Word Vectors / Language Modeling

CSE 5525: Foundations of Speech and Language Processing

https://shocheen.github.io/cse-5525-spring-2026/

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.
  - You can use your late days if you like (max 3 per homework, total 5)
- Homework 2 is going to be released tonight.
  - Topic: Language Modeling with Transformers
- Project details will also be released tonight.
  - Did you receive email(s) from Thinking Machines/Tinker API?
  - Default Project – post-train a language model using Tinker API + proposal additional explorations.
  - Or, custom project – you design/propose the project, free to use Tinker.
  - Both have same work expectation but we give you the main idea in default.

How to represent the meaning of a word?

Desiderata

Let's look at some desiderata from lexical semantics, the linguistic study of word meaning

Word senses

lemma: the canonical form, dictionary form, or citation form of a set of word forms

basin (*plural* **basins**)

1. A wide bowl for washing, sometimes affixed to a wall. [quotations ▼] [synonym ▲]

   Synonym: sink

2. (*obsolete*) A shallow bowl used for a single serving of a drink or liquidy food. [quotations ▼]
3. A depression, natural or artificial, containing water. [quotations ▼]
4. (*geography*) An area of land from which water drains into a common outlet; drainage basin. [quotations ▼]
5. (*geography*) A shallow depression in a rock formation, such as an area of down-folded rock that has accumulated a thick layer of sediments

Source: wiktionary

word senses: meanings of the word

Polysemous words: words having multiple senses

Word sense disambiguation

Word Senses

Who Cares?

- Capturing such sense distinctions is important for many NLP problems
- Including very practical ones:
  - Information retrieval / question answering
    - bat care / how do I care for my bat?
  - Machine translation
    - bat: murciélago (animal) or bate (for baseball)
  - Text-to-speech
    - bass (stringed instrument) vs. bass (fish)

Relation: synonymity

Synonyms have the same meaning in some or all contexts.

- filbert / hazelnut
- big / large
- automobile / car
- vomit / throw up
- Water / H_2o

Two words are synonymous if they are substitutable for one another in any sentence without changing the truth conditions of the sentence [the situations in which the sentence would be true]

Word similarity

Not synonyms, but sharing some element of meaning

- belief, impression
- skiing, snowboarding

How similar two words are? ⇒ How similar the meaning of two sentences are?

Ask humans how similar two words are

word1	word2	similarity
vanish	disappear	9.8
behave	obey	7.3
belief	impression	5.95
muscle	bone	3.65
modest	flexible	0.98
hole	agreement	0.3

SimLex-999 dataset (Hill et al., 2015)

Antonyms

Senses that are opposites with respect to only one feature of meaning

Antonyms can

- Define a binary opposition or be at opposite ends of a scale
  - hot/cold
- Be reversives:
  - ascend/descend

Relation: word relatedness

Also called "word association"

- Words be related in any way, perhaps via a semantic frame or field
  - car, bicycle: similar
  - car, gasoline: related, not similar

Lexical semantics

- How should we represent the meaning of the word?
  - Dictionary definition
  - Lemma and wordforms
  - Senses
  - Relationships between words or senses
  - Word similarity, word relatedness
  - Semantic frames and roles
  - Connotation and sentiment

Lexical semantics

- How should we represent the meaning of the word?
  - Dictionary definition
  - Lemma and wordforms
  - Senses
  - Relationships between words or senses
  - Word similarity, word relatedness
  - Semantic frames and roles
    - *John hit Bill*
    - *Bill was hit by John*

Lexical Semantics

- How should we represent the meaning of the word?
  - Dictionary definition
  - Lemma and wordforms
  - Senses
  - Relationships between words or senses
  - Word similarity, word relatedness
  - Semantic frames and roles
  - Connotation and sentiment
    - *valence*: the pleasantness of the stimulus
    - *arousal*: the intensity of emotion
    - *dominance*: the degree of control exerted by the stimulus

	Valence	Arousal	Dominance
courageous	8.05	5.5	7.38
music	7.67	5.57	6.5
heartbreak	2.45	5.65	3.58
cub	6.71	3.95	4.24
life	6.68	5.59	5.89

Lexical Semantics are discrete and sparse

- Manually designed – need knowledge of the language under consideration.
- Hard to use in machine learning models which expect continuous inputs

Distributional Semantics

Artemia

A cluster of ________ is floating in the lake.

Biologists study the adaptation of ________ in saline environments.

The population of ________ fluctuates with the salinity of the water.

You can observe ________ in the shallows of the Great Salt Lake.

Other words that can appear in this context: *algae*, *microorganisms*, *shrimp*

Other words that can appear in this context: *algae*, *microorganisms*, *shrimp*

We can conclude:

➔ Artemia is a simpler form of life found in aquatic environments like the Great Salt Lake similar to algae, microorganisms, shrimp

Distributional hypothesis

[Joos, 1950; Harris, 1994; Firth, 1957]

Words that occur in **similar contexts** tend to have **similar meanings**

Distributional Semantics

The Distributional Hypothesis

- Words that are used and occur in the same **context** tend to have similar meaning
- Similarity-based generalization: children can figure out how to **use** words by generalizing about their **use** from distributions of similar words
- The more semantically similar words are, the more distributionally similar they are
- **What is context**? Informally: whatever you can get your hands on that makes sense!

A Sparse Representation

Counting contexts

- Given a vocabulary of V words
- Let $f_i, i = 1$ …n be a binary (or count) indicator for the presence (or count) of the i-th word in the vocabulary in the context
- Represent a word w as:

$$w = (f_1, f_2, f_3, \dots, f_n)$$

where f_i are computed in contexts of all uses of w

- For example:

Artemia = $(1,1,0,\dots)$

lake saline biologists

Learning from Raw Data

Word Vectors

Raw Data

- Raw text = human-created language without any additional annotation
- A natural by-product of human use of language
- Abundant in text form for many domains and scenarios, but not for all
- How can we learn without any annotation? What kind of representations can we get? How can we use them?
- Key idea: self-supervised learning

Raw Data

Self-supervised Learning

- Given: raw data without any annotation
- Formalize a prediction training objective that is using this data only
- Common approach: given one piece of the data, predict another
- The prediction task is often not interesting on its own
- But the learned representations are!
- Big advantage: can use as much data as you can find and have compute for
- In contrast, supervised learning relies on enriching the data with human annotations

word2vec

word2vec is a **software** package (https://code.google.com/archive/p/word2vec/) that includes **two algorithms** [Mikolov et al., 2013a; Mikolov et al., 2013b]

1. **Skip-gram** with negative sampling (SGNS) [now]
2. Continuous Bag-Of-Words (**CBOW**) [in the readings]

These algorithms are often loosely referred to as word2vec

The intuition behind word2vec

Instead of counting how often each word `w` occurs near another word, *artemia*, train a classifier on a binary prediction task:

➔ Is word `w` likely to show up near *artemia*?

Specifically, with skip-gram

- Use the target word & a neighboring context word (from a corpus) as positive examples
- Randomly sample other words as negative examples
- Train a classifier to distinguish those two cases
- Use the learned weights as the embeddings

Skip-gram classifier – Intuition

```
... lemon,   a [tablespoon of apricot jam,      a] pinch ...
               c1          c2      w      c3       c4
```

$$p(+|w,c) = 1$$

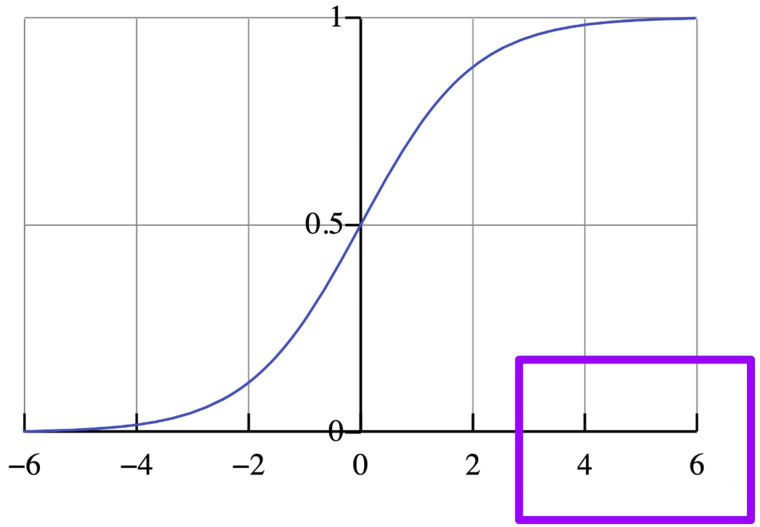
$$f(z) = \frac{e^z}{1+e^z} \text{ ... logistic function}$$

$$p(+|\text{apricot,tablespoon}) = 1$$

$$p(+|\text{apricot,of}) = 1$$

$$p(+|\text{apricot,jam}) = 1$$

$$p(+|\text{apricot,a}) = 1$$



embedding similarity high
⇒ probability high too

Skip-gram classifier – Intuition

```
... lemon,   a [tablespoon of apricot jam,      a] pinch ...
               c1          c2      w      c3        c4
```

$$p(+|w,c) = 1$$

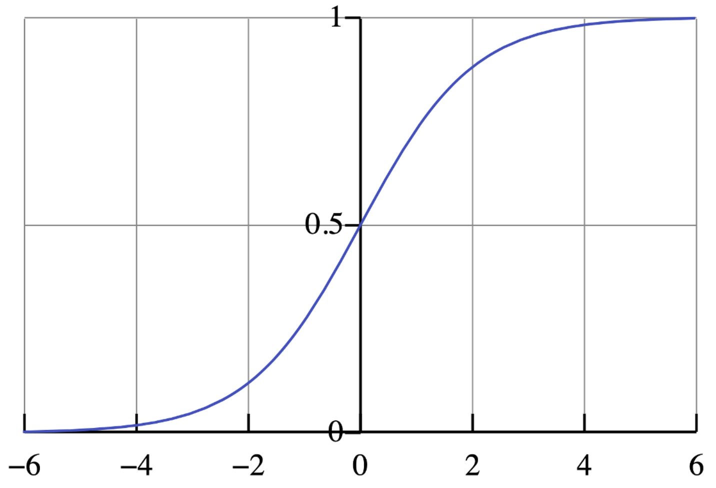
$$f(z) = \frac{e^z}{1+e^z} \quad \ldots \text{logistic function}$$

$p(+|\text{apricot,tablespoon}) = 1$

$p(+|\text{apricot,of}) = 1$

$p(+|\text{apricot,jam}) = 1$

$p(+|\text{apricot,a}) = 1$



$$\text{similarity}(w,c) \approx c \cdot w$$

$$p(+|w,c) = \frac{e^{c \cdot w}}{1+e^{c \cdot w}}$$

$$c \cdot w \to \infty \Rightarrow p(+|w,c) \to 1$$

Skip-gram learning algorithm

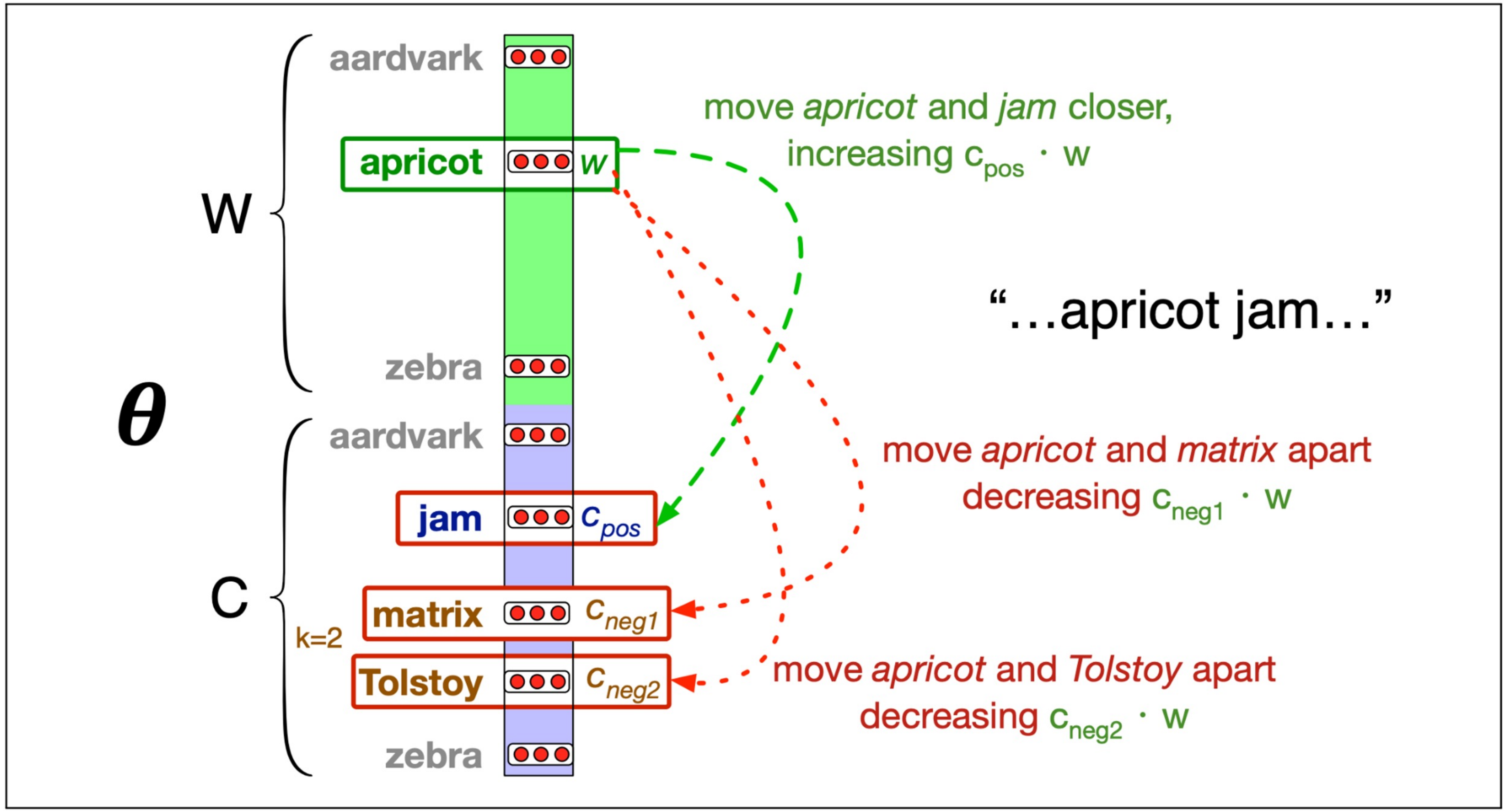
Given:

- Set of **positive** and **negative examples**
- An **initial** set of **random embeddings**

The goal of the learning algorithms it to **adjust** those embeddings to:

- Maximize the similarity of the target word, context word pairs `(w,c_pos)` drawn from the positive examples
- Minimize the similarity of `(w,c_neg)` pairs from the negative examples

Skip-gram learning algorithm – Stochastic gradient descent



Word Embeddings

How to Use Them?

- Word embeddings are often input to models of various end applications
- They provide lexical information beyond the annotated task datasets, which is often small
- Can be kept fixed or fine tuned (i.e. trained) with the task network
- Can also be input to sentence embedding models

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

Visualizations

Project embeddings to a 2D space and visualize them

- [How to Use t-SNE Effectively]()

Check k-nearest neighbors

[Li et al., 2016]

Extrinsic Evaluation

Initialize an NLP model's embedding layer and train

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

Instrinsic: Measuring Vector Similarity

- Similarity can be measured using vector distance measures
- Two typical examples: Euclidean distance and cosine similarity
- Cosine similarity:

$$\text{similarity}(w, u) = \frac{w \cdot u}{\parallel w \parallel \parallel u \parallel} = \frac{\sum_{i=1}^{n} w_i u_i}{\sqrt{\sum_{i=1}^{n} w_i^2} \sqrt{\sum_{i=1}^{n} u_i^2}}$$

which gives values between -1 (completely different), 0 (orthogonal), and 1 (completely identical)

Intrinsic Evaluation

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

similarity (embeddings)
1.1
0.5
0.3
1.7
0.98
0.3

Spearman's rho (human ranks, model ranks)

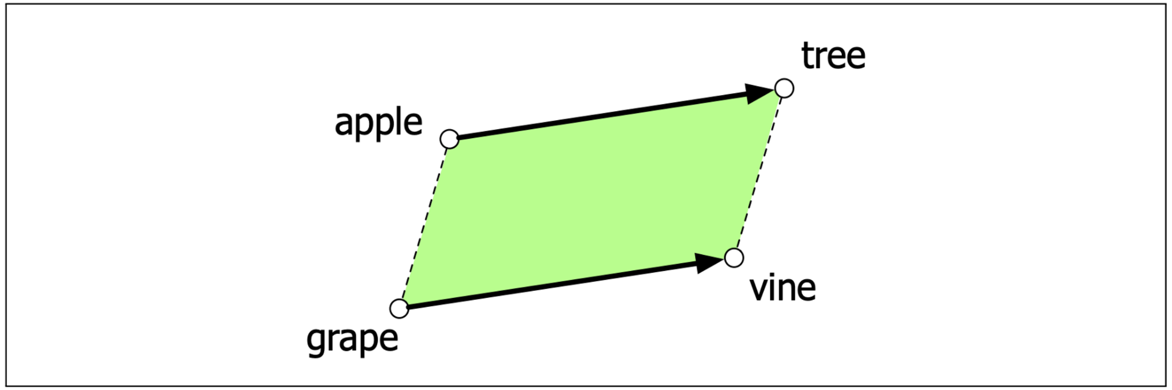
- WS-353 (Finkelstein et al. '02)
- MEN-3k (Bruni et al. '12)
- SimLex-999 dataset (Hill et al., 2015)

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

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

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

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]

Other kinds of static embeddings

Fasttext [Bojanowski et al, 2017]

- Limitation of word2vec: a distinct vector representation for each word, but words may share information even if they don't appear in context with each other.
- An extension which takes into account subword information
- https://fasttext.cc/

GloVe [Pennington et al., 2014]

And many, many more...

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

The

The cat

The cat sat

The cat sat on

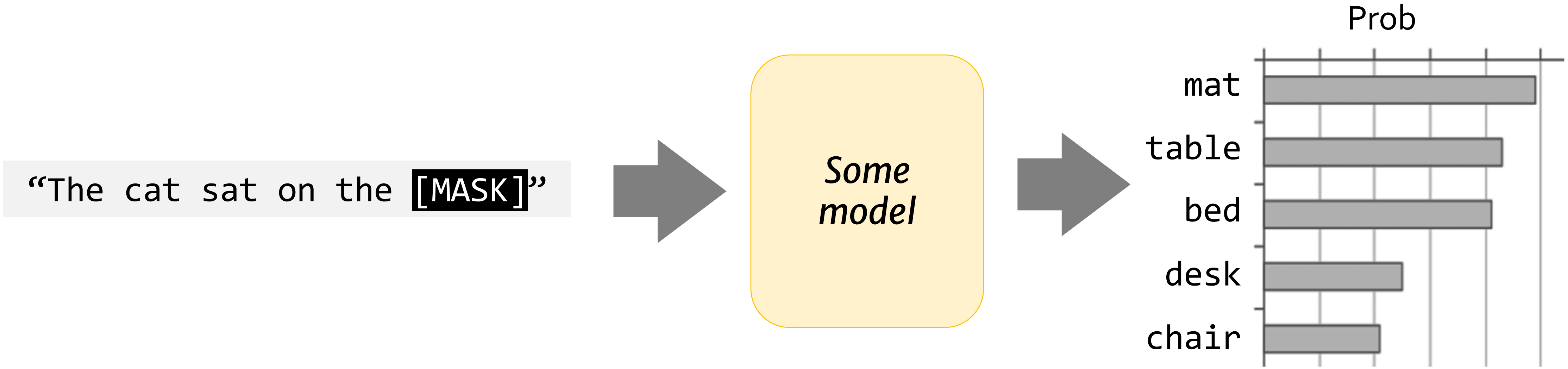
The cat sat on __?__

The cat sat on the mat.

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

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

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



But more broadly, we want to model

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

Apply chain rule

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

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.

- 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!

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

- 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

inverse ⇒ higher likelihood means lower perplexity

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

held-out test set

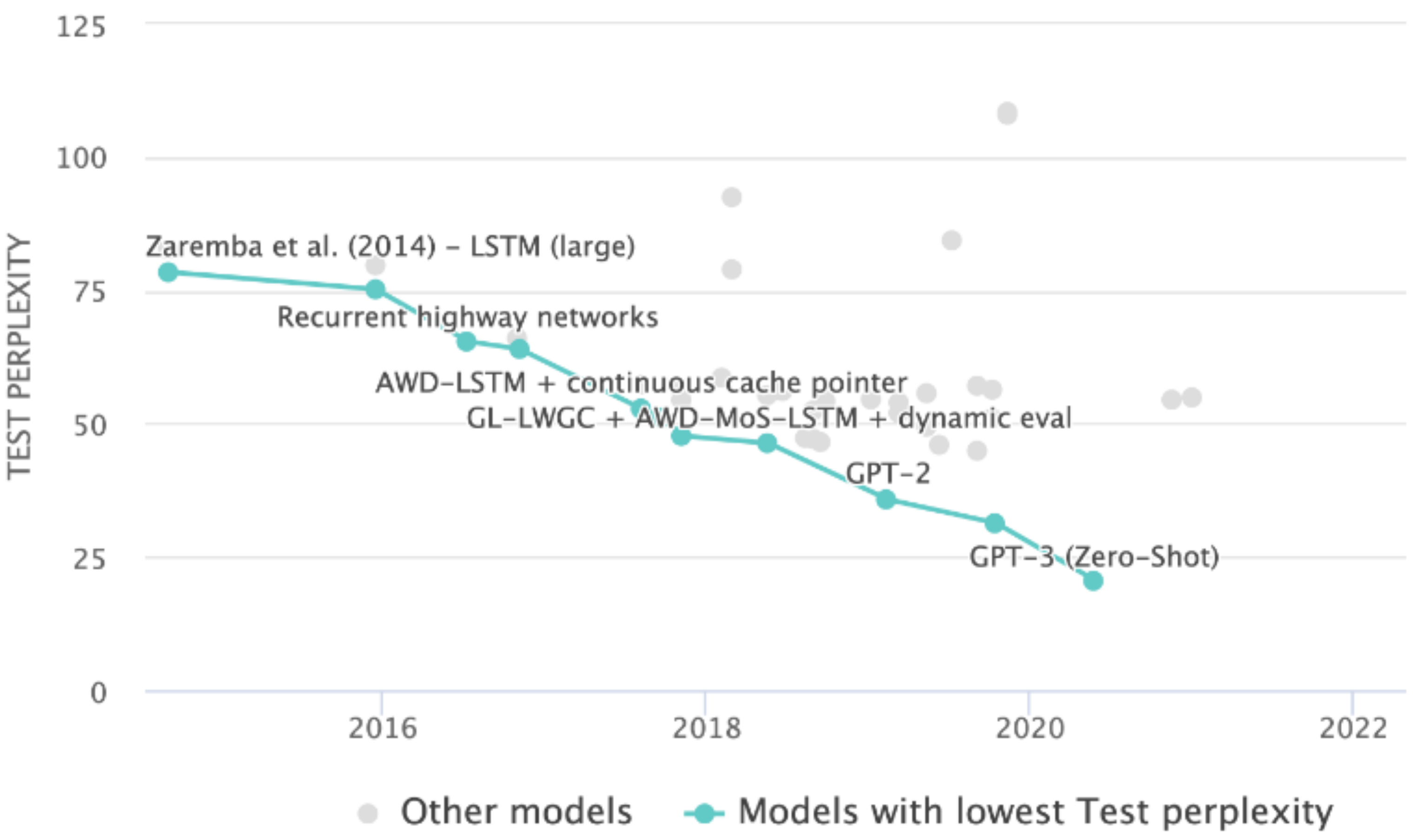
n-th root to normalize by the number of words; the likelihood gets smaller the longer the text (unwanted)

- 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 5–200
- 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.

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

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

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

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

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

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

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):

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

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

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

Smoothing & Discounting

unknown bigram

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

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

unknown bigram

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

known word

$\Rightarrow$ Zero probability and hence of the entire sequence

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

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

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

[adopted from Chris Manning]

N-Gram Models in Practice

- Now we can sample from this mode:

condition on this

$\overbrace{\texttt{the price}}^{\text{condition on this}}$

`today the price ____`

get probability distribution

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

N-Gram Models in Practice

- 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
...	

* 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

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

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):

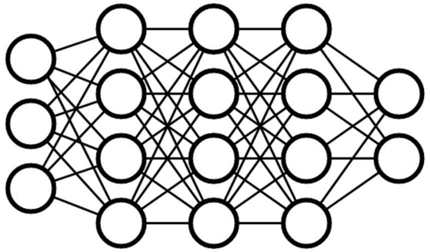
$$\mathbb{P}(w_n | w_{n-1}) = \frac{\text{count}(w_{n-1} w_n)}{\sum_w \text{count}(w_{n-1} w)} = \frac{\text{count}(w_{n-1} w_n)}{\text{count}(w_{n-1})}$$

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

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

Neural language modeling

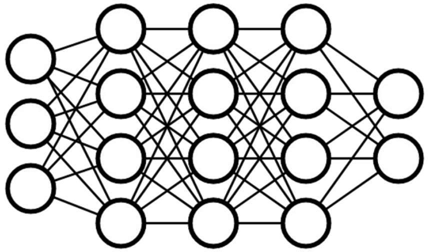
[BOS] →



→ Sylvester

Neural language modeling

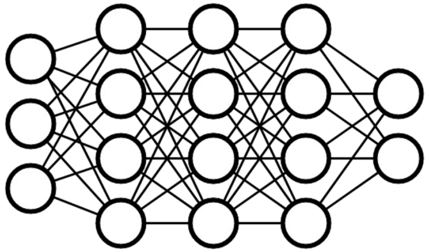
[BOS] Sylvester →



→ Stallone

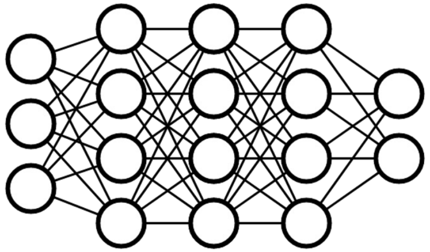
Neural language modeling

[BOS] Sylvester Stallone →



→ has

Neural language modeling

`[BOS] Sylvester Stallone has` →  → `made`

Reminder: Feedforward neural networks (FNNs)

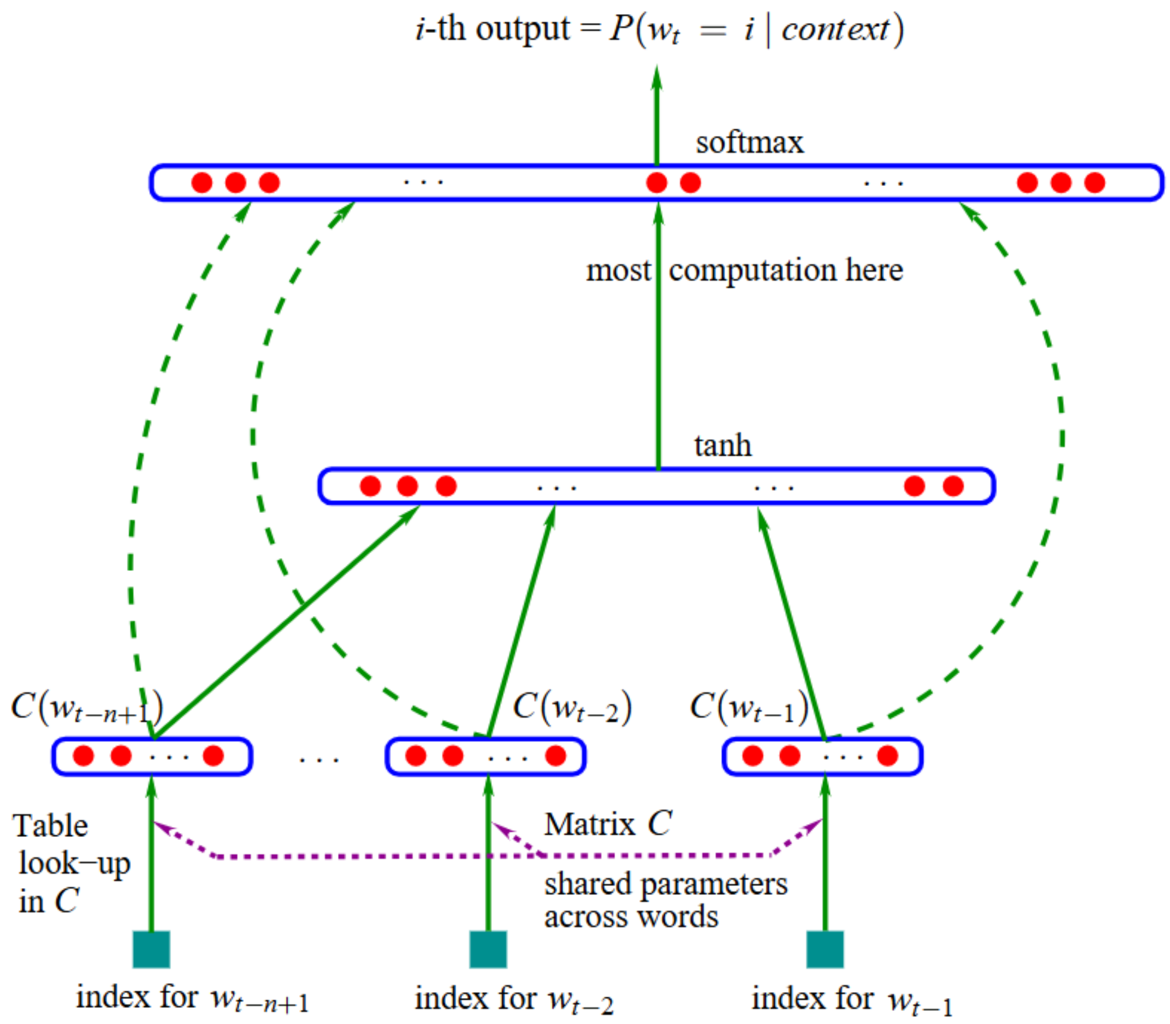
$\boxed{\boldsymbol{y} = [y_1, \ldots, y_m]}$ With neural LMs: The output space consists of all tokens in the vocabulary

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

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

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

Neural LMs with Feedforward neural networks (FNNs)



LMs w/ Recurrent Neural Nets

- Core idea: apply a model repeatedly

outputs — output distribution

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

hidden states

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

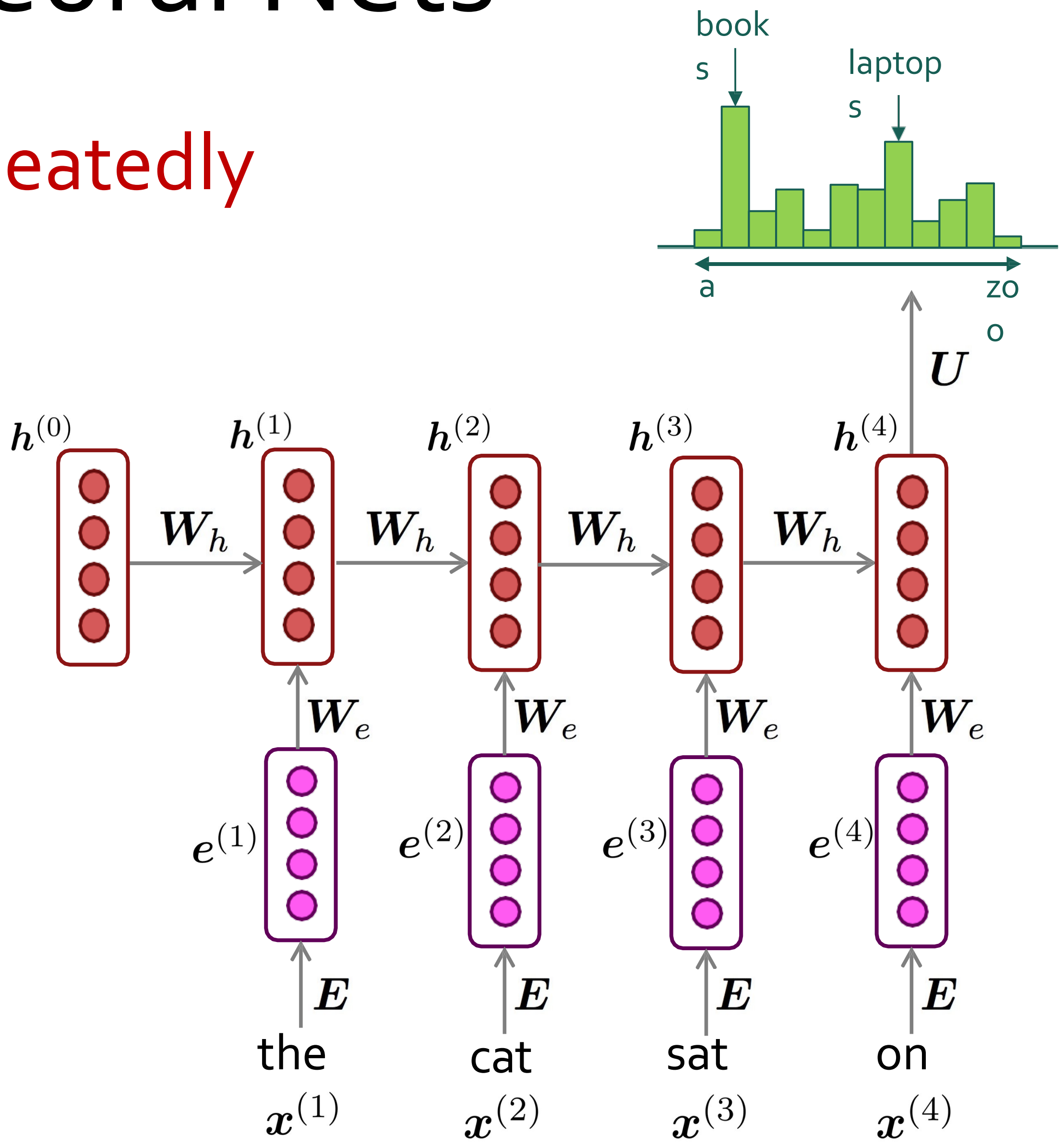
$\boldsymbol{h}^{(0)}$ is the initial hidden state

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