Word Vectors & word2vec

Robin Jia
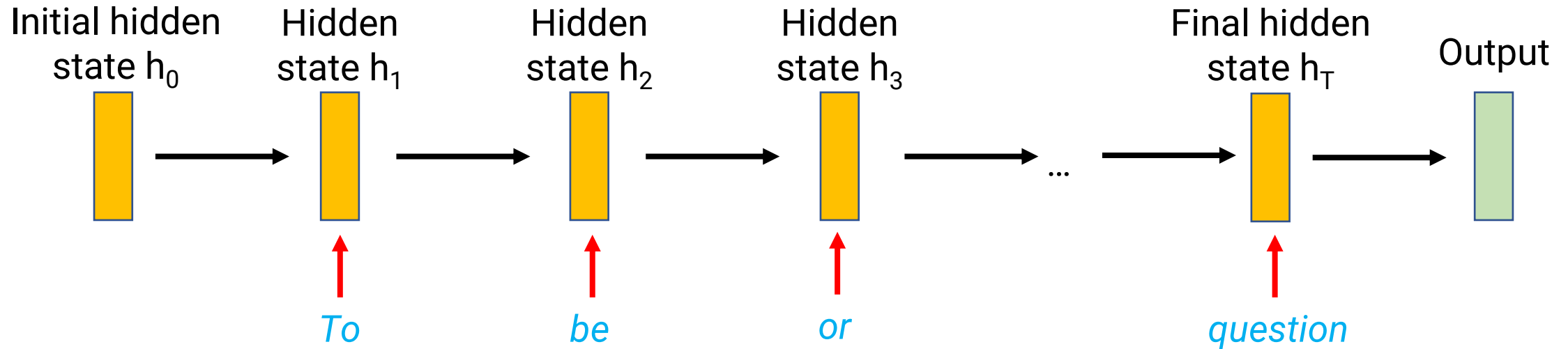USC CSCI 467, Spring 2023
April 4, 2023

With a lot borrowed from Jurafsky & Martin, "Speech and Language Processing"
https://web.stanford.edu/~jurafsky/slp3/

Previously: RNNs

Initial hidden state h_0 Hidden state h_1 Hidden state h_2 Hidden state h_3 Final hidden state h_T Output
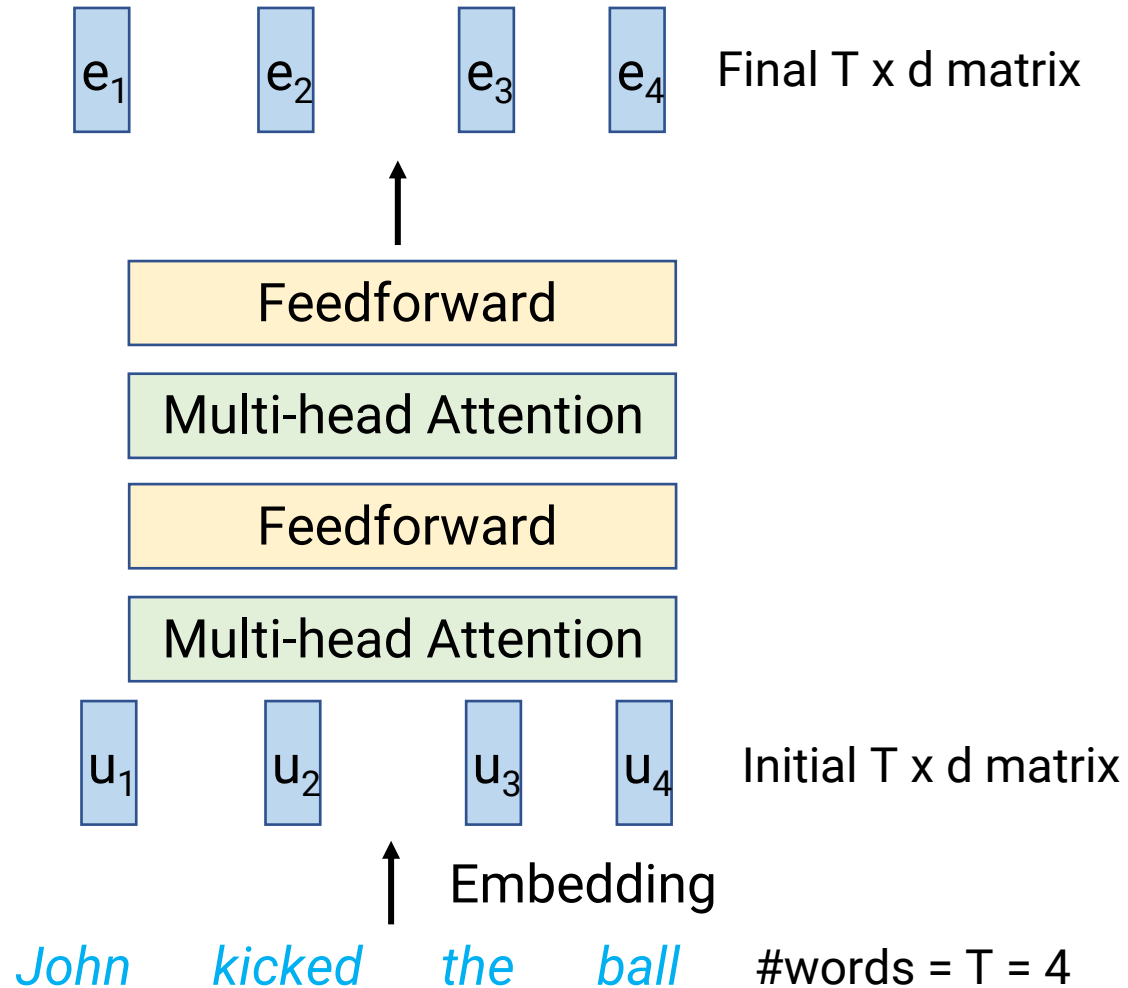
To   *be*   *or*   *question*

- Idea: Recurrence!
  - "Read" the input one word at a time
  - At each step, update the hidden state of the network
  - Model parameters to do this update are same for each step

Previously: Word Vectors in RNNs

- How do we "feed" the next word to the RNN?

- Want to learn a vector that represents each word
  - For each word w in vocabulary V, have vector v_w of size d
  - $|V| * d$ parameters needed

- Intuition: Similar words get similar vectors
  - More on learning word vectors later in the class!

Previously: Transformers

e₁ e₂ e₃ e₄   Final T x d matrix

Feedforward

Multi-head Attention

Feedforward

Multi-head Attention

u₁ u₂ u₃ u₄   Initial T x d matrix

↑ Embedding

John   kicked   the   ball   #words = T = 4

- One transformer consists of
  - **Embeddings for each token of size d**
    - Let T =#tokens, so initially T x d matrix
  - Alternating layers of
    - "Multi-headed" attention layer
    - Feedforward layer
    - Both take in T x d matrix and output a new T x d matrix
  - Plus some bells and whistles
    - Residual connections & LayerNorm
    - Byte pair encoding tokenization

Today: Unsupervised word vectors

- What do we want?

- word2vec

- Solving analogies

- Bias in word vectors

Lexical Semantics

- Goal: For each word *w*, have vector v_w that represents word's meaning
  - Lexical = word-level
  - Semantics = meaning
- What do we want to represent?
  - Synonymy (*car*/*automobile*) or antonymy (*cold*/*hot*)
  - Hypernymy/Hyponymy (*animal*/*dog*)
  - Similarity (*cat*/*dog*, *coffee*/*cup*, *waiter*/*menu*)
  - Various features
    - Sentiment (positive/negative)
    - Formality
    - All sorts of properties (Is a city? Is an action that a person can do?)

v_{cat} v_{dog} v_{hot} ...
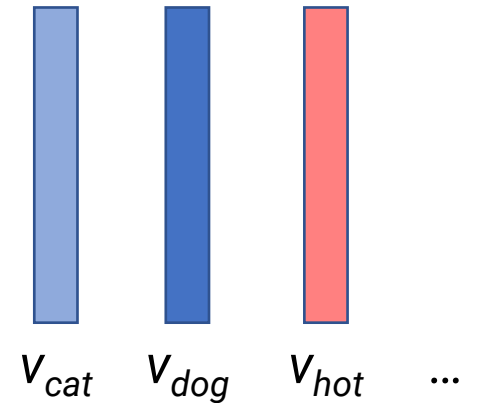
The Distributional Hypothesis

- You hear a new word, *ongchoi*
  - *Ongchoi is delicious sauteed with garlic.*
  - *Ongchoi is superb over rice.*
  - *...ongchoi leaves with salty sauces...*

- Compare with similar contexts:
  - *...spinach sauteed with garlic over rice...*
  - *...chard stems and leaves are delicious...*
  - *...collard greens and other salty leafy greens*

- Conclusion: **ongchoi** is probably a leafy green similar to spinach, chard, and collard greens
- Distributional Hypothesis: Words appearing in similar contexts have similar meanings!
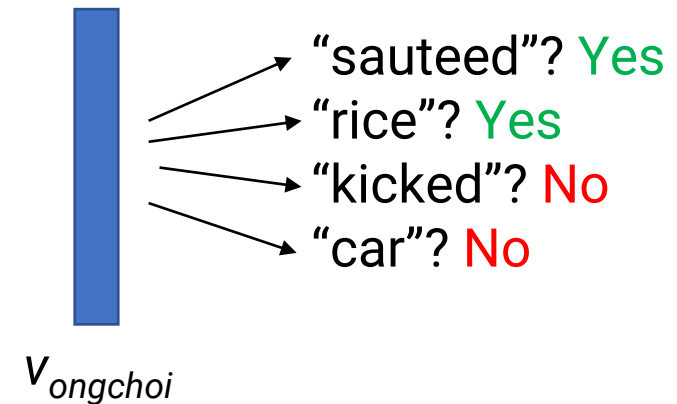- Firth 1957: "You Shall Know a Word by the Company It Keeps"

Today: Unsupervised word vectors

- What do we want?
- word2vec
- Solving analogies
- Bias in word vectors

Word vectors as a learning problem

- Want to learn vector v_w for each word w

- What makes a vector good?

- Idea: v_w should help you predict which words co-occur with w

  - Captures **distribution** of context words for w

  - Think of it as N binary classification problems, where N is size of vocabulary

$v_{ongchoi}$

"sauteed"? Yes
"rice"? Yes
"kicked"? No
"car"? No

Creating a dataset

- Given: Raw dataset of text (unsupervised)

- We will create N "fake" supervised learning problems!
  - We don't really care about these supervised learning problems
  - We just care that we learn good vectors

- Task i: Did word w co-occur with the i-th word?
  - Positive examples: Real co-occurrences within sliding window
  - Negative examples: Random samples

Word w

… lemon, a tablespoon of apricot jam, a pinch…

Window of radius 2
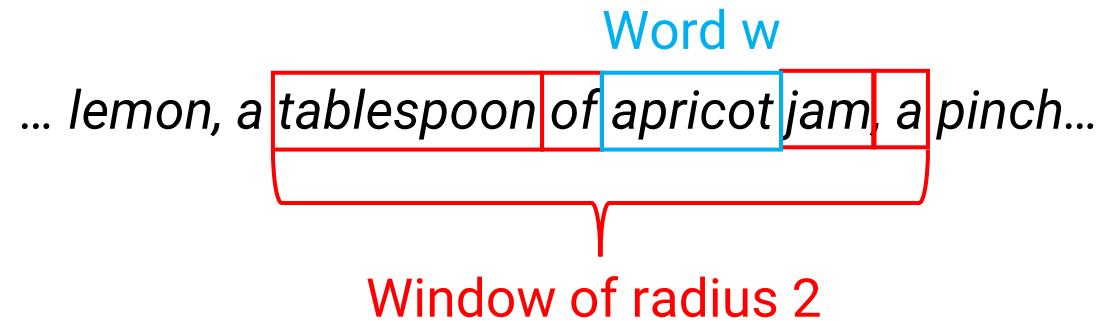
Word w ("input")	Context w' ("task")	y (label)
apricot	tablespoon	+1
apricot	of	+1
apricot	jam	+1
apricot	a	+1

Creating a dataset

- Given: Raw dataset of text (unsupervised)

- We will create N "fake" supervised learning problems!
  - We don't really care about these supervised learning problems
  - We just care that we learn good vectors

- Task i: Did word w co-occur with the i-th word?
  - Positive examples: Real co-occurrences within sliding window
 - Negative examples: Random samples

Word w

… lemon, a tablespoon *of* apricot jam*, a pinch…*

Window of radius 2
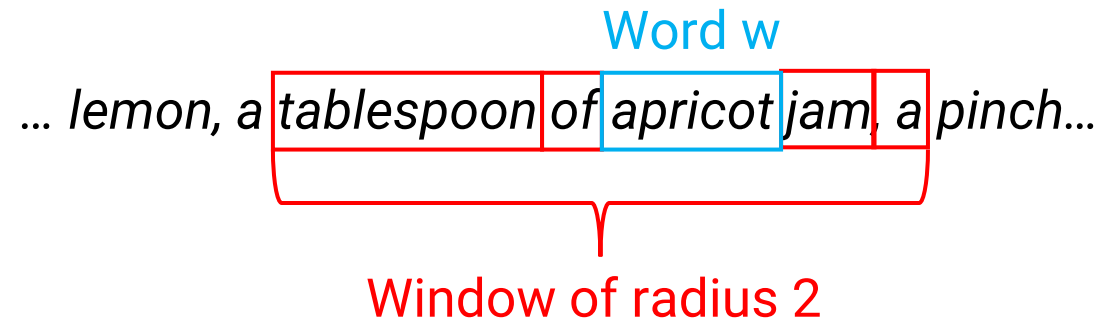
Word w ("input")	Context w' ("task")	y (label)
apricot	tablespoon	+1
apricot	of	+1
apricot	jam	+1
apricot	a	+1
apricot	seven	-1
apricot	forever	-1
apricot	dear	-1
apricot	if	-1

How to sample negatives?

- Choose a fixed ratio of negative:positive (e.g. 2)
- Baseline: Sample according to frequency of word p(w) in the data
  - Not ideal because very common words ("the") get sampled a lot
- Improvement: Sample according to α-weighted frequency

$$p_\alpha(w) = \frac{\text{count}(w)^\alpha}{\sum_{w' \in V} \text{count}(w')^\alpha}$$

  - For α < 1, high-frequency words get down-weighted
  - Typically choose around α=.75
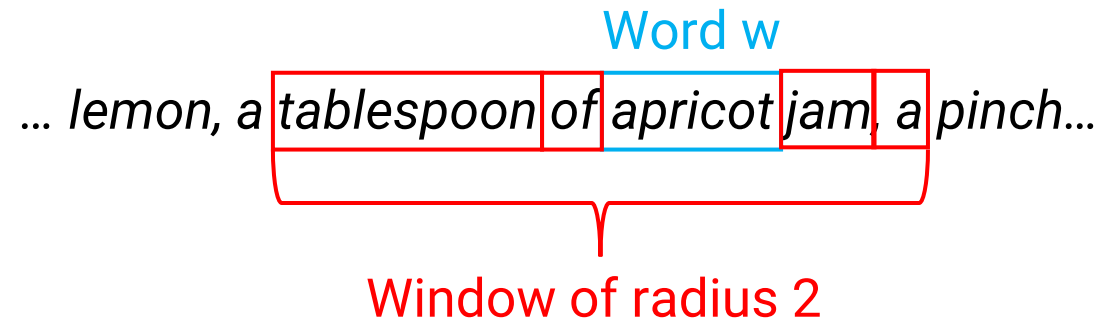
Word w

… lemon, a tablespoon of apricot jam, a pinch…

Window of radius 2

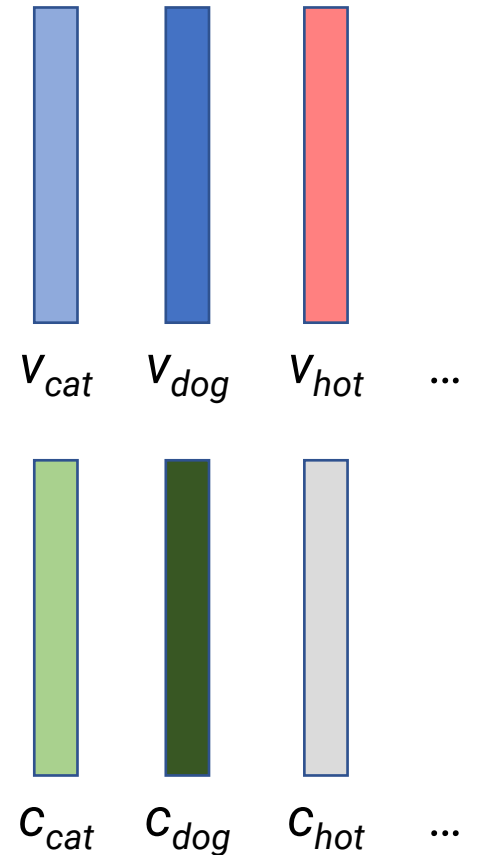| Word w ("input")	Context w' ("task")	y (label)
apricot	tablespoon	+1
apricot	of	+1
apricot	jam	+1
apricot	a	+1
apricot	seven	-1
apricot	forever	-1
apricot	dear	-1
apricot	if	-1

word2vec model

- Parameters (all of dimension d):
  - Word vector v_w for each word ("features"—the actual word vectors)
  - Context vector c_w for each word ("classifier weights" for task corresponding to w as context)
- Goal: v_w can be used by **linear classifier** to do **any** of the N "was this a context word" tasks
- Objective looks just like logistic regression:

$$L(v, c) = \sum_{(w, w', y)} -\log \sigma(y \cdot v_w^\top c_{w'})$$

word   context

"features" for word

"weight" for context

v_{cat} v_{dog} v_{hot} ...

c_{cat} c_{dog} c_{hot} ...

Training word2vec

- Strategy: Gradient descent

- Gradient updates essentially same as logistic regression
    - Gradient w.r.t. *c* holds *v* fixed, so it's like *v* are fixed features

$$\nabla_{c_u} L(v, c) = \sum_{(w, w', y): w' = u} -\sigma(y \cdot v_w^\top c_u) \cdot y \cdot v_w$$

Examples where w' = u

Same as logistic regression
where v_w is the input x

    - Gradient w.r.t. *v* is symmetrical

$$\nabla_{v_u} L(v, c) = \sum_{(w, w', y): w = u} -\sigma(y \cdot v_u^\top c_{w'}) \cdot y \cdot c_{w'}$$

Examples where w = u

Same as logistic regression
where $c_{w'}$ is the input x

Is this a convex problem?

- Looks a lot like logistic regression…
- But it's not convex!
- Why?
  - In logistic regression, we only optimize w.r.t. weights, features are constant
  - Now we optimize both at the same time!
- Fact to remember: f(x) = x_1 * x_2 is not convex
  - Consider points [-1, 1] and [1, -1]
  - f(x) = -1 at both points
  - But at the midpoint [0, 0], f(x) = 0
- Corollary: We need to randomly initialize
 - Must break symmetry, as in neural networks

$$L(v, c) = \sum_{(w, w', y)} -\log \sigma(y \cdot v_w^\top c_{w'})$$

Both are optimization variables

f([-1,1]) = -1

f([0,0]) = 0

f([1,-1]) = -1
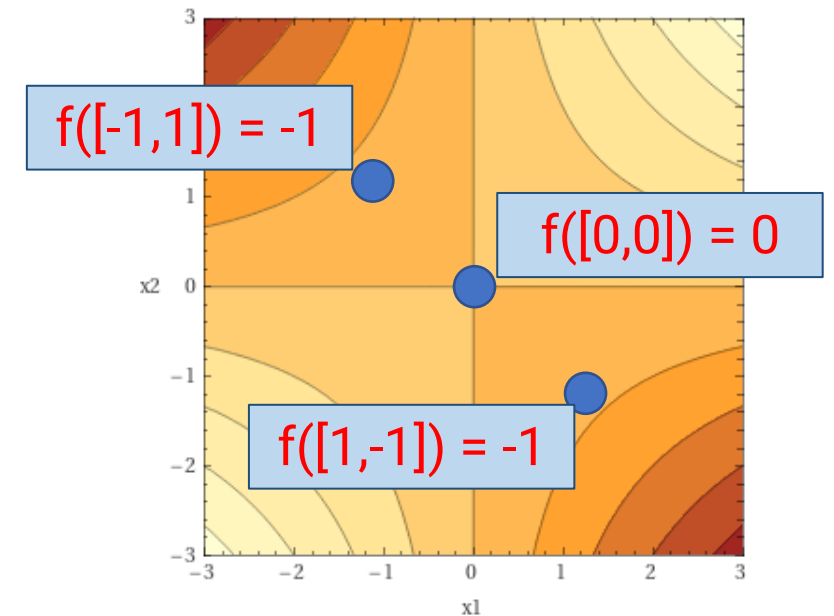
word2vec overview

- Acquire large unsupervised text corpus
- Create positive examples for every word by using sliding window
- Create negative examples by randomly sampling context word from weighted word frequency
- Randomly initialize all v and c vectors
- Train on logistic regression-like loss with gradient descent
- Return v vectors
  - c vectors not needed—just helpers
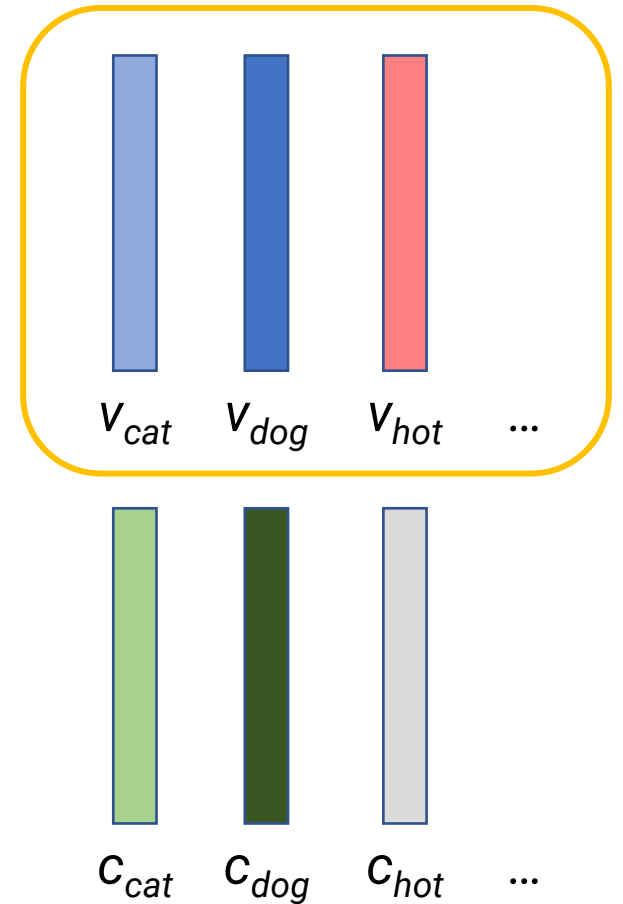
v_{cat} v_{dog} v_{hot} ...

c_{cat} c_{dog} c_{hot} ...

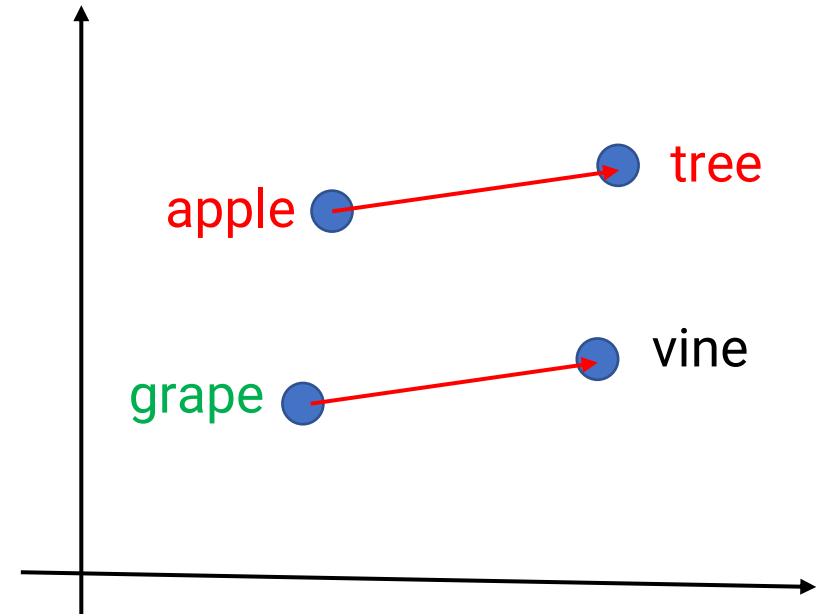# Today: Unsupervised word vectors

- What do we want?

- word2vec

- Solving analogies

- Bias in word vectors

Analogies in vector space

- *Apple is to tree as grape is to…*

- In vector space, resembles a parallelogram
  - Same relationship between apple and tree holds between grape and vine

- $v_{vine} \approx v_{tree} - v_{apple} + v_{grape}$
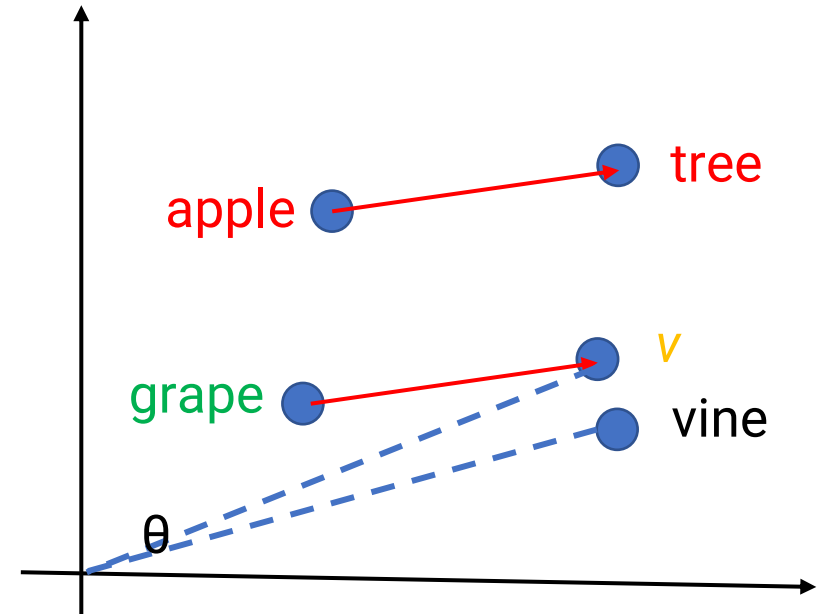
Represents the "grows on" relation

Query word

Answering analogy queries

- Compute $v = v_{tree} - v_{apple} + v_{grape}$

- Find word w in vocabulary whose v_w is most similar to v
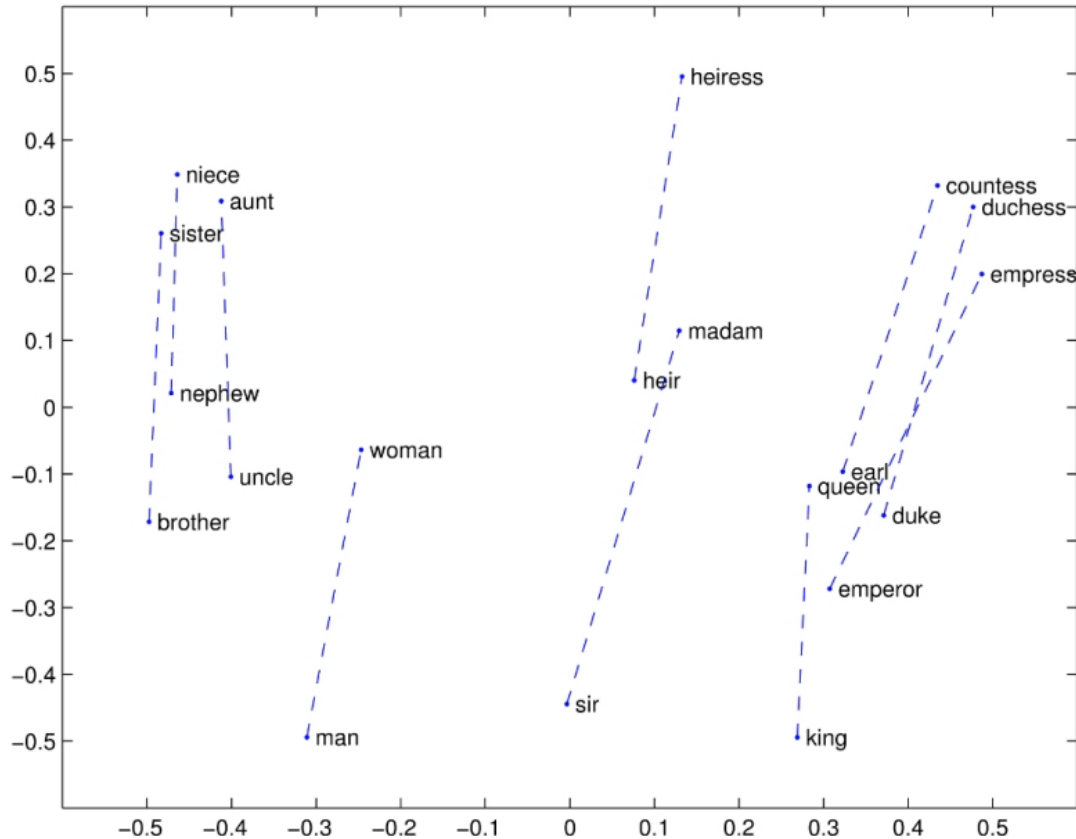
  - Common choice: Cosine similarity

  $$\text{cossim}(x, y) = \frac{x^\top y}{\|x\|\|y\|}$$

  (= cosine of angle between x and y)

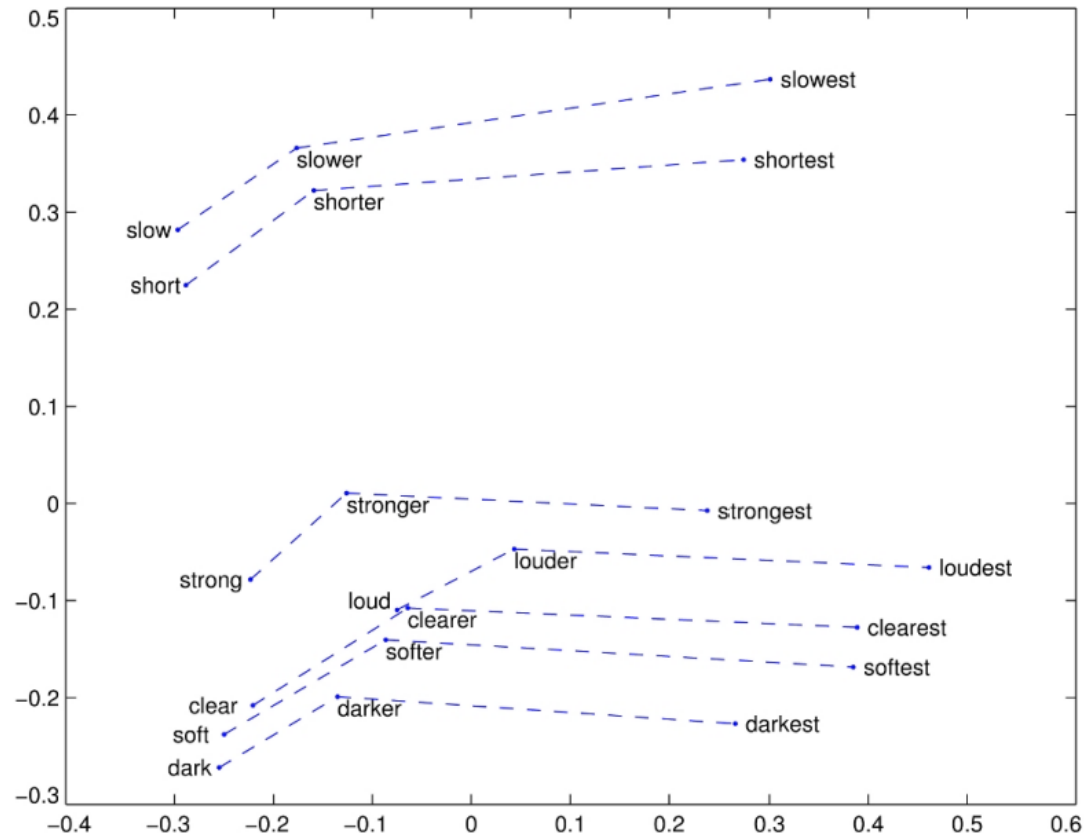 - Typically need to exclude words very similar to the query word (e.g. "*grapes*")

Visualizing Analogies

- Figure: Dimensionality reduction to 2D, then plot words with known relationship
- Roughly same difference between male/female versions of the same word

Visualizing Analogies

- Figure: Dimensionality reduction to 2D, then plot words with known relationship
- Roughly same difference between base, comparative, and superlative forms of adjectives

Today: Unsupervised word vectors

- What do we want?

- word2vec

- Solving analogies

- Bias in word vectors

Machine learning is a tornado

- …it picks up everything in its path

- Data has all sorts of associations we may not want to model

What word associations are out there?

- What is *programmer − man + woman*?
    - According to word vectors trained on news data, it's *homemaker*
    - Existing data has tons of correlations between occupation and gender
- word2vec doesn't know what is a semantic relationship and what is a historical correlation
    - "*queen*" is more related to "*she*" than "*he*" semantically
    - "*nurse*" may co-occur more with "*she*" than "*he*" in available data but not a semantic relationship!

Word vectors quantify gender stereotypes
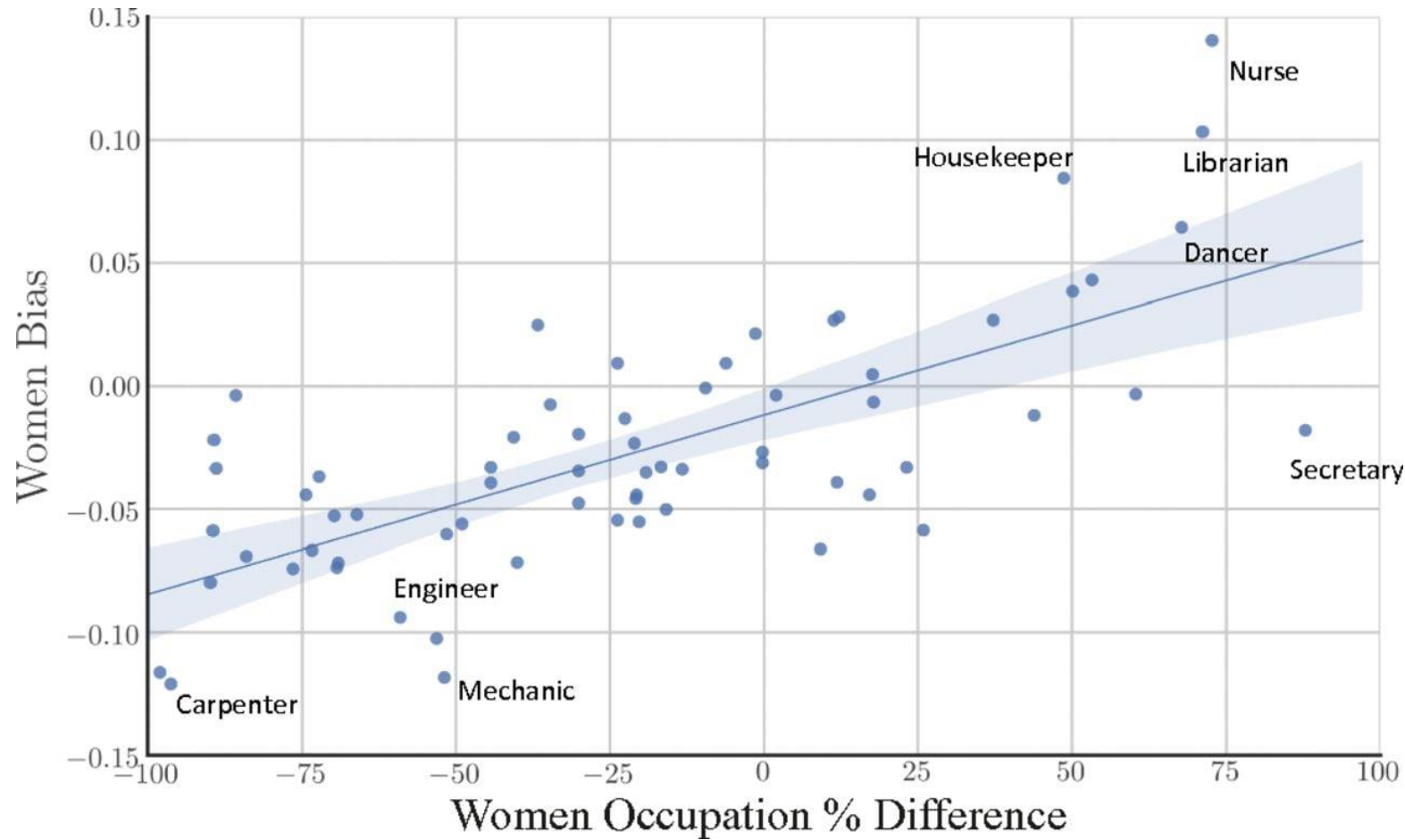
- X-axis: Real percentage difference in workforce between women & men
- Y-axis: Embedding bias = difference of distance from male-related words and female-related words
- Strong correlation!

Conclusion

- Distributional hypothesis: Words that appear in similar contexts have similar meanings

- word2vec: Learn vectors by inventing a prediction problem (did this word-context pair really occur in the text?)

- Vector arithmetic lets us complete relations

- Vectors capture both lexical semantics and historical biases

Word w

… lemon, a tablespoon *of* apricot jam *, a pinch…*

Window of radius 2