



JOHNS HOPKINS

WHITING SCHOOL
of ENGINEERING

Language Modeling

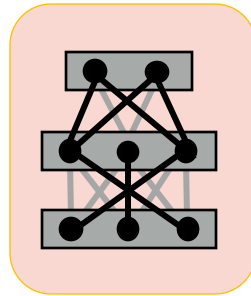
CSCI 601-471/671 (NLP: Self-Supervised Models)

<https://self-supervised.cs.jhu.edu/sp2025/>

Recap: Self-Supervised Models

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

“Wings over Kansas is [MASK]”



“Wings over Kansas is an aviation website founded in 1998 by Carl Chance owned by Chance Communications, Inc.”

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



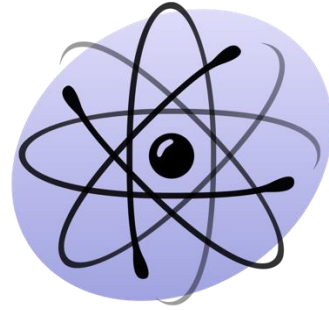
The cat sat on ___?___



The cat sat on ___?___



The cat sat on ___?___



The cat sat on ___?___

P(mat | The cat sat on the)

next word

context or prefix

Probability of Upcoming Word

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

next word context or prefix

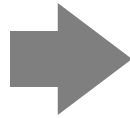
LMs as a Marginal Distribution

- Directly we train models on “marginals”:

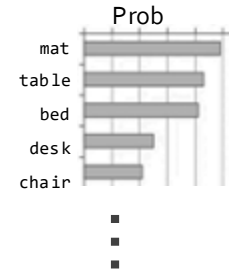
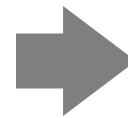
$$P(X_t | X_1, \dots, X_{t-1})$$

next word context

“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:

$$\mathbf{P}(X_1, \dots, X_t) = \mathbf{P}(X_1) \prod_{i=1}^t \mathbf{P}(X_i | X_1, X_2, \dots, X_i)$$

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

The Language Modeling Problem, Formally


- Learning to assign a probability to every sequence:
 - **Finite** vocabulary $\Sigma = \{x_1, x_2, x_3, \dots, x_n\}$
 - **Infinite** set of sequences in this language $\Sigma^* = \{x_1, x_1x_2, x_2x_3, x_2x_1x_3, x_4x_3x_1x_2, \dots\}$

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


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

Note that this is a (joint) distribution over sequences; different from (but related to) the conditional.

- Note, this statement does not specify the vocabulary (the atomic unit).
 - Example units: words, characters, bytes, etc.



Suppose we have a language model now.
What can we do with it? (how is it useful?)



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).

$$\mathbf{P}(\text{“I like Johns Hopkins University”}) = 10^{-5}$$

$$\mathbf{P}(\text{“like Hopkins I University Johns”}) = 10^{-15}$$

Doing Things with Language Model (2)

- We can rank sentences.

$$\mathbf{P}(X_t \mid \overbrace{X_1, \dots, X_{t-1}}^{\text{context}})$$

Diagram illustrating the probability function $\mathbf{P}(X_t \mid X_1, \dots, X_{t-1})$. The term X_t is labeled "next word" and the sequence X_1, \dots, X_{t-1} is labeled "context".

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

$\mathbf{P}(\text{"I like Johns Hopkins University."}) > \mathbf{P}(\text{"I like John Hopkins University"})$

$\mathbf{P}(\text{"I like Johns Hopkins University."}) > \mathbf{P}(\text{"University. I Johns EOS Hopkins like"})$

$\mathbf{P}(\text{"JHU is located in Baltimore."}) > \mathbf{P}(\text{"JHU is located in Virginia."})$

Doing Things with Language Model (3)

- Can also generate strings!

$$\text{P}(X_t \mid \overbrace{X_1, \dots, X_{t-1}}^{\text{context}})$$

next word

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

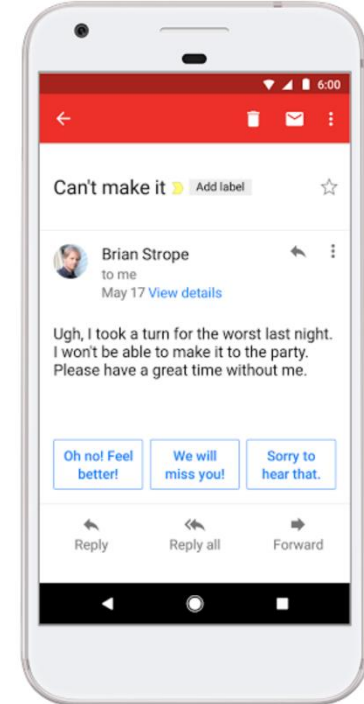
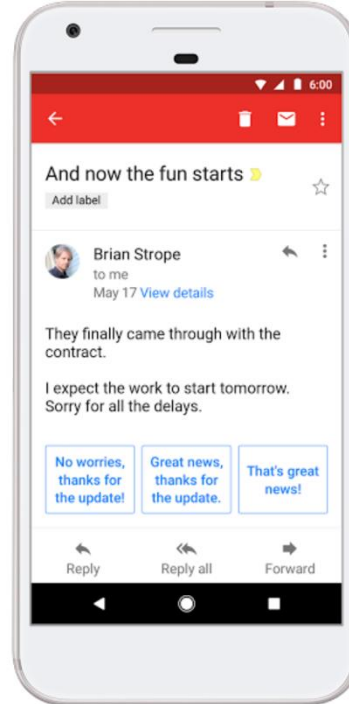
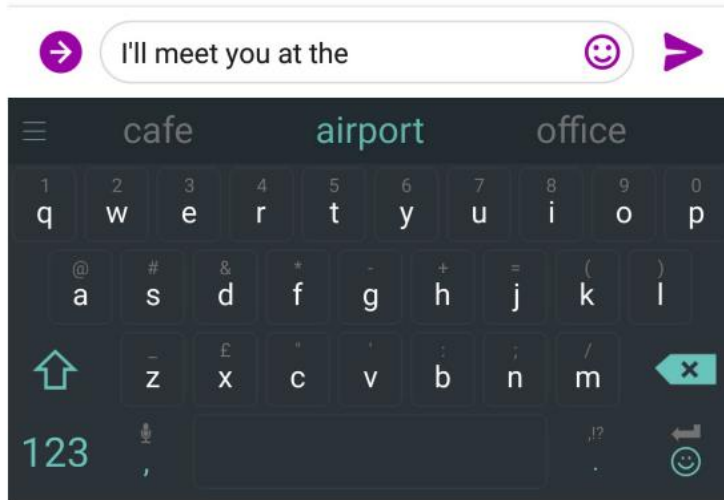
- Sample from $\text{P}(X \mid \text{"Johns Hopkins is "}) \rightarrow \text{"located"}$
- Sample from $\text{P}(X \mid \text{"Johns Hopkins is located"}) \rightarrow \text{"at"}$
- Sample from $\text{P}(X \mid \text{"Johns Hopkins is located at"}) \rightarrow \text{"the"}$
- Sample from $\text{P}(X \mid \text{"Johns Hopkins is located at the"}) \rightarrow \text{"state"}$
- Sample from $\text{P}(X \mid \text{"Johns Hopkins is located at the state"}) \rightarrow \text{"of"}$
- Sample from $\text{P}(X \mid \text{"Johns Hopkins is located at the state of"}) \rightarrow \text{"Maryland"}$
- Sample from $\text{P}(X \mid \text{"Johns Hopkins is located at the state of Maryland"}) \rightarrow \text{"EOS"}$

Notice this special word to indicate end of sentences

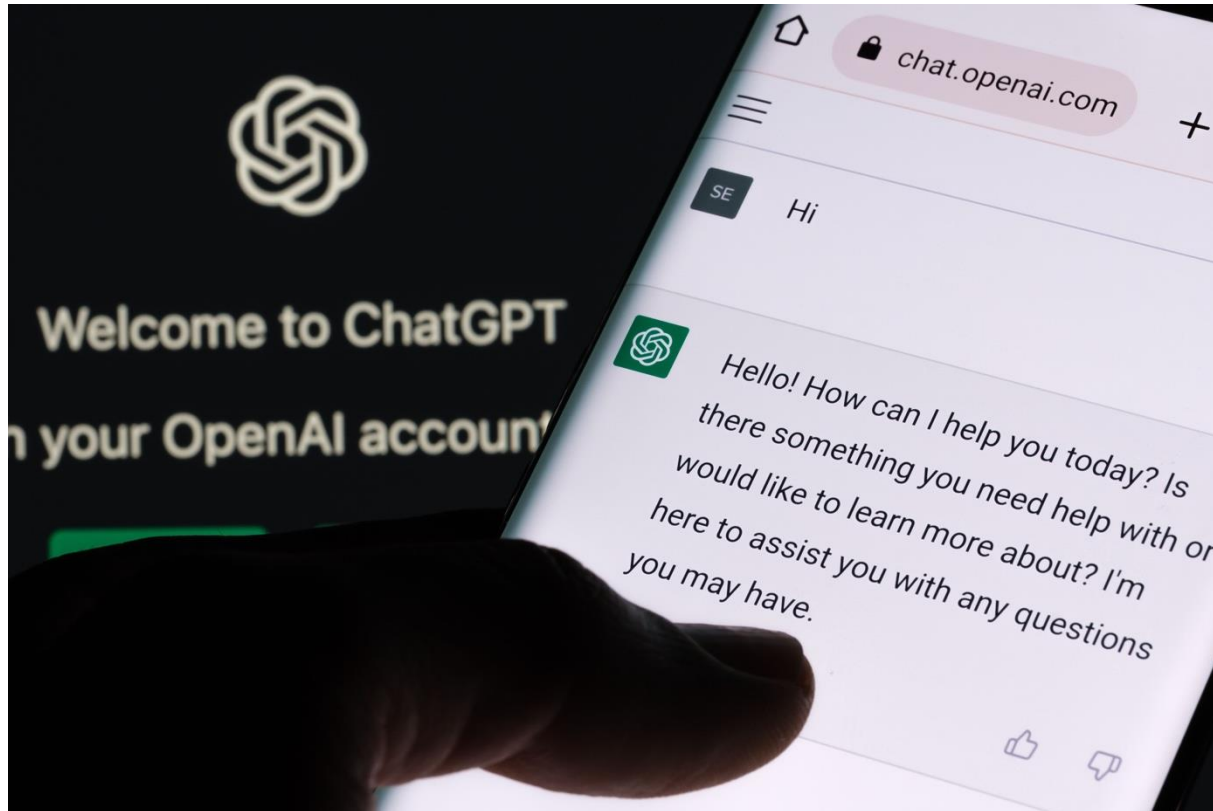
Why Care About Language Modeling?

- 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!



Summary

- **Language modeling:** building probabilistic distribution over language.
- An accurate distribution of language enables us to solve many important tasks that involve language communication.
- **Next question:** how do you actually estimate this distribution?

Language Modeling with Counting

LMs as a Marginal Distribution

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

Diagram illustrating the components of the probability distribution:

- The term X_t is labeled "next word" with a blue bracket above it.
- The terms X_1, \dots, X_{t-1} are collectively labeled "context" with a blue bracket above them.

- Now the question is, how to estimate this distribution.

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

How do we estimate these probabilities?

Let's just count!

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

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

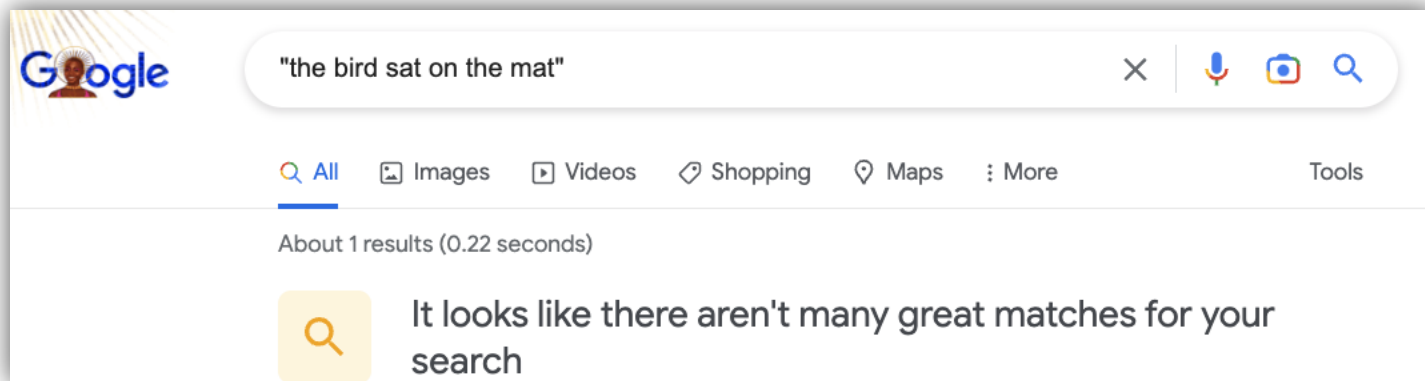
Divide that by, the count of
"the cat sat on the"
in the world (internet)!

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

How do we estimate these probabilities?

Let's just count!

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



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

How do we estimate these probabilities?

Let's just count!

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

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.

Understanding Sparsity: A Thought Experiment



- Consider Shakespeare's writing.
- Say, we want the count of **any word-pair** that Shakespeare knows
- The size vocab used by Shakespeare: $|V|=29,066$
- The number of potential word-pairs $29,066 \times 29,066 \sim 844$ million
 - (some of them don't make sense, but ok!)
- Of this, Shakespeare has used $\sim 300K$ word-pair combinations in his writings!
- So, **99.96%** of the possible bigrams are **never seen** (hence, have zero entries for bigram counts).

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.



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



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:

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

1 element



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



Andrey Markov

2nd order approximation:

$$\mathbf{P}(\text{mat} | \text{the cat sat on the}) \approx \mathbf{P}(\text{mat} | \underbrace{\text{on the}}_{2 \text{ elements}})$$

1st order approximation:

$$\mathbf{P}(\text{mat} | \text{the cat sat on the}) \approx \mathbf{P}(\text{mat} | \underbrace{\text{the}}_{1 \text{ element}})$$



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



Andrey Markov

3rd order approximation:

$$\mathbf{P}(\text{mat} | \text{the cat sat on the}) \approx \mathbf{P}(\text{mat} | \underbrace{\text{sat on the}}_{3 \text{ elements}})$$

2nd order approximation:

$$\mathbf{P}(\text{mat} | \text{the cat sat on the}) \approx \mathbf{P}(\text{mat} | \underbrace{\text{on the}}_{2 \text{ elements}})$$

1st order approximation:

$$\mathbf{P}(\text{mat} | \text{the cat sat on the}) \approx \mathbf{P}(\text{mat} | \underbrace{\text{the}}_{1 \text{ element}})$$



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



Andrey Markov

3rd order approximation:

$$\mathbf{P}(\text{mat} | \text{the cat sat on the}) \approx \mathbf{P}(\text{mat} | \underbrace{\text{sat on the}}_{\text{3 elements}})$$

Then, we can use counts of approximate conditional probability.


Using the 3rd order approximation, we can:

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


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:

$$P(X_t | X_1, \dots, X_{t-1}) \approx P(X_t | \overbrace{X_{t-n+1}, \dots, X_{t-1}}^{n-1 \text{ elements}}) = \frac{\text{count}(X_{t-n+1}, \dots, X_{t-1}, X_t)}{\text{count}(X_{t-n+1}, \dots, X_{t-1})}$$



How easy is it to build
an n-gram language model?



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*

Generation from N-Gram Models

- You can build a simple **trigram** 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!

Generation from N-Gram Models

- Now we can sample from this mode:

today the _____

get probability
distribution

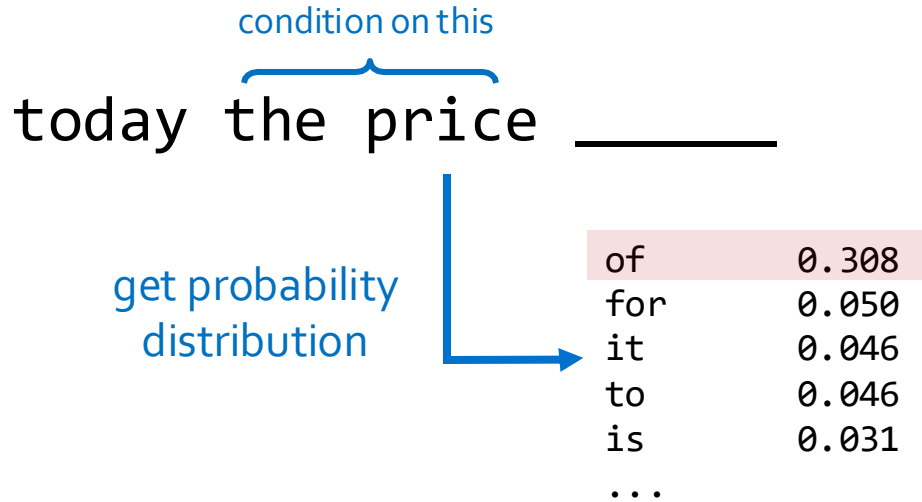
company	0.153
bank	0.153
price	0.077
italian	0.039
emirate	0.039
...	

Sparsity problem: not
much granularity in the
probability distribution

Otherwise, seems reasonable!

Generation from N-Gram Models

- Now we can sample from this mode:

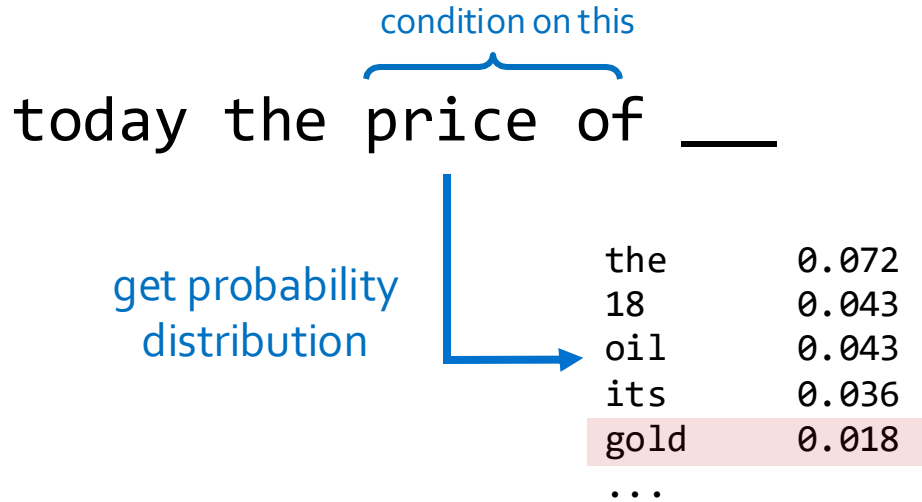


Sparsity problem: not much granularity in the probability distribution

Otherwise, seems reasonable!

Generation from N-Gram Models

- Now we can sample from this mode:



Sparsity problem: not much granularity in the probability distribution

Otherwise, seems reasonable!

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!**

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

532

PROCEEDINGS OF THE IEEE, VOL. 64, NO. 4, APRIL 1976

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

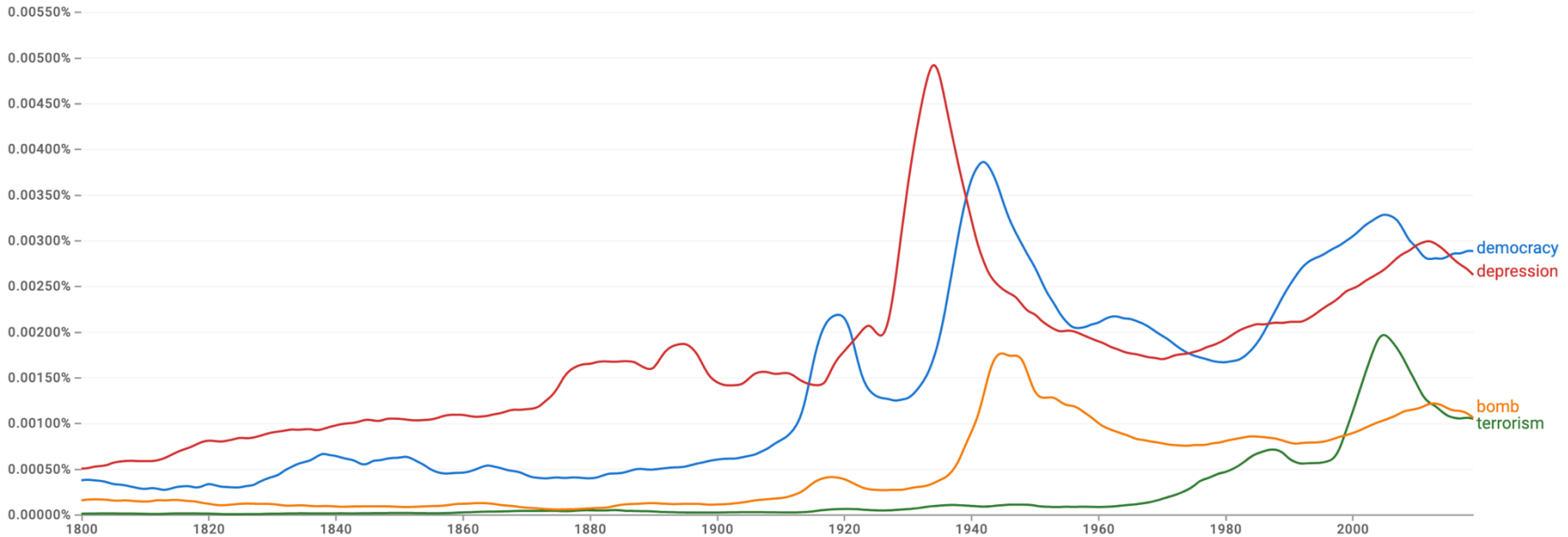


n-gram language models are enough
to give us many interesting insights!



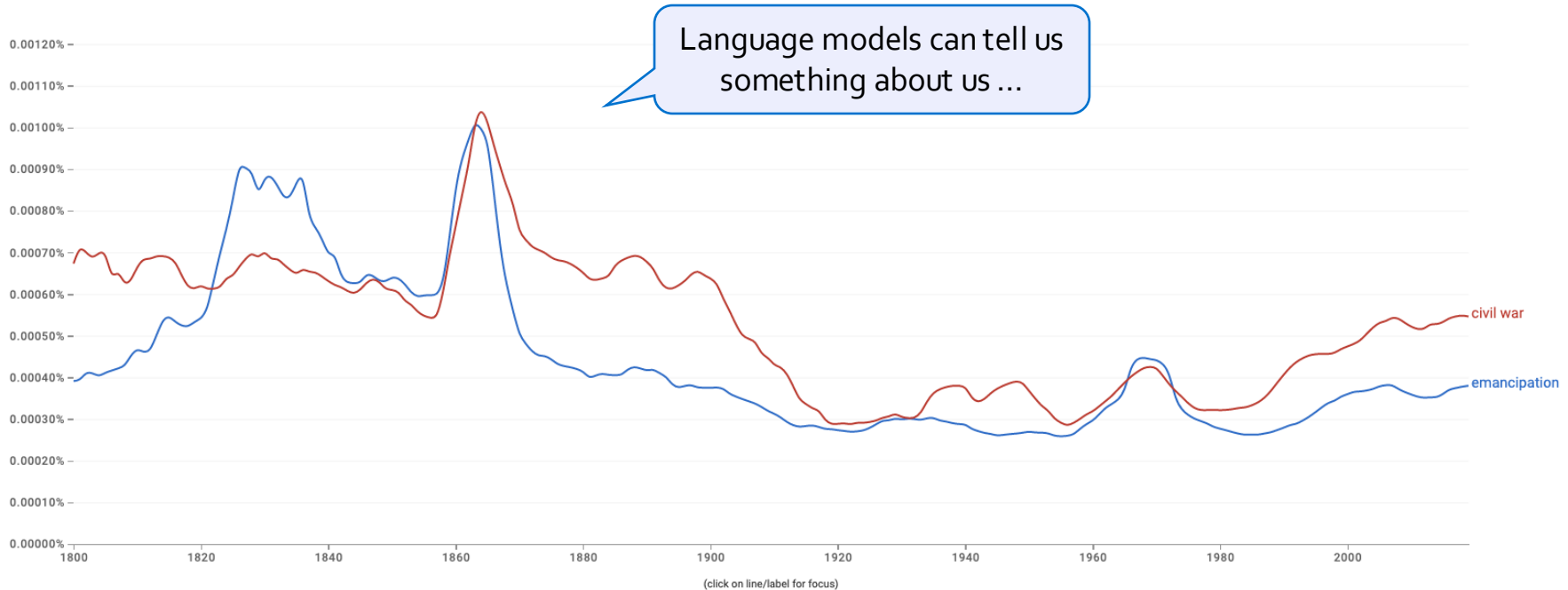
Pre-Computed N-Grams

Google Books Ngram Viewer



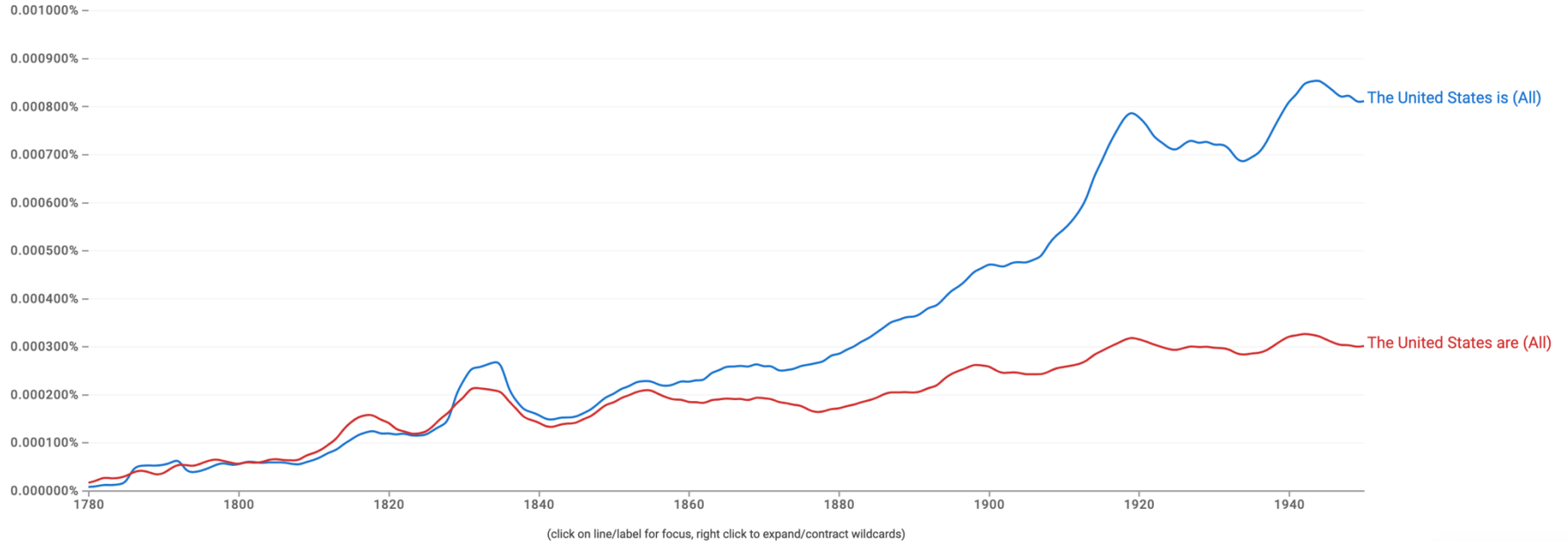
Pre-Computed N-Grams

Google Books Ngram Viewer



Pre-Computed N-Grams

Google Books Ngram Viewer



Limits of N-Grams LMs: Long-range Dependencies

- 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.**”

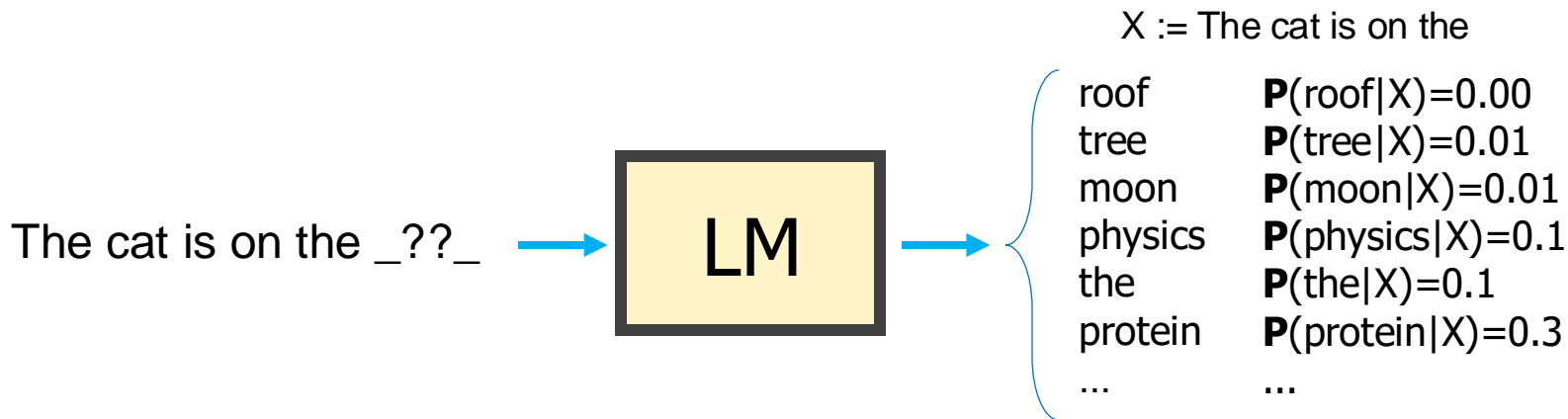
Summary

- Learning a language model \sim learning **conditional probabilities** over language.
- One approach to estimating these probabilities: **counting word co-occurrences**.
- Challenges:
 - 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?

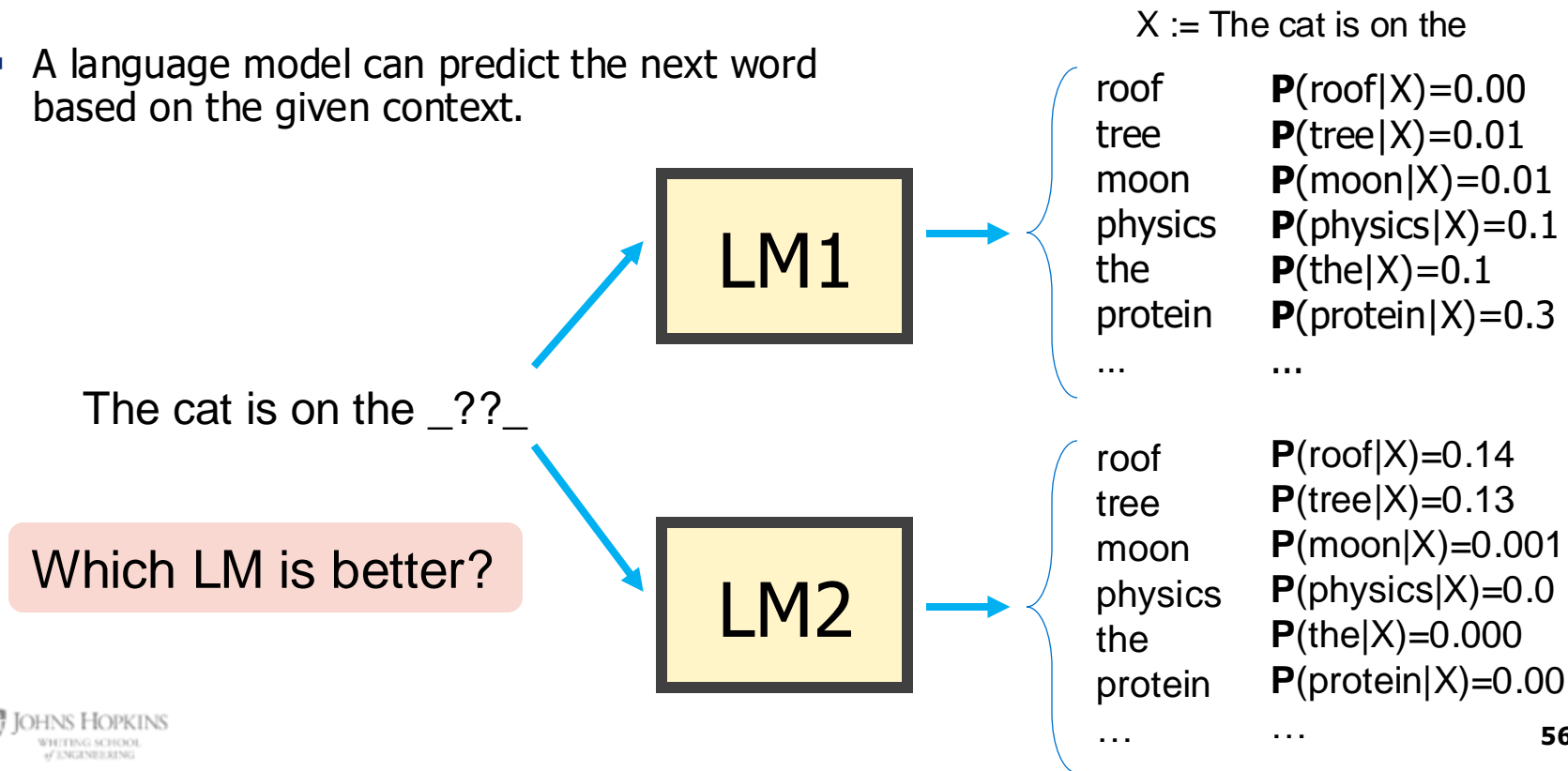
Large Language Models

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



Large Language Models

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



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.

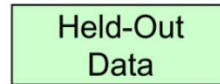
Evaluating Language Models

Setup:

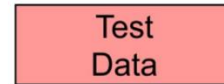
- **Train** it on a suitable training documents.
- **Evaluate** their **predictions** on different, unseen documents.
- An **evaluation metric** tells us how well our model does on the test set.



Counts / parameters from here



Hyperparameters from here



Evaluate here

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 →

A good language model should assign a high probability to held-out text!

count("on the mat")

→ 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 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 →

A good language model should assign a high probability to held-out text!

count("on the mat")

→ eval

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



Quiz: Building Intuition

- Sample a sentence $(w_1, w_2, \dots, w_n) = (\text{cat, sat, on, the, mat})$ from our natural data.
- We can show the probability that our language model assigns to this sentence with:

$$\mathbf{P}(w_1, w_2, \dots, w_n)$$

- A **strong** language model would assign a __ probability to this sentence. (**high or low?**)
- A **weak** language model would assign a __ probability to this sentence. (**high or low?**)

Next, we will define “perplexity”, a metric that quantifies LM’s uncertainty with respect to a corpus of natural sentences.

Evaluation Metric for Language Modeling: Perplexity

- Sample a sentence (w_1, w_2, \dots, w_n) from our natural data.
- **Perplexity** is the inverse probability of the test set, normalized by the number of words:

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

The negative power $(.)^{-}$ inverts the score. So, a small probability becomes a larger score – working with small numbers is tedious.

$\frac{1}{n}$ normalizes the probability as a function of length so that longer sequences are not assigned lower scores.

- A measure of **predictive quality** of a language model.
- A LM with **lower** perplexity is better because it assigns a **higher** probability to the unseen test corpus.

Evaluation Metric for Language Modeling: Perplexity

- Sample a sentence (w_1, w_2, \dots, w_n) from our natural data.
- **Perplexity** is the inverse probability of the test set, normalized by the number of words:

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

But wait, we usually have conditionals not the joint distribution! 🙄

Evaluation Metric for Language Modeling: Perplexity

- Sample a sentence (w_1, w_2, \dots, w_n) from our natural data.
- **Perplexity** is the inverse probability of the test set, normalized by the number of words:

$$\begin{aligned} \text{ppl}(w_1, \dots, w_n) &= \mathbf{P}(w_1, w_2, \dots, w_n)^{-\frac{1}{n}} \\ &= \sqrt[n]{\frac{1}{\mathbf{P}(w_1, w_2, \dots, w_n)}} = \sqrt[n]{\prod_{i=1}^n \frac{1}{\mathbf{P}(w_i | w_{<i})}} \quad \text{chain rule} \\ &= 2^H, \text{ where} \end{aligned}$$

$$H = -\frac{1}{n} \sum_{i=1}^n \log_2 \mathbf{P}(w_i | w_1, \dots, w_{i-1})$$

Putting Things Together: **Perplexity Definition**

- For a given a sampled sentence (w_1, w_2, \dots, w_n) from our natural data:

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

- Notice that this consists of probability assigned to all the partial sentences (i.e., next word probabilities).
- In practice, we prefer to use **log**-probabilities (also known as “logits”) since probabilities are too small and hard to understand (e.g., 10^{-18} vs -18).

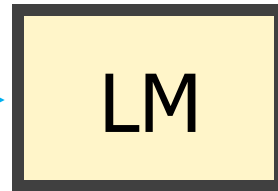
Intuition-building Quizzes (1)

- **Quiz:** let's we evaluate a **confused** (!!) model of language, i.e., our model has no idea what word should follow each context—it always chooses a uniformly random word. What is the perplexity of this model?
- Answer: $|V|$ (size of the vocabulary) – why?

Intuition-building Quizzes (1)

- **Quiz:** let's we evaluate a **confused** (!!) model of language, i.e., our model has no idea what word should follow each context—it always chooses a uniformly random word. What is the perplexity of this model?
- Sample a sentence from corpus: X ="The cat is on the mat."

For any partial sub-sentence:
 X =The cat is on the _??_ →



→ { roof $\mathbf{P}(\text{roof}|X)=1/|V|$
tree $\mathbf{P}(\text{tree}|X)=1/|V|$
moon $\mathbf{P}(\text{moon}|X)=1/|V|$
physics $\mathbf{P}(\text{physics}|X)=1/|V|$
...
... }

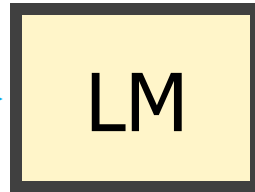
$$\forall w \in V: \mathbf{P}(w|w_{1:i-1}) = \frac{1}{|V|} \Rightarrow \text{ppl}(D) = 2^{-\frac{1}{n}n \log_2 \frac{1}{|V|}} = |V|$$

Intuition-building Quizzes (2)

- **Quiz:** let's suppose we have a sentence w_1, \dots, w_n and it's fixed. Our language is model is **mildly confused** because it narrows down the plausible continuations to 5 words, but it is confused among them. So it assigns probability $1/5$ to the correct next word. What is perplexity of our model?

A partial sentence:

X=The cat is on the _??_ →



$$\mathbf{P}(\text{roof}|X)=1/5$$

$$\mathbf{P}(\text{tree}|X)=1/5$$

$$\mathbf{P}(\text{table}|X)=1/5$$

$$\mathbf{P}(\text{mat}|X)=1/5$$

$$\mathbf{P}(\text{wall}|X)=1/5$$

$$\mathbf{P}(\text{physics}|X)=0$$

$$\mathbf{P}(\text{tesla}|X)=0$$

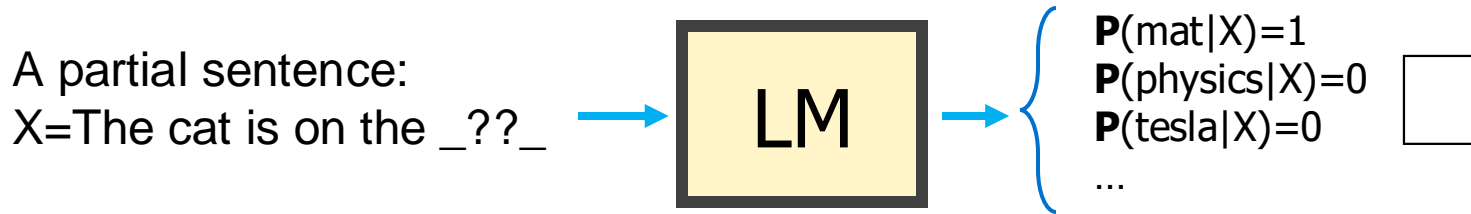
Our LM has narrowed down the right continuation to one of these five words.

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

Intuition: the model is indecisive among 5 choices.

Intuition-building Quizzes (2)

- **Quiz:** let's we evaluate an **exact** (!!) model of language, i.e., our model always knows what exact word should follow a given context. What is the perplexity of this model?



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

Intuition: the model is indecisive among 1 (the right!) choice!

Perplexity: Summary

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

- Perplexity is a measure of model's **uncertainty about next word** (aka "average branching factor").
 - The larger the number of vocabulary, the more options there to choose from.
 - (the choice of atomic units of language impacts PPL — more on this later)
- Perplexity ranges between **1** and **|V|**.
- We prefer LMs with **lower** perplexity.

Quiz: Perplexity for N-Grams

- Remember the ppl definition:

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

- Which expression corresponds to PPL of (1) unigram, (2) bigram and (3) trigram LM.

$$H = -\frac{1}{n} \sum_{i=1}^n \log_2 \mathbf{P}(w_i | w_{i-1}) \quad H = -\frac{1}{n} \sum_{i=1}^n \log_2 \mathbf{P}(w_i) \quad H = -\frac{1}{n} \sum_{i=1}^n \log_2 \mathbf{P}(w_i | w_{i-2}, w_{i-1})$$

Lower perplexity == Better Model

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

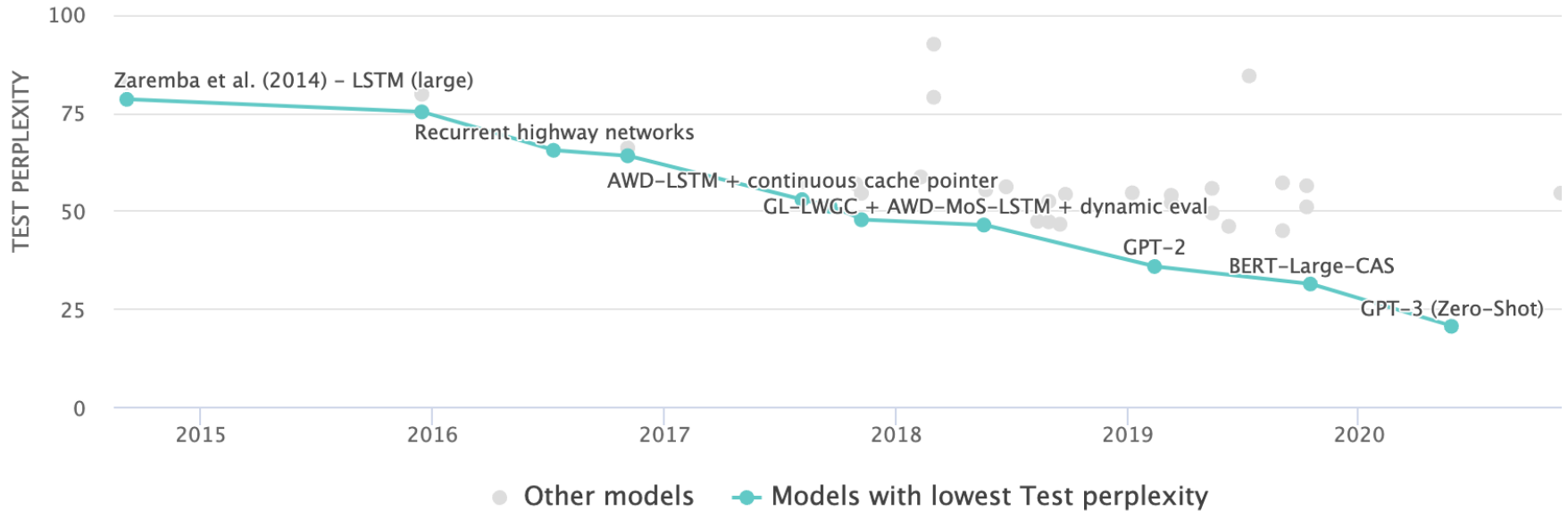
N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Lower is
better

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

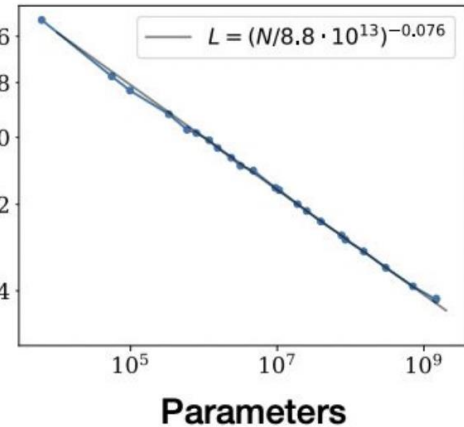
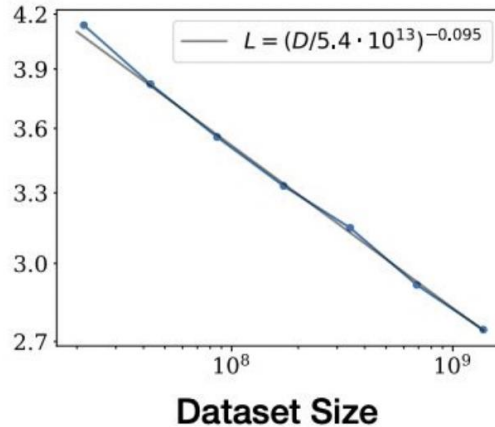
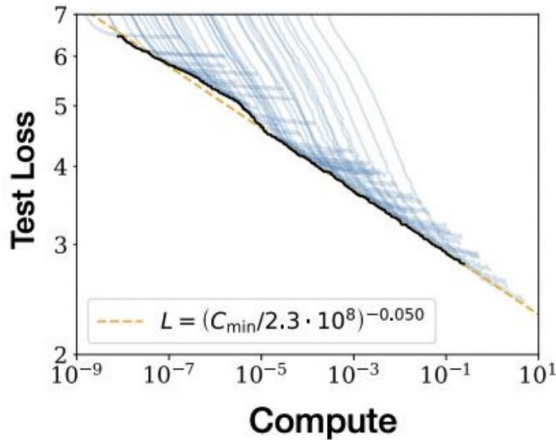
Lower perplexity == Better Model

The PPL of modern language models have consistently been going down.



Lower perplexity == Better Model

The PPL of modern language models have consistently been going down.



Perplexity as An Implicit Cross-Entropy

- Compare the definition of PPL and cross-entropy:

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

Cross-entropy between two distributions q, p :

$$\text{CE}(q, p) = - \sum_{x \in \mathcal{X}} q(x) \log p(x)$$

The H term in PPL can be interpreted as cross-entropy between LM distribution and the latent (unobserved) distribution of language. **Why?**

Hint: Monte Carlo Approximation + Law of Large Numbers.

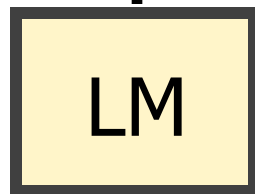
Summary

- **Language Models (LM):** distributions over language
- **Measuring LM quality:** use perplexity on held-out data.
- **Count-based LMs have limitations.**
 - **Challenge with large N's:** **sparsity** problem — many zero counts/probs.
 - **Challenge with small N's:** **lack of long-range** dependencies.
- **Next:** Rethinking language modeling as a statistical learning problem.

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



extrinsic
eval



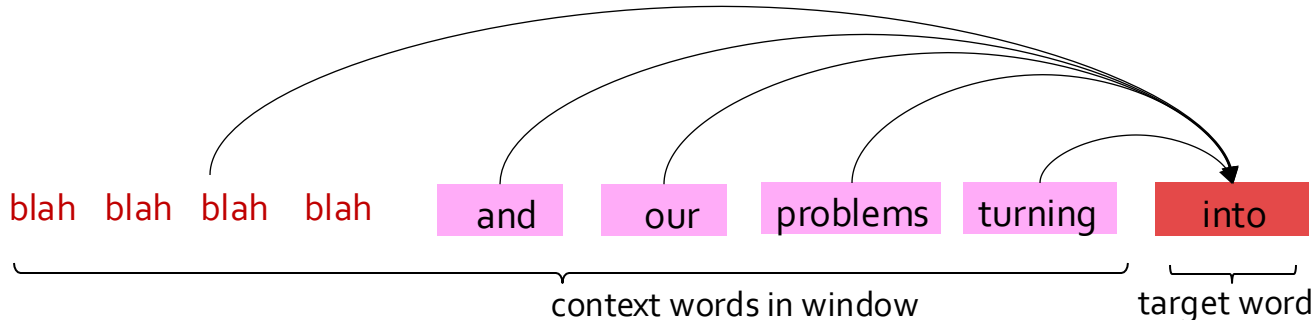
Now I'm going to evaluate the probability of some heldout data using our bigram table



Beyond Counting: Language Models as a Learning Problem

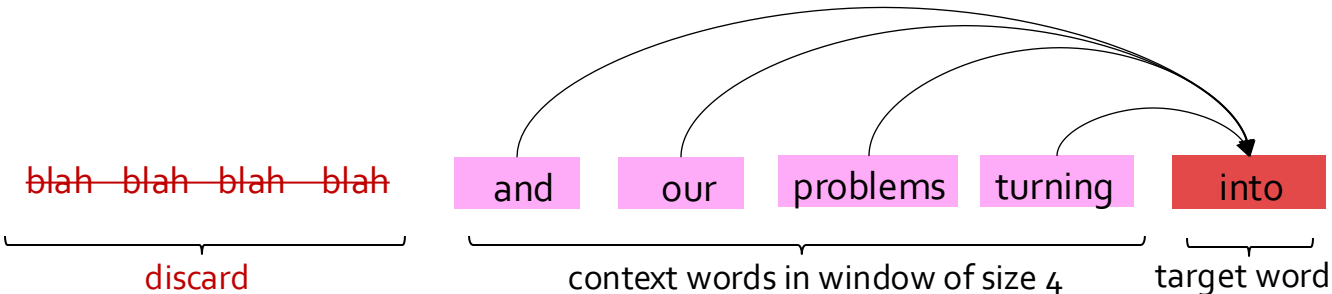
LM as a Machine Learning Problem

- 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



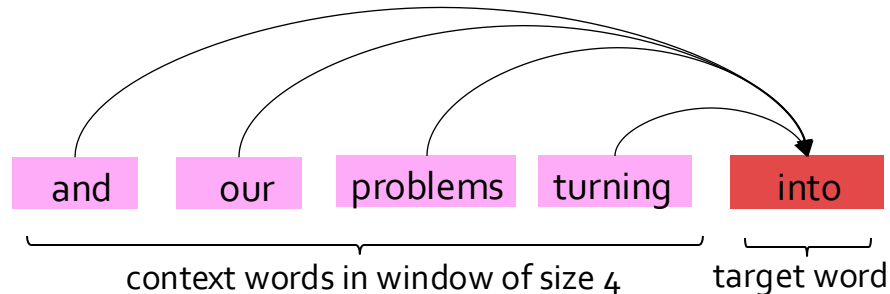
LM as a Machine Learning Problem

- 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



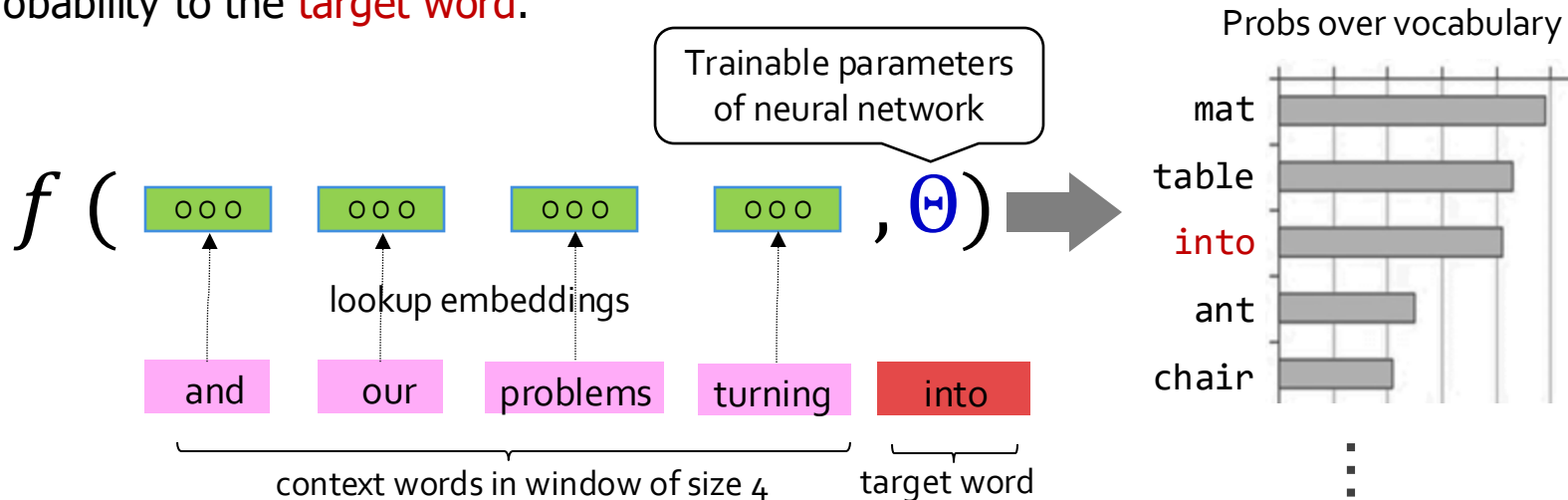
LM as a Machine Learning Problem

- 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**.



A Fixed-Window Neural LM

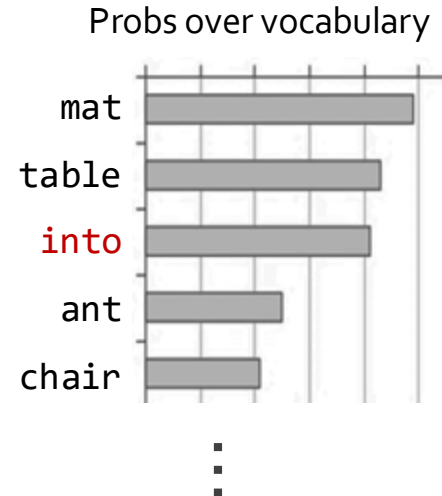
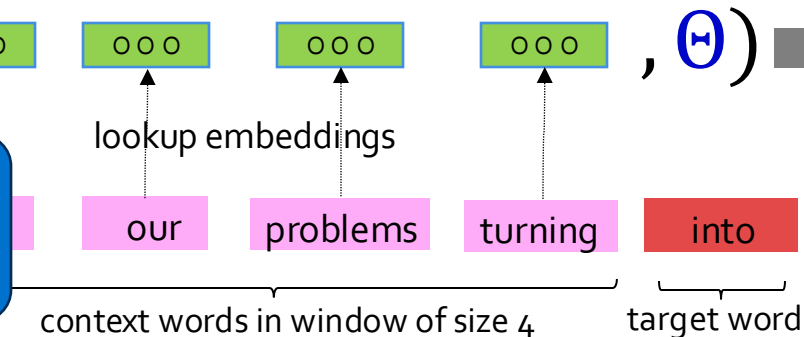
- It will also lay the foundation for the future models (recurrent nets, transformers, ...)
- But first we need to figure out how to train neural networks!

How do you build this function?

$f(\text{lookup embeddings}, \text{target word})$

Trainable parameters of neural network

Neural Networks for rescue!



From Counting (N-Gram) to Neural Models

- 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
- **Shallow neural LMs** (early and mid-2000's),
 - We will learn about build neural networks in the next chapter.
 - These will be effective predictive models based on feed-forward networks