JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

Language Modeling

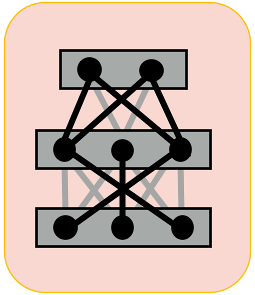
CSCI 601-471/671 (NLP: Self-Supervised Models)

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 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

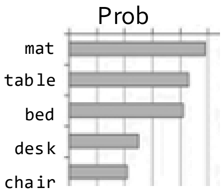
$$\mathbf{P}(\underbrace{X_t}_{\text{next word}} \mid \underbrace{X_1, \ldots, X_{t-1}}_{\text{context or prefix}})$$

LMs as a Marginal Distribution

- Directly we train models on "marginals":

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

`"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:

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

- Language Modeling ≜ 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.

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

- 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"*)

JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

Doing Things with Language Model (3)

- Can also generate strings!

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

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

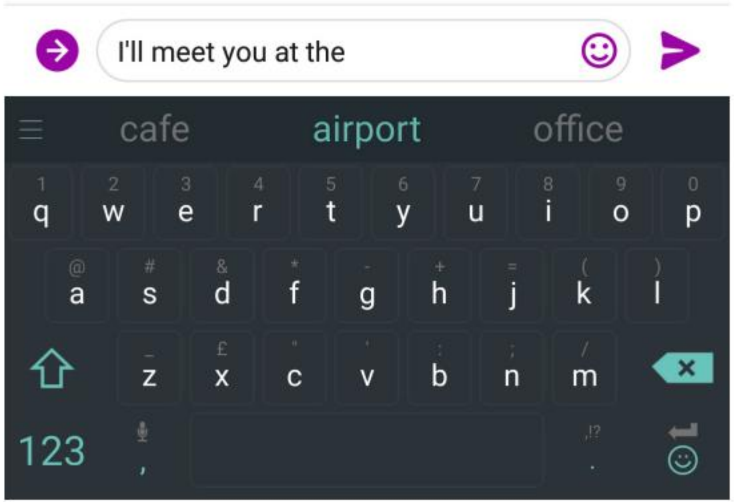
1. Sample from **P**(X| *"Johns Hopkins is "*) →"located"
2. Sample from **P**(X| *"Johns Hopkins is located"*) →"at"
3. Sample from **P**(X| *"Johns Hopkins is located at"*) →"the"
4. Sample from **P**(X| *"Johns Hopkins is located at the"*) →"state"
5. Sample from **P**(X| *"Johns Hopkins is located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

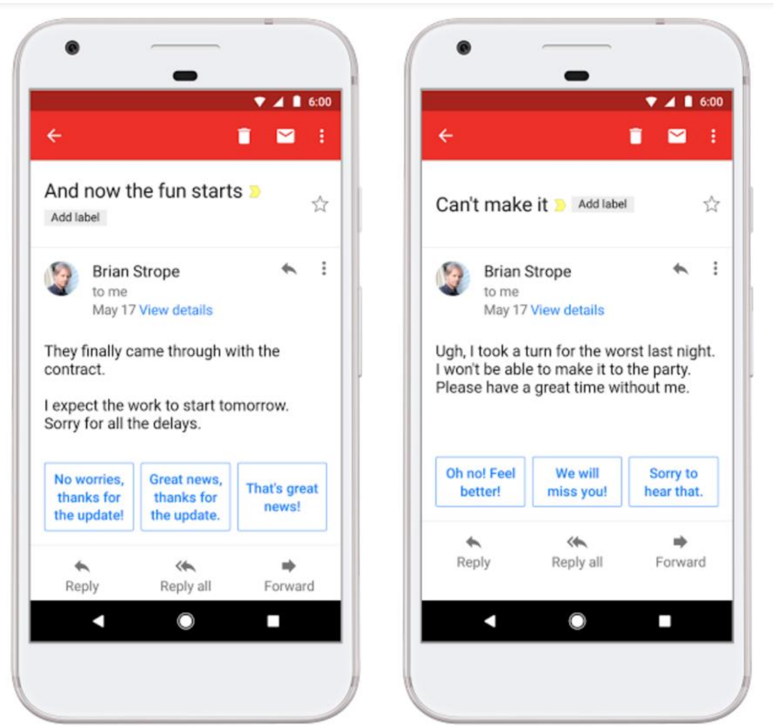
Why Care About Language Modeling?

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

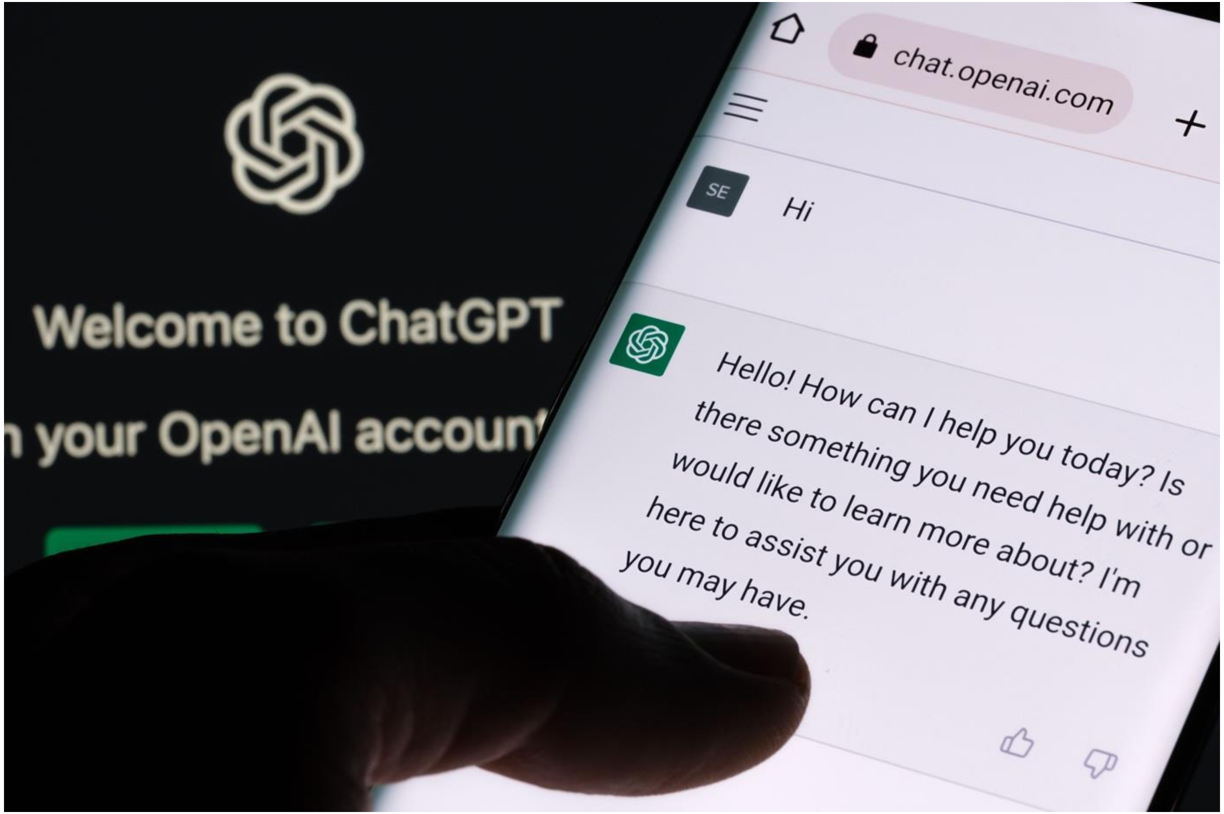
- Language Modeling is an effective proxy for language understanding.
  - Effective ability to predict forthcoming words requires on understanding of context/prefix.

You use Language Models every day!





You use Language Models every day!



Johns Hopkins Whiting School of Engineering

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

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

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

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

How do we estimate these probabilities?

Let's just count!

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

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

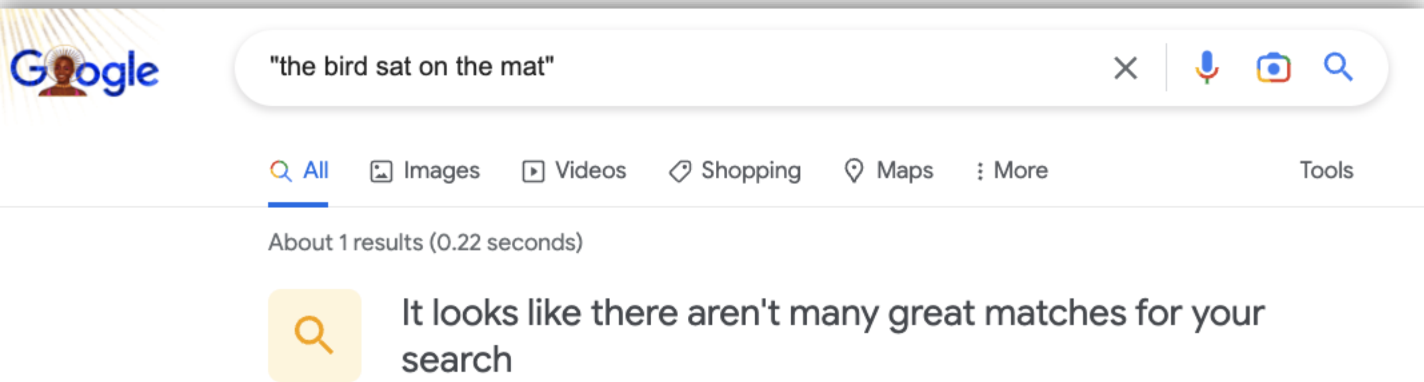
Divide that by, the count of
"the cat sat on the"
in the world (internet)!

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

How do we estimate these probabilities?

Let's just count!

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



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

How do we estimate these probabilities?

Let's just count!

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

<u>Challenge:</u> 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

- How common are zero-probabilities? 🤔
- **Example:** Shakespeare as a text corpus
  - The size vocab used by Shakespeare: |V|=29,066
  - 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 (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

(*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.

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

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:

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

[Prediction and Entropy of Printed English, Shanon 1950]

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

Andrey Markov

2nd order approximation:

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

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]

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

Andrey Markov

3rd order approximation:

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

2nd order approximation:

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

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]

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

3rd order approximation:

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

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:

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

[Prediction and Entropy of Printed English, Shanon 1950]

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*

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

[adopted from Chris Manning]

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/

[adopted from Chris Manning]

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!

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

[adopted from Chris Manning]

Generation from N-Gram Models

- Now we can sample from this mode:

condition on this

`today the price _____`

get probability distribution

of	0.308
for	0.050
it	0.046
to	0.046
is	0.031
...	

Sparsity problem: not much granularity in the probability distribution

Otherwise, seems reasonable!

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

[adopted from Chris Manning]

Generation from N-Gram Models

- Now we can sample from this mode:

condition on this

`today the price of ___`

get probability distribution

the	0.072
18	0.043
oil	0.043
its	0.036
gold	0.018
...	

<u>Sparsity problem</u>: not much granularity in the probability distribution

Otherwise, seems reasonable!

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

[adopted from Chris Manning]

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!

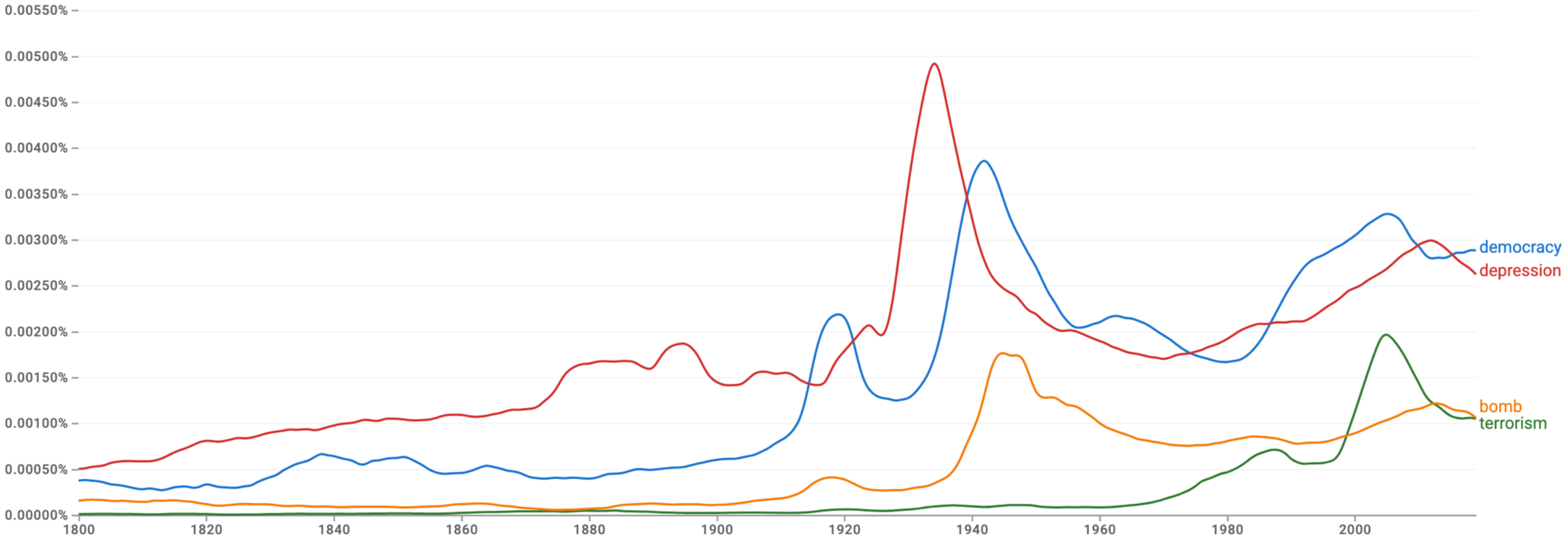
But quite incoherent! To improve coherence, one may consider increasing larger than 3-grams, but that would worsen the sparsity problem!

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

[adopted from Chris Manning]

Pre-Computed N-Grams

Google Books Ngram Viewer



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

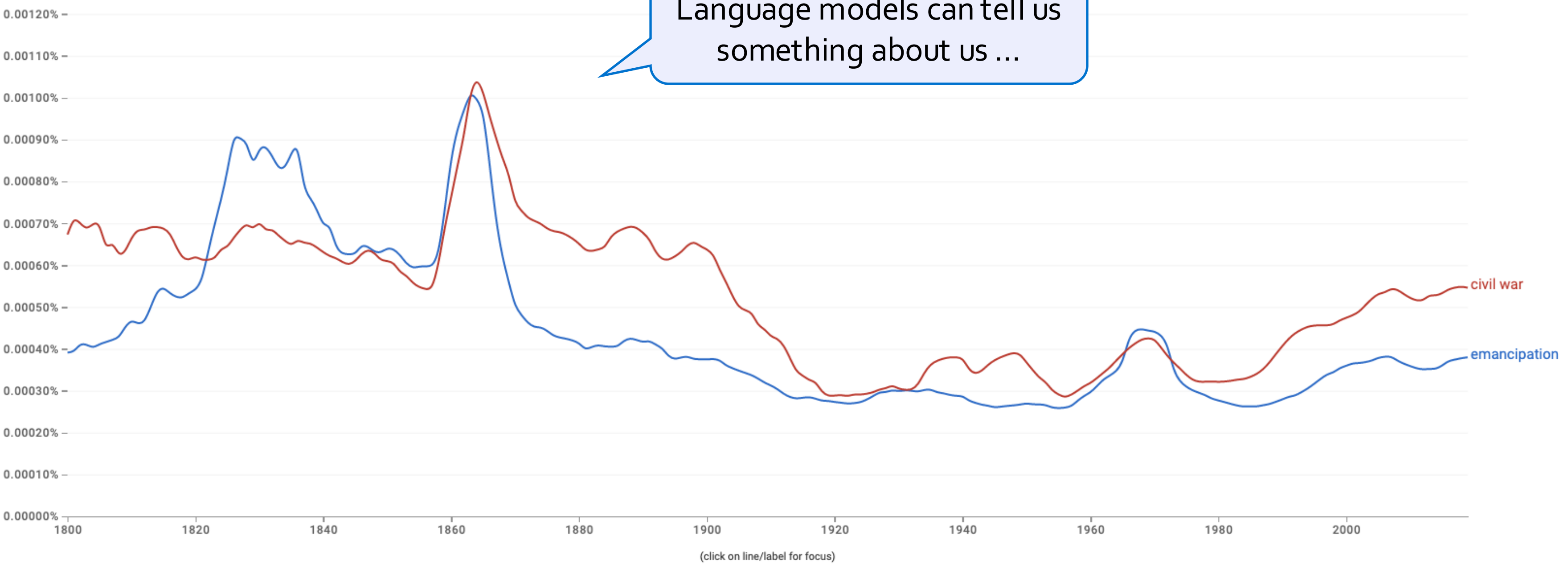
Data: http://storage.googleapis.com/books/ngrams/books/datasetsv2.html

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

Pre-Computed N-Grams

Google Books Ngram Viewer

Language models can tell us something about us ...

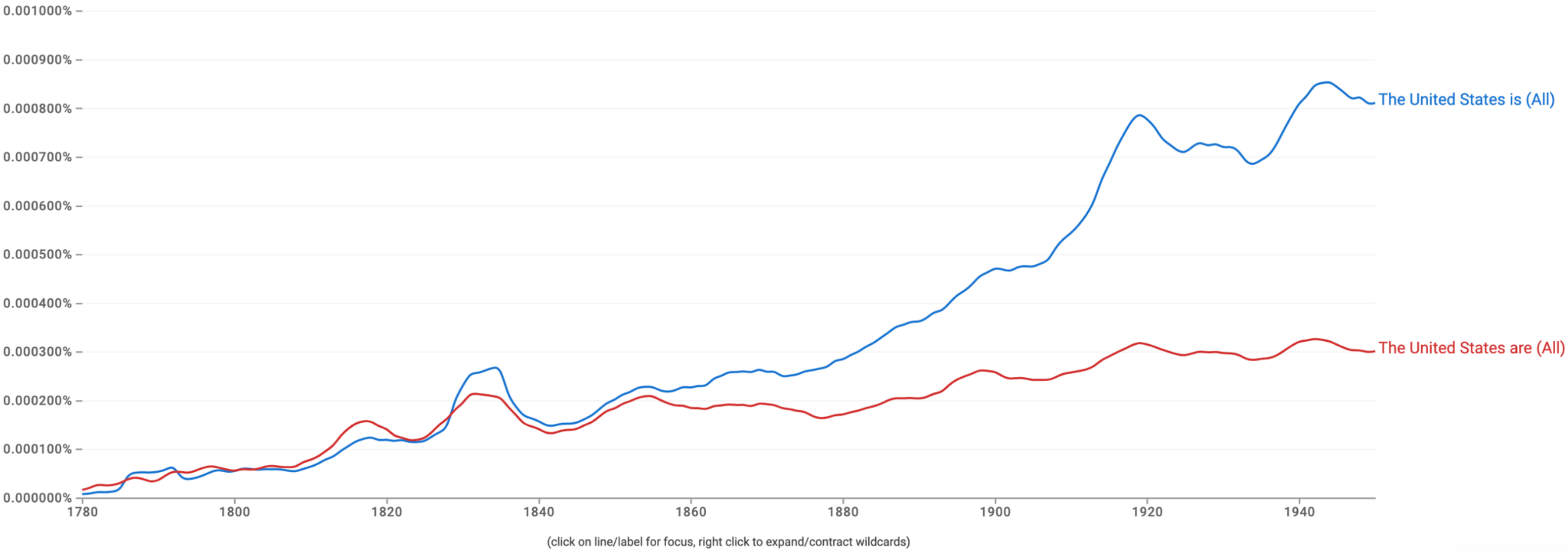


Google n-gram viewer https://books.google.com/ngrams/

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

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

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."

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

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

Summary

- Learning a language model ~ 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.

X := The cat is on the

The cat is on the _??_ → LM →

roof	**P**(roof|X)=0.00
tree	**P**(tree|X)=0.01
moon	**P**(moon|X)=0.01
physics	**P**(physics|X)=0.1
the	**P**(the|X)=0.1
protein	**P**(protein|X)=0.3
…	…

[slide credit: Arman Cohan]

Large Language Models

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

The cat is on the _??_

Which LM is better?

LM1

X := The cat is on the

roof	**P**(roof|X)=0.00
tree	**P**(tree|X)=0.01
moon	**P**(moon|X)=0.01
physics	**P**(physics|X)=0.1
the	**P**(the|X)=0.1
protein	**P**(protein|X)=0.3
…	…

LM2

roof	**P**(roof|X)=0.14
tree	**P**(tree|X)=0.13
moon	**P**(moon|X)=0.001
physics	**P**(physics|X)=0.0
the	**P**(the|X)=0.000
protein	**P**(protein|X)=0.00
…	…

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.

Training Data	Held-Out Data	Test Data
Counts / parameters from here	Hyperparameters from here	Evaluate here

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

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 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

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

train → count(“on the mat”) → eval

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.)

Google Translate

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

train →

count("on the mat")

extrinsic eval ↑

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:

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

- 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:

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

Evaluation Metric for Language Modeling: Perplexity

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

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

$$= \sqrt[n]{\frac{1}{\mathbf{P}(w_1, w_2, \dots, w_n)}} = \sqrt[n]{\prod_{i=1}^{n} \frac{1}{\mathbf{P}(w_i | w_{<i})}}$$

chain rule

$$= 2^{H}, \text{where}$$

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

Evaluation Metric for Language Modeling: Perplexity

- In practice, we prefer to use log-probabilities (also known as "logits")
- We can rewrite perplexity formula in terms of log-probs:

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

Recap: Definition of cross-entropy between two distributions:

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

Can be interpreted as cross-entropy between LM prob and language prob. **Why?**

Evaluation Metric for Language Modeling: Perplexity

- In practice, we prefer to use log-probabilities (also known as "logits")
- We can rewrite perplexity formula in terms of log-probs:

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

- **Perplexity** for n-grams:
  - Unigrams: $H = -\frac{1}{n}\sum_{i=1}^{n} \log_2 \mathbf{P}(w_i)$
  - Bigrams: $H = -\frac{1}{n}\sum_{i=1}^{n} \log_2 \mathbf{P}(w_i | w_{i-1})$
  - Trigrams: $H = -\frac{1}{n}\sum_{i=1}^{n} \log_2 \mathbf{P}(w_i | w_{i-2}, w_{i-1})$
  - …

Intuition-building Quizzes (1)

- In practice, we prefer to use log-probabilities (also known as "logits")
- We can rewrite perplexity formula in terms of log-probs:

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

- **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?

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

Intuition: the model is indecisive among 5 choices.

Intuition-building Quizzes (2)

- In practice, we prefer to use log-probabilities (also known as "logits")
- We can rewrite perplexity formula in terms of log-probs:

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

- **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?

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

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

Intuition-building Quizzes (3)

- In practice, we prefer to use log-probabilities (also known as "logits")
- We can rewrite perplexity formula in terms of log-probs:

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

- **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?

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

Intuition: the model is indecisive among all the vocabulary terms.

Perplexity: Summary

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

- 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.

Lower perplexity == Better Model

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

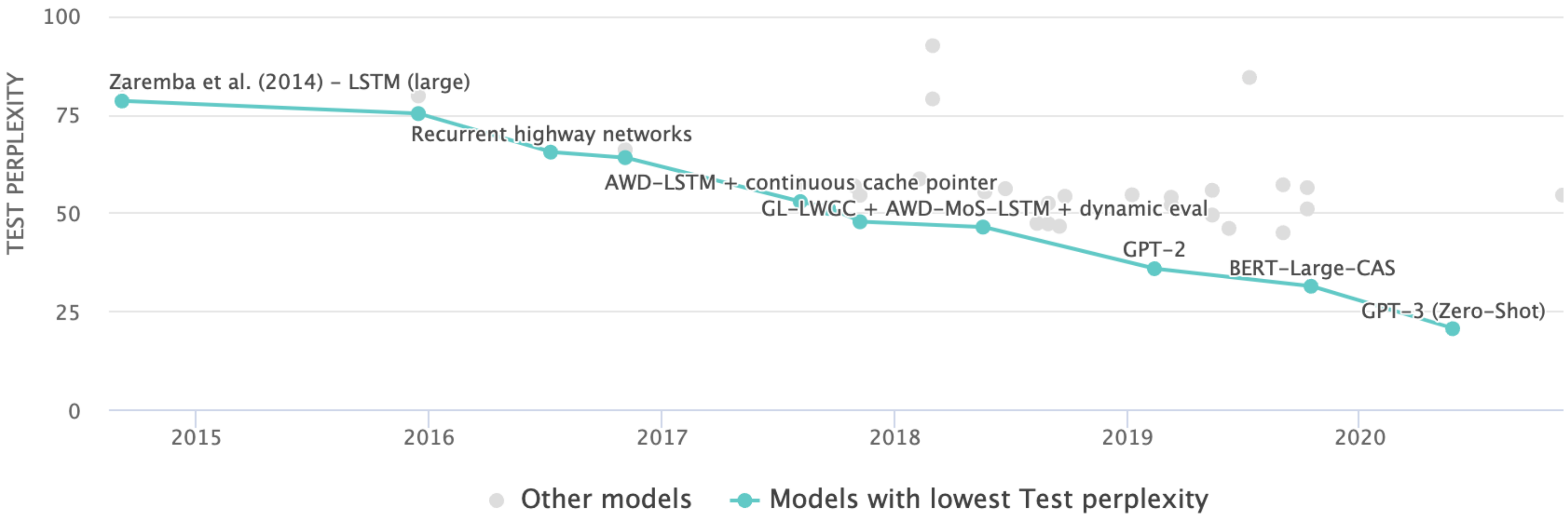
N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Lower is better

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

Lower perplexity == Better Model

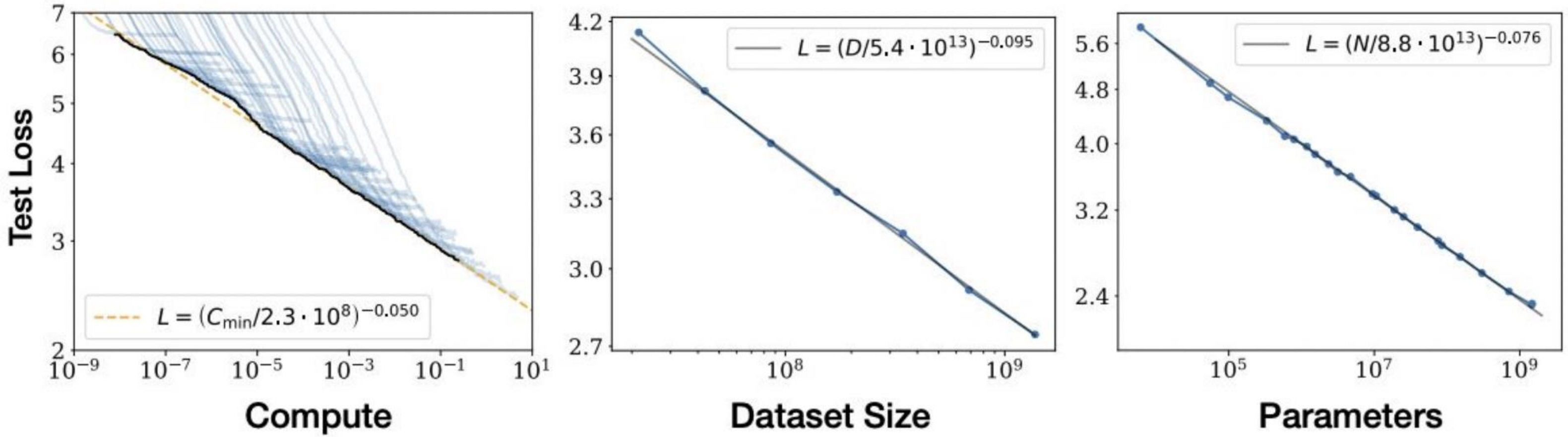
The PPL of modern language models have consistently been going down.



[Language Modelling on Penn Treebank (Word Level)]

Lower perplexity == Better Model

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

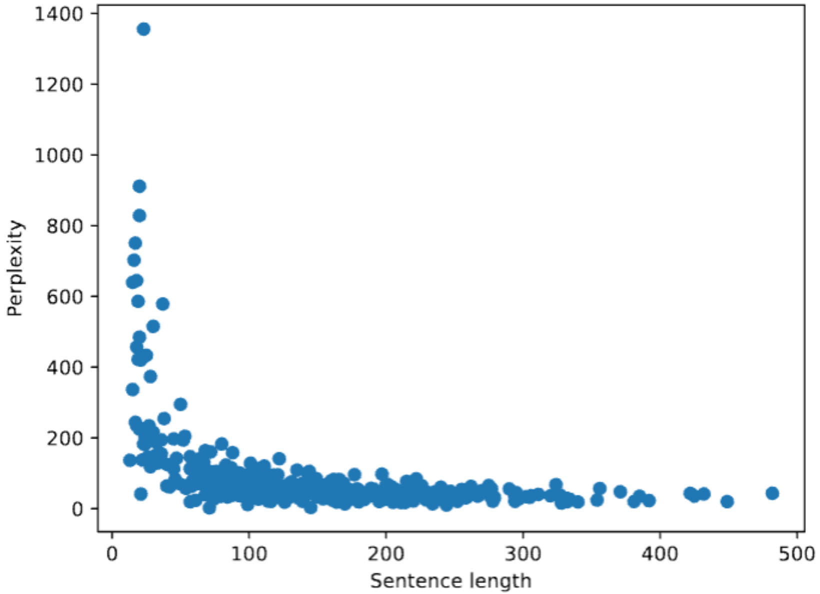


[Scaling Laws, Jared Kaplan et al.]

Quiz:

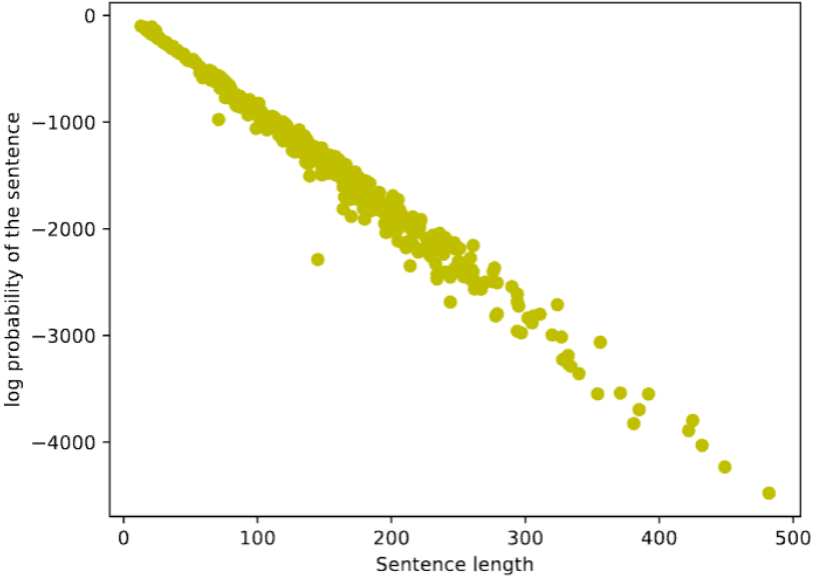
- Would you expect higher PPL for longer sentences?

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



- Would you expect a higher probability for longer sentences?

$$\mathbf{P}(w_1,\ldots,w_n) = \prod_{i=1}^{n}\mathbf{P}(w_i|w_1,\ldots,w_{i-1})$$



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.

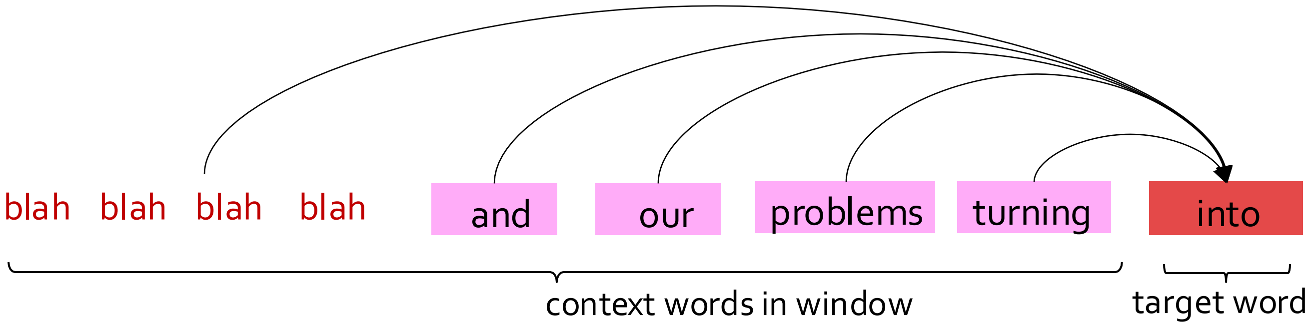
Beyond Counting:

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.

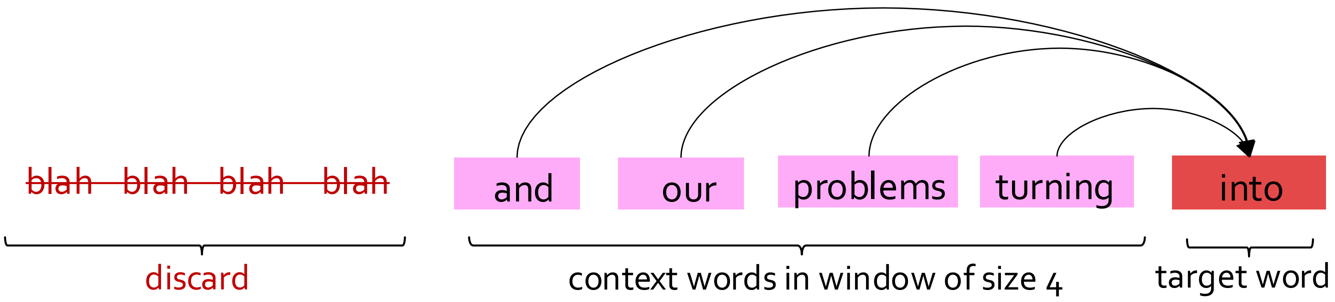
- Discard anything beyond its context window



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.

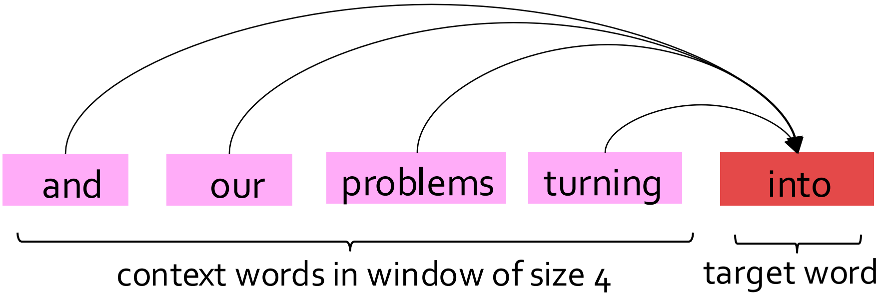
- Discard anything beyond its context window



LM as a Machine Learning Problem

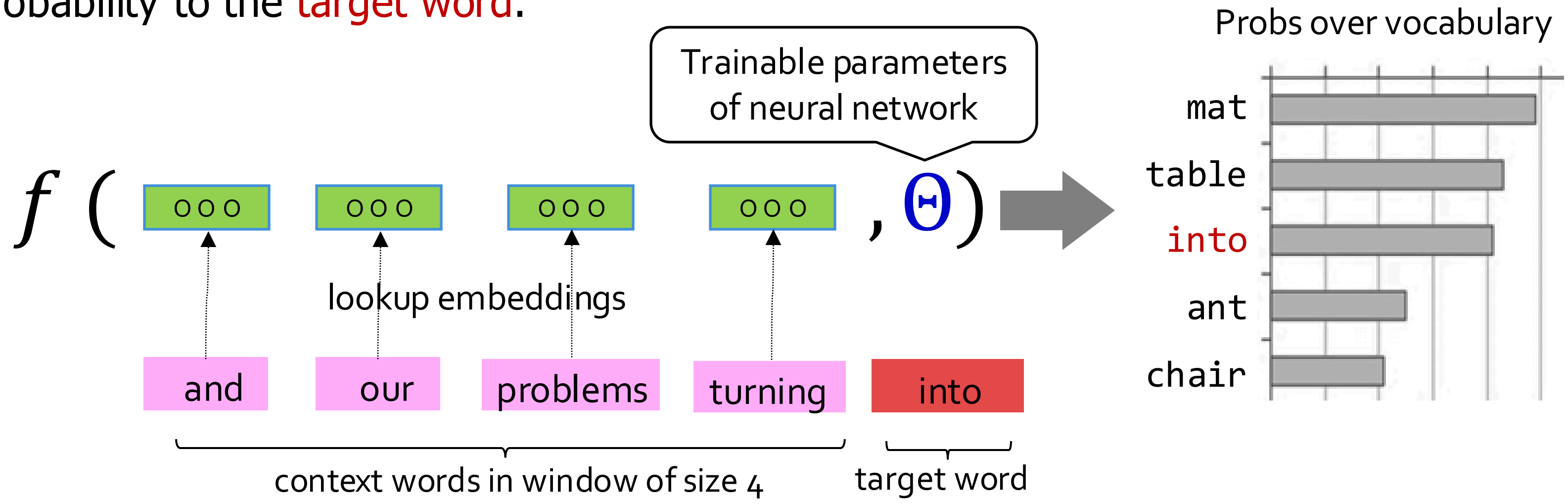
- Given the embeddings of the context, predict the word on the right side.
  - Dropping the right context for simplicity -- not a fundamental limitation.

- Discard anything beyond its context window



A Fixed-Window Neural LM

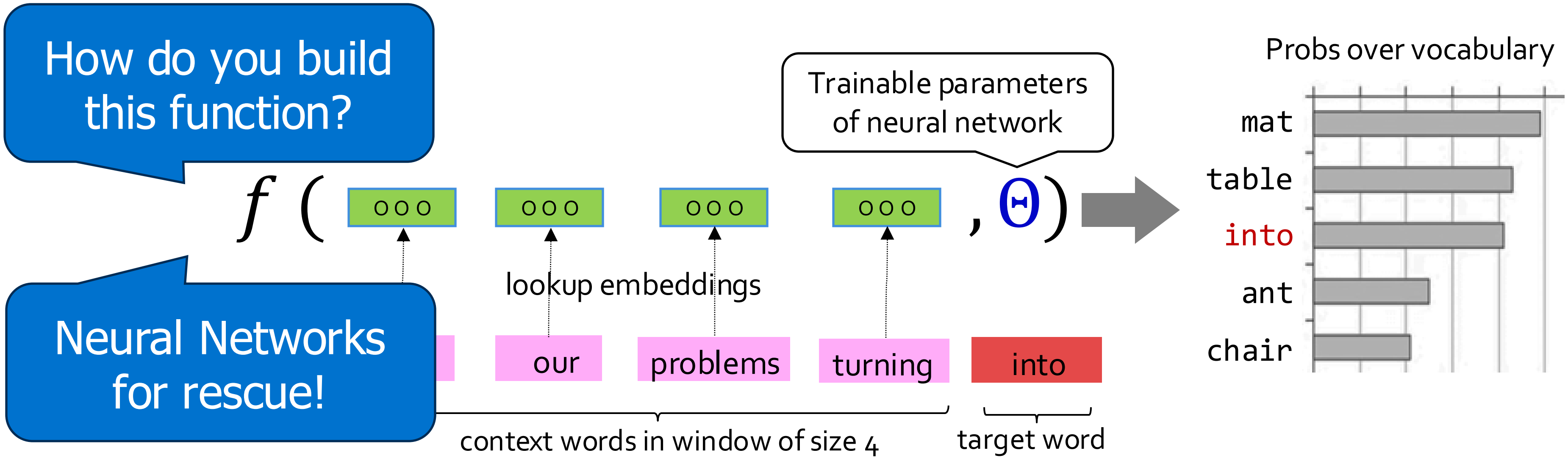
- Given the embeddings of the context, predict a target word on the right side.
  - Dropping the right context for simplicity -- not a fundamental limitation.
- Training this model is basically optimizing its parameters Θ such that it assigns high probability to the target word.



[Bengio et al. 2003]

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!



[Bengio et al. 2003]

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 need in coming sessions that one can build these models with neural networks.
  - These will be effective predictive models based on feed-forward networks