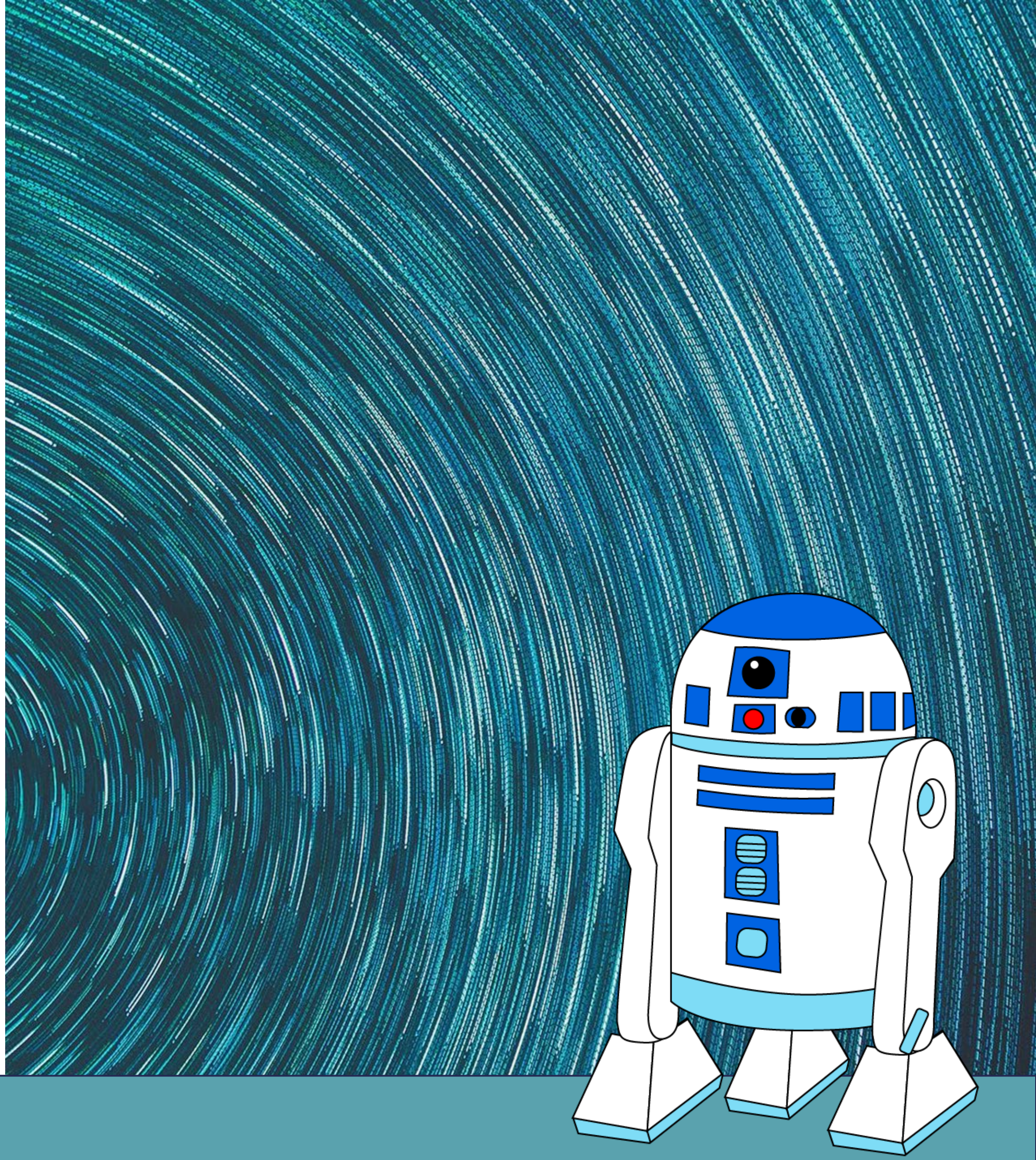
CIS 421/521:
ARTIFICIAL INTELLIGENCE

Vector Semantics part 2

Jurafsky and Martin Chapter 6



Penn Engineering
UNIVERSITY of PENNSYLVANIA

Sparse vectors

- Documents created by term-by-document or term-context matrices are
  - **long** (length |V|= 20,000 to 50,000)
  - **sparse** (most elements are zero)

Alternative: dense vectors

- vectors which are
  - **short** (length 50-1000)
  - **dense** (most elements are non-zero)

Sparse versus dense vectors

- Why dense vectors?
  - Short vectors may be easier to use as **features** in machine learning (fewer weights to tune)
  - Dense vectors may **generalize** better than storing explicit counts
  - They may do better at capturing synonymy:
    - *car* and *automobile* are synonyms; but are distinct dimensions in sparse vectors
      - a word with *car* as a neighbor and a word with *automobile* as a neighbor should be similar, but aren't
  - **In practice, they work better**

Dense embeddings you can download!

- **Word2vec** (Mikolov et al.)
- https://code.google.com/archive/p/word2vec/
- **Fasttext** http://www.fasttext.cc/
- **Glove** (Pennington, Socher, Manning)
- http://nlp.stanford.edu/projects/glove/
- **Magnitude** (Patel and Sands)
- https://github.com/plasticityai/magnitude

Word2vec

- 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))

Word2Vec: Skip-Gram Task

- Word2vec provides a variety of options. Let's do
  - "skip-gram with negative sampling" (SGNS)

Skip-gram algorithm

1. Treat the target word and a neighboring context word as positive examples.
2. Randomly sample other words in the lexicon to get negative samples
3. Use logistic regression to train a classifier to distinguish those two cases
4. Use the weights as the embeddings

Skip-Gram Training Data

- Training sentence:
- ... lemon, a tablespoon of **apricot** jam a pinch ...
- c1 c2 target c3 c4

Assume 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)?

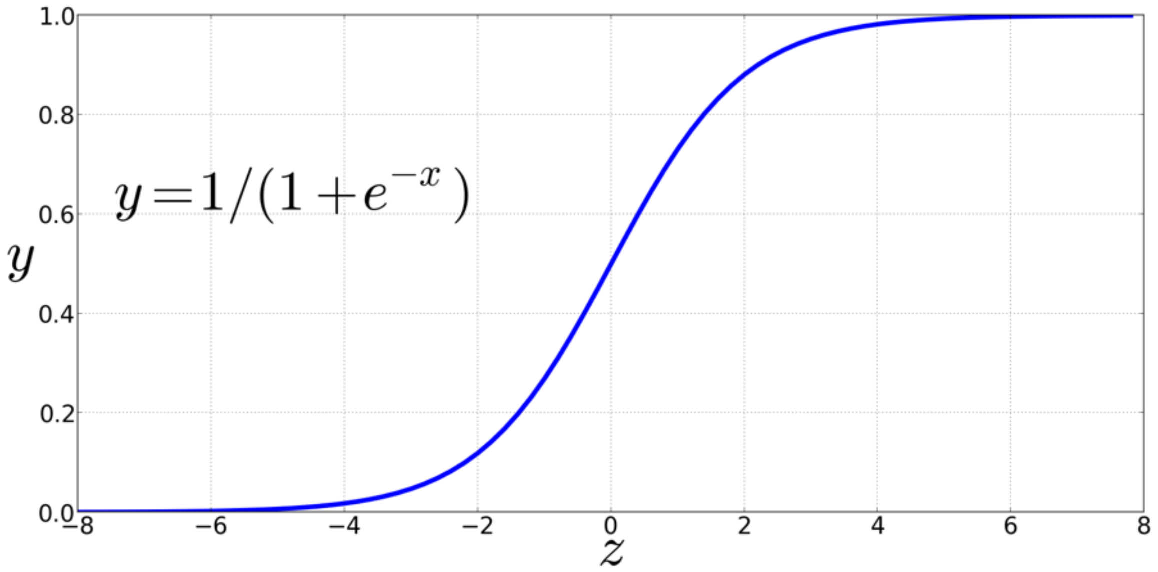
- Intuition:
  - Words are likely to appear near similar words
  - Model similarity with dot-product!
  - Similarity(t,c) ≈ t · c
- *Problem:*
  - *Dot product is not a probability!*
    - *(Neither is cosine)*

$$\text{dot-product}(\vec{v},\vec{w}) = \vec{v}\cdot\vec{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \ldots + v_N w_N$$

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

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

Turning dot product into a probability

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

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

For all the context words:

- Assume all context words are independent

$$P(+|t, c_{1:k}) = \prod_{i=1}^{\kappa} \frac{1}{1 + e^{-t \cdot c_i}}$$

For all the context words:

- Assume all context words are independent

$$P(+|t,c_{1:k}) = \prod_{i=1}^{k} \frac{1}{1+e^{-t\cdot c_i}}$$

$$\log P(+|t,c_{1:k}) = \sum_{i=1}^{k} \log \frac{1}{1+e^{-t\cdot c_i}}$$

Popping back up

- Now we have a way of computing the probability of p(+|t,c), which is the probability that c is a real context word for t.
- **But,** we need embeddings for **t** and **c** to do it.
- Where do we get those embeddings?
- Word2vec learns them automatically!
- It starts with an initial set of embedding vectors and then iteratively shifts the embedding of each word *w* to be more like the embeddings of words that occur nearby in texts, and less like the embeddings of words that don't occur nearby.

Skip-Gram Training Data

- Training sentence:
- … lemon, a tablespoon of **apricot** jam a pinch …
- c1 c2 t c3 c4

- Training data: input/output pairs centering on *apricot*
- Assume a +/- 2 word window

Skip-Gram Training

- Training sentence:
- ... lemon, a tablespoon of **apricot** jam a pinch ...
- c1 c2 t c3 c4

positive examples +	
t	c
apricot	tablespoon
apricot	of
apricot	preserves
apricot	or

For each positive example, we'll create *k* negative examples.
Using *noise* words
Any random word that isn't *t*

How many noise words?

- Training sentence:
- … lemon, a tablespoon of **apricot** jam a pinch …
- c1 c2 t c3 c4

k=2

positive examples +	
t	c
apricot	tablespoon
apricot	of
apricot	preserves
apricot	or

negative examples -			
t	c	t	c
apricot	aardvark	apricot	twelve
apricot	puddle	apricot	hello
apricot	where	apricot	dear
apricot	coaxial	apricot	forever

Choosing noise words

- Could pick w according to their unigram frequency P(w)
- More common to chosen then according to $p_\alpha(w)$

$$P_\alpha(w) = \frac{count(w)^\alpha}{\sum_w count(w)^\alpha}$$

- α= ¾ works well because it gives rare noise words slightly higher probability
- To show this, imagine two events p(a)=.99 and p(b) = .01:

$$P_\alpha(a) = \frac{.99^{.75}}{.99^{.75} + .01^{.75}} = .97$$

$$P_\alpha(b) = \frac{.01^{.75}}{.99^{.75} + .01^{.75}} = .03$$

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:

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.

Objective Criteria

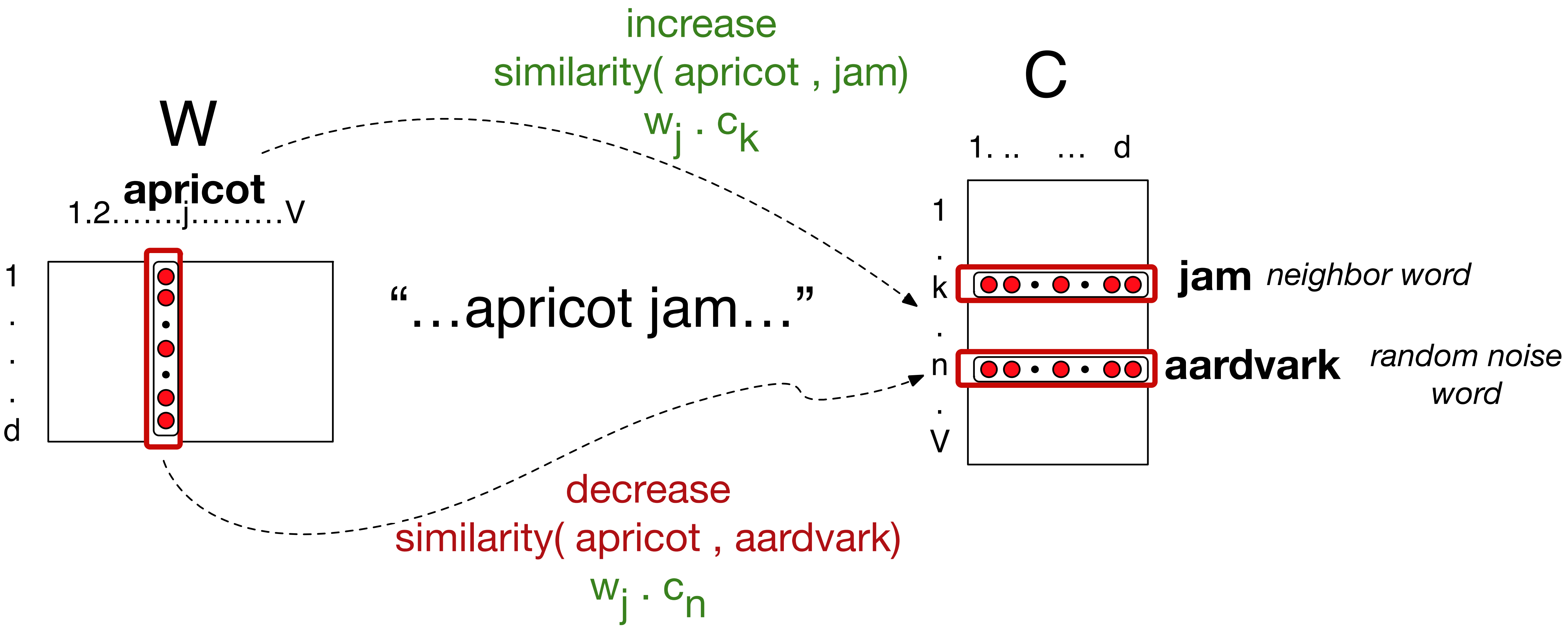
- We want to maximize…

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

- 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:

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



W
apricot
1.2…….j………V
1
.
.
.
d
increase
similarity(apricot , jam)
$w_j \cdot c_k$
C
1. ..   … d
1
.
k
.
n
.
V
jam neighbor word
aardvark random noise word
"…apricot jam…"
decrease
similarity(apricot , aardvark)
$w_j \cdot c_n$

Train using gradient descent

- Actually learns two separate embedding matrices W and C
- Can use W and throw away C, or merge them somehow

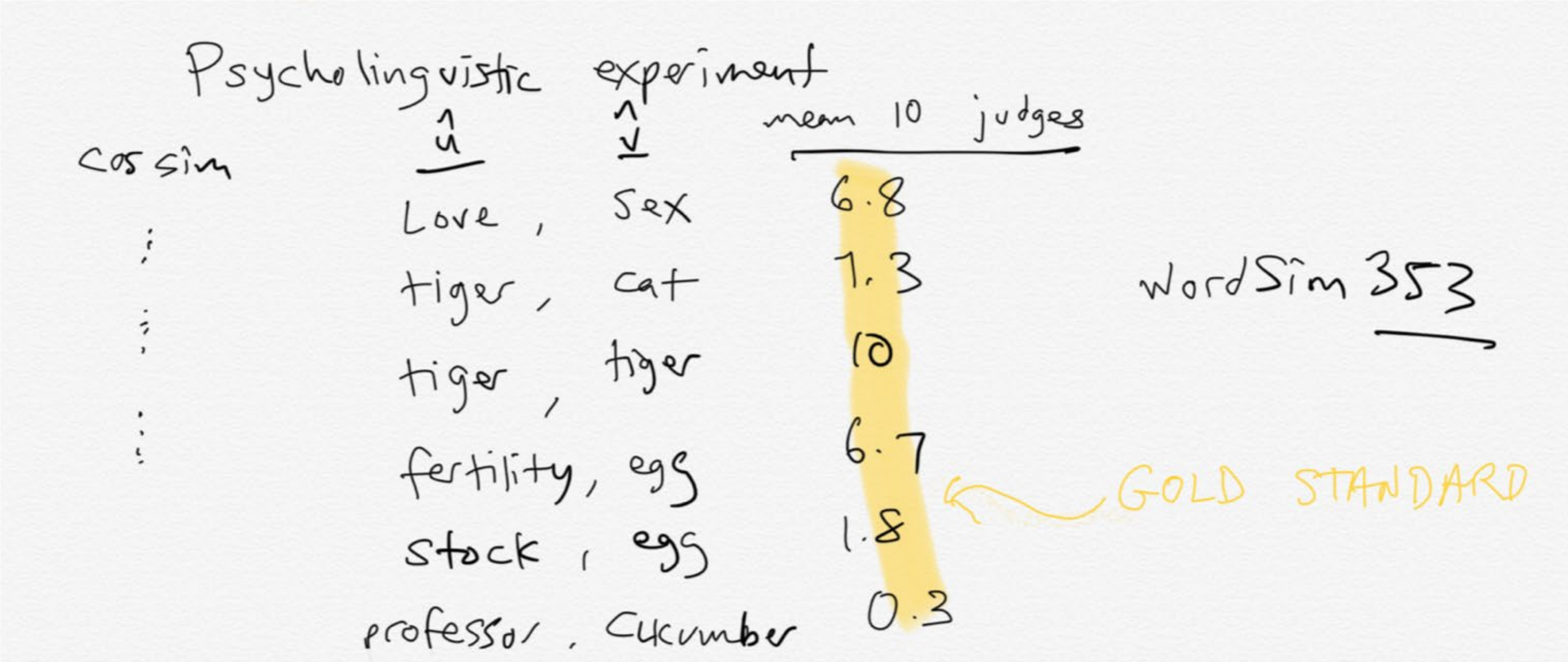
Summary: How to learn word2vec (skip-gram) embeddings

- 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.

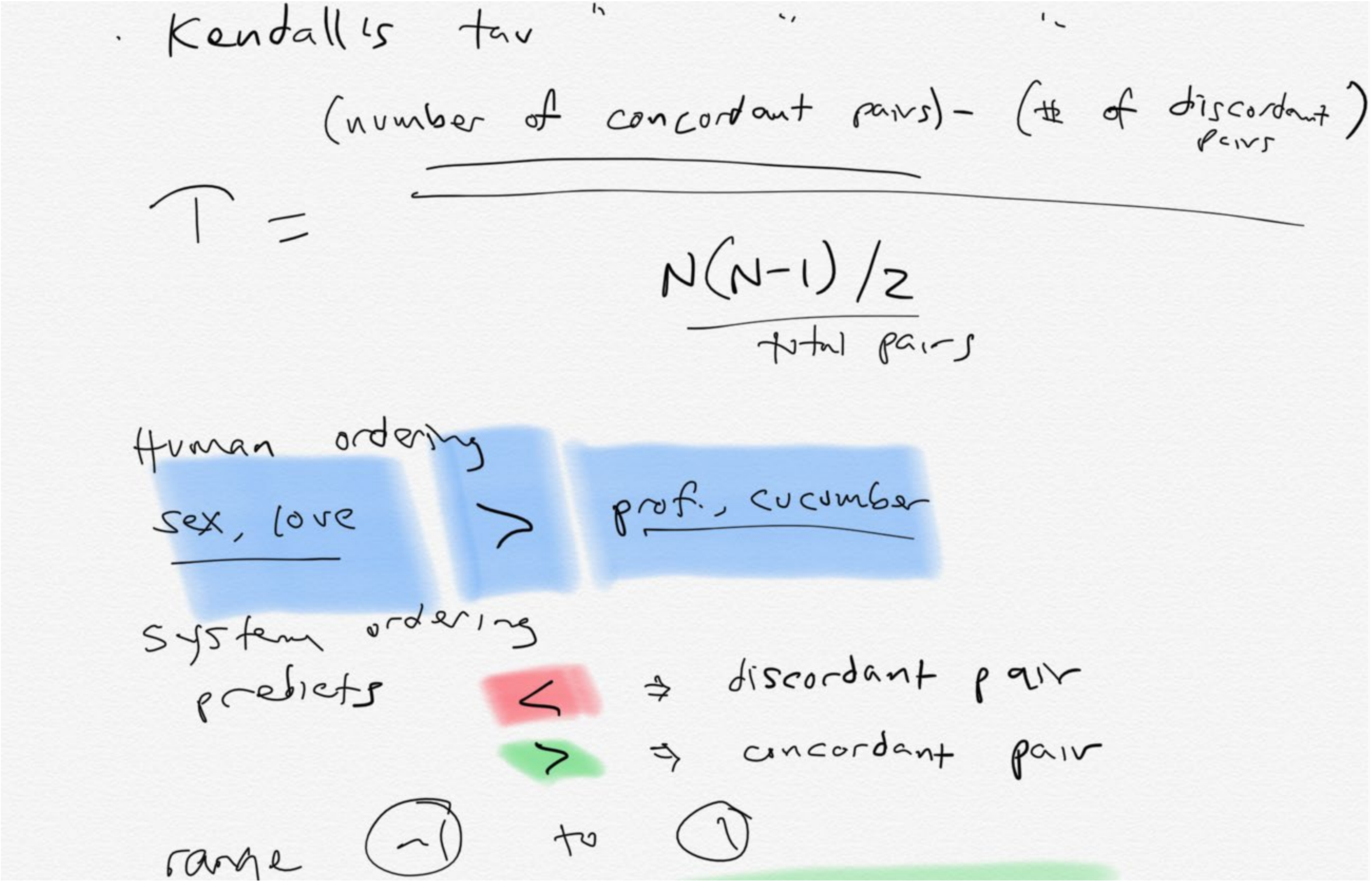
Evaluating embeddings

- 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:*
    (a) imposed, (b) believed, (c) requested, (d) correlated

Intrinsic evaluation



Compute correlation



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*

How does context window change word emeddings?

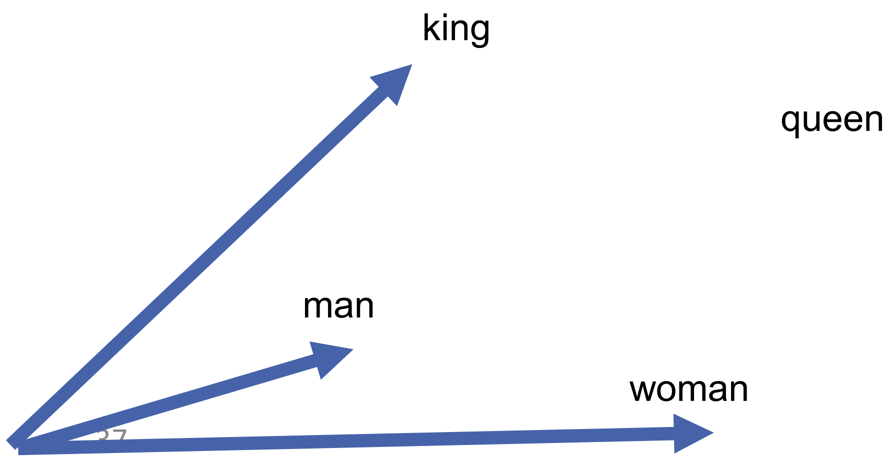
Target Word	BoW5	BoW2	Deps
batman	nightwing aquaman catwoman superman manhunter	superman superboy aquaman catwoman batgirl	superman superboy supergirl catwoman aquaman
hogwarts	dumbledore hallows half-blood malfoy snape	evernight sunnydale garderobe blandings collinwood	sunnydale collinwood calarts greendale millfield
florida	gainesville fla jacksonville tampa lauderdale	fla alabama gainesville tallahassee texas	texas louisiana georgia california carolina
	aspect-oriented	aspect-oriented	event-driven

Solving analogies with embeddings

- In a word-analogy task we are given two pairs of words that share a relation (e.g. "man:woman", "king:queen").
- The identity of the fourth word ("queen") is hidden, and we need to infer it based on the other three by answering
- "*man* is to *woman* as *king* is to — ?"
- More generally, we will say **a:a∗** as **b:b∗.**
- **Can we solve these with word vectors?**

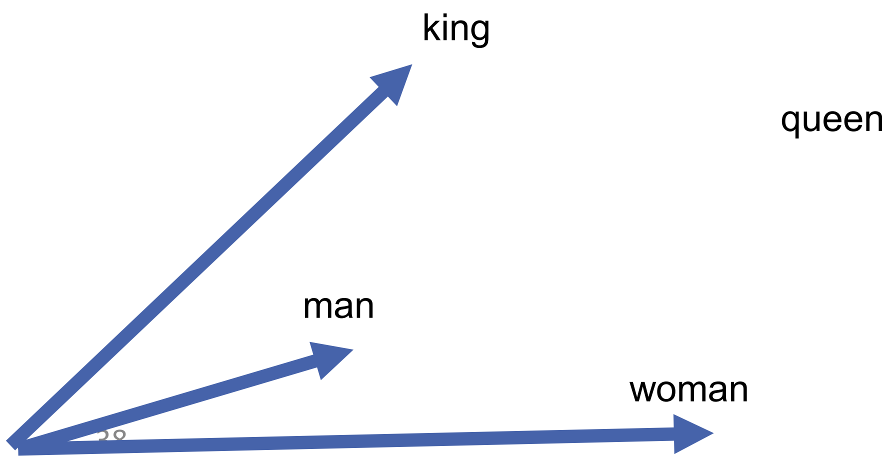
Vector Arithmetic

- **a:a∗** as **b:b∗. b∗** is a hidden vector.
- b∗ should be similar to the vector b − a + a∗
- vector(‘*king*’) - vector(‘*man*’) + vector(‘*woman*’) ≈ vector(‘queen’)



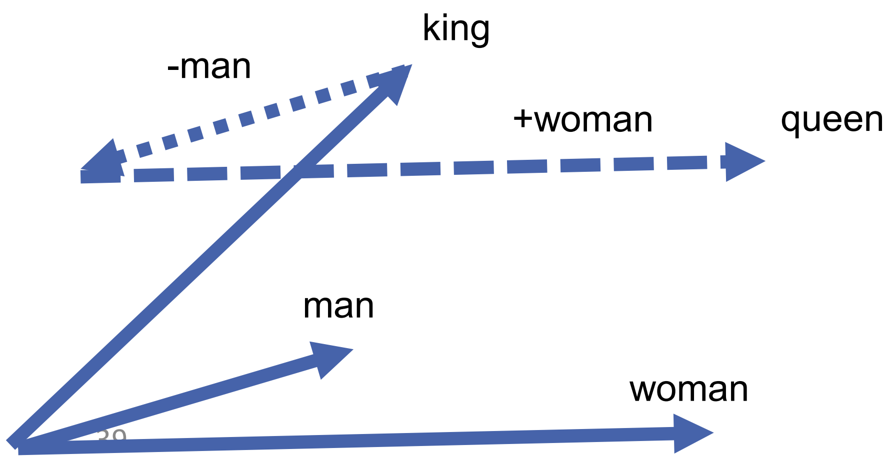
Vector Arithmetic

- **$a:a^*$** as **$b:b^*$**. **b^*** is a hidden vector.
- b^* should be similar to the vector $b - a + a^*$
- vector('*king*') - vector('*man*') + vector('*woman*') ≈ vector('queen')



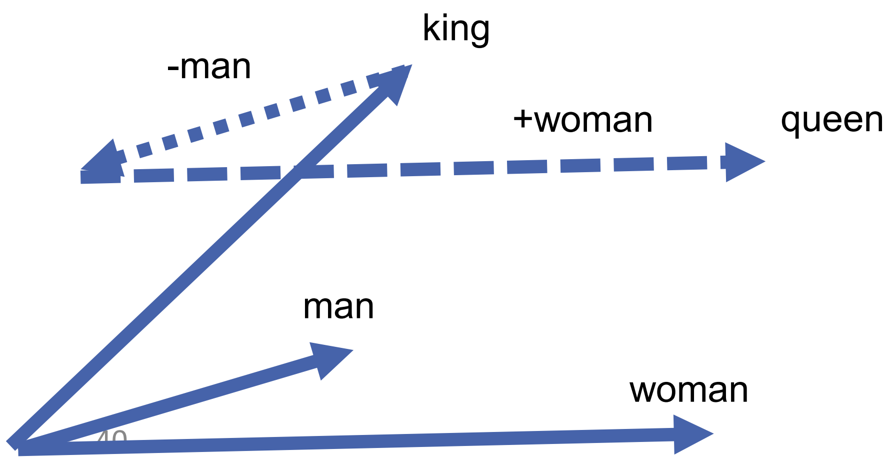
Analogy: Embeddings capture relational meaning!

- **a:a∗** as **b:b∗. b∗** is a hidden vector.
- b∗ should be similar to the vector $b - a + a*$
- vector(‘*king*’) - vector(‘*man*’) + vector(‘*woman*’) ≈ vector(‘queen’)



Vector Arithmetic

- **a:a∗** as **b:b∗.** **b∗** is a hidden vector.
- b∗ should be similar to the vector b − a + a∗
- vector(‘*king*’) - vector(‘*man*’) + vector(‘*woman*’) ≈ vector(‘queen’)

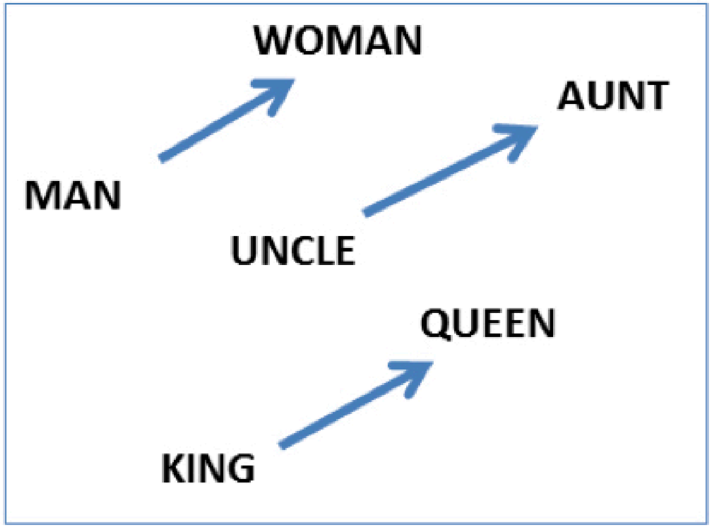


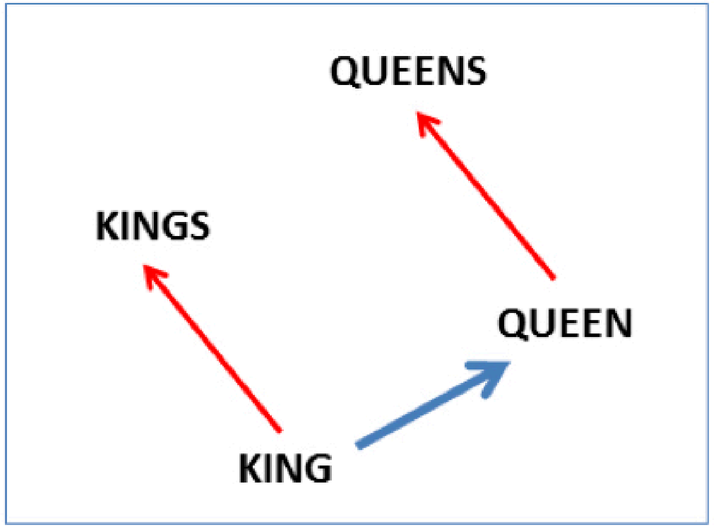
The analogy question can be solved by optimizing:

$$\arg\max_{b^* \in V} \left(\cos\left(b^*, b - a + a^*\right)\right)$$

Analogy: Embeddings capture relational meaning!

- vector(‘*king*’) - vector(‘*man*’) + vector(‘*woman*’) ≈ vector(‘queen’)
- V
  V





Vector Arithmetic

If all word-vectors are normalized to unit length then

$$\arg\max_{b^* \in V} \left(\cos\left(b^*, b - a + a^*\right)\right)$$

is equivalent to

$$\arg\max_{b^* \in V} \left(\cos\left(b^*, b\right) - \cos\left(b^*, a\right) + \cos\left(b^*, a^*\right)\right)$$

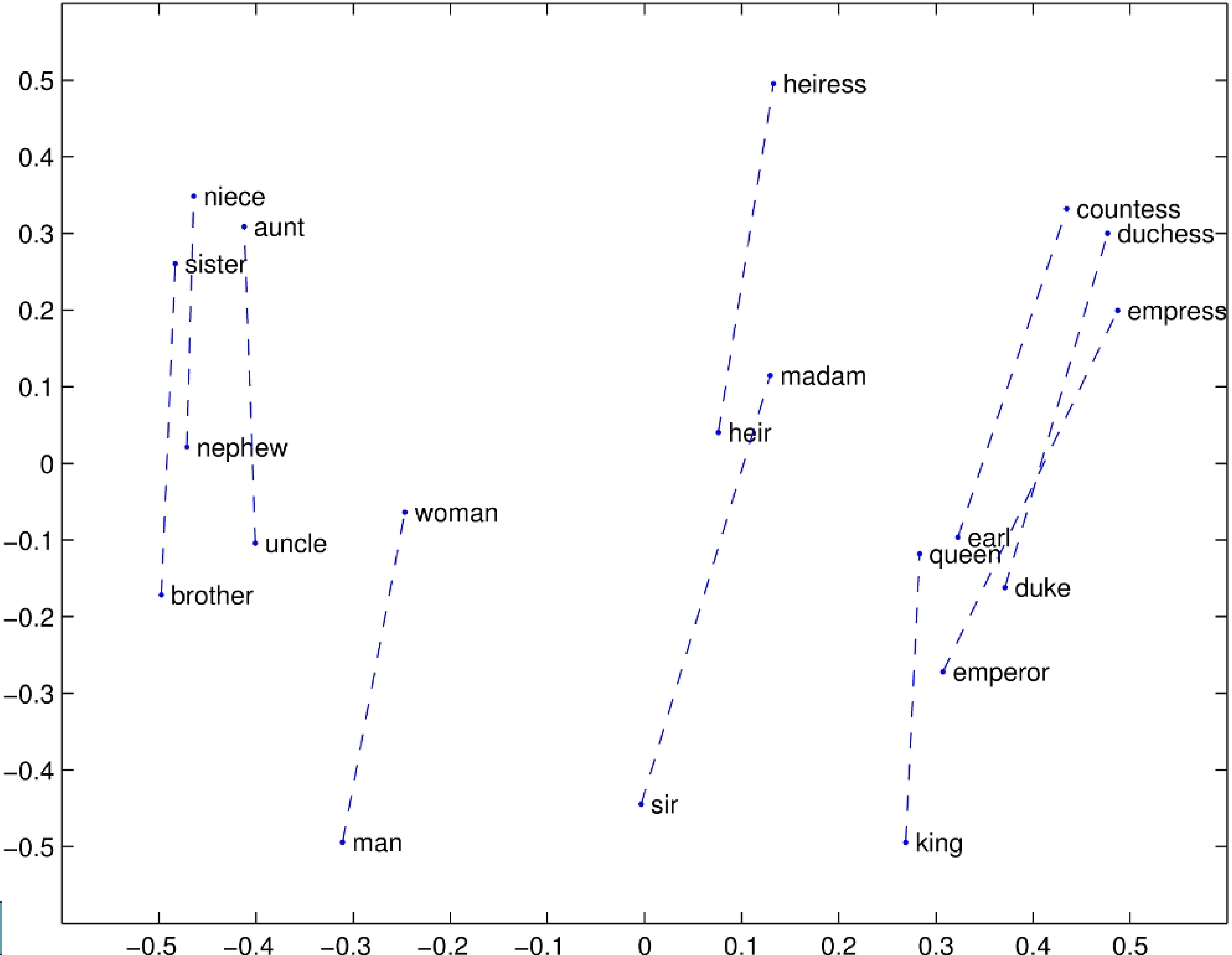
Vector Arithmetic

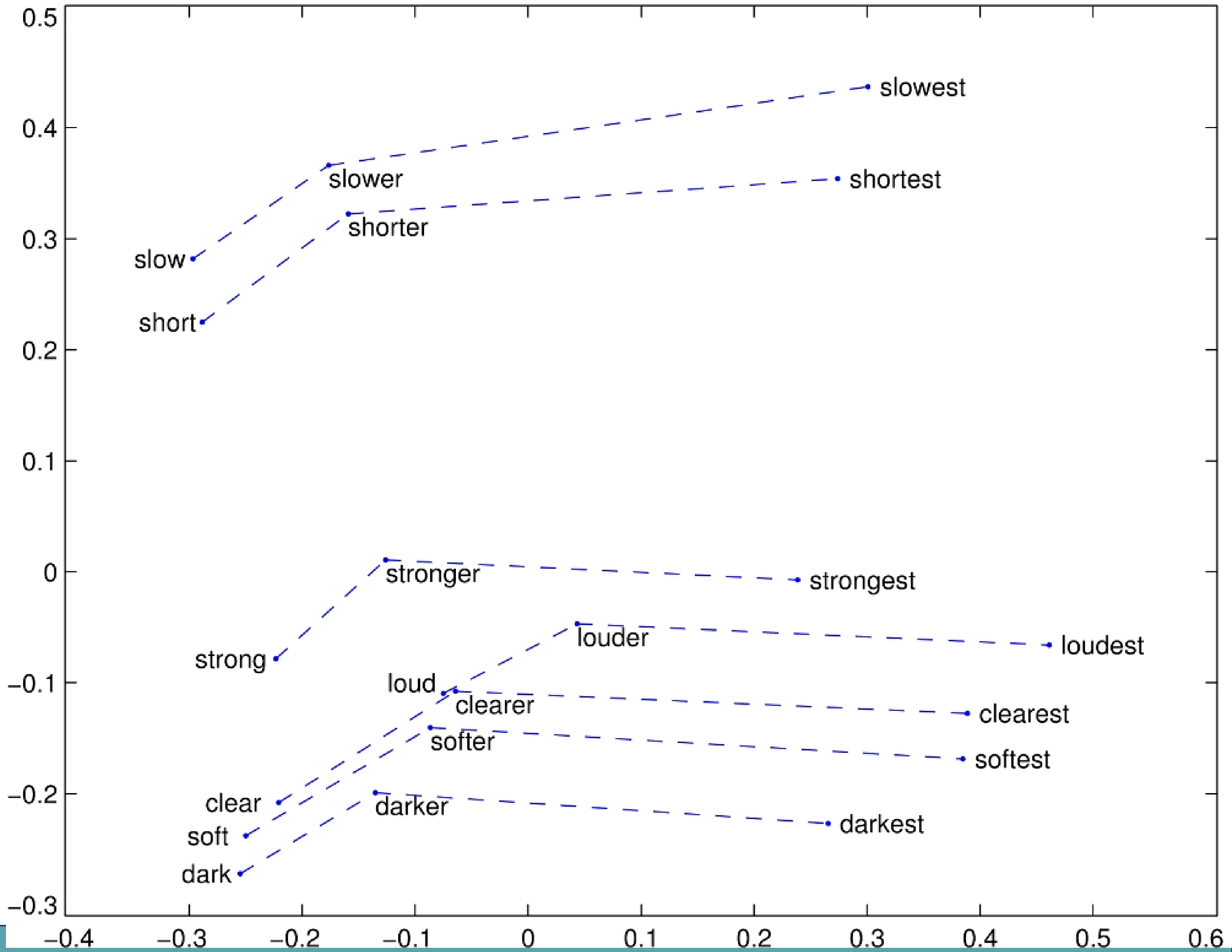
- Alternatively, we can require that the direction of the transformation be maintained.

$$\arg\max_{b^* \in V} \left(\cos\left(b^*, b - a + a^*\right)\right)$$

$$\arg\max_{b^* \in V} \left(\cos\left(b^* - b, a^* - a\right)\right)$$

- This basically means that $b_* - b$ **shares the same direction with** $a_* - a$, ignoring the distances


heiress
niece
aunt
sister
countess
duchess
empress
madam
heir
nephew
woman
uncle
earl
queen
duke
brother
emperor
sir
man
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
soft
darker
darkest
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

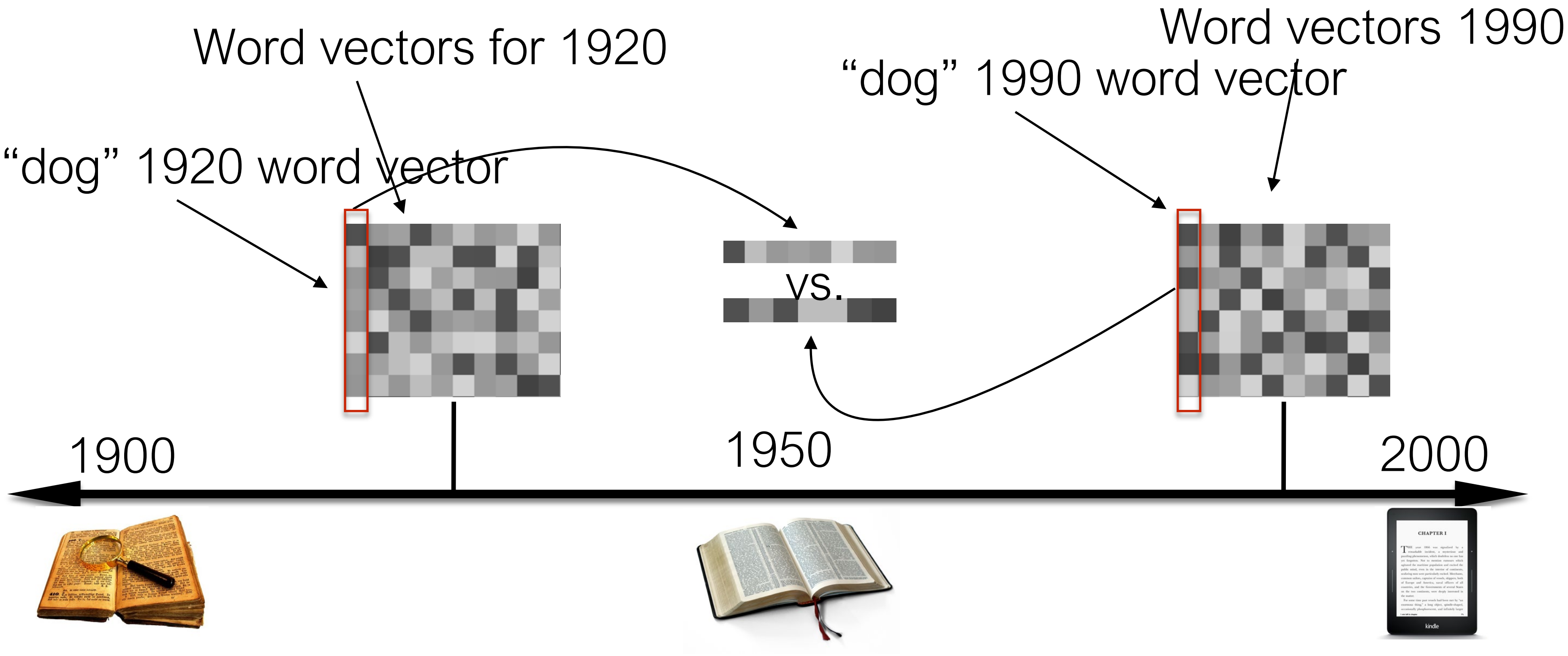
Embeddings can help study word history!

- Train embeddings on old books to study changes in word meaning!!

Dan Jurafsky

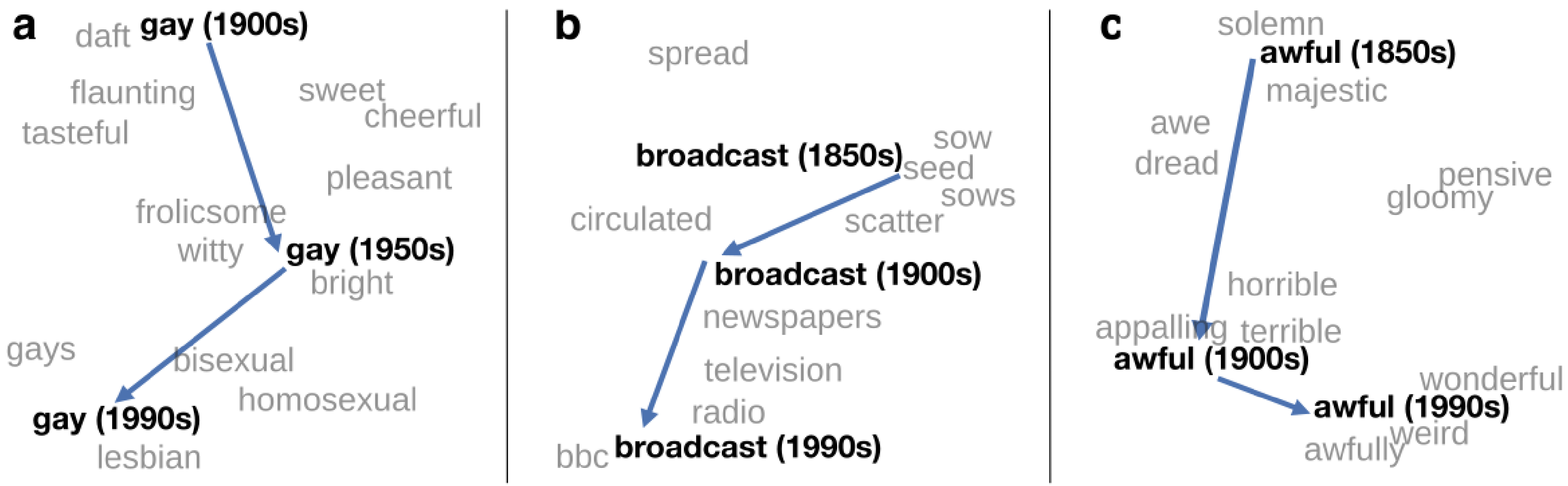
Will Hamilton

Diachronic word embeddings for studying language change!



Visualizing changes

Project 300 dimensions down into 2



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

gay | gā |

adjective (**gayer, gayest**)

1 (of a person) homosexual (used especially of a man): *that friend of yours, is he gay?*

- relating to or used by homosexuals: *a gay bar | the gay vote can decide an election.*

2 *dated* lighthearted and carefree: *Nan had a gay disposition and a very pretty face.*

- brightly colored; showy; brilliant: *a gay profusion of purple and pink sweet peas.*

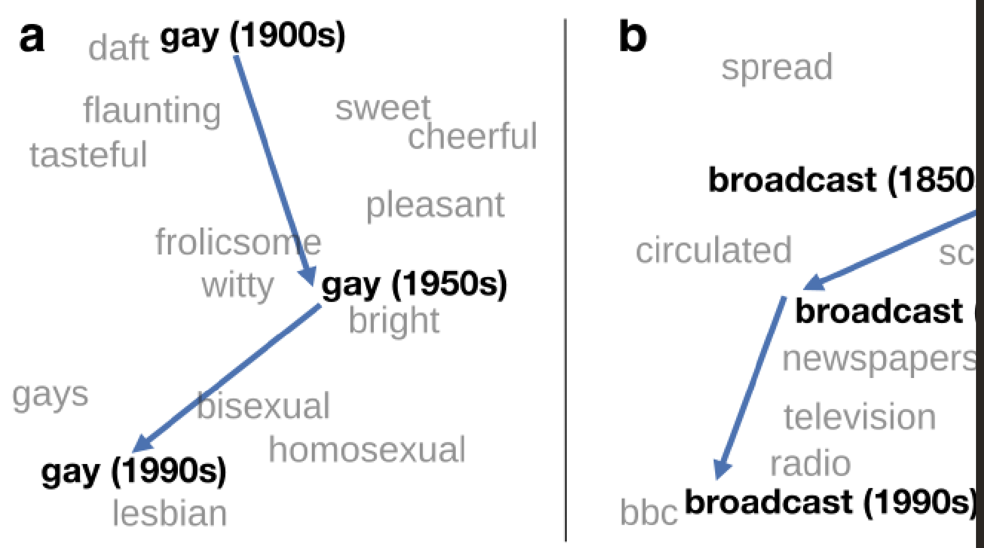
broadcast | 'brôdˌkast |

verb (past and past participle **broadcast**) *[with object]*

1 transmit (a program or some information) by radio or television: *the announcement was broadcast live* | (as noun **broadcasting**) : *the 1920s saw the dawn of broadcasting.*

- *[no object]* take part in a radio or television transmission: *the station broadcasts 24 hours a day.*
- tell (something) to many people; make widely known: *we don't want to broadcast our unhappiness to the world.*

2 scatter (seeds) by hand or machine rather than placing in drills or rows.



~30 million books, 1850-1990, Googl

awful | 'ôfəl |

adjective

1 very bad or unpleasant: *the place smelled awful | I look awful in a swimsuit | an awful speech.*

- extremely shocking; horrific: *awful, bloody images.*
- (of a person) very unwell, troubled, or unhappy: *I felt awful for being so angry with him | you look awful—you should go and lie down.*

2 *[attributive]* used to emphasize the extent of something, especially something unpleasant or negative: *I've made an awful fool of myself.*

3 *archaic* inspiring reverential wonder or fear.

Embeddings and bias

Embeddings reflect cultural bias

- Ask “Paris : France :: Tokyo : x”
  - x = Japan
- Ask “father : doctor :: mother : x”
  - x = nurse
- Ask “man : computer programmer :: woman : x”
  - x = homemaker

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.

Measuring cultural bias

- Implicit Association test (Greenwald et al 1998): How associated are
  - concepts (*flowers*, *insects*) & attributes (*pleasantness*, *unpleasantness*)?
  - Studied by measuring timing latencies for categorization.
- Psychological findings on US participants:
  - African-American names are associated with unpleasant words (more than European-American names)
  - Male names associated more with math, female names with arts
  - Old people's names with unpleasant words, young people with pleasant words.

Embeddings reflect cultural bias

- Caliskan et al. replication with embeddings:
  - African-American names (*Leroy, Shaniqua*) had a higher GloVe cosine with unpleasant words (*abuse, stink, ugly*)
  - European American names (*Brad, Greg, Courtney*) had a higher cosine with pleasant words (*love, peace, miracle*)
- **Embeddings reflect and replicate all sorts of pernicious biases.**

Aylin Caliskan, Joanna J. Bruson and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. Science 356:6334, 183-186.

Directions

- Debiasing algorithms for embeddings
- Use embeddings as a tool to study historical bias

Embeddings as a window onto history

- Use the Hamilton historical embeddings
- The cosine similarity of embeddings for decade X for occupations (like teacher) to male vs female names
  - Is correlated with the actual percentage of women teachers in decade X

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

History of biased framings of women

- Embeddings for competence adjectives are biased toward men
  - *Smart, wise, brilliant, intelligent, resourceful, thoughtful, logical, etc.*
- This bias is slowly decreasing

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

Princeton Trilogy experiments

- **Study 1: Katz and Braley (1933)**
- Investigated whether traditional social stereotypes had a cultural basis
- Ask 100 male students from Princeton University to choose five traits that characterized different ethnic groups (for example Americans, Jews, Japanese, Negroes) from a list of 84 word
- 84% of the students said that Negroes were superstitious and 79% said that Jews were shrewd. They were positive towards their own group.
- **Study 2: Gilbert (1951)**
  Less uniformity of agreement about unfavorable traits than in 1933.
-
  Study 3: Karlins et al. (1969)
- Many students objected to the task but this time there was greater agreement on the stereotypes assigned to the different groups compared with the 1951 study. Interpreted as a re-emergence of social stereotyping but in the direction more favorable stereotypical images.

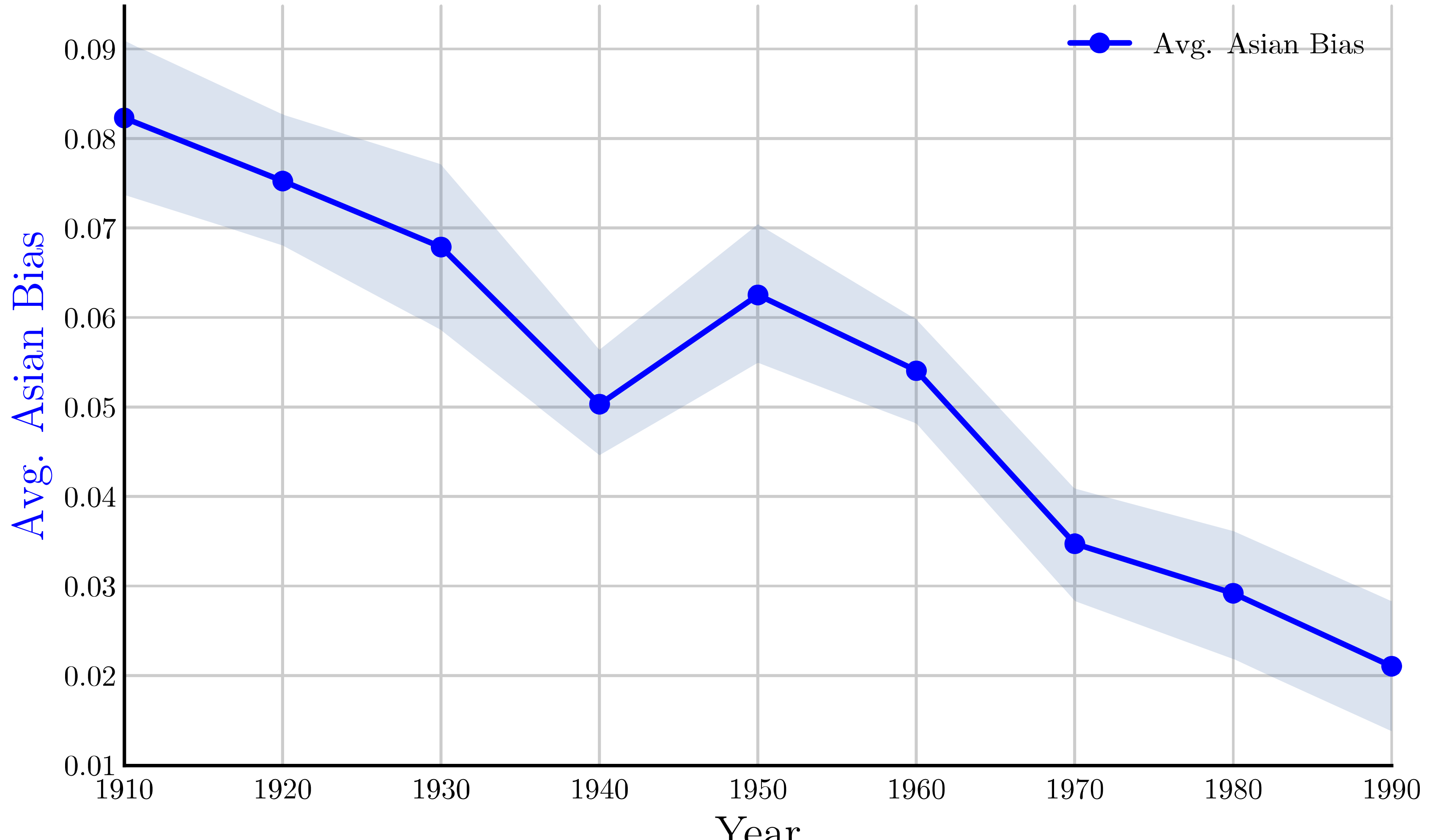
Embeddings reflect ethnic stereotypes over time

- Princeton trilogy experiments
- Attitudes toward ethnic groups (1933, 1951, 1969) scores for adjectives
  - *industrious, superstitious, nationalistic*, etc
- Cosine of Chinese name embeddings with those adjective embeddings correlates with human ratings.

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

Change in linguistic framing 1910-1990

Change in association of Chinese names with adjectives framed as "othering" (*barbaric*, *monstrous*, *bizarre*)



Changes in framing: adjectives associated with Chinese

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

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

Conclusion

- **Embeddings** = vector models of meaning
  - More fine-grained than just a string or index
  - Especially good at modeling similarity/analogy
    - Just download them and use cosines!!
  - Can use sparse models (tf-idf) or dense models (word2vec, GLoVE)

  - **Useful in practice but know they encode cultural stereotypes**