



COS 484/584

(Advanced) Natural Language Processing

# L5: Word Embeddings (I)

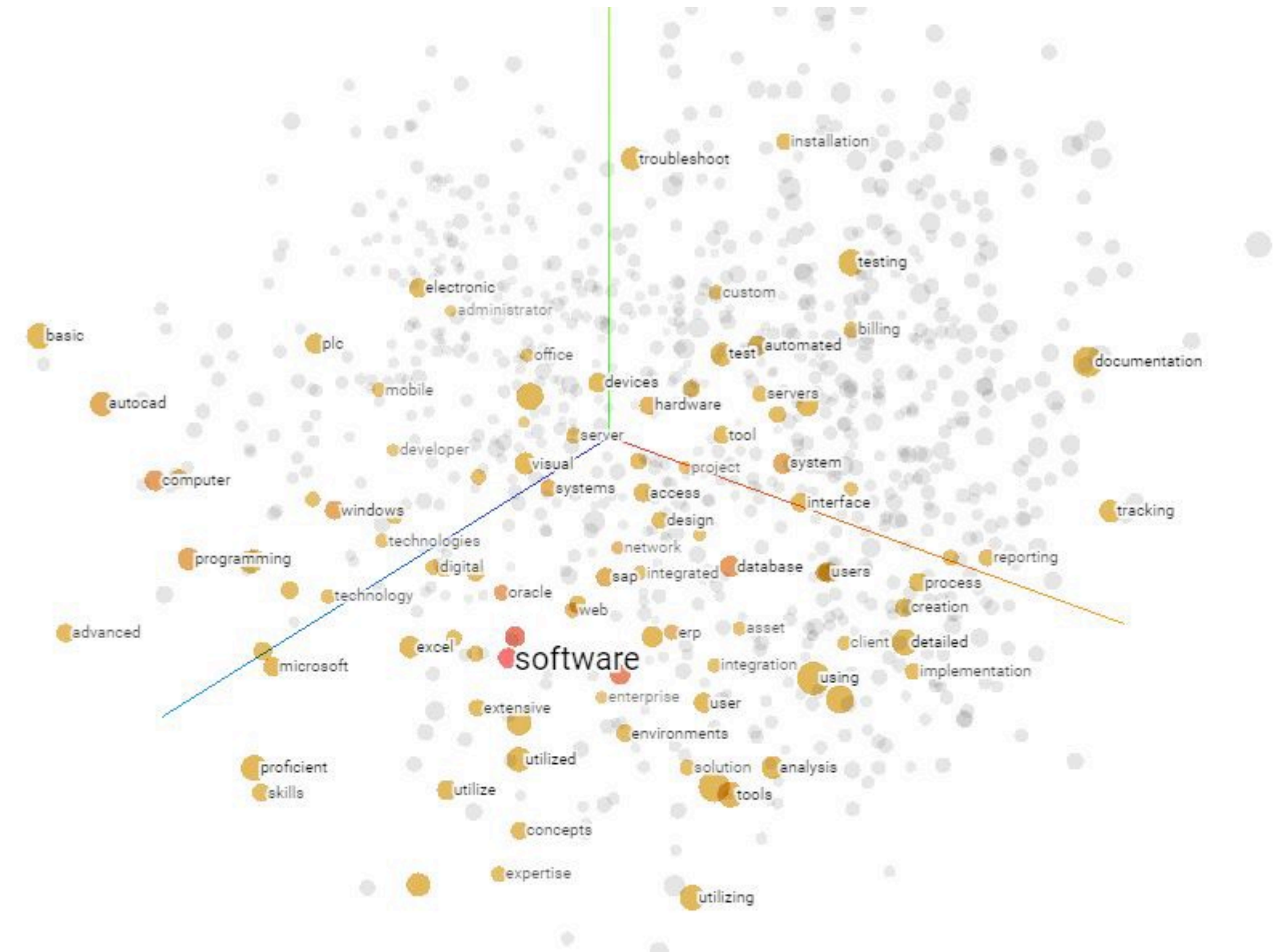
Spring 2021

(Some slides adapted from Chris Manning, Dan Jurafsky)

# Word embeddings

- Representing words as vectors
- Distributional hypothesis
- PPMI vector models
- word2vec
  - What is it?
  - How does it work?
  - More variants
- Evaluation

} Wednesday lecture



Every modern NLP algorithm uses word embeddings as the representation of word meaning...

# Before we get started...

- Question from feedback: 484 and 584 will be graded on different curves

- Zoom poll:



All the questions & answers will be on the slides

We will ask you to choose A/B/C/D in the poll

We heard that people like having more polls :)

- In-lecture questions:

Q: What is the dimension of  $W$ ?

Please type in chat if you have the answer!

- In the meantime, feel free to ask any question you may have in the chat

How should we represent the meaning of a word?

# Recap

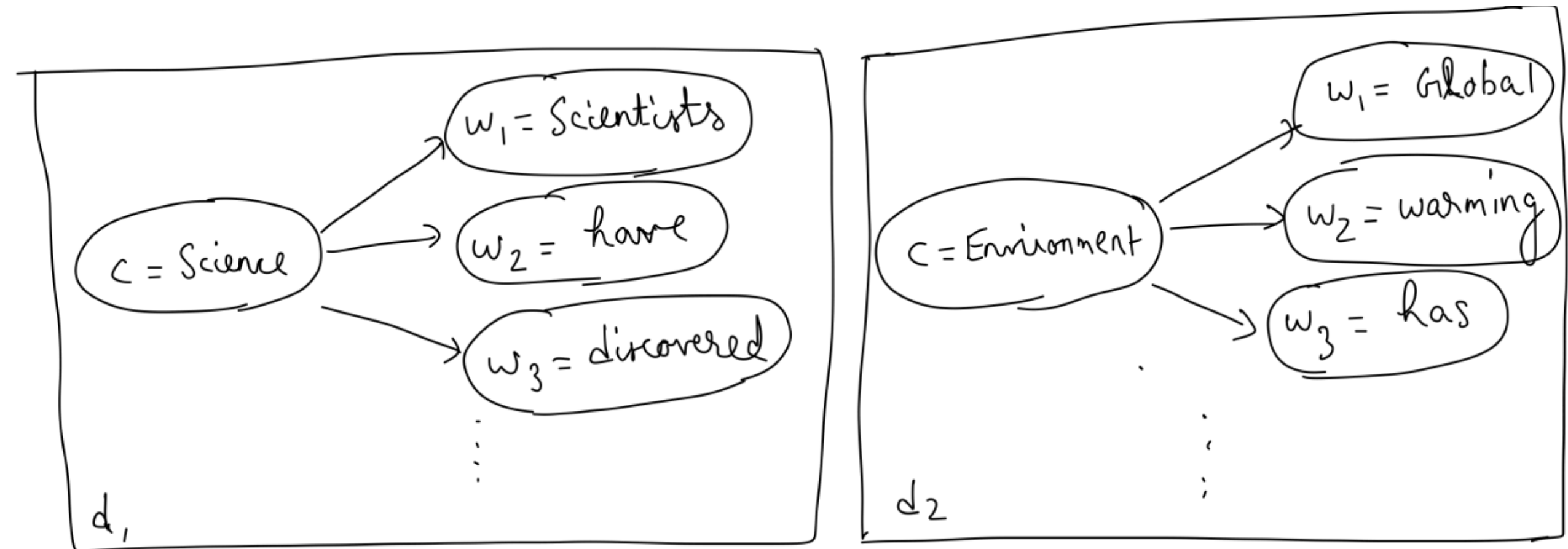
- n-gram models

$$P(\text{the cat sat on the mat}) \approx P(\text{the} \mid \text{START}) \times P(\text{cat} \mid \text{the}) \times P(\text{sat} \mid \text{cat}) \times P(\text{on} \mid \text{sat}) \times P(\text{the} \mid \text{on}) \times P(\text{mat} \mid \text{the})$$

$$P(w_i | w_{i-1}) = \frac{C(w_{i-1}, w_i) + \alpha}{C(w_{i-1}) + \alpha |V|}$$

- Naive Bayes

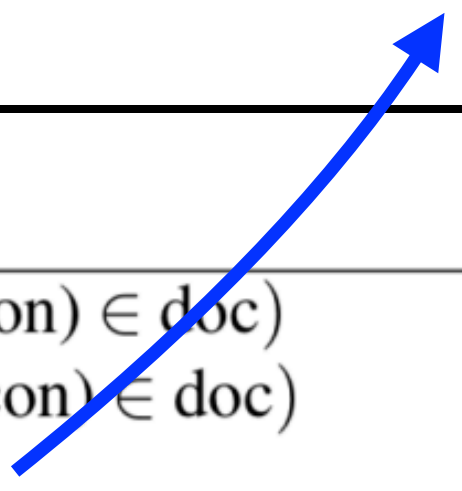
$$\hat{P}(w_i | c_j) = \frac{\text{Count}(w_i, c_j) + \alpha}{\sum_{w \in V} [\text{Count}(w, c_j) + \alpha]}$$



# Recap

- Logistic regression

Whether the word “no” appears in the document or not



Var	Definition	Value
$x_1$	count(positive lexicon) $\in$ doc)	3
$x_2$	count(negative lexicon) $\in$ doc)	2
$x_3$	$\begin{cases} 1 & \text{if “no”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
$x_4$	count(1st and 2nd pronouns $\in$ doc)	3
$x_5$	$\begin{cases} 1 & \text{if “!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
$x_6$	log(word count of doc)	$\ln(64) = 4.15$

Q: What is the notion of “word representations” in these models?

# Representing words as discrete symbols

In traditional NLP, we regard words as discrete symbols:

hotel, conference, motel — a localist representation

one 1, the rest 0's



Words can be represented by one-hot vectors:

hotel = [0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

motel = [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]

Vector dimension = number of words in vocabulary (e.g., 500,000)

Q: Why is this representation not good?

# Why is this representation not good?

If we use **word identity** as features,

⇒ it requires **exact same** word to be in training and test

Training	hotel = [0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
Test	motel = [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]

If we use **word vectors** as features,

⇒ We can generalize to **similar** but **unseen** words at testing time!!!

Training	hotel = [35, 22, 17, ..]
Test	motel = [34, 21, 14, ..]



# How do we know the meaning of a word?

- You can look up the word in a dictionary/thesaurus!

“Princeton”

1. a university town in central New Jersey
2. a university in New Jersey

The meaning of words can be defined by other words!

“tejuino”

Word to search for:

Display Options:

**Your search did not return any results.**

WordNet Search - 3.1  
- [WordNet home page](#) - [Glossary](#) - [Help](#)

Word to search for:

Display Options:

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations  
Display options for sense: (gloss) "an example sentence"

**Noun**

- [S:](#) (n) **mouse** (any of numerous small rodents typically resembling diminutive rats having pointed snouts and small ears on elongated bodies with slender usually hairless tails)
- [S:](#) (n) [shiner](#), [black eye](#), **mouse** (a swollen bruise caused by a blow to the eye)
- [S:](#) (n) **mouse** (person who is quiet or timid)
- [S:](#) (n) **mouse**, [computer mouse](#) (a hand-operated electronic device that controls the coordinates of a cursor on your computer screen as you move it around on a pad; on the bottom of the device is a ball that rolls on the surface of the pad) "a mouse takes much more room than a trackball"

**Verb**

- [S:](#) (v) [sneak](#), **mouse**, [creep](#), [pussyfoot](#) (to go stealthily or furtively) "..stead of sneaking around spying on the neighbor's house"
- [S:](#) (v) **mouse** (manipulate the mouse of a computer)

<http://wordnetweb.princeton.edu/>

Key idea: you can know the meaning of a word by looking at its context words

# Representing words by their context

**Distributional hypothesis:** words that occur in similar contexts tend to have similar meanings



J.R.Firth 1957

- “You shall know a word by the company it keeps”
- One of the most successful ideas of modern statistical NLP!

When a word  $w$  appears in a text, its context is the set of words that appear nearby (within a fixed-size window).

*...government debt problems turning into **banking** crises as happened in 2009...*  
*...saying that Europe needs unified **banking** regulation to replace the hodgepodge...*  
*...India has just given its **banking** system a shot in the arm...*

These context words will represent “*banking*”.

# Distributional hypothesis

Q: Guess the meaning of tejuino now?

“tejuino”



C1: A bottle of \_\_\_\_ is on the table.

C2: Everybody likes \_\_\_\_.

C3: Don't have \_\_\_\_ before you drive.

C4: We make \_\_\_\_ out of corn.

# Distributional hypothesis

C1: A bottle of \_\_\_\_ is on the table.

C2: Everybody likes \_\_\_\_.

C3: Don't have \_\_\_\_ before you drive.

C4: We make \_\_\_\_ out of corn.

	C1	C2	C3	C4
tejuino	1	1	1	1
loud	0	0	0	0
motor-oil	1	0	0	0
tortillas	0	1	0	1
choices	0	1	0	0
wine	1	1	1	0

Q: Which word is closest to “tejuino”?

“words that occur in similar contexts tend to have similar meanings”



Q: What is the dimension of these vectors?

# Words as vectors

We'll build a new model of meaning focusing on similarity

- Each word is a vector
- Similar words are “nearby in space”

A first solution: we can just use **word-word co-occurrence counts** to represent the meaning of words!

context words:

4 words to the left,

4 words to the right

is traditionally followed by **cherry** pie, a traditional dessert  
often mixed, such as **strawberry** rhubarb pie. Apple pie  
computer peripherals and personal **digital** assistants. These devices usually  
a computer. This includes **information** available on the internet

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

Most entries are 0s  $\implies$  sparse vectors

# Words as vectors

context words: 4  
words to the left, 4  
words to the right

is traditionally followed by **cherry** pie, a traditional dessert  
often mixed, such as **strawberry** rhubarb pie. Apple pie  
computer peripherals and personal **digital** assistants. These devices usually  
a computer. This includes **information** available on the internet

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

C1: A bottle of \_\_\_\_ is on the table.

C2: Everybody likes \_\_\_\_.

C3: Don't have \_\_\_\_ before you drive.

C4: We make \_\_\_\_ out of corn.

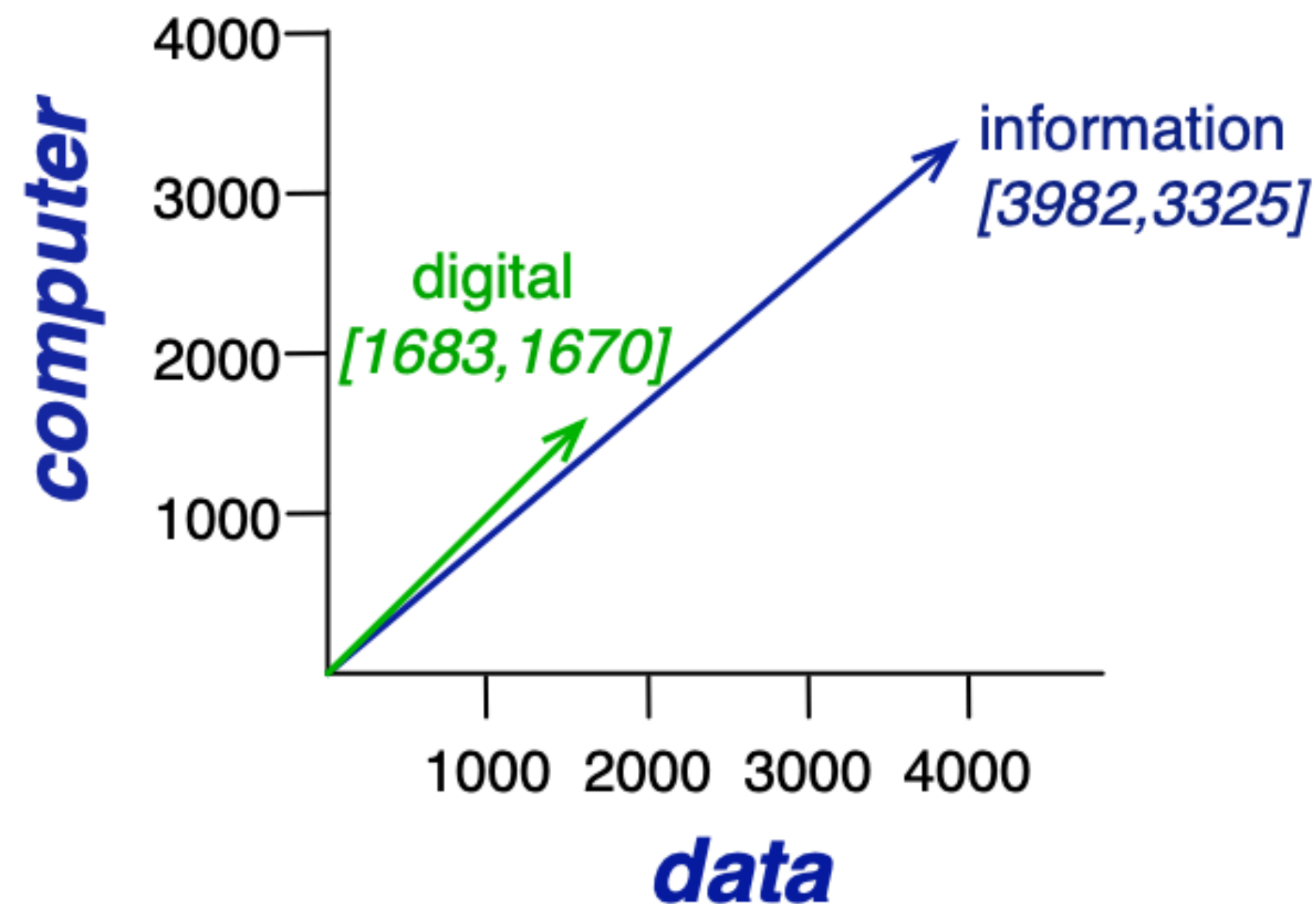
vs.

	C1	C2	C3	C4
tejuino	1	1	1	1
loud	0	0	0	0
motor-oil	1	0	0	0
tortillas	0	1	0	1
choices	0	1	0	0
wine	1	1	1	0

Using  $C_i$  is too sparse.

Word-word co-occurrence can be thought of as a simplification + frequency captures important information!

# Measuring similarity



A common similarity metric: **cosine** of the angle between the two vectors (the larger, the more similar the two vectors are)

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|}$$

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\sum_{i=1}^{|V|} u_i v_i}{\sqrt{\sum_{i=1}^{|V|} u_i^2} \sqrt{\sum_{i=1}^{|V|} v_i^2}}$$

Q: Why cosine similarity instead of dot product  $\mathbf{u} \cdot \mathbf{v}$ ?

# Zoom poll



What is the range of  $\cos(u, v)$  in this model?

- (a)  $[-1, 1]$
- (b)  $[0, 1]$
- (c)  $[0, +\infty)$
- (d)  $(-\infty, +\infty)$

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\sum_{i=1}^{|V|} u_i v_i}{\sqrt{\sum_{i=1}^{|V|} u_i^2} \sqrt{\sum_{i=1}^{|V|} v_i^2}}$$

The answer is (b). Cosine similarity ranges between -1 and 1. In this model, all the values of  $u_i, v_i$  are non-negative.



# Any issues with this model?

Raw frequency count is a bad representation!

- Frequency is clearly useful; if “pie” appears a lot near “cherry”, that's useful information.
- But overly frequent words like “the”, “it”, or “they” also appear a lot near “cherry”. They are not very informative about the context.

Solution: use a **weighting function** instead of raw counts!

Pointwise Mutual Information (PMI):

Do events  $x$  and  $y$  co-occur more or less than if they were independent?

$$\text{PMI}(x, y) = \log_2 \frac{P(x, y)}{P(x)P(y)} \qquad \text{PMI}(\text{word}_1, \text{word}_2) = \log_2 \frac{P(\text{word}_1, \text{word}_2)}{P(\text{word}_1)P(\text{word}_2)}$$

# Positive Pointwise Mutual Information (PPMI)

- PMI ranges from  $-\infty$  to  $+\infty$
- $\text{PMI}(w_1, w_2) > 0 \implies P(w_1, w_2) > P(w_1)P(w_2)$
- $\text{PMI}(w_1, w_2) < 0 \implies P(w_1, w_2) < P(w_1)P(w_2)$
- When one or both words are rare, there is high sampling error in their probabilities
- Negative values of PMI are frequently not reliable
- A simple fix: replace all the negative PMI values by 0s

$$\text{PPMI}(\text{word}_1, \text{word}_2) = \max \left( \log_2 \frac{P(\text{word}_1, \text{word}_2)}{P(\text{word}_1)P(\text{word}_2)}, 0 \right)$$

Warning: negative PMI values may be statistically significant, and informative in practice, if both words are quite common.

# PPMI - A running example

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^W \sum_{j=1}^C f_{ij}}$$

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

$$p(w=\text{information}, c=\text{data}) = 3982/11716 = .3399$$

$$p(w=\text{information}) = 7703/11716 = .6575$$

$$p(c=\text{data}) = 5673/11716 = .4842$$

$$p(w_i) = \frac{\sum_{j=1}^C f_{ij}}{N} \quad p(c_j) = \frac{\sum_{i=1}^W f_{ij}}{N}$$

	p(w,context)					p(w)
	computer	data	result	pie	sugar	p(w)
cherry	0.0002	0.0007	0.0008	0.0377	0.0021	0.0415
strawberry	0.0000	0.0000	0.0001	0.0051	0.0016	0.0068
digital	0.1425	0.1436	0.0073	0.0004	0.0003	0.2942
information	0.2838	0.3399	0.0323	0.0004	0.0011	0.6575
p(context)	0.4265	0.4842	0.0404	0.0437	0.0052	

# Zoom poll



$$\begin{aligned} p(w=\text{information}, c=\text{data}) &= 3982/111716 = .3399 \\ p(w=\text{information}) &= 7703/111716 = .6575 \\ p(c=\text{data}) &= 5673/111716 = .4842 \end{aligned}$$

$$p(w_i) = \frac{\sum_{j=1}^C f_{ij}}{N} \quad p(c_j) = \frac{\sum_{i=1}^W f_{ij}}{N}$$

Assume that we have a text corpus of 1M tokens, we use the left 4 words and the right 4 words as context words, what is N (the denominator for computing these probabilities) approximately?

- (a) 1M
- (b) 4M
- (c) 8M
- (d) not enough information

The answer is (c). For every word  $w_i$  in the corpus, we need to collect  $(w_i, w_{i+j})$  pairs,  $j = -4, -3, -2, -1, 1, 2, 3, 4$ .

# Zoom poll



Which of the following statements is correct:

- (a)  $\text{PPMI}(\text{cherry, pie}) > 0$ ,  $\text{PPMI}(\text{cherry, result}) < 0$ ,  $\text{PPMI}(\text{digital, result}) > 0$
- (b)  $\text{PPMI}(\text{cherry, pie}) > 0$ ,  $\text{PPMI}(\text{cherry, result}) = 0$ ,  $\text{PPMI}(\text{digital, result}) > 0$
- (c)  $\text{PPMI}(\text{cherry, pie}) > 0$ ,  $\text{PPMI}(\text{cherry, result}) = 0$ ,  $\text{PPMI}(\text{digital, result}) = 0$
- (d)  $\text{PPMI}(\text{cherry, pie}) > 0$ ,  $\text{PPMI}(\text{cherry, result}) < 0$ ,  $\text{PPMI}(\text{digital, result}) < 0$

The answer is (c). PPMI never take negative values! See the next slide.



# PPMI - A running example

	p(w,context)					p(w)
	computer	data	result	pie	sugar	p(w)
cherry	0.0002	0.0007	0.0008	0.0377	0.0021	0.0415
strawberry	0.0000	0.0000	0.0001	0.0051	0.0016	0.0068
digital	0.1425	0.1436	0.0073	0.0004	0.0003	0.2942
information	0.2838	0.3399	0.0323	0.0004	0.0011	0.6575
p(context)	0.4265	0.4842	0.0404	0.0437	0.0052	

$$\text{PMI}(\text{cherry}, \text{pie}) = \log_2(0.0377/0.0415/0.0437) = 4.38$$

$$\text{PMI}(\text{cherry}, \text{result}) = \log_2(0.0008/0.0415/0.0404) = -1.07$$

$$\text{PMI}(\text{digital}, \text{result}) = \log_2(0.0073/0.2942/0.0404) = -0.70$$

Resulting PPMI matrix (negatives replaced by 0)

	computer	data	result	pie	sugar
cherry	0	0	0	4.38	3.30
strawberry	0	0	0	4.10	5.51
digital	0.18	0.01	0	0	0
information	0.02	0.09	0.28	0	0

# From sparse vectors to dense vectors

- Still, the vectors we get from word-word occurrence matrix are sparse (most are 0's) & long (vocabulary size)
- Alternative: we want to represent words as **short** (50-300 dimensional) & **dense** (real-valued) vectors
  - The basis of all the modern NLP systems

$$\text{employees} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 10.109 \\ -0.542 \\ 0.349 \\ 0.271 \\ 0.487 \end{pmatrix}$$



# Why dense vectors?

- Short vectors are easier to use as features in ML systems
- Dense vectors may generalize better than explicit counts
- Sparse vectors can't capture high-order co-occurrence
  - $w_1$  co-occurs with “car”,  $w_2$  co-occurs with “automobile”
  - They should be similar but they aren't because “car” and “automobile” are distinct dimensions
- In practice, they work better!



# How to get dense vectors?

Singular value decomposition (SVD) of PPMI weighted co-occurrence matrix

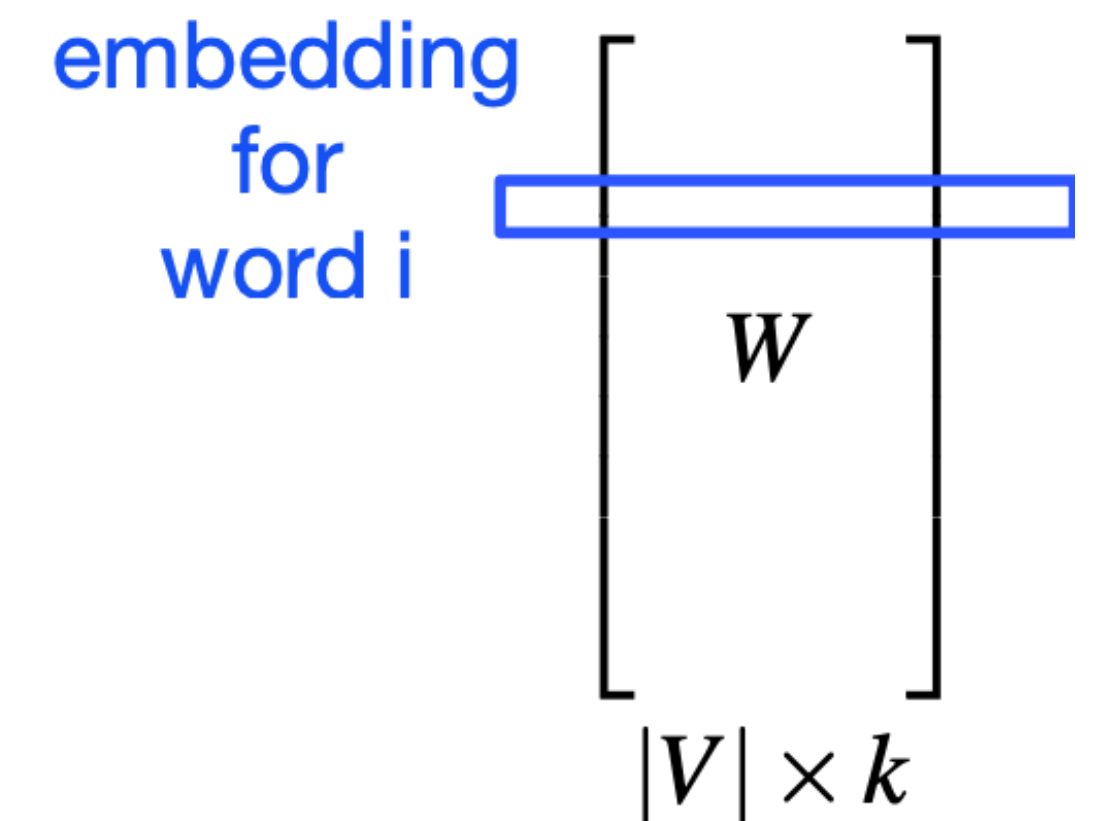
$$\begin{bmatrix} X \\ |V| \times |V| \end{bmatrix} = \begin{bmatrix} W \\ |V| \times |V| \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_V \end{bmatrix} \begin{bmatrix} C \\ |V| \times |V| \end{bmatrix}$$

$$\begin{bmatrix} X \\ |V| \times |V| \end{bmatrix} = \begin{bmatrix} W \\ |V| \times k \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_k \end{bmatrix} \begin{bmatrix} C \\ k \times |V| \end{bmatrix}$$

Only keep the top k (e.g., 100) singular values!

# How to get dense vectors?

- Singular value decomposition (SVD) of PPMI weighted co-occurrence matrix
  - Each row of the matrix  $W$  is a  $k$ -dimensional vector for each word  $w$
  - This idea originates from Latent Semantic Analysis (Deerwester et al., 1990) (applied on word-document matrix)
- Alternative approach: **learning** word vectors directly from text
  - Popular methods: **word2vec** (Mikolov et al., 2013), Glove (Pennington et al., 2014), FastText (Bojanowski et al., 2017)
  - Key idea: Instead of counting how often each word  $w$  co-occurs with another word  $v$  and perform matrix factorization, we use the dense vector of  $w$  to predict  $v$  (a machine learning problem!)



# Count-based vs prediction-based word vectors

- Recommended reading: [\(Baroni et al., 2014\)](#)

***Don't count, predict!* A systematic comparison of  
context-counting vs. context-predicting semantic vectors**

**Marco Baroni** and **Georgiana Dinu** and **Germán Kruszewski**

Center for Mind/Brain Sciences (University of Trento, Italy)

`(marco.baroni|georgiana.dinu|german.kruszewski)@unitn.it`

These context-predicting word vectors are also called word embeddings...

# Word2vec and other variants

# Word embeddings

- = **Learned** representations from text for representing words

- Input: a large text corpora,  $V, d$

- $V$ : a pre-defined vocabulary
- $d$ : dimension of word vectors (e.g. 300)
- Text corpora:
  - Wikipedia + Gigaword 5: 6B tokens
  - Twitter: 27B tokens
  - Common Crawl: 840B tokens

- Output:  $f : V \rightarrow \mathbb{R}^d$

$$v_{\text{cat}} = \begin{pmatrix} -0.224 \\ 0.130 \\ -0.290 \\ 0.276 \end{pmatrix} \quad v_{\text{dog}} = \begin{pmatrix} -0.124 \\ 0.430 \\ -0.200 \\ 0.329 \end{pmatrix}$$

$$v_{\text{the}} = \begin{pmatrix} 0.234 \\ 0.266 \\ 0.239 \\ -0.199 \end{pmatrix} \quad v_{\text{language}} = \begin{pmatrix} 0.290 \\ -0.441 \\ 0.762 \\ 0.982 \end{pmatrix}$$

Each word is represented by a low-dimensional (e.g.,  $d = 300$ ), real-valued vector

Each coordinate/dimension of the vector doesn't have a particular interpretation

# Word embeddings

- Basic property: similar words have similar vectors

word = “sweden”

Word	Cosine distance
norway	0.760124
denmark	0.715460
finland	0.620022
switzerland	0.588132
belgium	0.585835
netherlands	0.574631
iceland	0.562368
estonia	0.547621
slovenia	0.531408

$\cos(u, v)$  ranges between -1 and 1



# Word embeddings

- Basic property: similar words have similar vectors

Nearest words to  
**frog:**

1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus



litoria



leptodactylidae



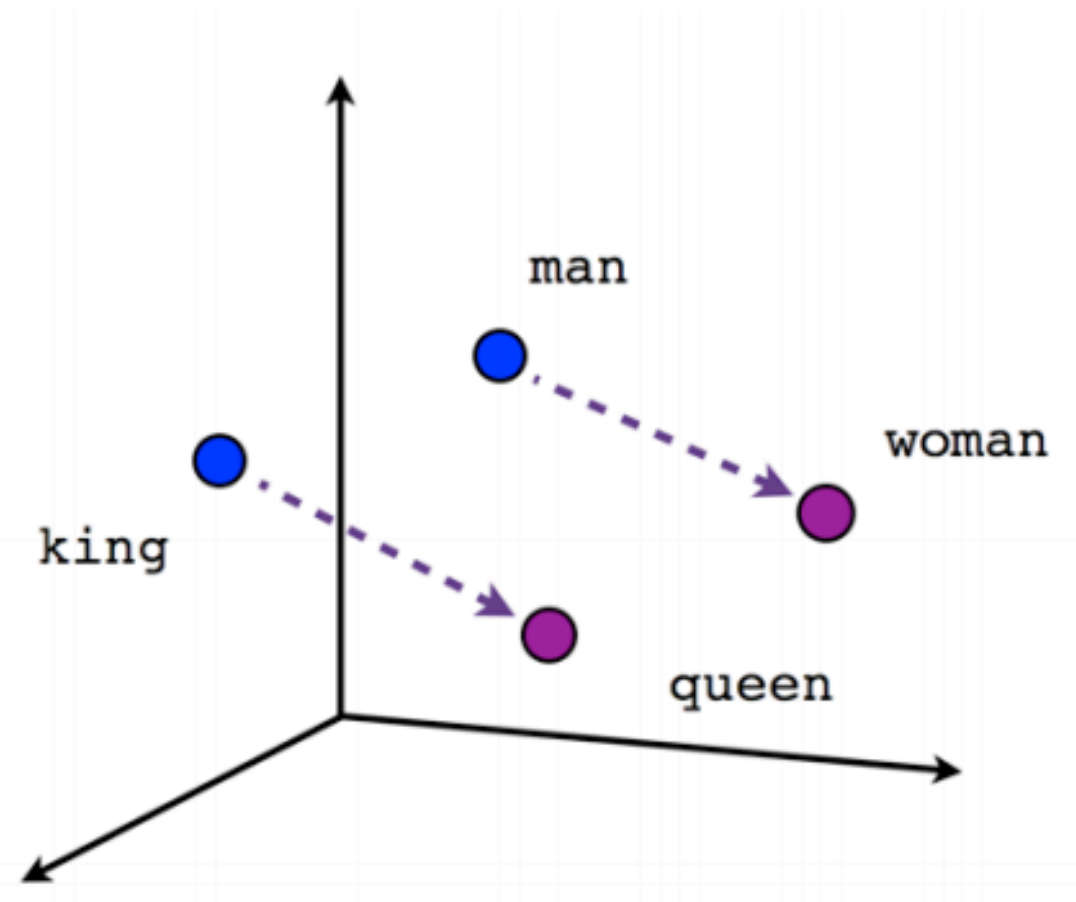
rana



eleutherodactylus

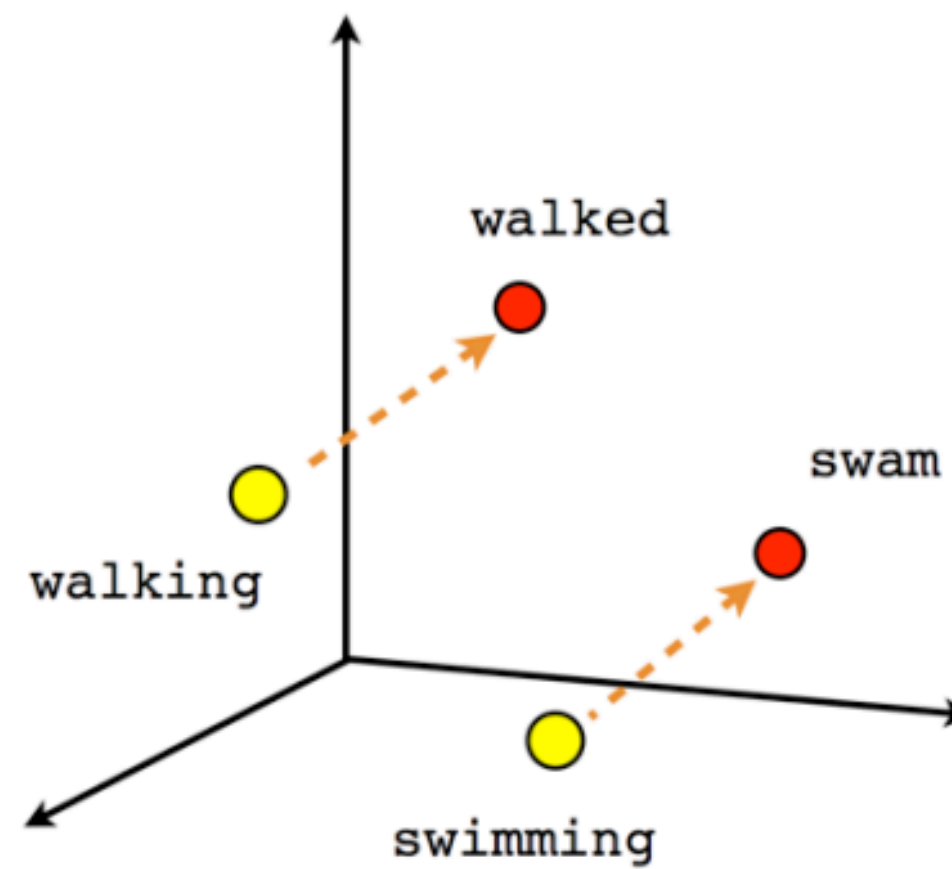
# Word embeddings

- They have some other nice properties too!

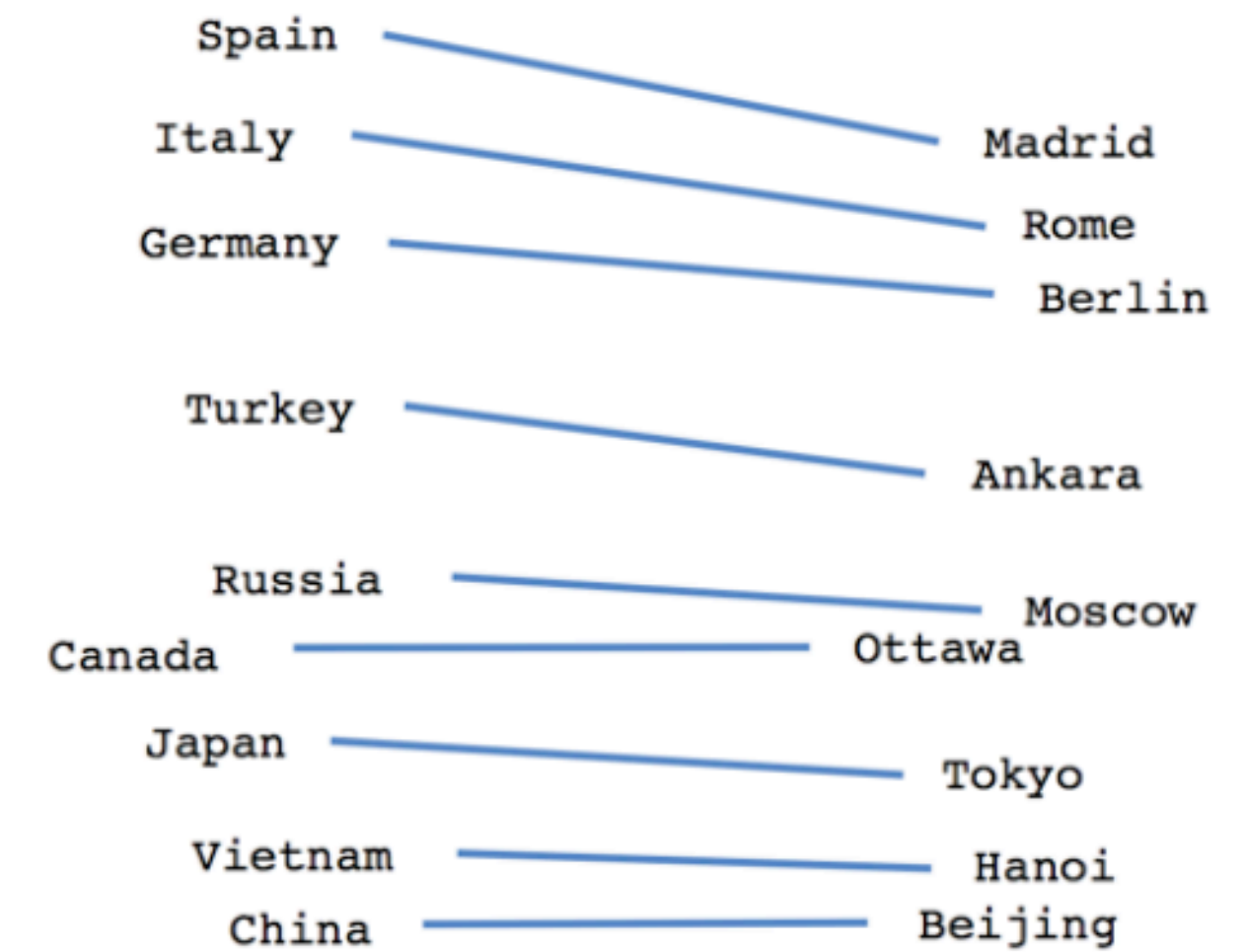


Male-Female

$$v_{\text{man}} - v_{\text{woman}} \approx v_{\text{king}} - v_{\text{queen}}$$



Verb tense



Country-Capital



# Word embeddings

- They have some other nice properties too!

$$v(\text{cuatro}) \approx Wv(\text{four})$$

