Neural LMs and Word Embeddings

CS 685, Fall 2025

Advanced Natural Language Processing

https://people.cs.umass.edu/~brenocon/cs685_f25/

Brendan O'Connor

College of Information and Computer Sciences

University of Massachusetts Amherst

- Office hours start this week
- No live zoom; lecture recordings instead (Echo360/Canvas)

- Hope HW1 is going well!

- Approximate midterm dates
  - Midterm #1: Early October
  - Midterm #2: Mid-November
  - *(might be in-class, TBD)*

Problems with n-gram Language Models

Sparsity Problem 1

Problem: What if *"students opened their* $\boldsymbol{w}_j$*"* never occurred in data? Then $\boldsymbol{w}_j$ has probability 0!

(Partial) Solution: Add small δ to count for every $\boldsymbol{w}_j \in V$. This is called *smoothing*.

$$p(w_j \mid \text{students opened their}) = \frac{\text{count(students opened their } w_j)}{\text{count(students opened their)}}$$

Sparsity problem is really bad for n-grams!

Smoothing doesn't address the real problem: *composition* of language

Problems with n-gram Language Models

- We treat all words / prefixes independently of each other

students opened their ___

pupils opened their ___

scholars opened their ___

undergraduates opened their ___

students turned the pages of their ___

students attentively perused their ___

…

Shouldn't we *share information* across these semantically-similar prefixes?

Today

- *Question*: how to flexibly represent word meanings?
  - *Word embeddings*, a key tool for all major NLP models
  - ... which work because of the principle of *distributional similarity*

- *Key idea*: automatically induce word meanings from unlabeled text

- Two-ish models that use or produce word embeddings
  - 1. Markov neural LM (left-to-right)
  - 2. Skip-gram LM ("word2vec")

- Why?
  - Better left-to-right LMs
  - Word embeddings can be used directly (e.g. for lexical semantics or text classification; more on Thursday)

What is a *pawpaw* ?

I. Look it up in a dictionary

https://www.merriam-webster.com/

https://www.oed.com/

https://en.wiktionary.org/

pawpaw noun

Save Word

paw·paw

variants: *or less commonly* **papaw**

Definition of *pawpaw*

1 \ pə-ˈpȯ \ **:** PAPAYA

2 \ ˈpä-(ˌ)pȯ , ˈpȯ- \ **:** a North American tree (*Asimina triloba*) of the custard-apple family with purple flowers and an edible green-skinned fruit

also **:** its fruit


Lemma
Word Senses
Definition
pawpaw noun
Save Word
paw·paw
variants: or less commonly papaw
Definition of pawpaw
1 \ pə-ˈpȯ \ : PAPAYA
2 \ ˈpä-(ˌ)pȯ, ˈpȯ- \ : a North American tree (Asimina triloba) of the custard-apple family with purple flowers and an edible green-skinned fruit
also : its fruit

II. Look it at how its used

“ **<u>Pawpaw</u>**, Most Neglected American Fruit.” — NYTimes <u>1922</u>

“ **<u>Pawpaw</u>** Recommended by U.S. Food Experts, Along With Persimmon, as War Nutrition” — NYTimes <u>1942</u>

“ The **<u>pawpaw</u>** is also pollinated by flies and other insects rather than by honeybees…”— NYTimes <u>2020</u>

“Many people also cook with ripe **<u>pawpaws</u>**, making bread, beer, ice cream, or this **<u>pawpaw</u>** pudding…” — NYTimes <u>2020</u>

II. Look it at how its used

“ *Pawpaw*, Most Neglected **American Fruit** .” — NYTimes 1922

“ *Pawpaw* Recommended by U.S. Food Experts, Along With **Persimmon** , as War **Nutrition** ” — NYTimes 1942

“ The *pawpaw* is also **pollinated** by **flies** and other insects rather than by honeybees…”— NYTimes 2020

“Many people also **cook** with **ripe** *pawpaws* , making **bread** , **beer**, **ice cream** , or this *pawpaw* **pudding** …” — NYTimes 2020

Aspects of word meaning

Synonyms

- couch / sofa
- oculist / eye - doctor
- car / automobile
- water / H_2O
- draft / draught

Antonyms

- yes / no
- dark / light
- hot / cold
- up / down
- clip / clip

Aspects of word meaning

Similarity

· cat / dog

· cardiologist / pulmonologist

· car / bus

· sheep / goat

· glass / mug

Relatedness

· coffee / cup

· waiter / menu

· farm / cow

· house / roof

· theater / actor

Aspects of word meaning

- Connotation: the affective meaning of a word
- Osgood (1957)'s three-dimensional model:
  - Valence
    - unhappy, annoyed <---------------> happy, satisfied
  - Arousal
    - calm <---------------> excited
  - Dominance
    - awed, influences <---------------> controlling

	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

Learning word representations

- How to get word meanings?
  - Lexical resources like WordNet: dictionary-like databases of word synonyms & other word-to-word relationships, constructed manually
    - Can sometimes help, but typically don't cover all words or meanings any particular task needs
- Landauer and Dumais (citing many philosophers, etc.): it's crazy how much knowledge humans have. You can't look it all up in a dictionary!
- OK, can we *learn* the word representations instead?

Distributional Semantics

“You shall know a word by the company it keeps!” — Firth (1957)

Intuitions: Harris (1954)

“If A and B have almost identical environments except chiefly sentences which contain both, we say they are synonyms: *oculist* and *eye- doctor* .”

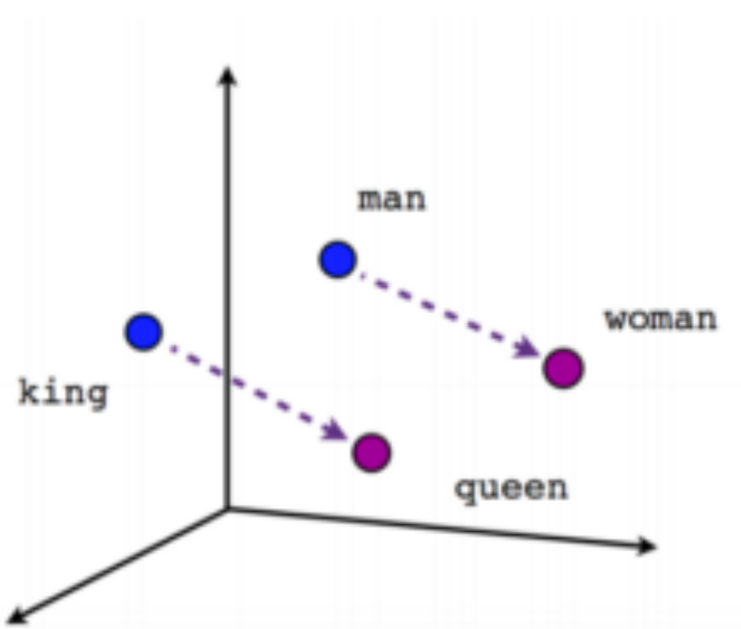
Learning word representations

- Could we automatically *learn* word meanings?
  - 1. We'd like to generalize word meanings beyond individual words, and
  - 2. Information from nearby words gives information about a word

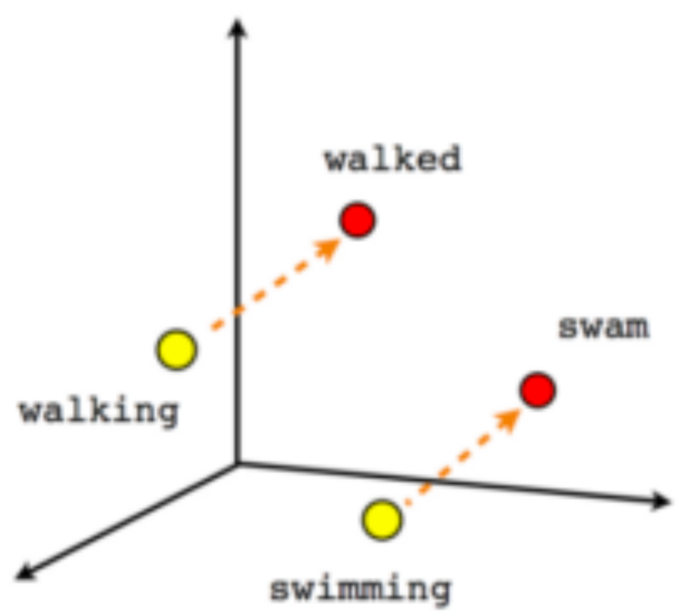
Word embeddings

- Represent words with low(ish)-dimensional vectors called **embeddings**
- Every word in vocabulary has a vector — these are model parameters.
    - Ideally: semantically similar words get similar vectors. Or other semantic properties??

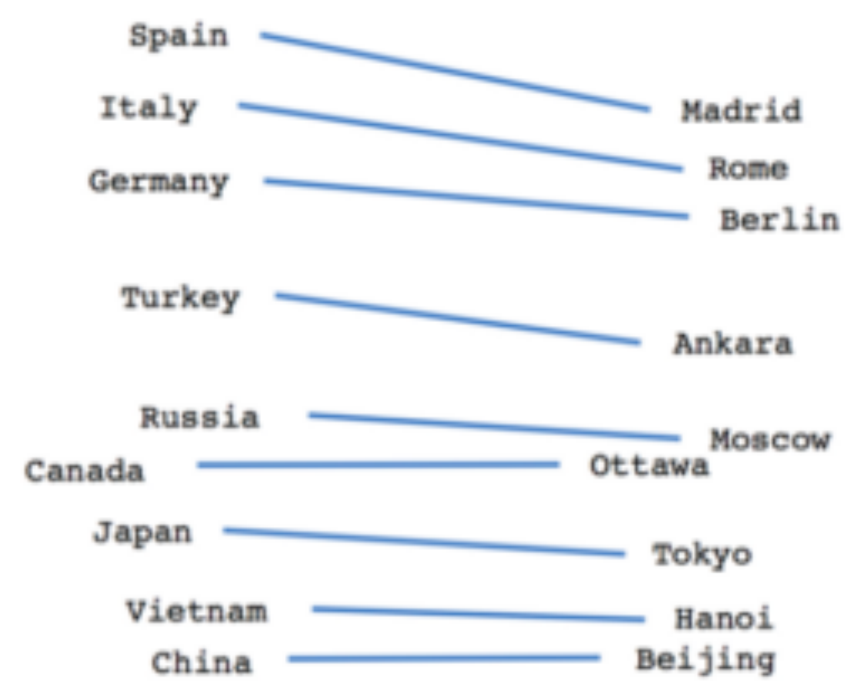
king =
[0.23, 1.3, -0.3, 0.43]



Male-Female



Verb tense



Country-Capital

Left-to-right LM as linear softmax

- Instead of only n-gram count ratios, model the next-word as softmax over the vocabulary.
- We can use anything to help predictions: features (Rosenfeld 1996) or MLP neural net (Bengio et al. 2003) or weird neural net (Vaswani 2017: self-attention) to compose ***y***

Output layer (softmax)

$$\hat{P}(w_t|w_{t-1},\cdots w_{t-n+1}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}.$$

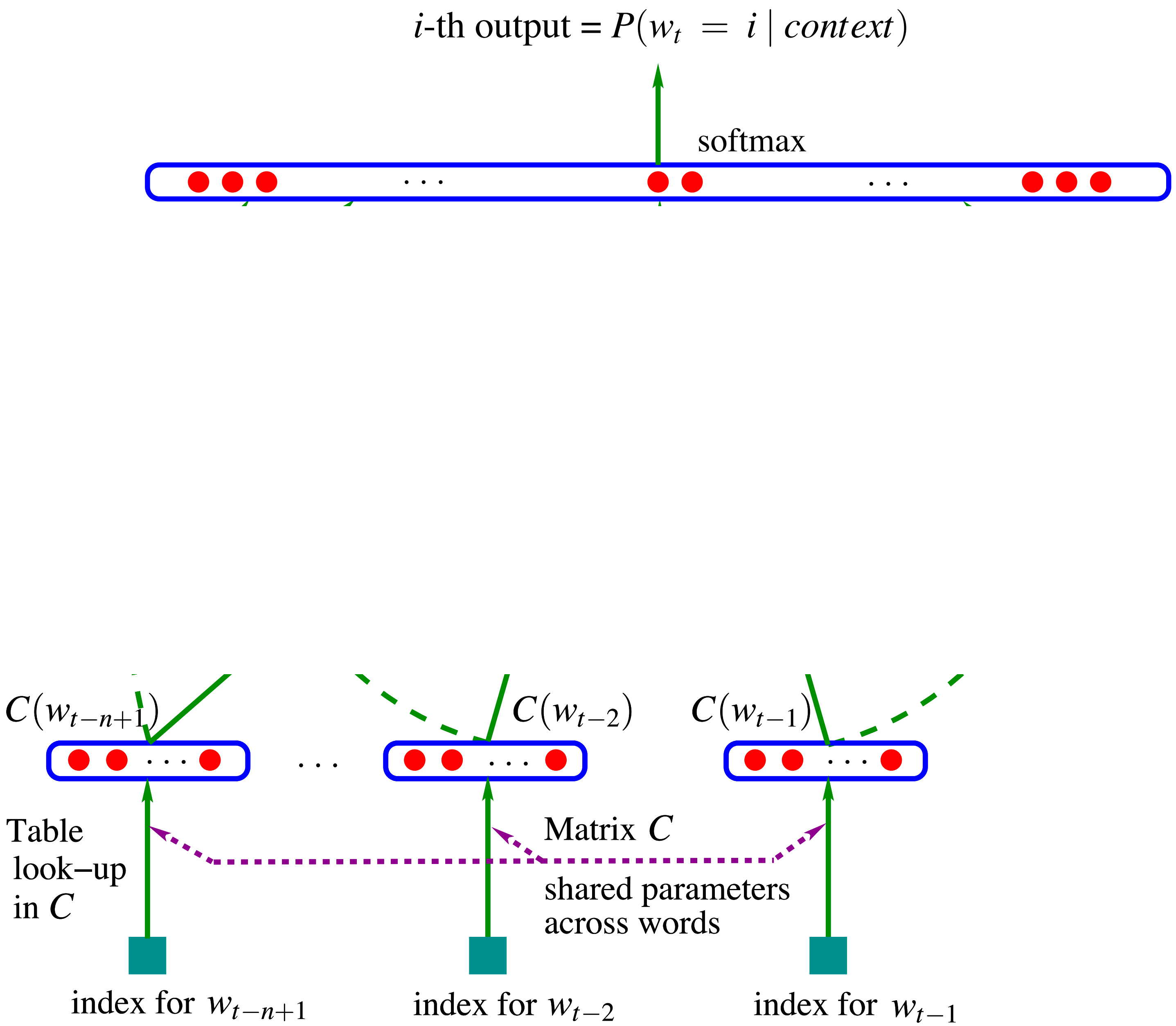
y: length V vector

$y = f(w_{t-1}, \ldots w_{t-n+1})$

- Can use any information from the left context

Bengio et al. 2003: Markov word embedding LM

Key idea: represent words on left as **vectors.** Learn a vector for each word in the vocabulary. Better perplexity than an n-gram LM!



Output layer (softmax)

$$\hat{P}(w_t \mid w_{t-1}, \cdots w_{t-n+1}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}.$$

linear layer

(ignore today) hidden layer, size h

$$y = b + Wx \; + \; U \tanh(d + Hx)$$

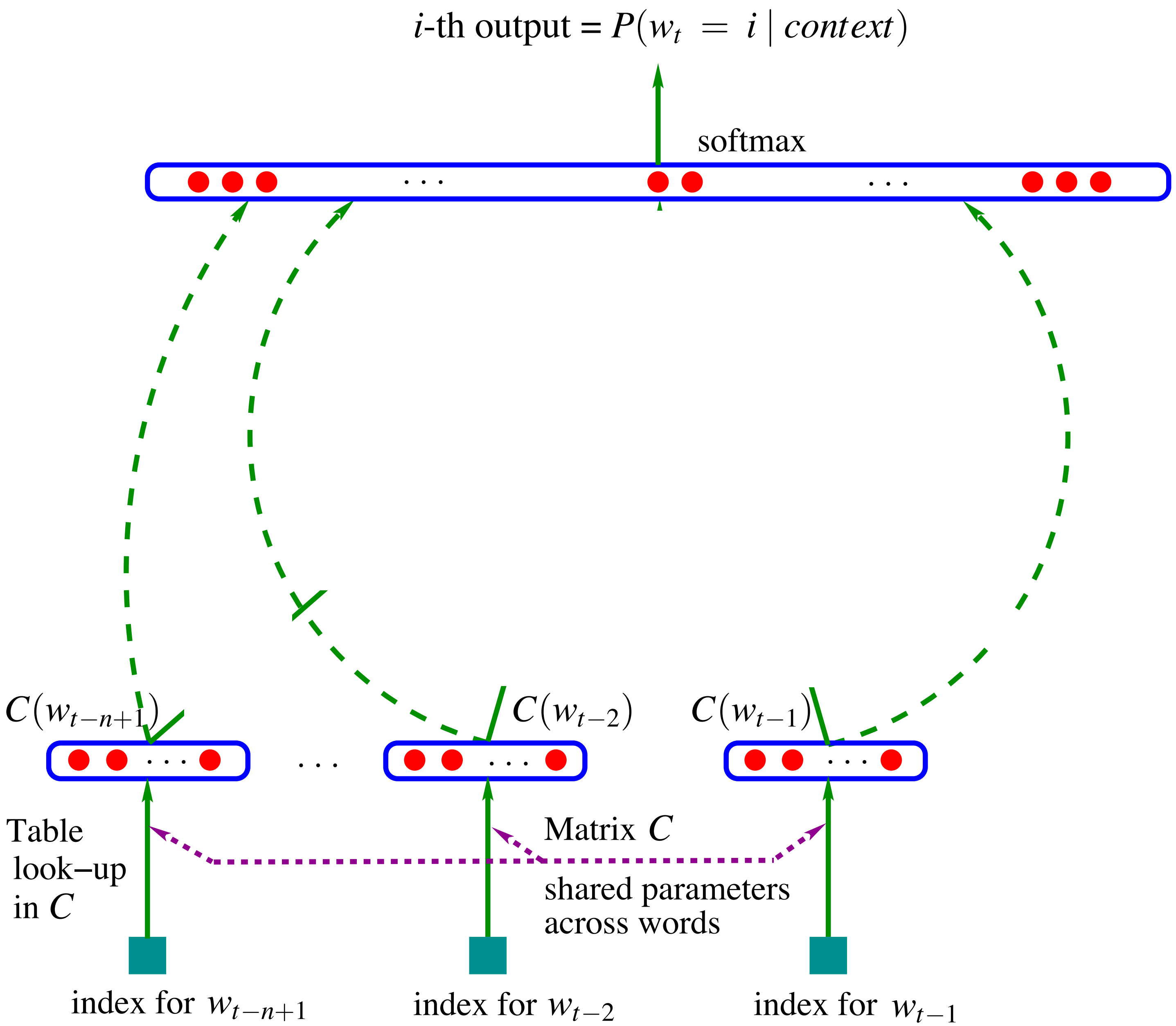
$$x = (C(w_{t-1}), C(w_{t-2}), \cdots, C(w_{t-n+1}))$$

Word vector lookup layer with concatenation

$$C(i) \in \mathbb{R}^m$$ Word embedding parameters

Bengio et al. 2003: Markov word embedding LM

Key idea: represent words on left as **vectors.**
Learn a vector for each word in the vocabulary.



Output layer (softmax)

$$\hat{P}(w_t | w_{t-1}, \cdots w_{t-n+1}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}.$$

linear layer

(ignore today) hidden layer, size h

$$y = \underbrace{b + Wx}_{\text{linear layer}} + \underbrace{U \tanh(d+Hx)}_{\text{hidden layer}}$$

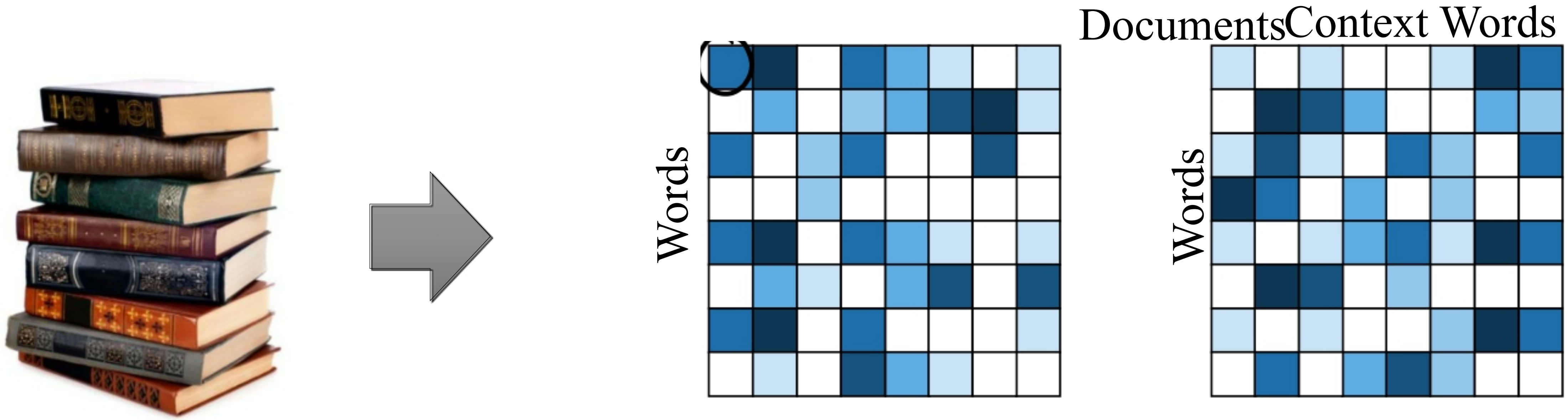
$$x = (C(w_{t-1}), C(w_{t-2}), \cdots, C(w_{t-n+1}))$$

Word vector lookup layer with concatenation

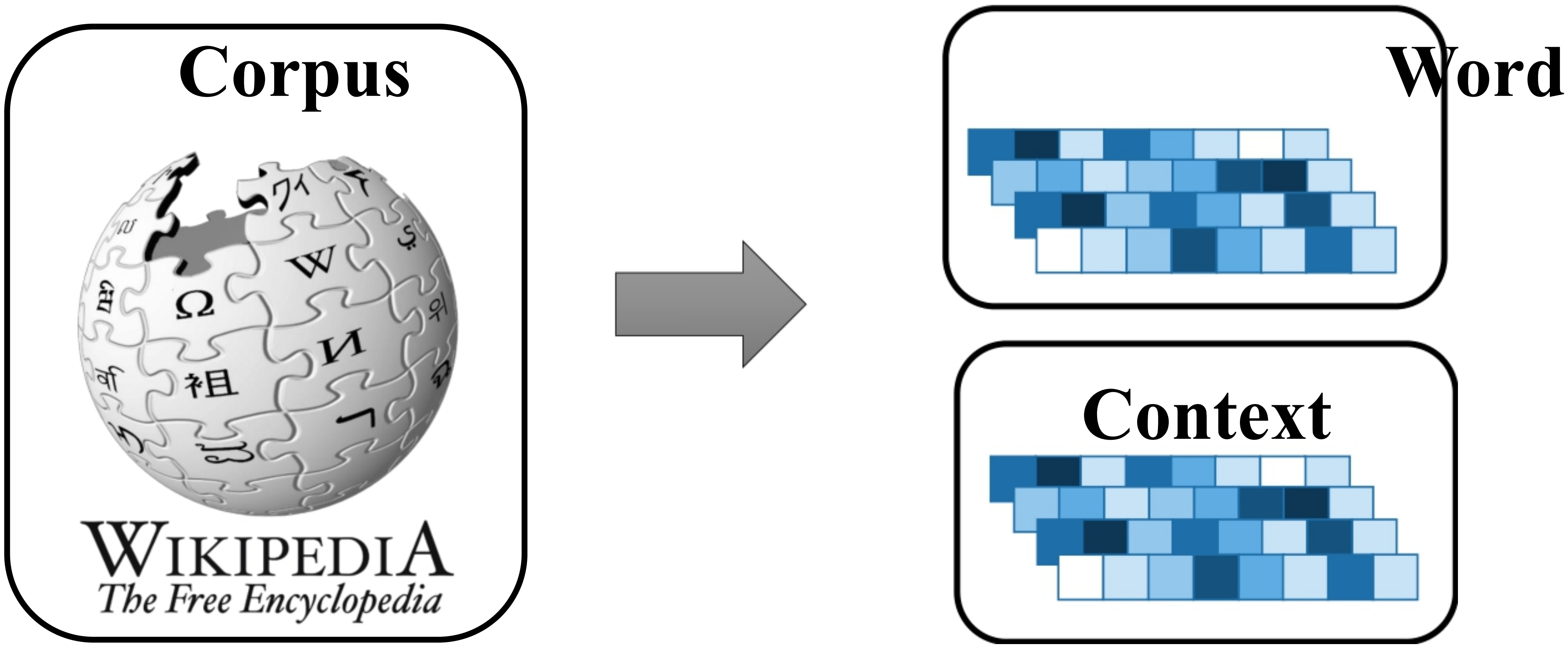
$C(i) \in \mathbb{R}^m$ Word embedding parameters

- Learning: follow the gradient of the negative log-likelihood (more on this in coming weeks)

Build vectors based on context



Neural Word Embeddings



Skip- Gram with Negative Sampling (SGNS)

The brown fox **jumps** over the lazy dog

<u>SG</u> NS: Skip- Gram Model

The brown fox **jumps** over the lazy dog.

Context Window Size = 2

<u>SG</u> NS: Skip- Gram Model

The brown fox **jumps** over the lazy dog.

Context Window Size = 2

Simple idea: from a word, predict its context words!
(A funny type of language model.)
Learn a vector that's good at that. Similar words should get similar vectors.

Key idea: use unlabeled text as *implicitly supervised data*

- A word *s* near *apricot*
  - Acts as gold ‘correct answer’ to the question
  - “Is word *w* likely to show up near *apricot*?”
- No need for hand-labeled supervision
- The idea comes from **neural language modeling**
  - Bengio et al. (2003)
  - Collobert et al. (2011)

Modeling goal

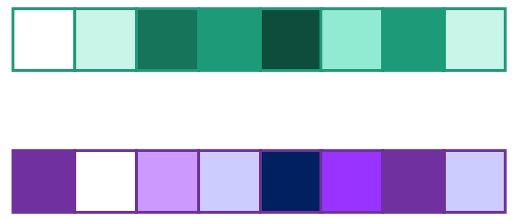
- Given a (word, context) tuple
  - [+] (apricot, jam) <- observed
  - [–] (apricot, aardvark) <- unseen
- Want binary probability
  - P(c | w) for a real context [+])
  - 1-P(c | w) for a “fake”, unseen context [–])
- Let u_t and v_c be their vectors.
- P(c | w) = $\sigma(u_w' v_c)$: logistic in their *affinity/similarity*
- Maximize P(c | w) for all (w, c) pairs

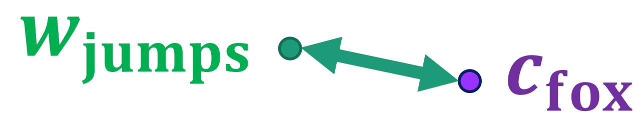
SGNS : Negative Sampling

Co-occurrence **jumps** , **fox**:

$w_{\mathbf{jumps}}$

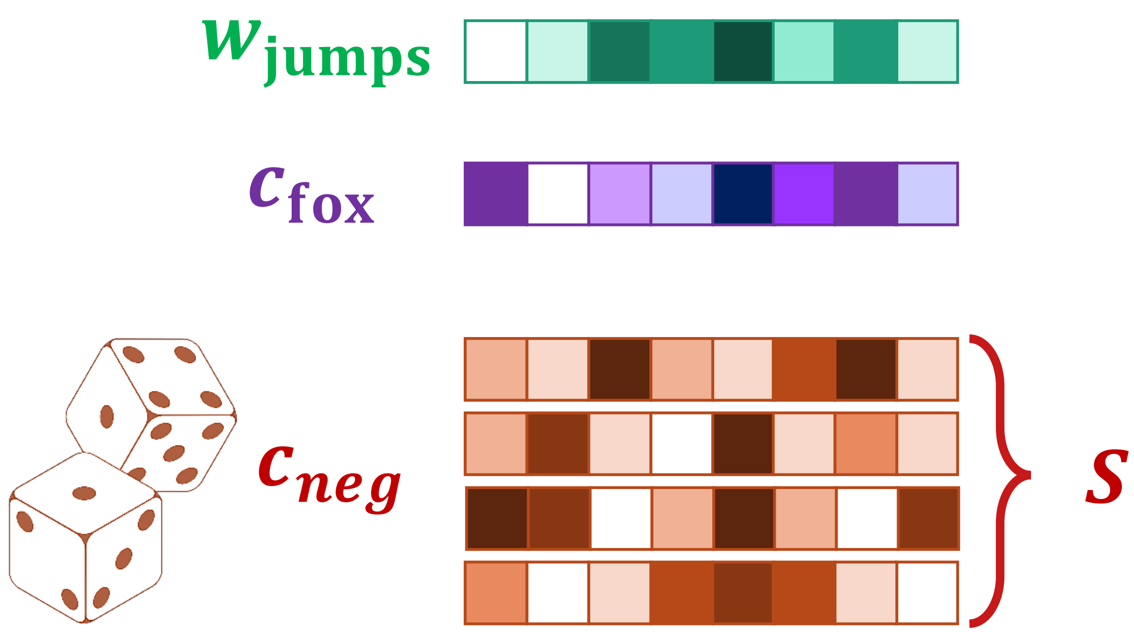
$c_{\mathbf{fox}}$

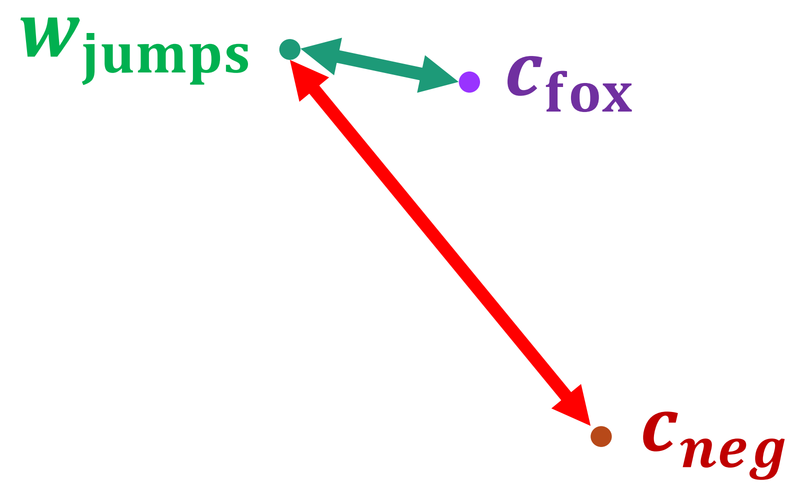




SG**NS** : Negative Sampling

Co-occurrence **jumps** , **fox**:





- Can word embeddings be directly used?
- Most basic type of information: word-to-word similarity!

Euclidean Distance

$$d(x, y) = \sqrt{\sum_i (x_i - y_i)^2}$$

Issue: Vector length depends on frequency. More frequent words will have longer vectors.

Cosine Similarity

$$s(x, y) = \frac{x \cdot y}{|x||y|}$$

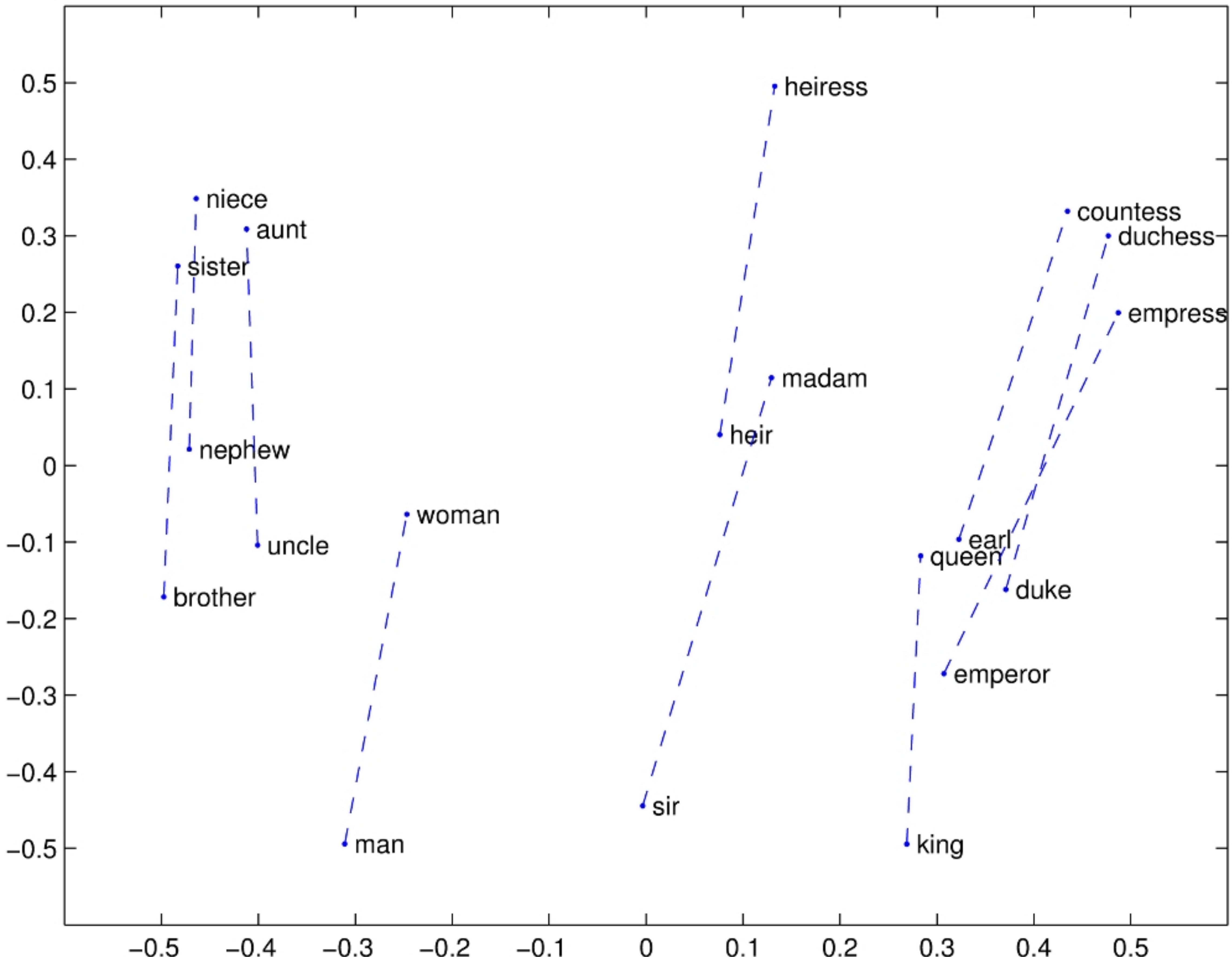
Only depends on vector angle

Range:

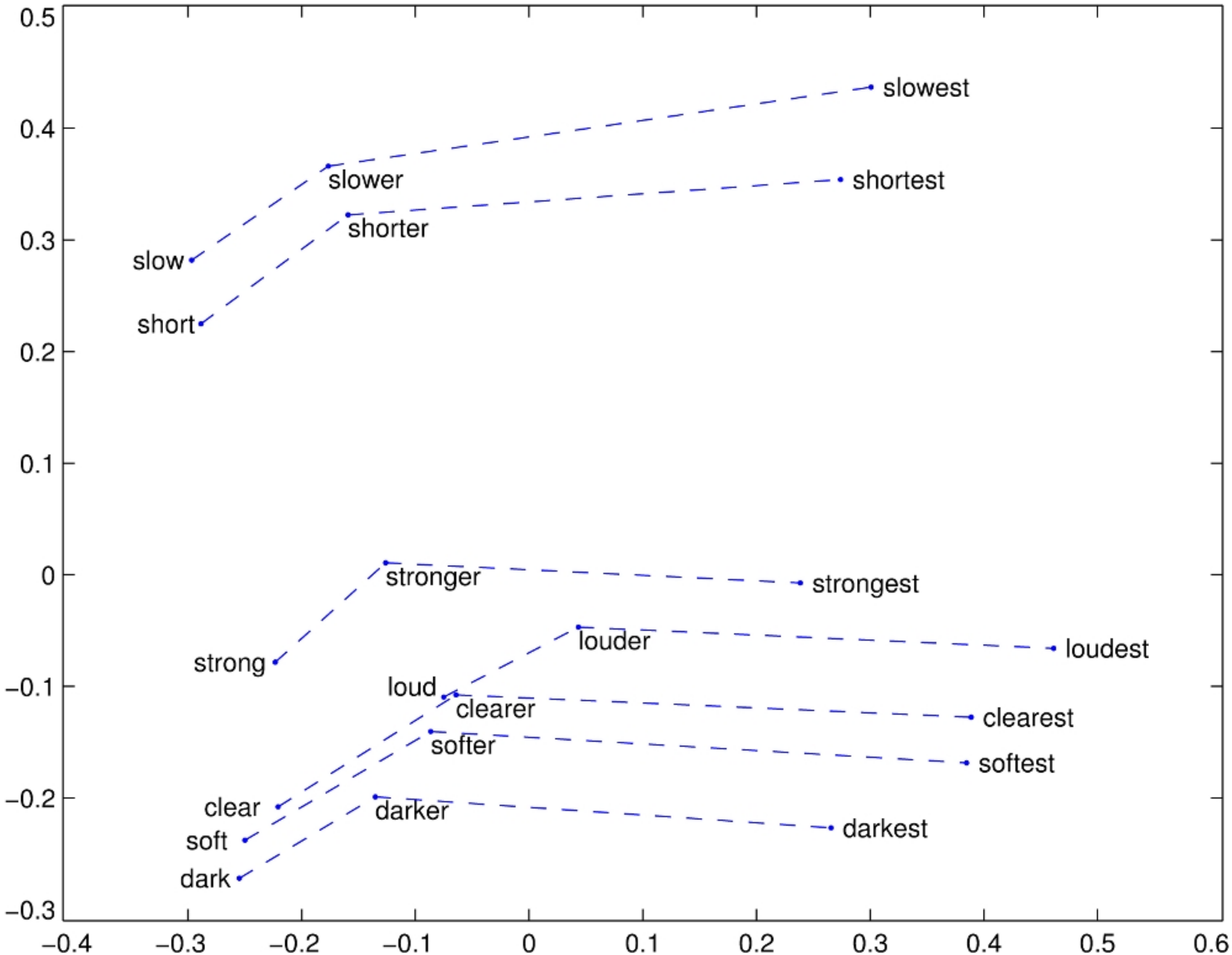
What does it learn?

- Demo: GLOVE embedding similarities
  - fasttext, glove, and word2vec are most-often used pretrained word embeddings

https://nlp.stanford.edu/projects/glove/



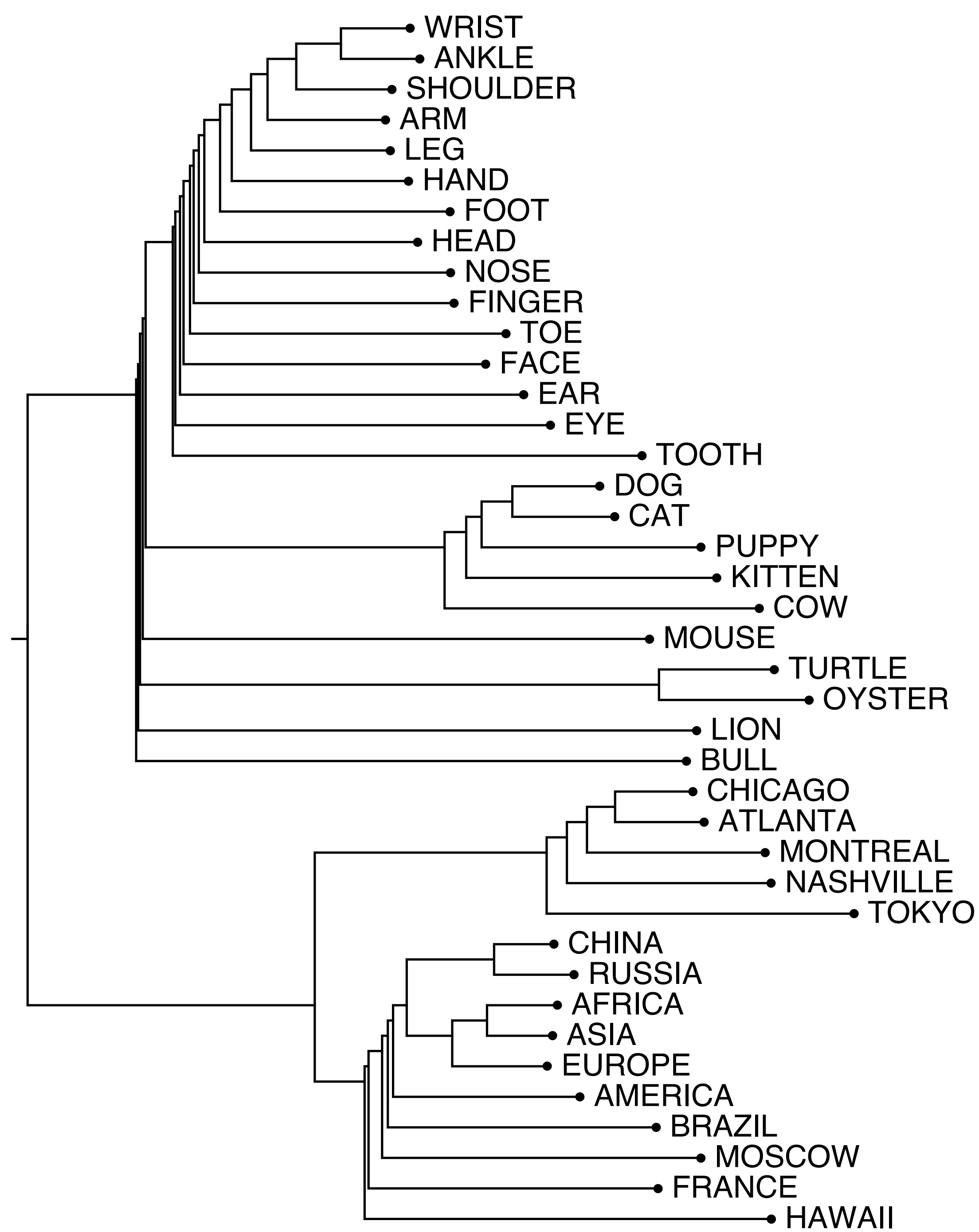
https://nlp.stanford.edu/projects/glove/



embeddings may have larger-scale semantic structure?

- Hierarchical distributional word clusters, trained from tweets: http://www.cs.cmu.edu/~ark/TweetNLP/cluster_viewer.html
- What distinctions is it learning?

embeddings may have larger-scale semantic structure?



SLP3 ch. 6