the NLP: Self-supervised learning class will be the final concept test on Saturday, June 19th at 3pm GMT. Since we hired a recent NET programer, whose brain I couldn't even understand at first, I had increased difficultly finding solution to brain teaser type questions.

`babbage-002`

A > B > C

[slide inspiration: Arman Cohan]

Rank the LMs!

What is the result of (1 + 9 - 2)*2? The answer is 10.

10 = 10.46%	
18 = 9.30%	
16 = 7.13%	
12 = 6.15%	
8 = 6.00%	

Total: -2.26 logprob on 1 tokens
(39.04% probability covered in top 5 logits)

A

davinci-002

What is the result of (1 + 9 - 2)*2? The answer is 27.

27 = 7.90%	
11 = 5.78%	
18 = 5.43%	
17 = 5.06%	
21 = 3.70%	

Total: -2.54 logprob on 1 tokens
(27.88% probability covered in top 5 logits)

C

babbage-002

What is the result of (1 + 9 - 2)*2? The answer is 16.

16 = 99.71%	
8 = 0.16%	
= 0.06%	
4 = 0.01%	
2 = 0.01%	

Total: -0.00 logprob on 1 tokens
(99.96% probability covered in top 5 logits)

B

gpt-3.5-turbo-instruct

B > A > C

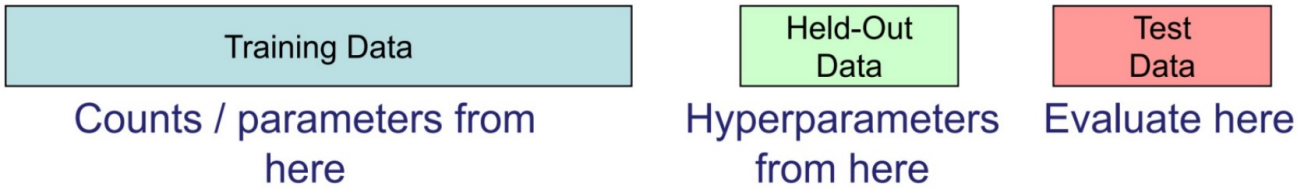
Evaluating Language Models

- 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 test the model's performance on data we haven't seen.

Evaluating Language Models

Setup:

- Train it on a suitable training documents.
- Evaluate their predictions on different, unseen documents.
- An evaluation metric tells us how well our model does on the test set.



train

count("on the mat")

Evaluating Language Models: Example

Setup:

- Train it on a suitable training documents.
- Evaluate their predictions on different, unseen documents.
- An evaluation metric tells us how well our model does on the test set.

Example: I use a bunch of New York Times articles to build a bigram probability table

train

count("on the mat")

A good language model should assign a high probability to heldout text!

eval

Now I'm going to evaluate the probability of some heldout data using our bigram table

Be Careful About Data Leakage!

Advice from a grandpa👴:

- Don't allow test sentences to leak into into training set.
- Otherwise, you will assign it an artificially high probability (==cheating).

Example: I use a bunch of New York Times articles to build a bigram probability table

train

count("on the mat")

eval

A good language model should assign a high probability to heldout text!

Now I'm going to evaluate the probability of some heldout data using our bigram table

Evaluating Language Models: Intrinsic vs Extrinsic

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

Example: I use a bunch of New York Times articles to build a bigram probability table

train

count("on the mat")

extrinsic eval

Google Translate

eval

Now I'm going to evaluate the probability of some heldout data using our bigram table

Evaluation Metric for Language Modeling: Perplexity

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

$$\text{ppl}(w_1, \ldots, w_n) = \mathbf{P}(w_1, w_2, \ldots, w_n)^{-\frac{1}{N}}$$

- A measure of predictive quality of a language model.
- Minimizing perplexity is the same as maximizing probability

Evaluation Metric for Language Modeling: Perplexity

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

$$\text{ppl}(w_1, \dots, w_n) = \mathbf{P}(w_1, w_2, \dots, w_n)^{-\frac{1}{N}}$$

- **Quiz:** let's suppose we have a sentence $w_1, \dots, w_n$ and it's fixed. Our model will correctly guess each word with probability 1/5. What is perplexity of our model?

$$\text{ppl}(w_1, \dots, w_n) = \left((1/5)^N\right)^{-\frac{1}{N}} = 5$$

Intuition: the model is indecisive among 5 choices.

Evaluation Metric for Language Modeling: Perplexity

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

$$\text{ppl}(w_1, \ldots, w_n) = \mathbf{P}(w_1, w_2, \ldots, w_n)^{-\frac{1}{N}}$$

$$= \sqrt[n]{\frac{1}{\mathbf{P}(w_1, w_2, \ldots, w_n)}}$$

$$= \sqrt[n]{\prod_{i=1}^{n} \frac{1}{\mathbf{P}(w_i | w_{<i})}}$$

(the chain rule)

Evaluation Metric for Language Modeling: Perplexity

- **Perplexity** for n-grams:

Bi-grams:

$$\text{ppl}(w_1, \ldots, w_n) = \sqrt[n]{\prod_{i=1}^{n} \frac{1}{\mathbf{P}(w_i | w_{i-1})}}$$

Tri-grams:

$$\text{ppl}(w_1, \ldots, w_n) = \sqrt[n]{\prod_{i=1}^{n} \frac{1}{\mathbf{P}(w_i | w_{i-1}, w_{i-2})}}$$

Evaluation Metric for Language Modeling: Perplexity

- In practice, we prefer to use log-probabilities
- We can rewrite perplexity formula in terms of log-probs:

$$\text{ppl}(w_1, \ldots, w_n) = 2^H, \text{ where } H = -\frac{1}{n}\sum_{i=1}^{n} \log_2 \mathbf{P}(w_i | w_1, \ldots, w_{i-1})$$

Can be interpreted as cross-entropy between LM prob and language prob

Evaluation Metric for Language Modeling: Perplexity

- In practice, we prefer to use log-probabilities
- We can rewrite perplexity formula in terms of log-probs:

$$\text{ppl}(w_1, \ldots, w_n) = 2^H, \text{ where } H = -\frac{1}{n}\sum_{i=1}^{n} \log_2 \mathbf{P}(w_i | w_1, \ldots, w_{i-1})$$

- **Quiz:** let's suppose we have a sentence $w_1, \ldots, w_n$ and it's fixed. Our model will correctly guess each word with probability 1/5. What is perplexity of our model?

$$H = -\frac{1}{n}\left[\log_2\left(\frac{1}{5}\right) + \cdots + \log_2\left(\frac{1}{5}\right)\right] = -\log\left(\frac{1}{5}\right) \Rightarrow \text{ppl(D)} = 5$$

Evaluation Metric for Language Modeling: Edge Cases

$$\text{ppl}(w_1, \ldots, w_n) = 2^H, \text{ where } H = -\frac{1}{n}\sum_{i=1}^{n} \log_2 \mathbf{P}(w_i | w_1, \ldots, w_{i-1})$$

- If $P(.)$ uninformative:
$$\forall w \in V: \mathbf{P}(w | w_{1:i-1}) = \frac{1}{|V|} \Rightarrow \text{ppl}(D) = 2^{-\frac{1}{2} n \log_2 \frac{1}{|V|}} = |V|$$

- If $P(.)$ is exact:
$$\forall w \in V: \mathbf{P}(w_i | w_{1:i-1}) = 1 \Rightarrow \text{ppl}(D) = 2^{-\frac{1}{2} n \log_2 1} = 1$$

Perplexity is a measure of model's uncertainty about next word (aka "average branching factor")

Perplexity ranges between 1 and $|V|$.

Lower perplexity == Better Model

- Training on 38 million words, test 1.5 million words, Wall Street Journal

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Lower is better

Note these evaluations are done on data that was not used for "counting." (no cheating!!)

[Mohit Iyyer]

How Should One Deal With Zeros?

$$\text{ppl}(w_1, \ldots, w_n) = \sqrt[n]{\prod_{i=1}^{n} \frac{1}{\mathbf{P}(w_i | w_{<i})}}$$

- If $\mathbf{P}(w_i | w_{<i}) = 0$, ppl would go 🤯 !! (division by zero)

How Should One Deal With Zeros?

Training set:

… denied the allegations
… denied the reports
… denied the claims
… denied the request

Test set:

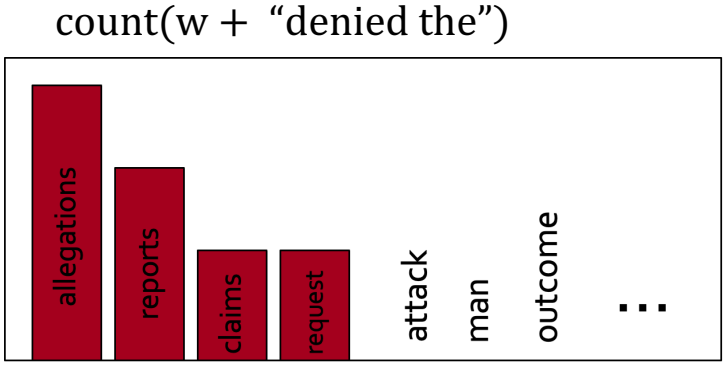
… denied the offer
… denied the load

P(offer| denied the) = o

How Should One Deal With Zeros? Smoothing

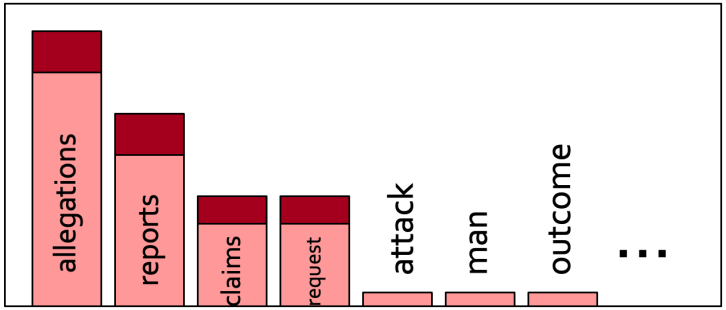
- When we have sparse statistics:

  3 allegations, 2 reports, 1 claims, 1 request = 7 total



- Steal probability mass to generalize better

  2.5 allegations, 1.5 reports, 0.5 claims, 0.5 request, 2 other = 7 total



Summary

- Language Models (LM): distributions over language

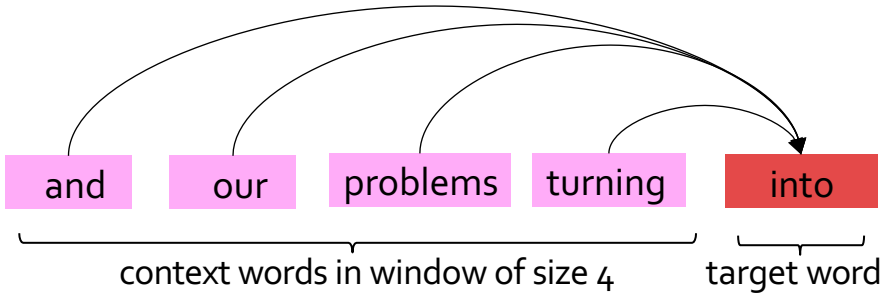
- N-gram: language modeling via counting

- Challenge with large N's: sparsity problem — many zero counts/probs.
- Challenge with small N's: not very informative and lack of long-range dependencies.

Beyond Counting:
Language Models as a Learning Problem

A Fixed-Window Neural LM

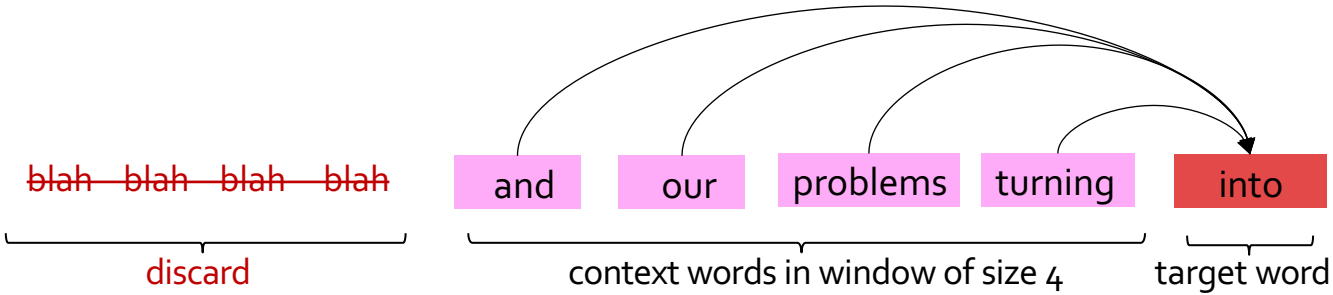
- Given the embeddings of the context, predict the word on the right side.
  - Dropping the right context for simplicity -- not a fundamental limitation.



A Fixed-Window Neural LM

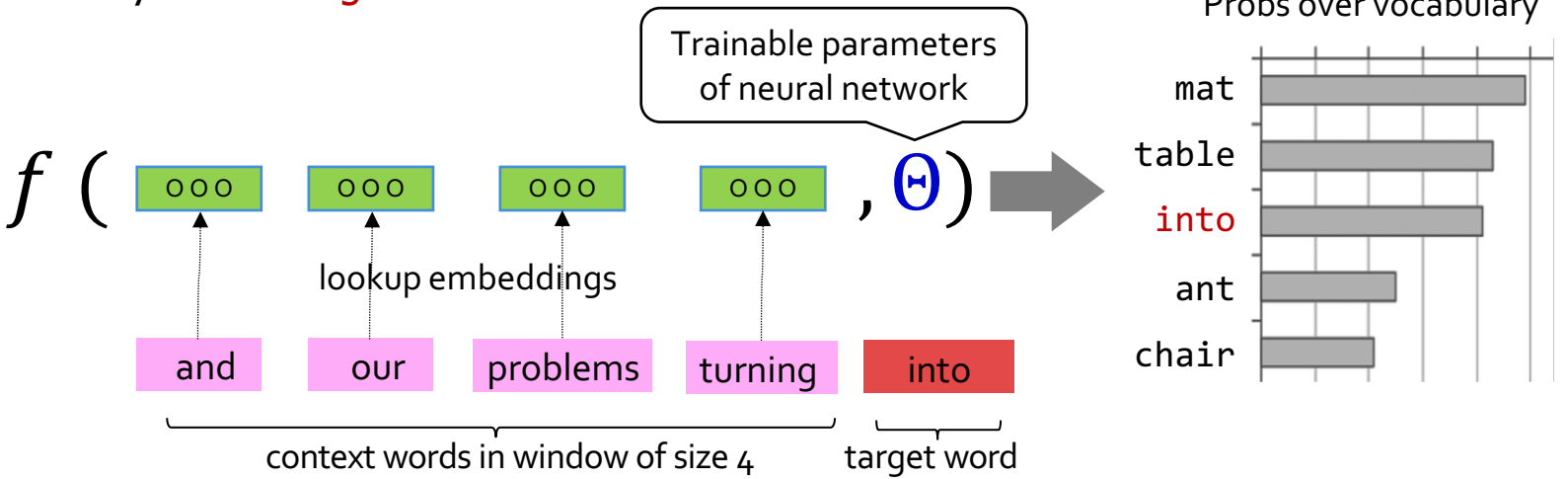
- Given the embeddings of the context, predict the word on the right side.
  - Dropping the right context for simplicity -- not a fundamental limitation.

- Discard anything beyond its context window



A Fixed-Window Neural LM

- Given the embeddings of the context, predict a target word on the right side.
  - Dropping the right context for simplicity -- not a fundamental limitation.
- Training this model is basically optimizing its parameters θ such that it assigns high probability to the target word.



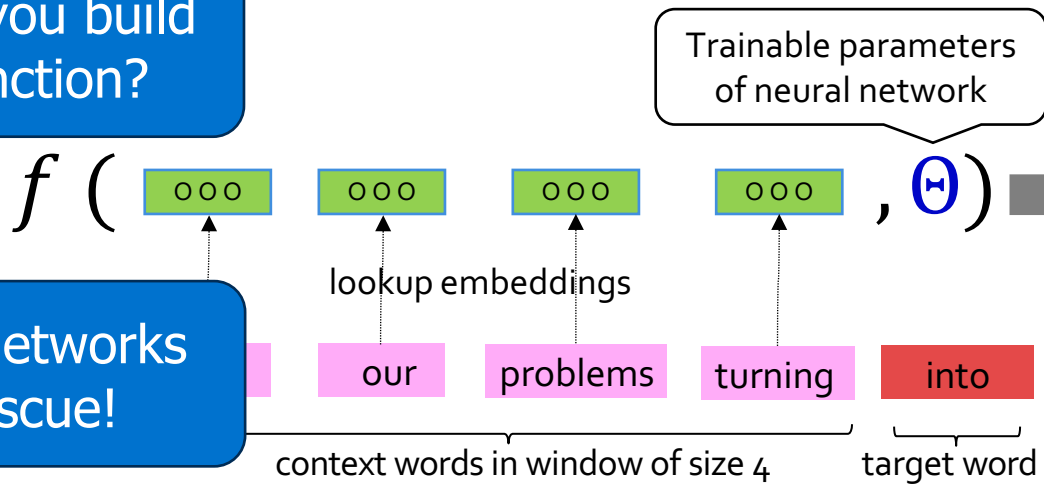
[Bengio et al. 2003]

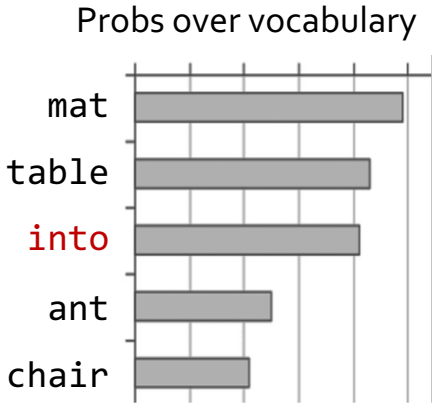
A Fixed-Window Neural LM

- This is actually a pretty good model!
- It will also lay the foundation for the future models (recurrent nets, transformers, ...)
- But first we need to figure out how to train neural networks!

How do you build this function?

Neural Networks for rescue!



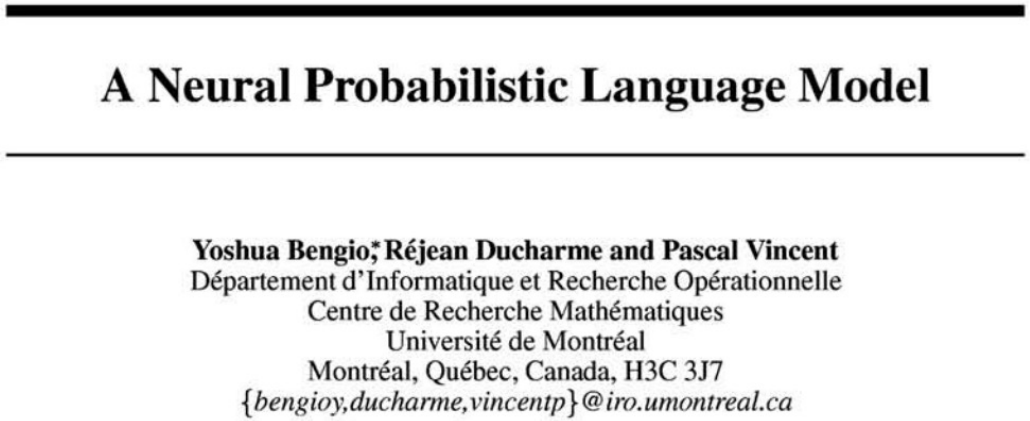


[Bengio et al. 2003]

From Counting (N-Gram) to Neural Models

- Probabilistic n-gram models of text generation [Jelinek+ 1980's, …]
  - Applications: Speech Recognition, Machine Translation
- "Shallow" statistical/neural language models (2000's) [Bengio+ 1999 & 2001, …]

NeurIPS 2000



A Neural Probabilistic Language Model

Yoshua Bengio;* Réjean Ducharme and Pascal Vincent
Département d'Informatique et Recherche Opérationnelle
Centre de Recherche Mathématiques
Université de Montréal
Montréal, Québec, Canada, H3C 3J7
{*bengioy,ducharme,vincentp*}*@iro.umontreal.ca*

Summary

- Language Modeling (LM), a useful predictive objective for language
- Perplexity, a measure of an LM's predictive ability
- N-gram models (~1980 to early 2000's),
  - Early instances of LMs
  - Difficult to scale to large window sizes
- FFN-LMs (early and mid-2000's),
  - We will need in coming sessions that one can build these models with neural networks.
  - These will be effective predictive models based on feed-forward networks