neural language models I: word embeddings

CS 585, Fall 2019

Introduction to Natural Language Processing

http://people.cs.umass.edu/~miyyer/cs585/

Mohit Iyyer

College of Information and Computer Sciences

University of Massachusetts Amherst

many slides from Brendan O'Connor and Jurafsky & Martin Ed. 3

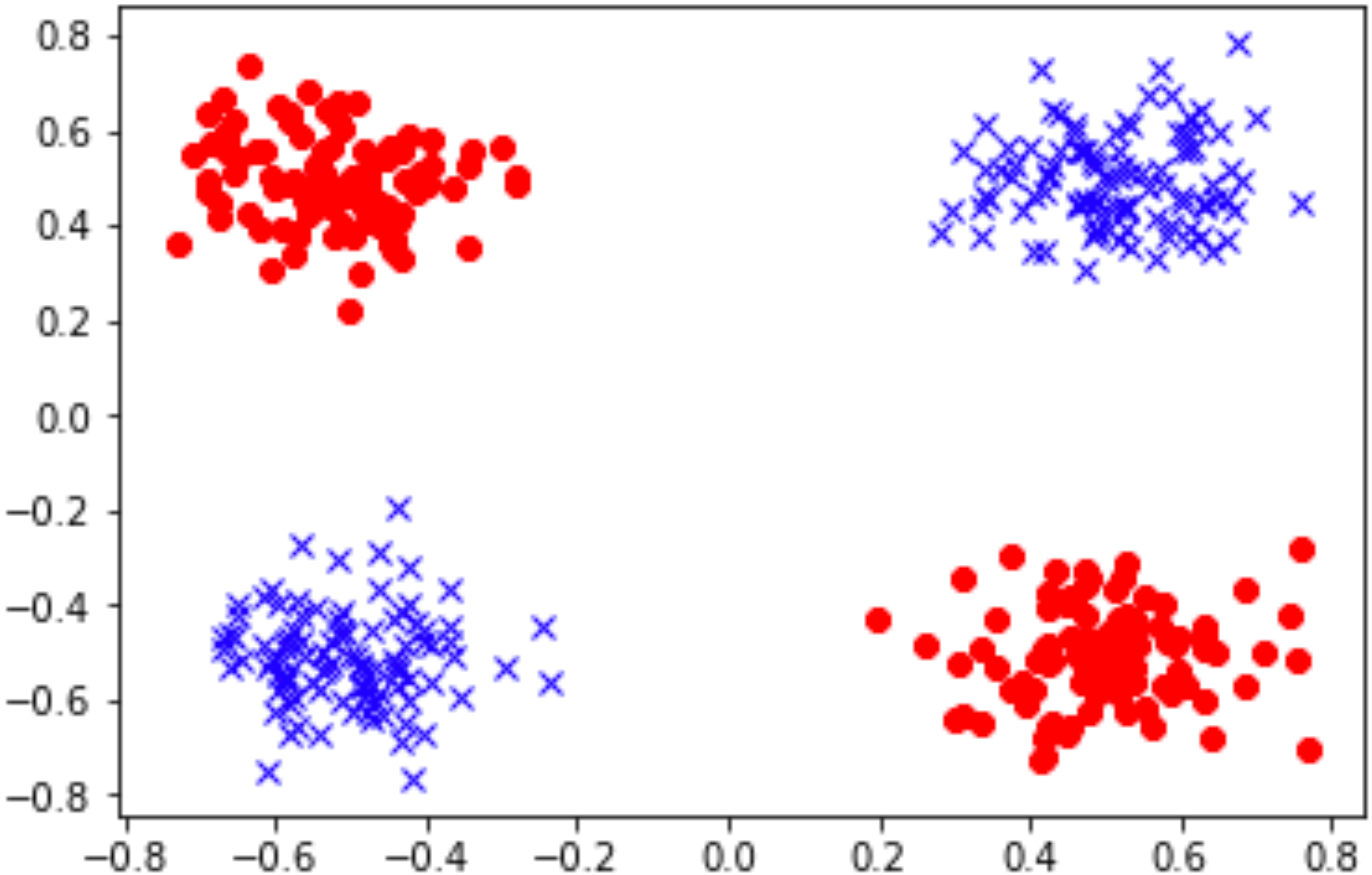
Questions from last time

- HW1?
- Midterm? Tentative plan is Oct 31. Will be finalized next week
- Sometimes you're writing stuff that's off the camera… tell me when this happens!

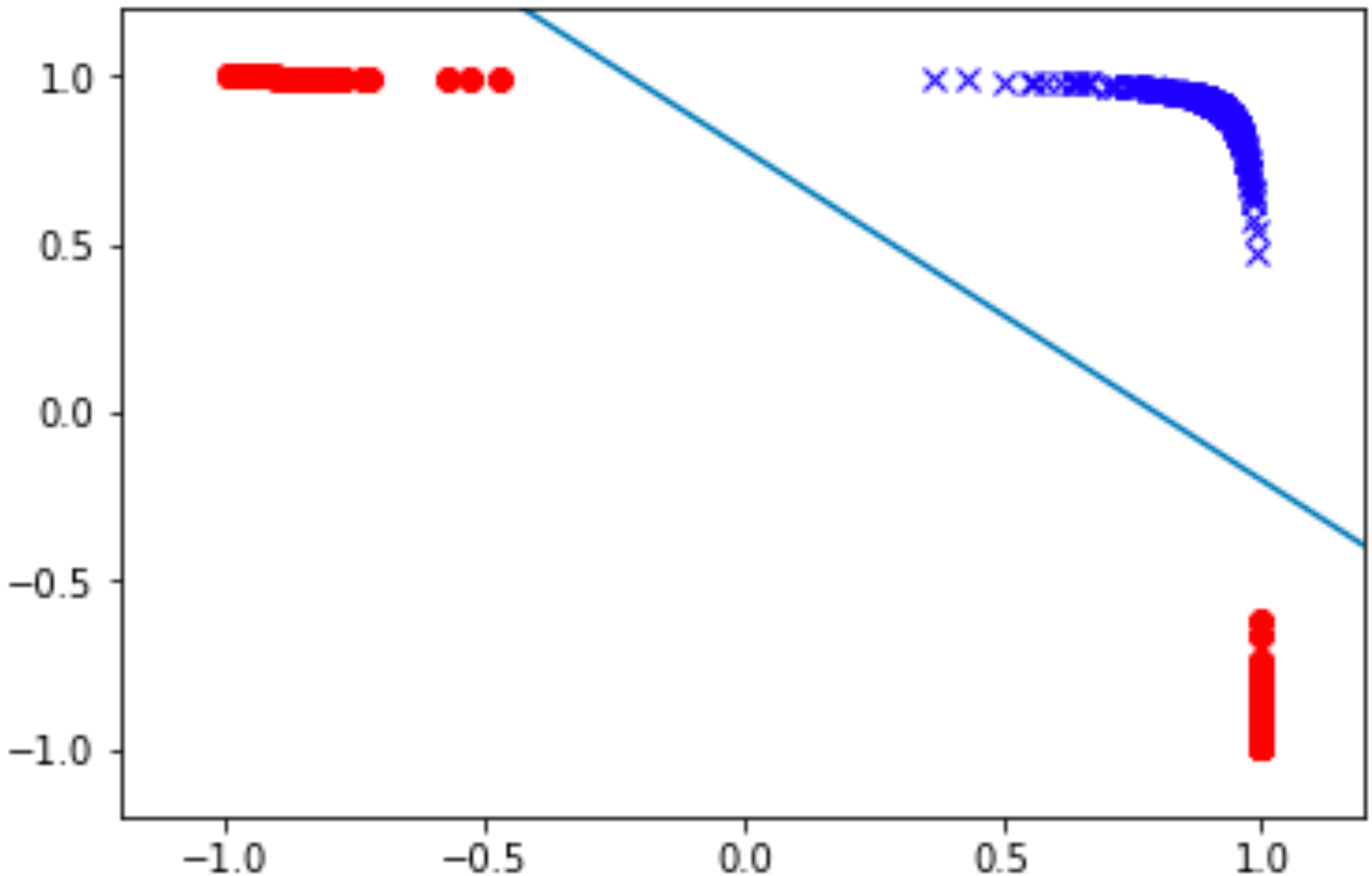
why do neural networks work better?

- multiple layer and *nonlinearities* allow feature combinations that a linear model can't get
  - e.g., XOR function
- the learned representations of words and contexts are tuned to the prediction problem
  - unlike one-hot vectors

are blue and red points linearly separable?



after transforming with tanh(Wx+b)…



this class: focus on the input layer of neural language models:
word embeddings

- we'll talk about what properties a good word representation should encode
- and we'll discuss some algorithms for learning good word representations
- next class: we'll plug this concept of representing vectors with words into our language modeling framework

What do words mean?

First thought: look in a dictionary

http://www.oed.com/

Words, Lemmas, Senses, Definitions

lemma **sense** **definition**

pepper, *n.*

Pronunciation: Brit. /ˈpɛpə/ , U.S. /ˈpɛpər/

Forms: OE **peopor** (*rare*), OE **pipcer** (transmission error), OE **pipor**, OE **pipur** (*rare* .

Frequency (in current use):

Etymology: A borrowing from Latin. **Etymon:** Latin *piper*.

< classical Latin *piper*, a loanword < Indo-Aryan (as is ancient Greek *πέπερι*); compare Sar

I. The spice or the plant.

1.

a. A hot pungent spice derived from the prepared fruits (peppercorns) of the pepper plant, *Piper nigrum* (see sense 2a), used from early times to season food, either whole or ground to powder (often in association with salt). Also (locally, chiefly with distinguishing word): a similar spice derived from the fruits of certain other species of the genus *Piper*; the fruits themselves.

The ground spice from *Piper nigrum* comes in two forms, the more pungent *black pepper*, produced from black peppercorns, and the milder *white pepper*, produced from white peppercorns: see BLACK *adj.* and *n.* Special uses 5a, PEPPERCORN *n.* 1a, and WHITE *adj.* and *n.*[1] Special uses 7b(a).

2.

a. The plant *Piper nigrum* (family Piperaceae), a climbing shrub indigenous to South Asia and also cultivated elsewhere in the tropics, which has alternate stalked entire leaves, with pendulous spikes of small green flowers opposite the leaves, succeeded by small berries turning red when ripe. Also more widely: any plant of the genus *Piper* or the family Piperaceae.

b. Usu. with distinguishing word: any of numerous plants of other families having hot pungent fruits or leaves which resemble pepper (1a) in taste and in some cases are used as a substitute for it.

c. *U.S.* The California pepper tree, *Schinus molle*. Cf. PEPPER TREE *n.*

3. Any of various forms of capsicum, esp. *Capsicum annuum* var. *annuum*. Originally (chiefly with distinguishing word): any variety of the *C. annuum* Longum group, with elongated fruits having a hot, pungent taste, the source of cayenne, chilli powder, paprika, etc., or of the perennial *C. frutescens*, the source of Tabasco sauce. Now frequently (more fully ***sweet pepper***): any variety of the *C. annuum* Grossum group, with large, bell-shaped or apple-shaped, mild-flavoured fruits, usually ripening to red, orange, or yellow and eaten raw in salads or cooked as a vegetable. Also: the fruit of any of these capsicums.

Sweet peppers are often used in their green immature state (more fully ***green pepper***), but some new varieties remain green when ripe.

Relation: Synonymity

Synonyms have the same meaning in some or all contexts.

- couch / sofa
- big / large
- automobile / car
- vomit / throw up
- Water / $H_2 0$

Relation: Antonymy

Senses that are opposites with respect to one feature of meaning

Otherwise, they are very similar!

dark/light short/long fast/slow rise/fall

hot/cold up/down in/out

Relation: Similarity

Words with similar meanings. Not synonyms, but sharing some element of meaning

car, bicycle

cow, horse

Ask humans how similar two words are on scale of 1-10

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)

in NLP, we commonly represent word *types* with **vectors**!

why use vectors to encode meaning?

- computing the similarity between two words (or phrases, or documents) is *extremely* useful for many NLP tasks

- Q: how **tall** is Mount Everest?

  A: The official **height** of Mount Everest is 29029 ft

Word similarity for plagiarism detection

MAINFRAMES

Mainframes are primarily referred to large computers with rapid, advanced processing capabilities that can execute and perform tasks equivalent to many Personal Computers (PCs) machines networked together. It is characterized with high quantity Random Access Memory (RAM), very large secondary storage devices, and high-speed processors to cater for the needs of the computers under its service.

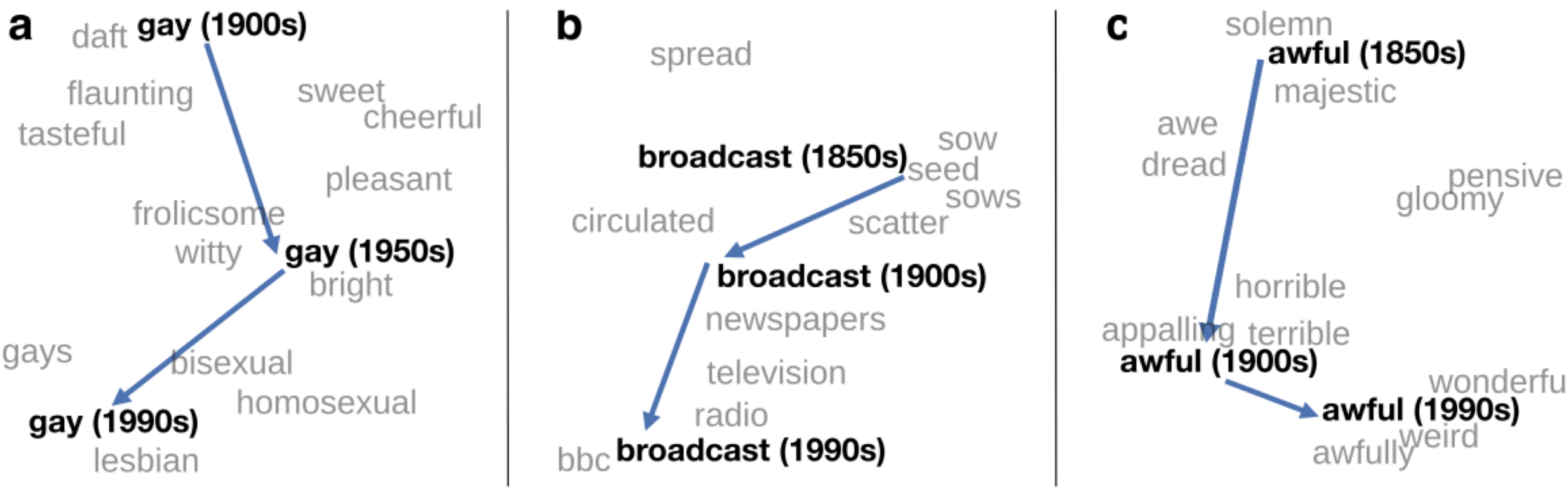
Consisting of advanced components, mainframes have the capability of running multiple large applications required by many and most enterprises and organizations. This is one of its advantages. Mainframes are also suitable to cater for those applications (programs) or files that are of very high

MAINFRAMES

Mainframes usually are referred those computers with fast, advanced processing capabilities that could perform by itself tasks that may require a lot of Personal Computers (PC) Machines. Usually mainframes would have lots of RAMs, very large secondary storage devices, and very fast processors to cater for the needs of those computers under its service.

Due to the advanced components mainframes have, these computers have the capability of running multiple large applications required by most enterprises, which is one of its advantage. Mainframes are also suitable to cater for those applications or files that are of very large demand

visualizing semantic word change over time



~30 million books, 1850-1990, Google Books data

Distributional models of meaning
= vector-space models of meaning
= vector semantics

Intuitions: Zellig Harris (1954):

- “oculist and eye-doctor … occur in almost the same environments”
- “If A and B have almost identical environments we say that they are synonyms.”

Firth (1957):

- “You shall know a word by the company it keeps!”

Intuition of distributional word similarity

```
A bottle of tesgüino is on the table
Everybody likes tesgüino
Tesgüino makes you drunk
We make tesgüino out of corn.
```

- From context words humans can guess **tesgüino** means...

Intuition of distributional word similarity

```
A bottle of tesgüino is on the table
Everybody likes tesgüino
Tesgüino makes you drunk
We make tesgüino out of corn.
```

- From context words humans can guess **tesgüino** means...
- an alcoholic beverage like **beer**
- Intuition for algorithm:
  - Two words are similar if they have similar word contexts.

one-hot vectors

- we’ve already seen these before in bag-of-words models (e.g., naive Bayes)!
- represent each word as a vector of zeros with a single 1 identifying the index of the word

vocabulary
i
hate
love
the
movie
film

movie = <0, 0, 0, 0, 1, 0>

film = <0, 0, 0, 0, 0, 1>

what are the issues of representing a word this way?

all words are equally (dis)similar!

movie = <0, 0, 0, 0, 1, 0>

film = <0, 0, 0, 0, 0, 1>

dot product is zero!

these vectors are **orthogonal**

how can we compute a vector representation such that the dot product correlates with word similarity?

We'll introduce 2 kinds of embeddings

Tf- idf

- A common baseline model
- Sparse vectors
- Words are represented by a simple function of the counts of nearby words

Word2vec

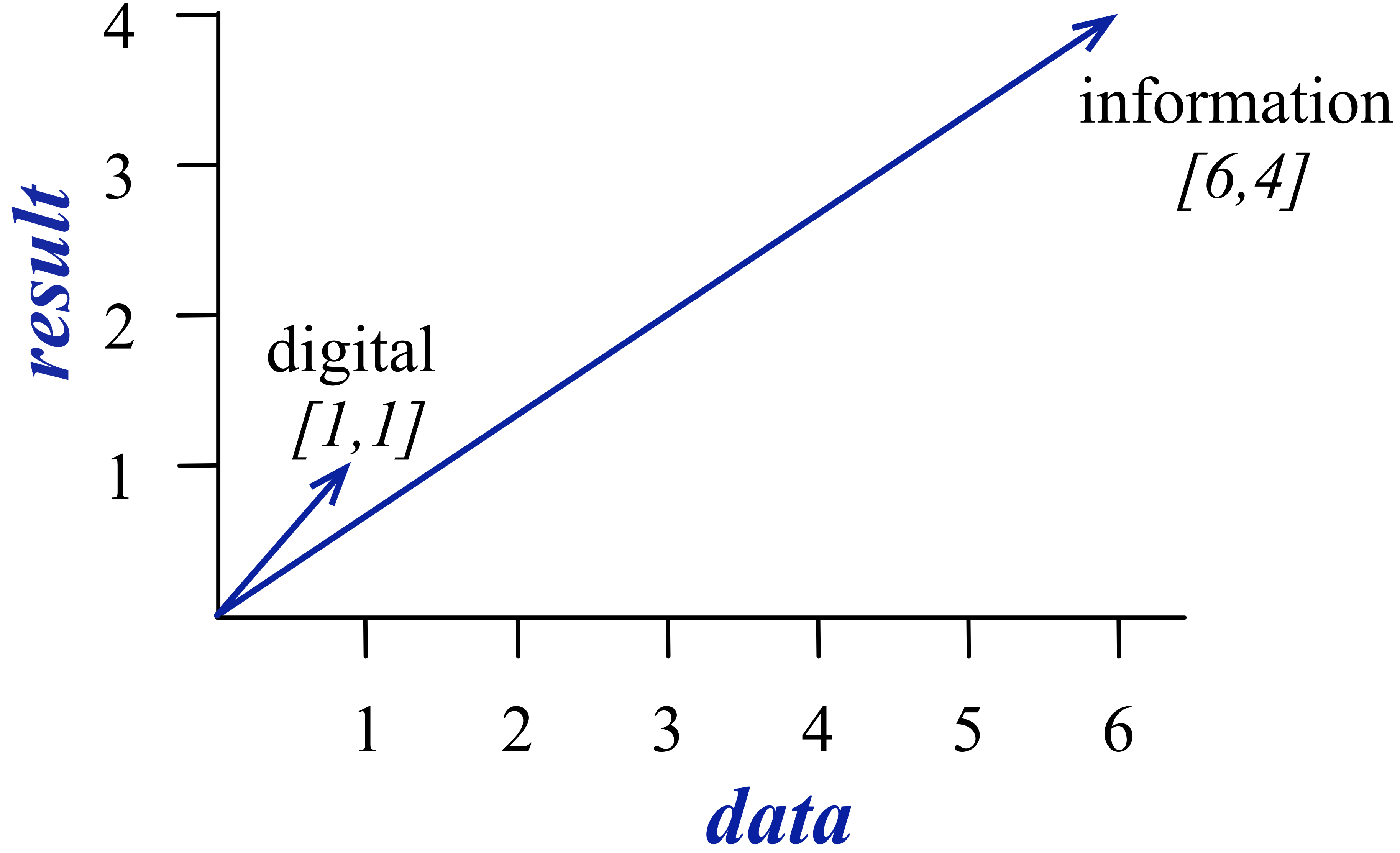
- Dense vectors
- Representation is created by training a classifier to distinguish nearby and far-away words

Word-word co-occurence matrix

Two **words** are similar in meaning if their context vectors are similar

sugar, a sliced lemon, a tablespoonful of	**apricot**	jam, a pinch each of,
their enjoyment. Cautiously she sampled her first	**pineapple**	and another fruit whose taste she likened
well suited to programming on the digital	**computer**.	In finding the optimal R-stage policy from
for the purpose of gathering data and	**information**	necessary for the study authorized in the

	aardvark	computer	data	pinch	result	sugar	…
apricot	0	0	0	1	0	1	
pineapple	0	0	0	1	0	1	
digital	0	2	1	0	1	0	
information	0	1	6	0	4	0	



4
3
2
1
result
information
[6,4]
digital
[1,1]
1
2
3
4
5
6
data

cosine similarity of two vectors

$$\text{cosine}(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

v_i is the count for word *v* in context *i*

w_i is the count for word *w* in context *i*.

$$\vec{a}\cdot\vec{b} = |\vec{a}||\vec{b}|\cos\theta$$

$$\frac{\vec{a}\cdot\vec{b}}{|\vec{a}||\vec{b}|} = \cos\theta$$

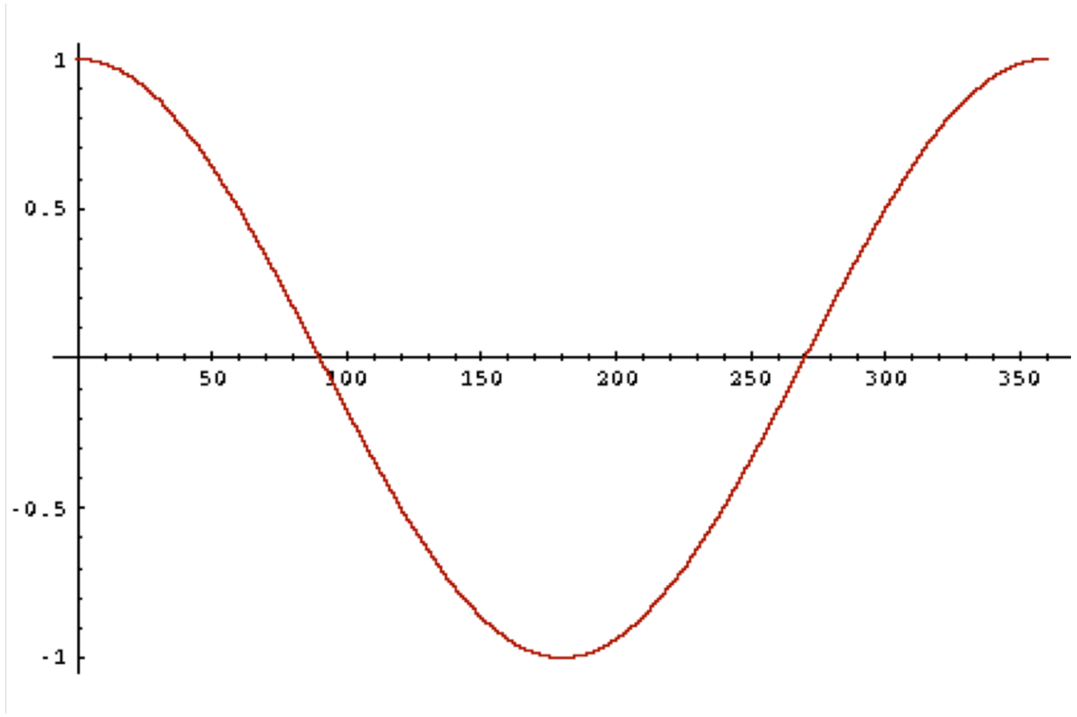
Cos($\vec{v}$,$\vec{w}$) is the cosine similarity of $\vec{v}$ and $\vec{w}$

Cosine as a similarity metric

-1: vectors point in opposite directions

+1: vectors point in same directions

0: vectors are orthogonal



Frequency is non-negative, so cosine range 0-1

$$\cos(\vec{v},\vec{w}) = \frac{\vec{v} \bullet \vec{w}}{|\vec{v}||\vec{w}|} = \frac{\vec{v}}{|\vec{v}|} \bullet \frac{\vec{w}}{|\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

	large	data	computer
apricot	1	0	0
digital	0	1	2
information	1	6	1

Which pair of words is more similar?

cosine(apricot,information) =

cosine(digital,information) =

cosine(apricot,digital) =

	large	data	computer
apricot	1	0	0
digital	0	1	2
information	1	6	1

$$\cos(\vec{v},\vec{w}) = \frac{\vec{v}\bullet\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\vec{v}}{|\vec{v}|}\bullet\frac{\vec{w}}{|\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

Which pair of words is more similar?

cosine(apricot,information) =

$$\frac{1+0+0}{\sqrt{1+0+0}\ \sqrt{1+36+1}} = \frac{1}{\sqrt{38}} = .16$$

cosine(digital,information) =

$$\frac{0+6+2}{\sqrt{0+1+4}\ \sqrt{1+36+1}} = \frac{8}{\sqrt{38}\sqrt{5}} = .58$$

cosine(apricot,digital) =

$$\frac{0+0+0}{\sqrt{1+0+0}\ \sqrt{0+1+4}} = 0$$

But raw frequency is a bad representation

Frequency is clearly useful; if *sugar* appears a lot near *apricot*, that's useful information.

But overly frequent words like *the*, *it,* or *they* are not very informative about the context

Need a function that resolves this frequency paradox!

tf-idf: combine two factors

tf: term frequency. frequency count (usually log-transformed):

$$\mathrm{tf}_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{count}(t,d) & \text{if } \mathrm{count}(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

count(t, d) =
# of occurrences
of word *t* in doc *d*

Idf: inverse document frequency: tf-

Total # of docs in collection

$$\mathrm{idf}_i = \log\left(\frac{N}{\mathrm{df}_i}\right)$$

df_i =
# of documents
containing word *i*

# of docs that have word i

Words like "the" or "good" have very low idf

tf-idf value for word t in document d:

$$w_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t$$

dense word vectors

- model the meaning of a word as an **embedding** in a vector space
  - this vector space is commonly low dimensional (e.g., 100-500d).
  - what is the dimensionality of a one-hot word representation?
- embeddings are real-valued vectors (not binary or counts)

how can we learn embeddings?

Sparse vector representations

1. Mutual-information weighted word co-occurrence matrices

Dense vector representations:

2. Singular value decomposition (and Latent Semantic Analysis)
3. Neural-network-inspired models (skip-grams, CBOW)
4. Brown clusters

Word2vec (Mikolov et al., 2013)

Popular embedding method

Very fast to train

Code available on the web

Idea: **predict** rather than **count**

Word2vec

- Instead of **counting** how often each word *w* occurs near "*apricot*"
- Train a classifier on a binary **prediction** task:
  - Is *w* likely to show up near "*apricot*"?

- We don't actually care about this task
  - But we'll take the learned classifier weights as the word embeddings

Brilliant insight: Use running text as implicitly supervised training 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)

Setup

Let's represent words as vectors of some length (say 300), randomly initialized.

So we start with 300 * V random parameters

Over the entire training set, we’d like to adjust those word vectors such that we

- Maximize the similarity of the target word, context word pairs (t,c) drawn from the positive data
- Minimize the similarity of the (t,c) pairs drawn from the negative data.

word	**dim0**	**dim1**	**dim2**	**dim3**	...	dim300
today	0.35	-1.3	2.2	0.003		
cat	-3.1	-1.7	1.1	-0.56		
sleep	0.55	3.0	2.4	-1.2		
watch	-0.09	0.8	-1.8	2.9		
bird	2.0	0.16	-1.9	2.3		

...

Skip-gram with negative sampling (SGNS)

1. From a large source of text (e.g., Wikipedia), generate *positive examples* by pairing a target word with a word in its neighboring context
2. Create *negative examples* for the context words by randomly sampling other words in the vocabulary
3. Train a *logistic regression* model to identify whether a given pair of words is a positive or negative example
4. Use the weights of this model as the *embeddings*

Skip-Gram Training Data

Training sentence:

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

c1 c2 target c3 c4

Asssume context words are those in +/- 2 word window

Skip-Gram Goal

Given a tuple (t,c) = target, context

- (*apricot*, *jam*)
- (*apricot*, *aardvark*)

Return probability that c is a real context word:

$$P(+|t,c)$$

$$P(-|t,c) = 1 - P(+|t,c)$$

How to compute p(+|t,c)?

Intuition:

- Words are likely to appear near similar words
- Model similarity with dot-product!
- Similarity(t,c) $\propto t \cdot c$

t and **c** here are vectors for **t**arget and **c**ontext!

Problem:

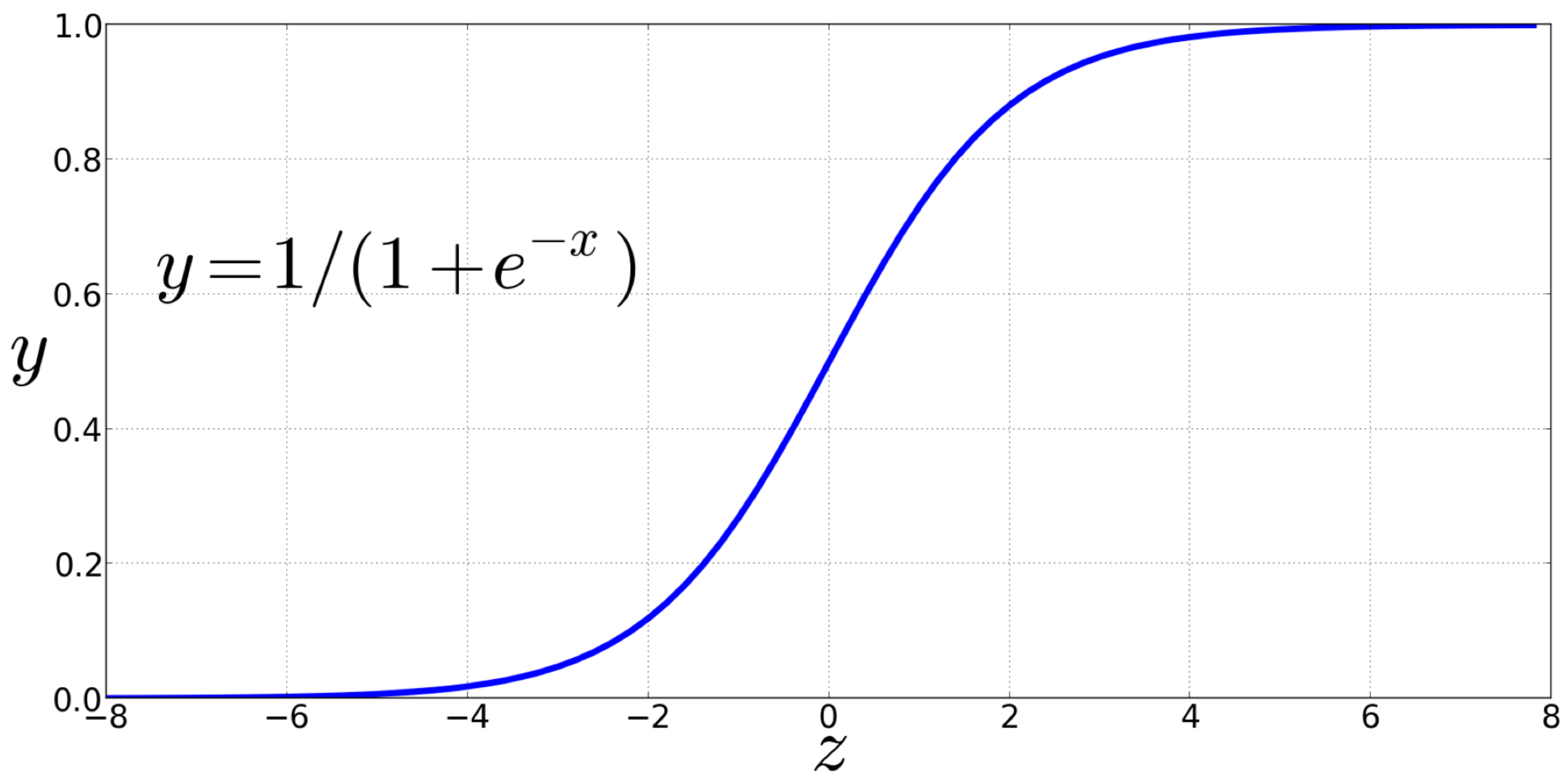
- *Dot product is not a probability!*
  - *(Neither is cosine)*

Turning dot product into a probability

Turning dot product into a probability

The sigmoid lies between 0 and 1:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



Turning dot product into a probability

think back to last class…
what are our features and weights here???

both target and context vectors are *learned*, so we have no explicit featurization!

$$P(+|t,c) = \frac{1}{1+e^{-t\cdot c}}$$

$$\begin{aligned} P(-|t,c) &= 1 - P(+|t,c) \\ &= \frac{e^{-t\cdot c}}{1+e^{-t\cdot c}} \end{aligned}$$

Learning the classifier

Iterative process.

We'll start with 0 or random weights

Then adjust the word weights to

- make the positive pairs more likely
- and the negative pairs less likely

over the entire training set

guess what algorithm we'll use to make this happen?

gradient descent!!!!!!!!

Objective Criteria

We want to maximize…

$$\sum_{(t,c)\in +} logP(+|t,c) + \sum_{(t,c)\in -} logP(-|t,c)$$

Maximize the + label for the pairs from the positive training data, and the – label for the pairs sample from the negative data.

Focusing on one target word t:

n_i is the vector for the negative sample

$$
\begin{aligned}
L(\theta) &= \log P(+|t,c) + \sum_{i=1} \log P(-|t,n_i) \\
&= \log \sigma(c \cdot t) + \sum_{i=1}^{k} \log \sigma(-n_i \cdot t) \\
&= \log \frac{1}{1+e^{-c \cdot t}} + \sum_{i=1}^{k} \log \frac{1}{1+e^{n_i \cdot t}}
\end{aligned}
$$

Summary: How to learn word2vec (skip-gram) embeddings

Start with V random 300-dimensional vectors as initial embeddings

Use logistic regression, the second most basic classifier used in machine learning after naïve bayes

- Take a corpus and take pairs of words that co-occur as positive examples
- Take pairs of words that don't co-occur as negative examples
- Train the classifier to distinguish these by slowly adjusting all the embeddings to improve the classifier performance
- Throw away the classifier code and keep the embeddings.

Evaluating embeddings

Compare to human scores on word similarity-type tasks:

- WordSim-353 (Finkelstein et al., 2002)
- SimLex-999 (Hill et al., 2015)
- Stanford Contextual Word Similarity (SCWS) dataset (Huang et al., 2012)
- TOEFL dataset: *Levied is closest in meaning to: imposed, believed, requested, correlated*

Properties of embeddings

Similarity depends on window size C

C = ±2 The nearest words to *Hogwarts:*

- *Sunnydale*
- *Evernight*

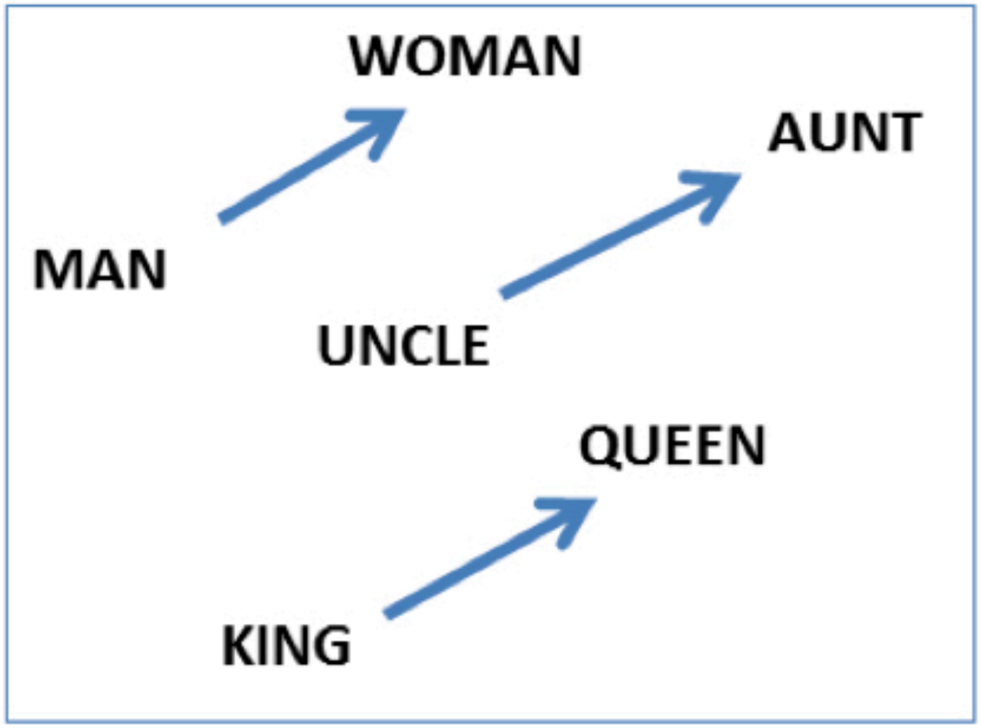
C = ±5 The nearest words to *Hogwarts:*

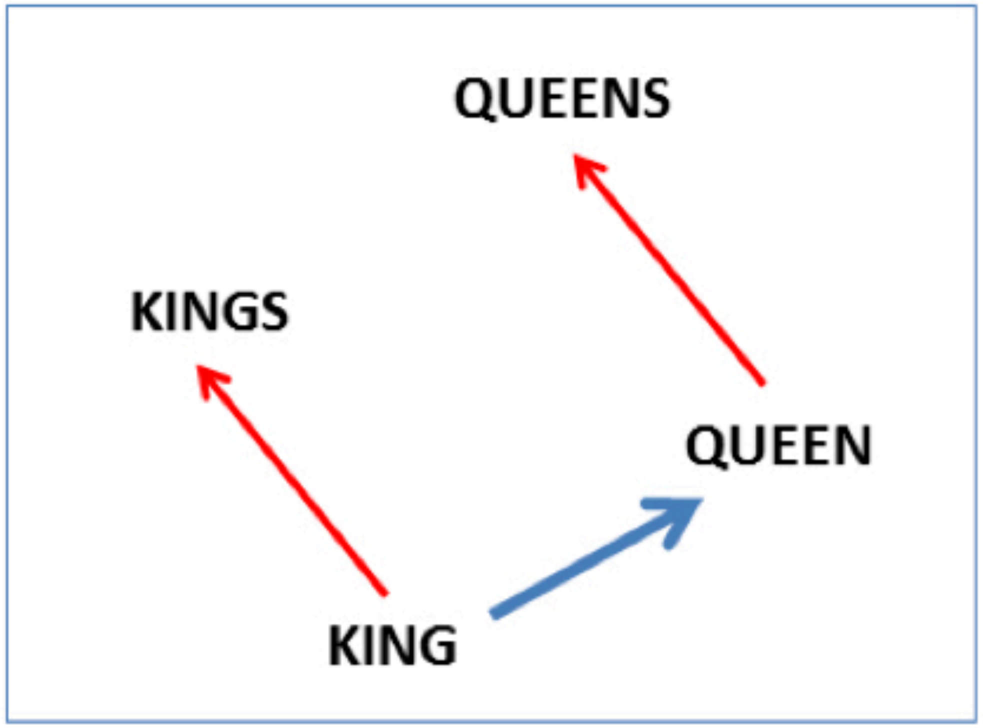
- *Dumbledore*
- *Malfoy*
- *halfblood*

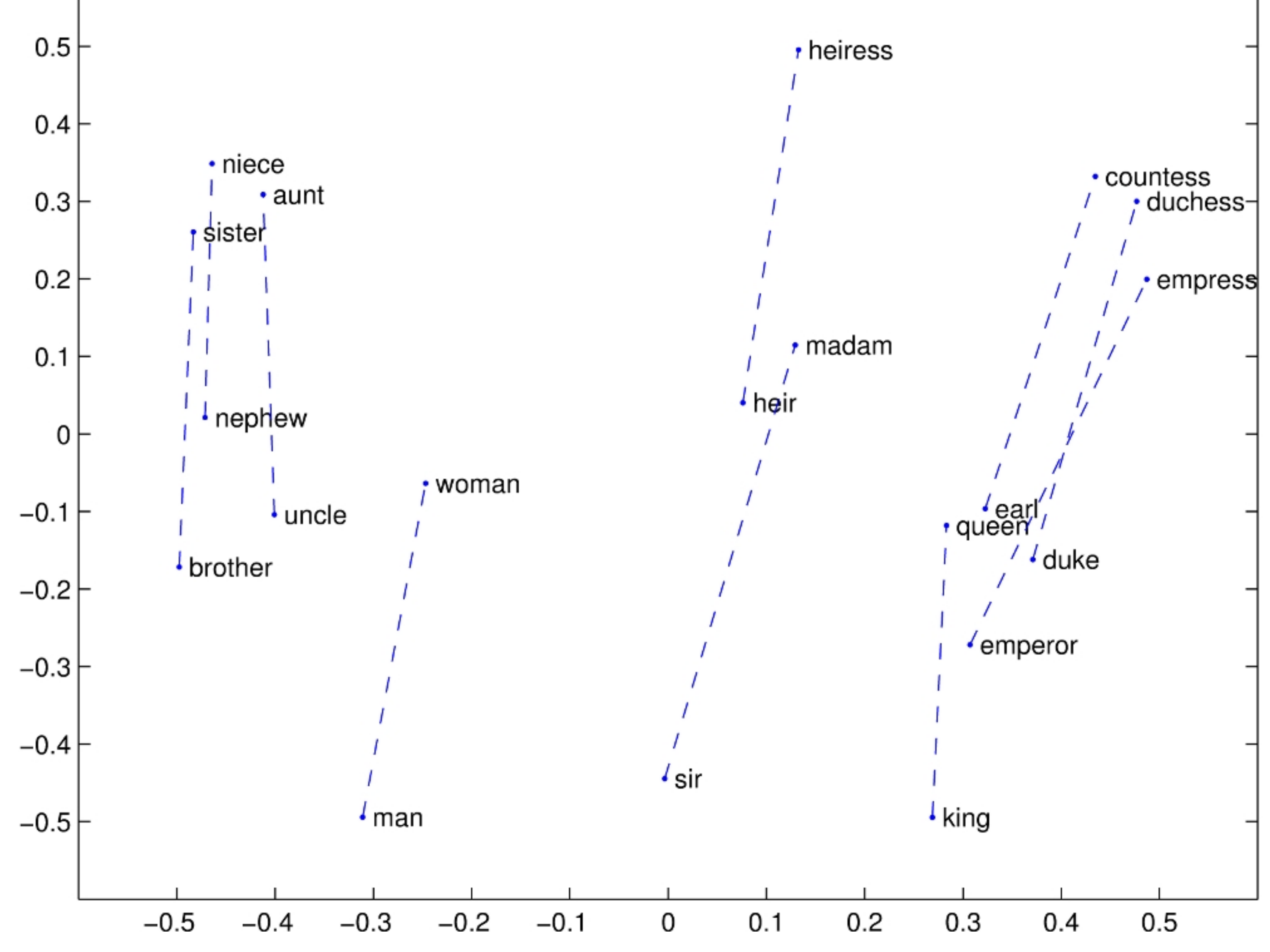
Analogy: Embeddings capture relational meaning!

vector(*'king'*) - vector(*'man'*) + vector(*'woman'*) ≈ vector('queen')

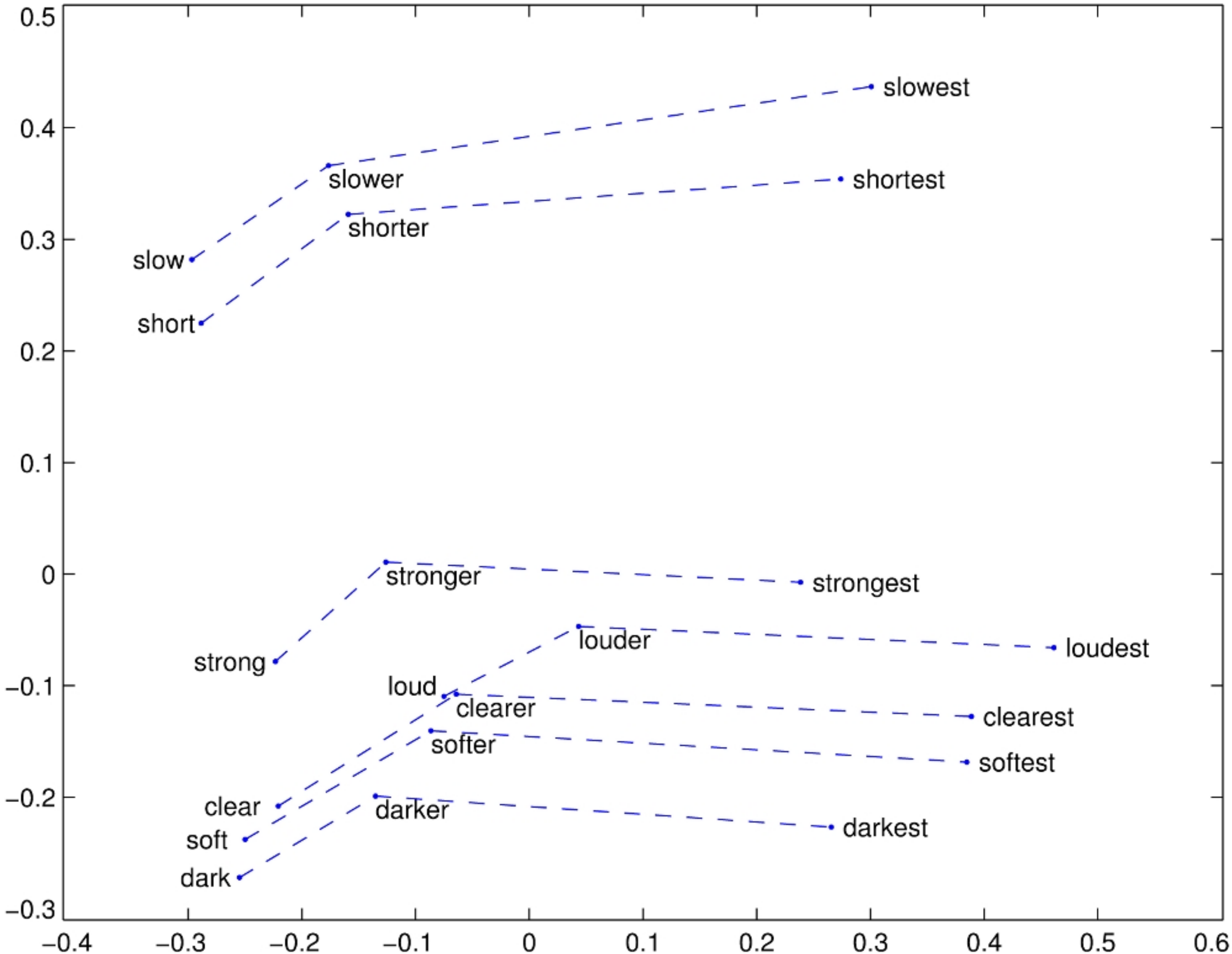
vector(*'Paris'*) - vector(*'France'*) + vector(*'Italy'*) ≈ vector('Rome')







heiress
niece
aunt
sister
nephew
uncle
brother
woman
man
madam
heir
sir
countess
duchess
empress
earl
queen
duke
emperor
king
0.5
0.4
0.3
0.2
0.1
0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.5
−0.4
−0.3
−0.2
−0.1
0
0.1
0.2
0.3
0.4
0.5



slowest
slower
shortest
shorter
slow
short
stronger
strongest
louder
loudest
strong
loud
clearer
clearest
softer
softest
clear
darker
darkest
soft
dark
0.5
0.4
0.3
0.2
0.1
0
−0.1
−0.2
−0.3
−0.4
−0.3
−0.2
−0.1
0
0.1
0.2
0.3
0.4
0.5
0.6

Embeddings reflect cultural bias

Bolukbasi, Tolga, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam T. Kalai. "Man is to computer programmer as woman is to homemaker? debiasing word embeddings." In *Advances in Neural Information Processing Systems*, pp. 4349-4357. 2016.

Ask "Paris : France :: Tokyo : x"

- x = Japan

Ask "father : doctor :: mother : x"

- x = nurse

Ask "man : computer programmer :: woman : x"

- x = homemaker

huge concern for NLP systems deployed in the real world that use embeddings!

Occupations		Adjectives	
Man	Woman	Man	Woman
carpenter	nurse	honorable	maternal
mechanic	midwife	ascetic	romantic
mason	librarian	amiable	submissive
blacksmith	housekeeper	dissolute	hysterical
retired	dancer	arrogant	elegant
architect	teacher	erratic	caring
engineer	cashier	heroic	delicate
mathematician	student	boyish	superficial
shoemaker	designer	fanatical	neurotic
physicist	weaver	aimless	attractive

Table 7: Top occupations and adjectives by gender in the Google News embedding.

Changes in framing: adjectives associated with Chinese

Garg, Nikhil, Schiebinger, Londa, Jurafsky, Dan, and Zou, James (2018). Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, *115*(16), E3635–E3644

1910	1950	1990
Irresponsible	Disorganized	Inhibited
Envious	Outrageous	Passive
Barbaric	Pompous	Dissolute
Aggressive	Unstable	Haughty
Transparent	Effeminate	Complacent
Monstrous	Unprincipled	Forceful
Hateful	Venomous	Fixed
Cruel	Disobedient	Active
Greedy	Predatory	Sensitive
Bizarre	Boisterous	Hearty

Rachel Thomas
@math_rachel

Follow

Biased word embeddings in action: a rating system ranked Mexican restaurants worse, bc Mexican had neg connotations blog.conceptnet.io/2017/04/24/con ...

I had tried building an algorithm for sentiment analysis based on word embeddings — evaluating how much people like certain things based on what they say about them. When I applied it to restaurant reviews, I found it was ranking Mexican restaurants lower. The reason was not reflected in the star ratings or actual text of the reviews. It's not that people don't like Mexican food. The reason was that the system had learned the word "Mexican" from reading the Web.

Directions

Debiasing algorithms for embeddings

- Bolukbasi, Tolga, Chang, Kai-Wei, Zou, James Y., Saligrama, Venkatesh, and Kalai, Adam T. (2016). Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems*, pp. 4349–4357.

Use embeddings as a historical tool to study bias exercise!