Language Modeling

CSE 5525: Foundations of Speech and Language Processing

https://shocheen.github.io/cse-5525-spring-2025/

THE OHIO STATE UNIVERSITY

Sachin Kumar (kumar.1145@osu.edu)

Slide Credits: Yoav Artzi, Yulia Tsvetkov, Ana Marasović

Logistics

- Homework 1 is due TODAY at midnight

- Homework 2 is going to be released tomorrow morning.
  - Topic: Language Modeling with Transformers

Previous lecture recap: Lexical Semantics / Word Vectors

- Linguistics provides different ways of representing meanings of words
  - Requires a large amount of manual effort and not easily operationalizable in NLP models.

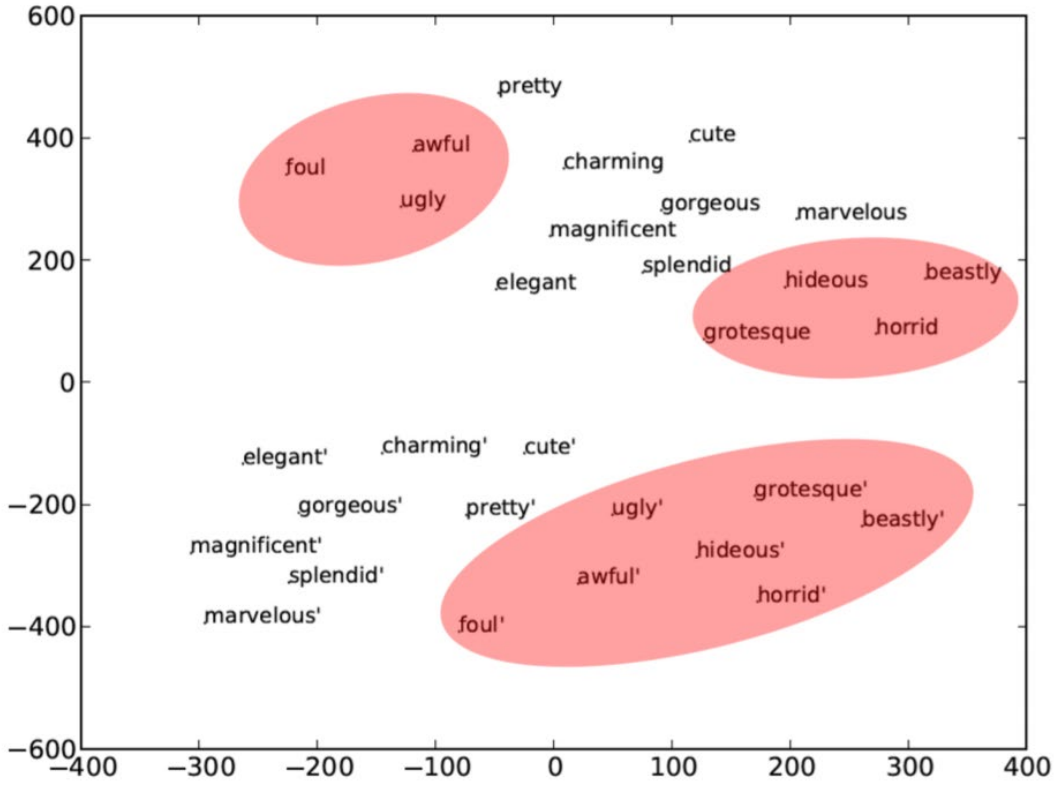
- Distributional semantics: allows to build representations of words using an unlabeled corpus.
  - One way to learn them is wordvec (skipgram). It is a shallow neural network built on the idea of converting word co-occurrences into a binary classification task.

Word Vectors Evaluation

- Qualitative
- Intrinsic
- Extrinsic

WORD	d1	d2	d3	d4	d5	...	d50
summer	0.12	0.21	0.07	0.25	0.33	...	0.51
spring	0.19	0.57	0.99	0.30	0.02	...	0.73
fall	0.53	0.77	0.43	0.20	0.29	...	0.85
light	0.00	0.68	0.84	0.45	0.11	...	0.03
clear	0.27	0.50	0.21	0.56	0.25	...	0.32
blizzard	0.15	0.05	0.64	0.17	0.99	...	0.23

Visualisation



[Faruqui et al., 2015]

Figure 6.5: Monolingual (top) and multilingual (bottom; marked with apostrophe) word projections of the antonyms (shown in red) and synonyms of "beautiful".

- Visualizing Data using t-SNE (van der Maaten & Hinton'08)

Extrinsic Evaluation

Initialize an NLP model's embedding layer and train

- Topic categorization
- Sentiment analysis
- Machine Translation
- Document summarization
- …

Intrinsic Evaluation

word1	word2	similarity (humans)	similarity (embeddings)
vanish	disappear	9.8	1.1
behave	obey	7.3	0.5
belief	impression	5.95	0.3
muscle	bone	3.65	1.7
modest	flexible	0.98	0.98
hole	agreement	0.3	0.3

- WS-353 (Finkelstein et al. '02)
- MEN-3k (Bruni et al. '12)
- SimLex-999 dataset (Hill et al., 2015)

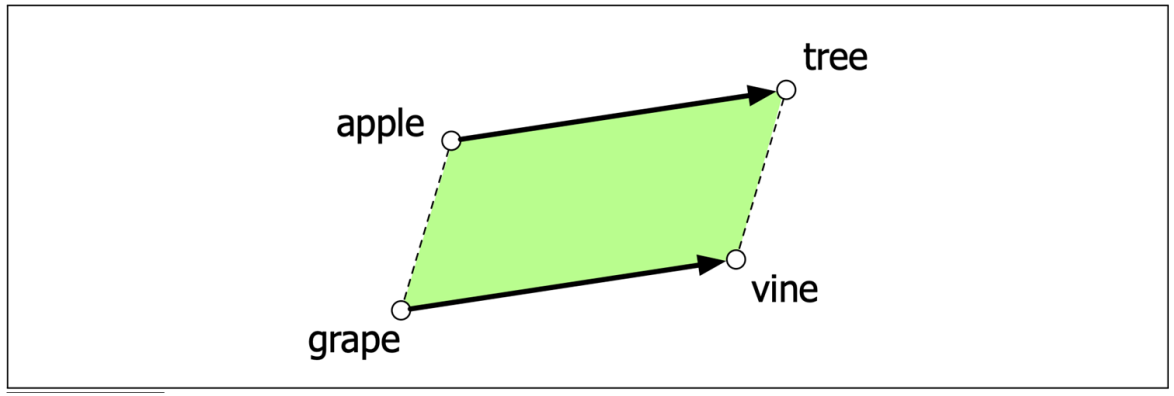
Spearman's rho (human ranks, model ranks)

Analogy/Relational Similarity

Embeddings capture relational meanings

Analogy problems:

- *a is to b as a* is to what?*
- *a:b::a*:b**
- *apple:tree::grape:?*
- *king:man::woman:?*
- *Paris:Frace::Italy:?*



Add the vector from the word *apple* to the word *tree*, v(tree)-v(apple), to the vector of the grape, v(grape)

The nearest word to that point is returned

[The (too Many) Problems of Analogical Reasoning with Word Vectors]

$$\hat{b} = \text{argmin}_x \text{distance}(x, b - a + a^*)$$

Societal biases

computer programmer - man + woman

= homemaker [Bolukbasi et al., 2016]

doctor - man + woman=nurse

Downstream impact: A tool for hiring doctor or programmers downweights documents with women's names

Allocation harm: a system allocates resources (jobs or credit) unfairly to different groups [Blodgett et al., 2020]

Bias amplification: gendered terms become more gendered in embeddings spaces than they were in the input text statics [Jia et al., 2020]

Representational harm: Harm caused by a system demeaning or even ignoring some social groups

- Names like "Leroy" have a higher cosine similarity with unpleasant words while names like Brad, Greg, Courtney have a higher cosine with pleasant words [Zhou et al., 2022]

Debiasing is very hard [Gonen and Goldberg, 2019]

Language Models

Goals & Overview of Today's Lecture

Goal: Understand the language modeling task, which will be used for pretraining and the modern approach to treating many NLP applications as text generation

- Language modeling
- n-gram language modeling
- Smoothing
- Neural language modeling
- Intrinsic evaluation of language models

The

The cat

The cat sat

The cat sat on

The cat sat on __?__

The cat sat on the mat.

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

next word

context or prefix

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

next word

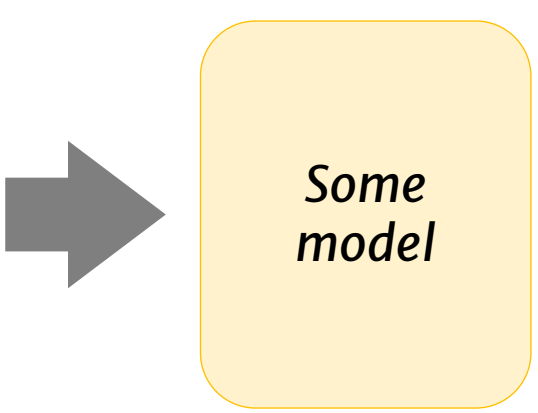
context

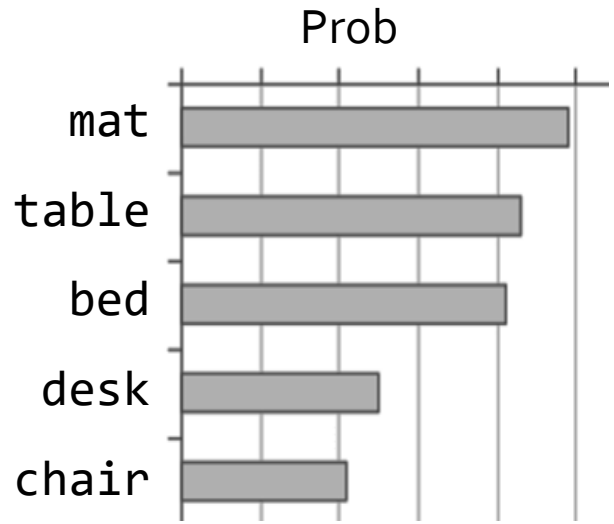
$$\mathbf{P}(X_t \mid X_1, \ldots, X_{t-1})$$

next word

context

"The cat sat on the [MASK]"





But more broadly, we want to model

$$\mathrm{P}(X_1, \ldots, X_t)$$

Apply chain rule

$$\mathrm{P}(X_1)P(X_2|\mathrm{X}_1) \ldots \mathrm{P}(\mathrm{X}_\mathrm{t}|\mathrm{X}_1, \ldots, X_{t-1})$$

The Original #1 Mad Libs

MAD LIBS®

World's Greatest Word Game

A super silly way to fill in the _____!
PLURAL NOUN

The Original #1 Mad Libs

MAD LIBS®

World's Greatest Word Game

If you just heard someone say . . .

"Every morning before washing your ___CAT___, massage it
NOUN
gently with a/an ___DOORKNOB___ that has been soaked overnight
NOUN
in a/an ___PENCIL BOX___ full of warm ___CORN OIL___."
TYPE OF CONTAINER TYPE OF LIQUID

. . . you've obviously been playing
The Original #1 Mad Libs!
Play them with friends or enjoy them by yourself!

Download the awesome
FREE Mad Libs app!

Available Now

$3.99 US
($5.50 CAN)

PSS!
PRICE STERN SLOAN
www.penguin.com/youngreaders
www.madlibs.com

ISBN 978-0-8431-0055-6

EAN 9 780843 100556 50399>

Ridiculously simple directions inside!

Doing Things with Language Model

- What is the probability of ….

"I like The Ohio State University"

"like State I University The Ohio State"

Doing Things with Language Model

- What is the probability of ….

  "I like The Ohio State University"

  "like State I University The Ohio State"

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

  P("I like The Ohio State University") = 10^-5

  P("like State I University The Ohio State") = 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 The Ohio State University. EOS*") > **P**("*I like Ohio State University EOS*")
P("*OSU is located in Columbus. EOS*") > **P**("*OSU is located in Pittsburgh. EOS*")

Doing Things with Language Model (3)

- Can also generate strings!

next word context

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

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

1. Sample from **P**(X | *"Ohio State is "*) → "located"
2. Sample from **P**(X | *"Ohio State is located"*) → "in"
3. Sample from **P**(X | *"Ohio State is located in"*) → "the"
4. Sample from **P**(X | *"Ohio State is located in the"*) → "state"
5. Sample from **P**(X | *"Ohio State is located in the state"*) → "of"
6. Sample from **P**(X | *"Ohio State is located in the state of"*) → "Ohio"
7. Sample from **P**(X | *"Ohio State is located in the state of Ohio"*) → "EOS"

Why Care About Language Modeling?

- Language Modeling is a part of many tasks:
  - Summarization
  - Machine translation
  - Spelling correction
  - Dialogue etc.
  - General purpose Instruction following (ala ChatGPT)

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

Summary so far

- **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?

Goals & Overview of Today's Lecture

Goal: Understand the language modeling task, which will be used for pretraining and the modern approach to treating many NLP applications as text generation

- Language modeling
- Intrinsic evaluation of language models
- n-gram language modeling
- Smoothing
- Neural language modeling

How good is our language model

- How good is our model?
  - At what?

- We want our model to prefer good sentences over bad ones
  - Higher probability to real or frequent sentences
    - Than ungrammatical or rare ones
    - Without overfitting to a training corpus

  - How does this relate to how we use the language model?

Evaluation

- We must test the model on data it hasn't seen during learning
  - Otherwise — overfitting! 😱

- We need an evaluation metric — two options:
  - Extrinsic: focused on however the model will be used
  - Intrinsic: focused on the language model task — how good can the model assign probabilities to real unseen data?
  - Ideally, the two correlate, but reality is more complex

Perplexity (PPL) – Instrinsic Evaluation

Train the language model on a train corpus, then evaluate on a held-out test set

Using the **likelihood of held-out data**
… actually, using a function of the likelihood

$$\text{perplexity}(w_1 w_2 \ldots w_n) = \mathbb{P}(w_1 w_2 \ldots w_n)^{-\frac{1}{n}} = \left(\prod_{i=1}^{n} \mathbb{P}(w_i | w_1 \ldots w_{i-1})\right)^{-\frac{1}{n}}$$

held-out test set

inverse ⇒ higher likelihood means lower perplexity

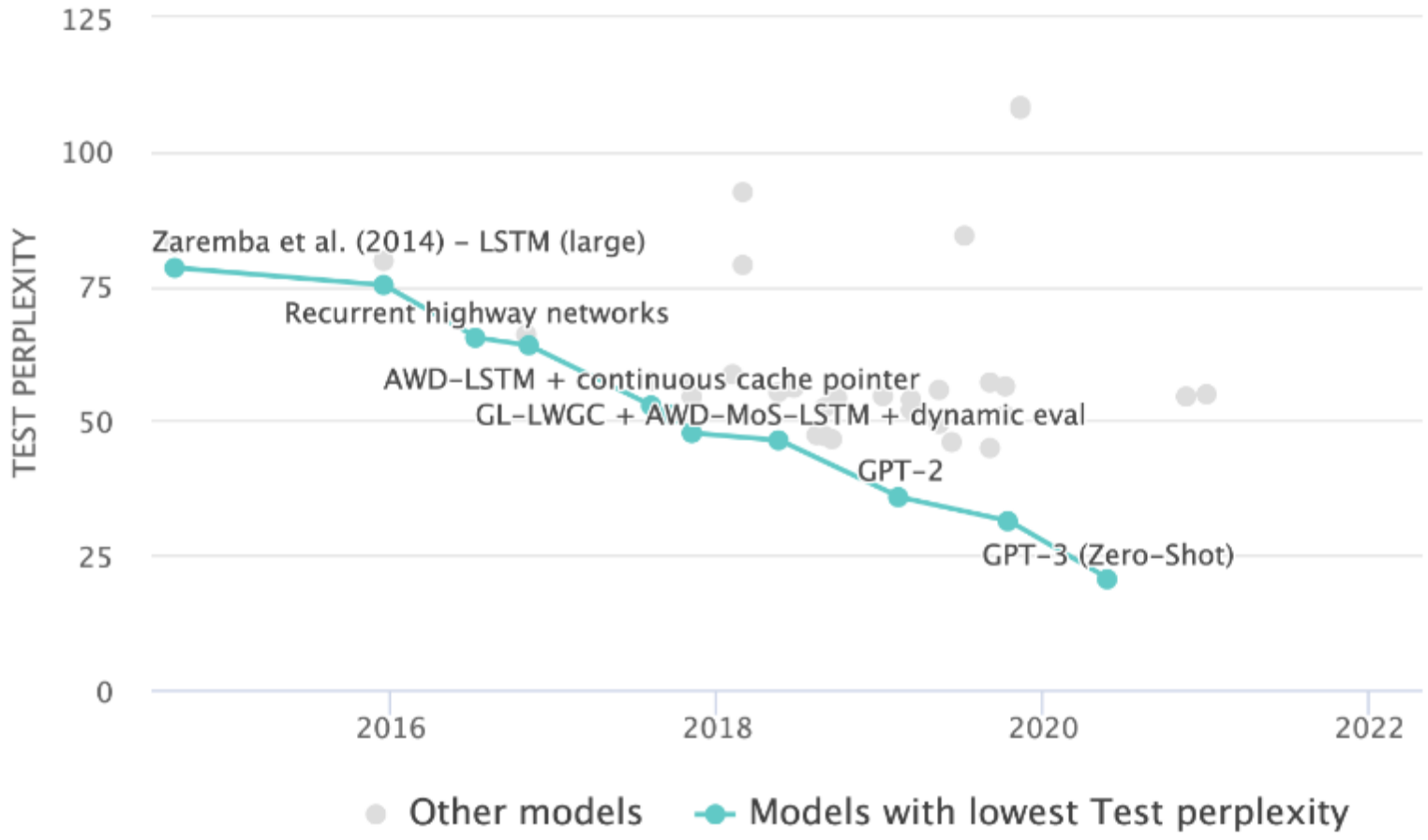
n-th root to normalize by the number of words; the likelihood gets smaller the longer the text (unwanted)

- Inverse comes from the original definition of perplexity in information theory
- Perplexity usually only reported in LM research papers: an (intrinsic) improvement in perplexity does not guarantee an (extrinsic) improvement in the downstream tak performance
- Typically ranges from 10–200
- Perplexity of 2 LMs is only comparable if they use identical vocabularies

Perplexity of a Uniform Model

- Assume sentences consisting of random digits
- Assume M sentences with m random digits. Vocabulary size = 10
- What is the perplexity of this data for a model that assigns p=1/10 to each digit

Perplexity of contemporary models



Goals & Overview of Today's Lecture

Goal: Understand the language modeling task, which will be used for pretraining and the modern approach to treating many NLP applications as text generation

- Language modeling
- Intrinsic evaluation of language models
- n-gram language modeling
- Smoothing
- Neural language modeling

Language Models: A History

- Shannon (1950): The predictive difficulty (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.

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”, …

https://books.google.com/ngrams/

- *n*-gram language model:

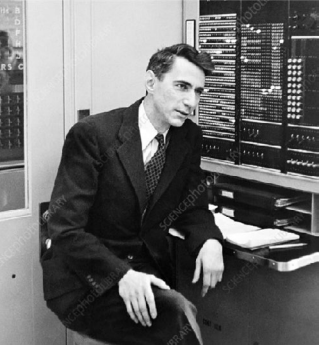
Goal: What's the probability of a word w given some history h?

$$\mathrm{P}(X_1)P(X_2|\mathrm{X}_1) \ldots \mathrm{P}(\mathrm{X}_\mathrm{t}|\mathrm{X}_1, \ldots, X_{t-1})$$

$$\mathbb{P}(\text{the}|\text{its water is so transparent that}) = \frac{\text{count}(\text{its water is so transparent that the})}{\text{count}(\text{its water is so transparent that})}$$

Even the Web isn't big enough to give us good estimates in most cases

Simple extensions of the example sentence may have counts zero



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

Andrey Markov

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

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

1st order approximation: $\mathbf{P}(\text{mat} \mid \text{the cat sat on the}) \approx \mathbf{P}(\text{mat} \mid \text{the})$

2nd order approximation: $\mathbf{P}(\text{mat} \mid \text{the cat sat on the}) \approx \mathbf{P}(\text{mat} \mid \text{on the})$

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

Estimating N-gram probabilities

The probabilities can be computed by **relative frequency** estimation

E.g., for *bigram* model (N=2):

$$\mathbb{P}(w_n|w_{n-1}) = \frac{\text{count}(w_{n-1}w_n)}{\sum_w \text{count}(w_{n-1}w)} = \frac{\text{count}(w_{n-1}w_n)}{\text{count}(w_{n-1})}$$

The general case (with any N):

$$\mathbb{P}(w_n|w_{n-N+1:n-1}) = \frac{\text{count}(w_{n-N+1:n-1}w_n)}{\text{count}(w_{n-N+1:n-1})}$$

Goals & Overview of Today's Lecture

Goal: Understand the language modeling task, which will be used for pretraining and the modern approach to treating many NLP applications as text generation

- Language modeling
- Intrinsic evaluation of language models
- n-gram language modeling
- Smoothing
- Neural language modeling

Increasing N-gram order & Sparsity

- **Gorillas** always like to groom **their** friends.
  - The likelihood of "their" depends on knowing that "gorillas" is plural
- The **computer** that's on the 3rd floor of our office building **crashed**.
  - The likelihood of "crashed" depends on knowing that the subject is a "computer"

With a low N, the resulting LM would offer probabilities that are too low for these sentences, and too high for sentences that fail basic linguistic tests like number agreement

In these examples we need a 6-gram model, but to estimate the probability of 6-grams, they must occur a sufficient number of times in our corpus

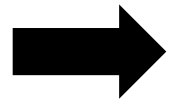
Sparsity: having many cases of putative "zero probability n-grams" that should really have some non-zero probability

Smoothing & Discounting

unknown bigram

$$\mathbb{P}(w_n|w_{n-1}) = \frac{\text{count}(w_{n-1}w_n)}{\sum_w \text{count}(w_{n-1}w)} = \frac{\text{count}(w_{n-1}w_n)}{\text{count}(w_{n-1})}$$

known word



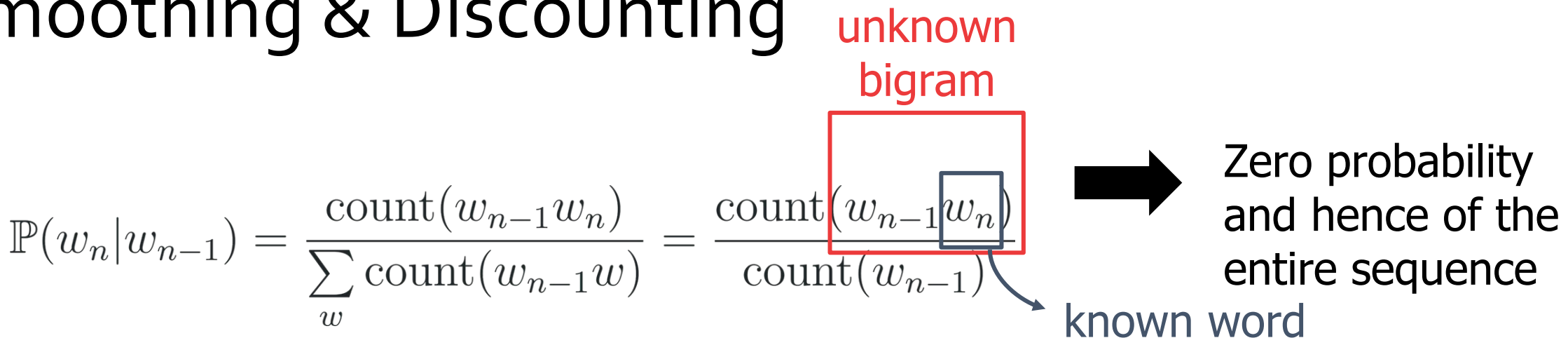
Zero probability and hence of the entire sequence

Lidstone smoothing: Add imaginary "pseudo" counts

- α=1 ⇒ **Laplace smoothing**
- α=0.5 ⇒ **Jeffreys-Perks law**
- V is vocabulary
- The probability mass is re-distributed equally

$$\mathbb{P}(w_n|w_{n-1}) = \frac{\text{count}(w_{n-1}w_n) + \alpha}{\text{count}(w_{n-1}) + |V| \cdot \alpha}$$

Smoothing & Discounting



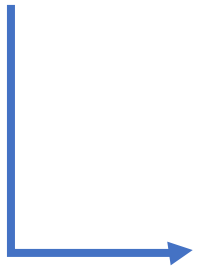
Absolute discounting: "shave off" a bit of probability from some more frequent n-grams and give it to the n-grams we've never seen

N-Gram Models in Practice

- 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/

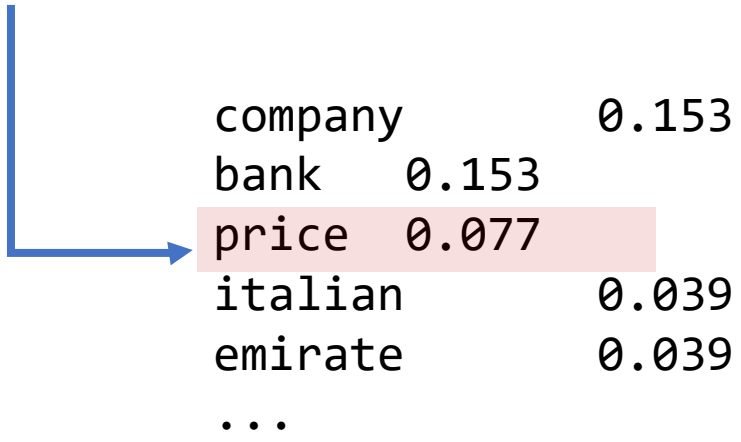
[adopted from Chris Manning]

N-Gram Models in Practice

- Now we can sample from this mode:

`today the ____`

get probability distribution



Sparsity problem: not much granularity in the probability distribution

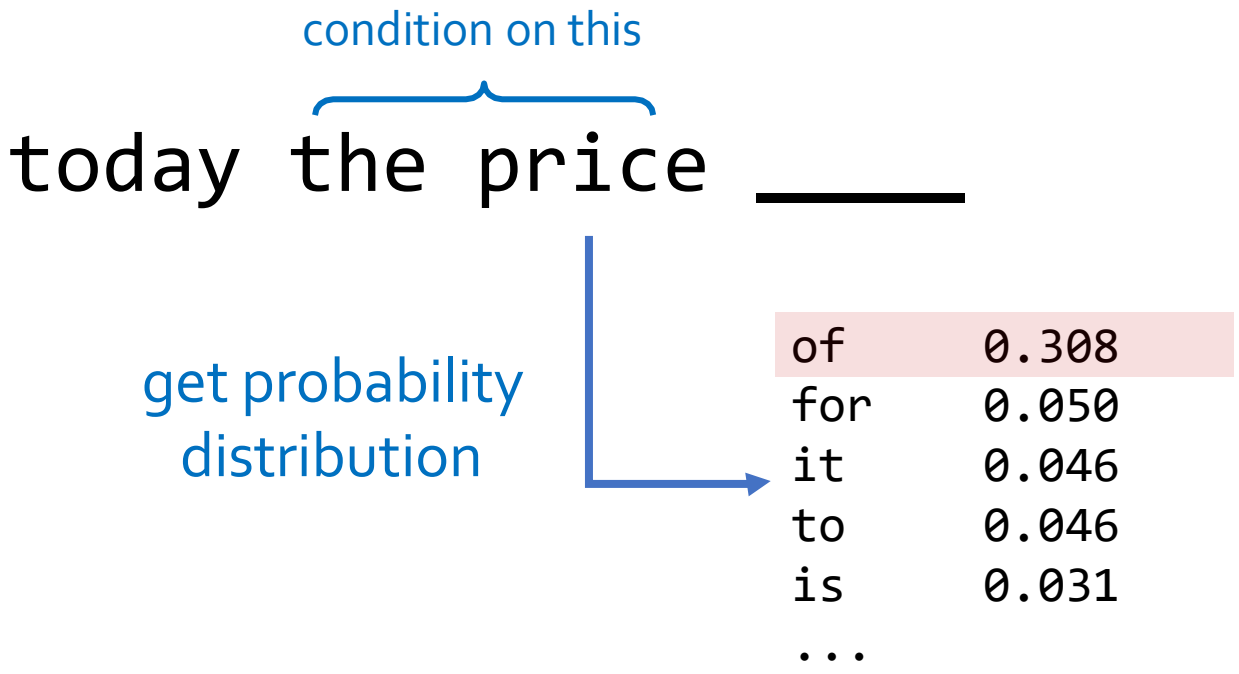
Otherwise, seems reasonable!

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

[adopted from Chris Manning]

N-Gram Models in Practice

- Now we can sample from this mode:



Sparsity problem: not much granularity in the probability distribution

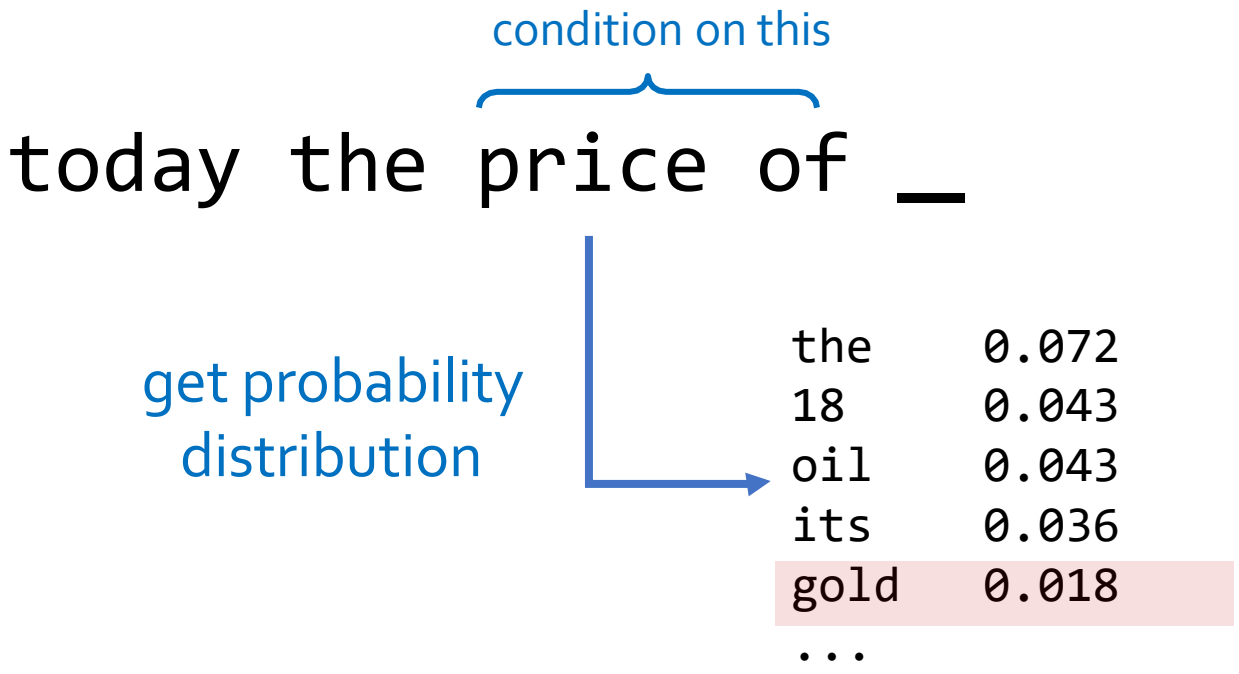
Otherwise, seems reasonable!

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

[adopted from Chris Manning]

N-Gram Models in Practice

- 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/

[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]

N-gram language models in practice

- 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

Goals & Overview of Today's Lecture

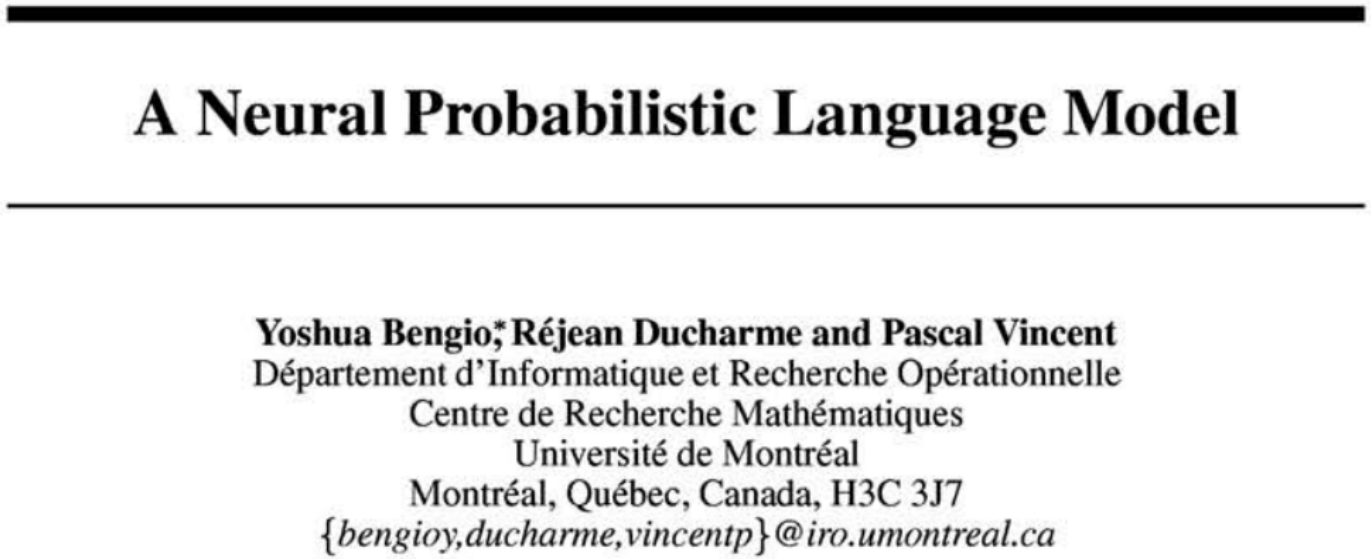
Goal: Understand the language modeling task, which will be used for pretraining and the modern approach to treating many NLP applications as text generation

- Language modeling
- Intrinsic evaluation of language models
- n-gram language modeling
- Smoothing
- Neural language modeling

Language Models: A History

- "Shallow" statistical 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

Reminder: Estimating N-gram probabilities

The probabilities can be computed by **relative frequency** estimation

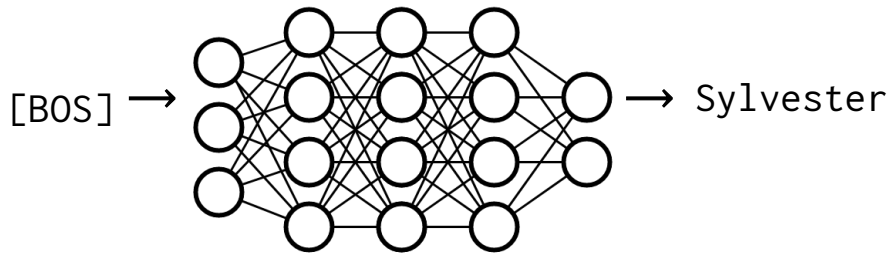
E.g., for *bigram* model (N=2):

$$\mathbb{P}(w_n|w_{n-1}) = \frac{\text{count}(w_{n-1}w_n)}{\sum_w \text{count}(w_{n-1}w)} = \frac{\text{count}(w_{n-1}w_n)}{\text{count}(w_{n-1})}$$

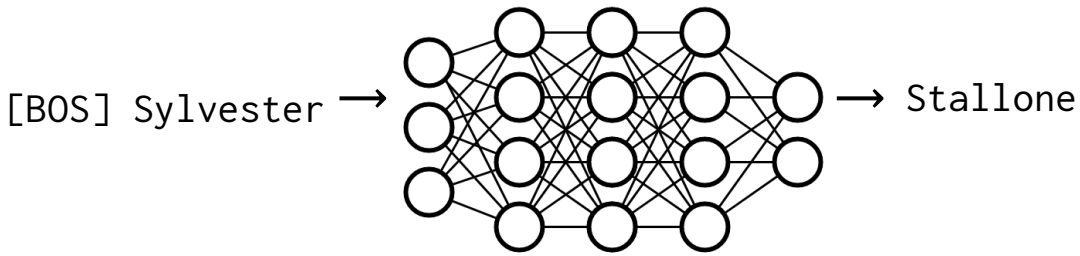
$$\mathbb{P}(w_1, \ldots, w_n) = \prod_{k=1}^{n} \mathbb{P}(w_k|w_{k-1})$$

Can we use more history by having a neural network predicting the next word?

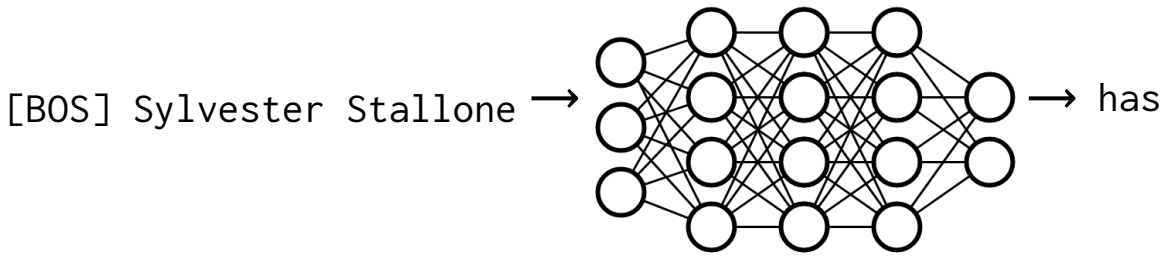
Neural language modeling



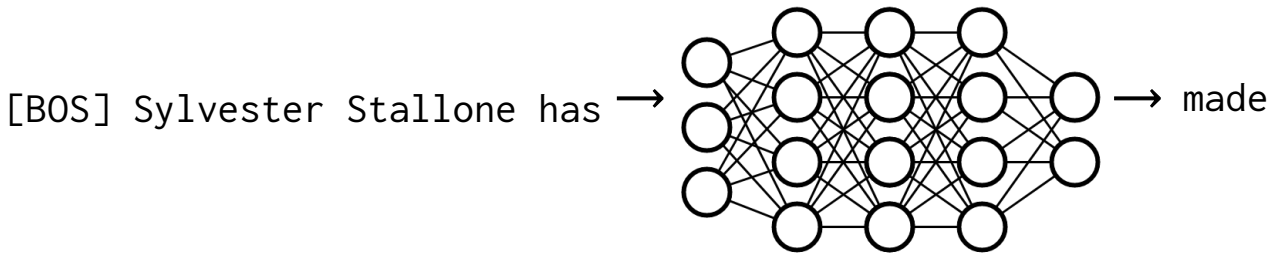
Neural language modeling



Neural language modeling



Neural language modeling



Reminder: Feedforward neural networks (FNNs)

$$\boldsymbol{y} = [y_1, \ldots, y_m]$$

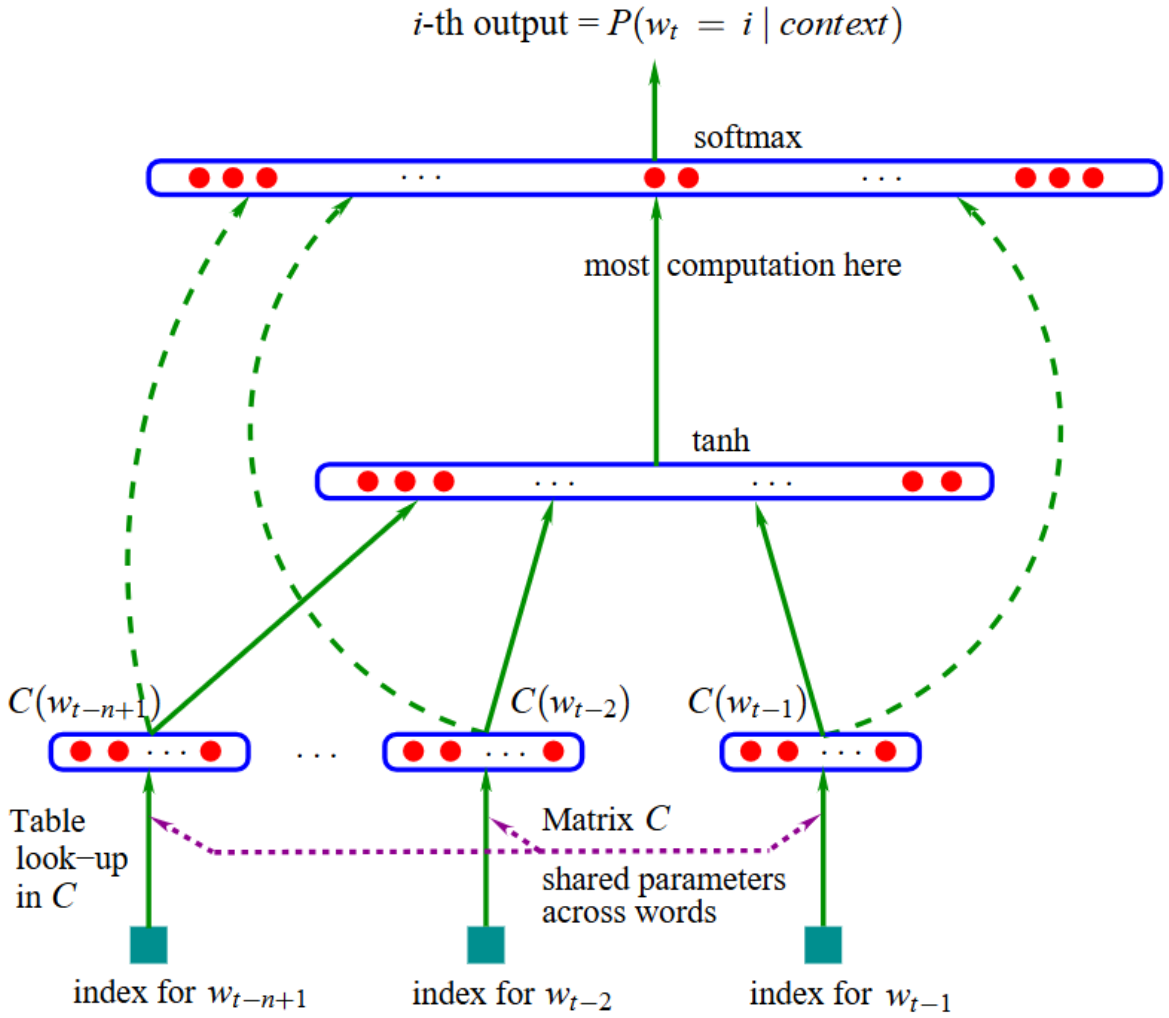
With neural LMs: The output space consists of all tokens in the vocabulary

$$W_o = \begin{bmatrix} w_{y_1}^T \\ \vdots \\ w_{y_m}^T \end{bmatrix} \in \mathbb{R}^{m \times d}$$

$$p(\boldsymbol{y}|x) = \text{softmax}(W_o \cdot g(W_1 f(x)))$$

$$y_{\text{pred}} = \text{argmax}_i \, p(\boldsymbol{y}|x)$$

Neural LMs with Feedforward neural networks (FNNs)



LMs w/ Recurrent Neural Nets

- Core idea: apply a model repeatedly

outputs — output distribution

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

hidden states

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

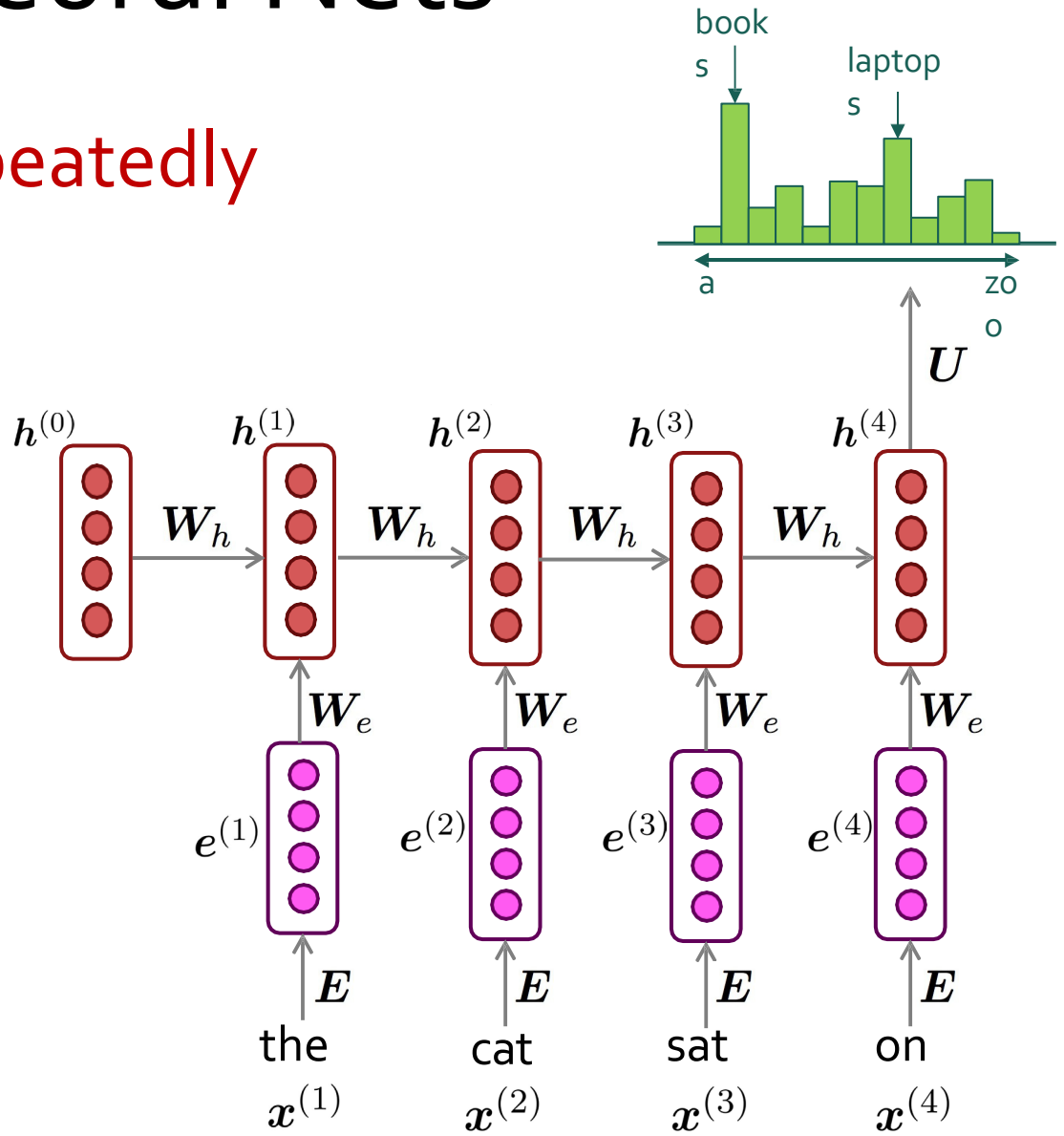
$\boldsymbol{h}^{(0)}$ is the initial hidden state

Input embedding — word embeddings

$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

words / one-hot vectors

$$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$$



RNNs in Practice

- RNN-LM trained on Obama speeches:

```
The United States will step up to the cost of a new
challenges of the American people that will share the fact
that we created the problem. They were attacked and so that
they have to say that all the task of the final days of war
that I will not be able to get this done.
```

https://medium.com/@samim/obama-rnn-machine-generated-political-speeches-c8abd18a2ea0