EECS 498-004: Introduction to Natural Language Processing

Instructor: Prof. Lu Wang

Computer Science and Engineering

University of Michigan

https://web.eecs.umich.edu/~wangluxy/

Time to discuss the progress!
Project progress report

- The project progress report is expected to cover the following aspects:
- 1- What changes you have made for the task compared to the proposal, including problem/task, models, datasets, or evaluation methods? If there is any change, please explain why.
- 2- Describe data preprocessing process. This includes data cleaning, selection, feature generation or other representation you have used, etc. Justify the decisions you've made, e.g. the reasons behind using a certain dataset or building a new dataset.
- 3- What methods or models you have tried towards the project goal? And why do you choose the methods (you can include related work on similar tasks or relevant tasks)? Justify the usage of a model or a certain method you choose.
- 4- What results you have achieved up to now based on your proposed evaluation methods? What worked and what didn't work? Try to explain why something didn't work as expected.
- 5- How can you improve your models? What are the next steps?

Grading: For 2-5, each aspect will take 25 points.

Length: 2 page (or more if necessary). Single space if MS word is used. Or you can choose latex templates, e.g. https://www.acm.org/publications/proceedings-template.

- Feel free to reach out to instructors for discussions!

Sparse versus dense vectors

- PPMI vectors are
  - **long** (length |V|= 20,000 to 50,000)
  - **sparse** (most elements are zero)
- Alternative: learn vectors which are
  - **short** (length 200-1000)
  - **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 (less 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 represented as distinct dimensions; this fails to capture similarity between a word with *car* as a neighbor and a word with *automobile* as a neighbor

Outline

- Neural language models with skip-grams (Word2vec)
  - Task, training algorithm, training data construction, training objective
- Properties of embeddings
- Embeddings and bias

Neural language models

- Skip-grams
- Continuous Bag of Words (CBOW)

Neural language models

- Skip-grams
- Continuous Bag of Words (CBOW)

Prediction-based models to get dense vectors

- **Skip-gram** (Mikolov et al. 2013a), **CBOW** (Mikolov et al. 2013b)
- **Idea**: Learn embeddings as part of the process of word prediction
- **Implementation**: Train a neural network to predict neighboring words
- Advantages:
  - Fast, easy to train
  - Available online in the `word2vec` package
  - Including sets of pretrained embeddings!

Word2Vec: **Skip-Gram** Task

- Given a sentence:

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

- 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 weights (will be discussed later) as the word embeddings

Brilliant insight: Use running text as implicitly supervised training data!

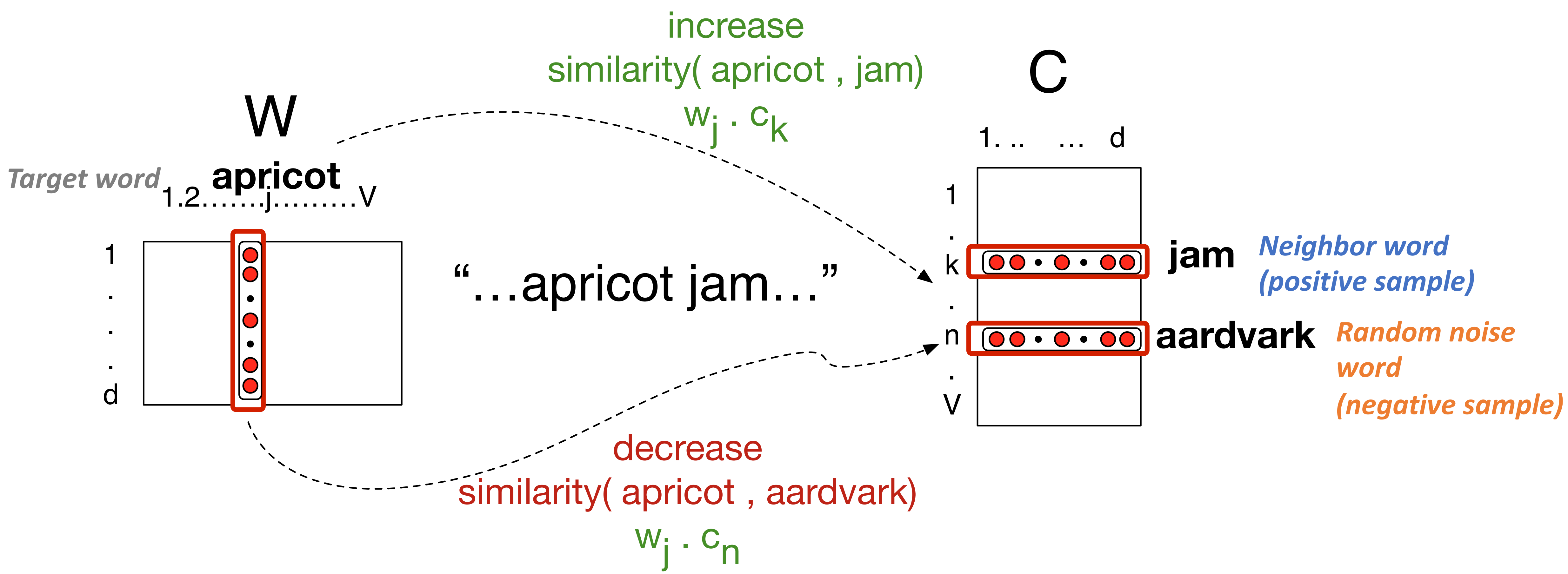
- A word near *apricot*
  - Acts as gold 'correct answer' to the question
  - "Is word *w* likely to show up near *apricot*?"
- No need for hand-labeled supervision
  - The idea comes from **neural language modeling**
    - Bengio et al. (2003)
    - Collobert et al. (2011)

Word2Vec: **Skip-Gram** Task

- Now we have positive samples.
- Where do the "negative samples" come from?

Skip-gram algorithm

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



increase
similarity(apricot , jam)
$w_j \cdot c_k$
W
C
Target word
apricot
1.2…….j………V
1
.
.
.
d
1. ..   … d
1
.
k
.
n
.
V
jam
Neighbor word
(positive sample)
aardvark
Random noise
word
(negative sample)
"…apricot jam…"
decrease
similarity(apricot , aardvark)
$w_j \cdot c_n$

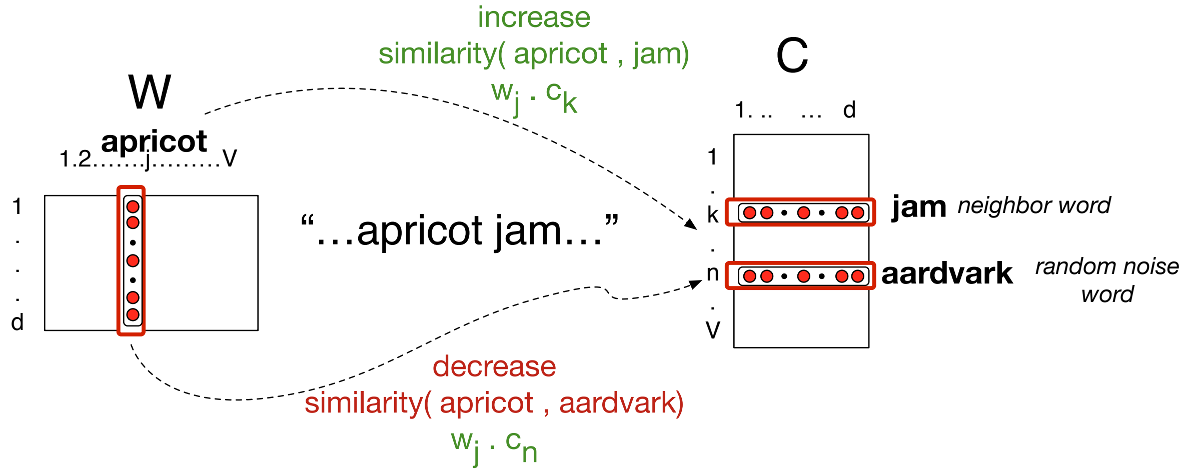
Skip-gram Training Data

- Training sentence:

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

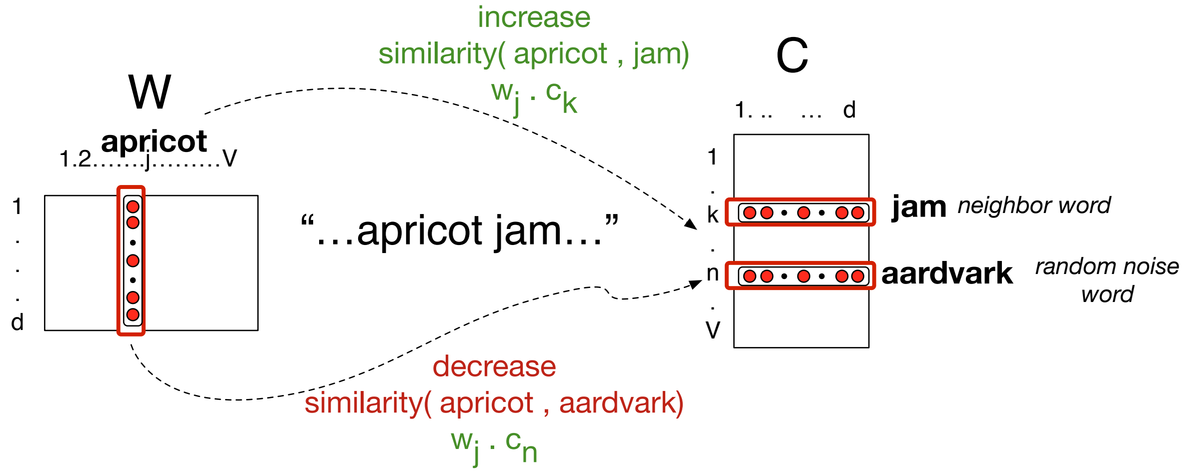
c1 c2 target c3 c4

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 (or not):
  - P(+|t,c)-> positive
  - $P(-|t,c) = 1 - P(+|t,c)$ -> negative



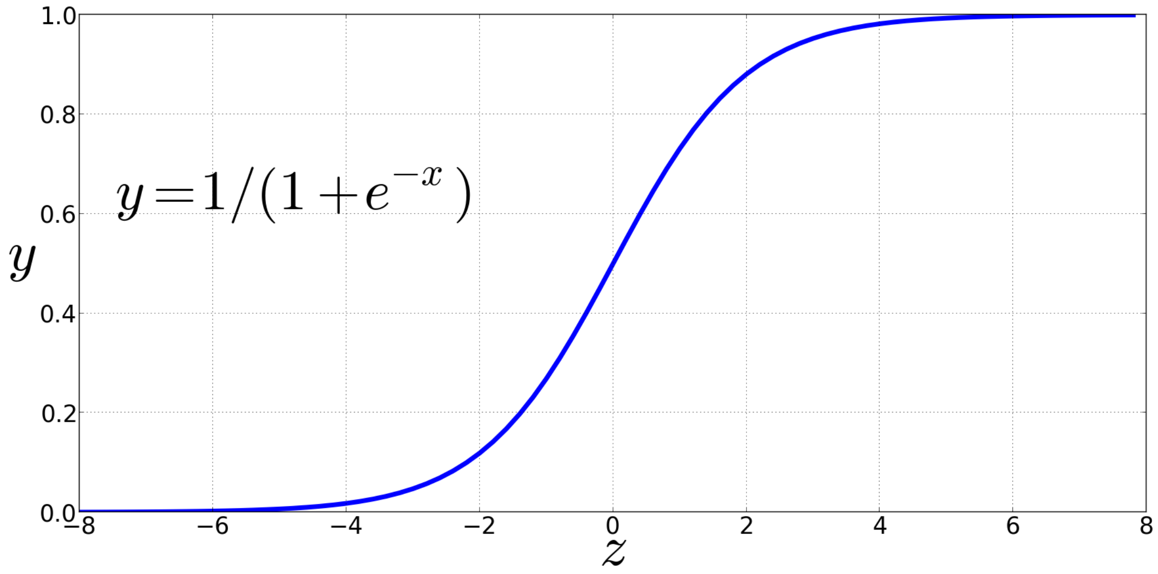
How to compute p(+|t,c)?

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

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

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

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

For all the context words:

- Assume all context words are independent

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

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

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 Data

- 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

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

Skip-gram Training Data

- 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.
- Any random word that isn't *t*

Skip-gram Training Data

- Training sentence:

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

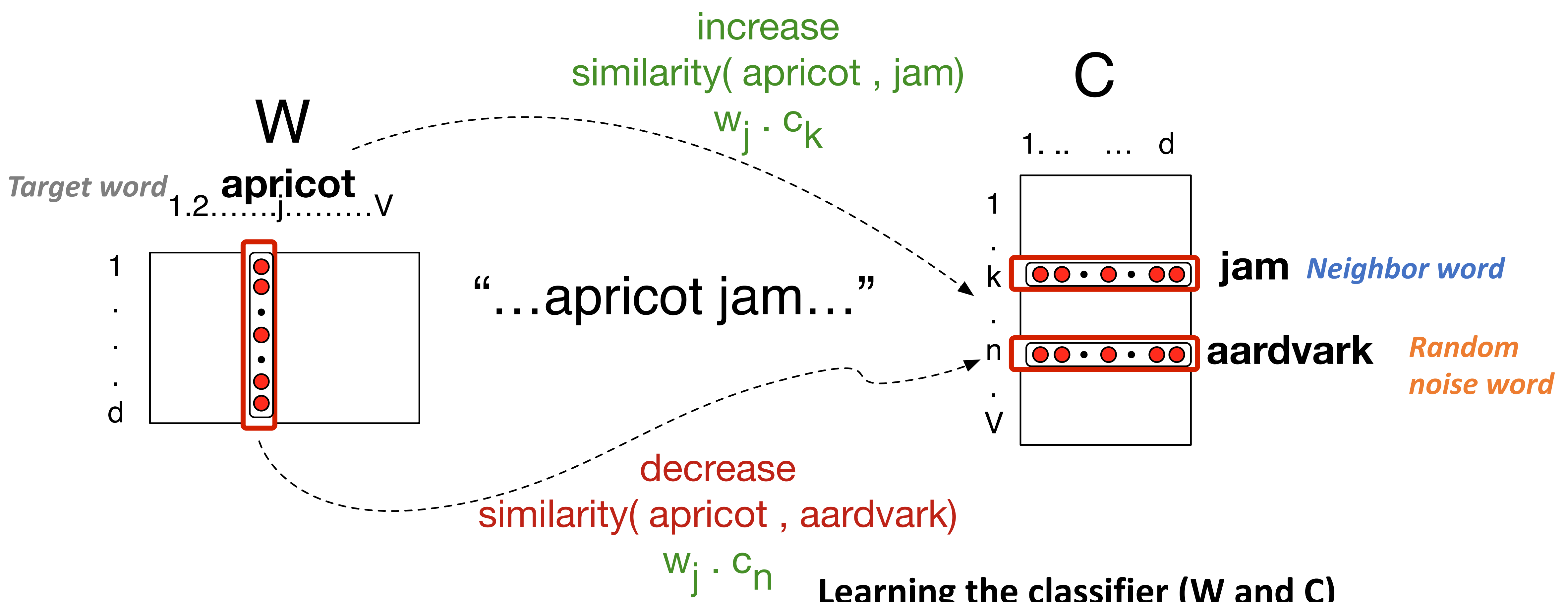
c1 c2 t c3 c4

positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	preserves
apricot	or

negative examples - *k=2*

t	c	t	c
apricot	aardvark	apricot	twelve
apricot	puddle	apricot	hello
apricot	where	apricot	dear
apricot	coaxial	apricot	forever



Learning the classifier (W and C)

Iterative process on training data. Then adjust the word weights to make the positive pairs more likely and the negative pairs less likely.

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

Formally

- We want to maximize the following objective

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

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

Focusing on one target word t:

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

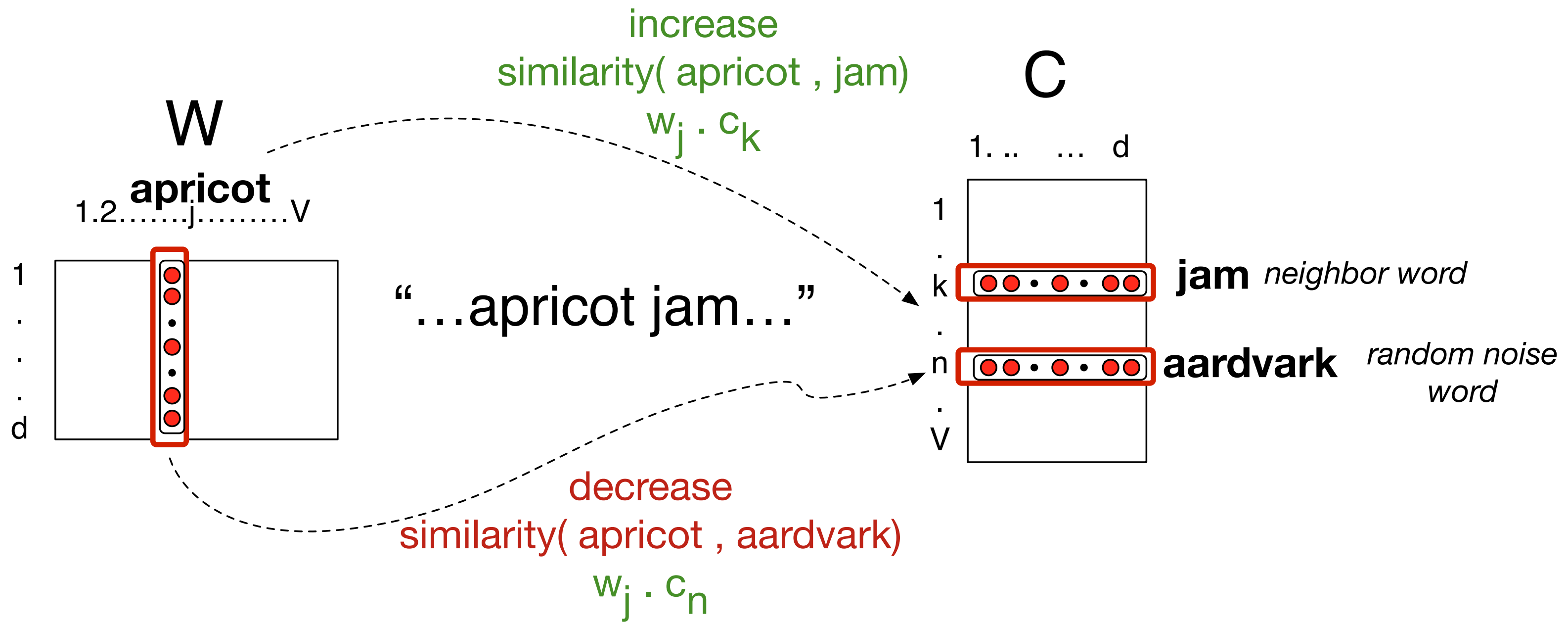
Focusing on one target word t:

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

Logistic regression

Train using gradient descent

- **Idea**: gradually changing W and C
- Finally learns two separate embedding matrices W and C
- Can use W and throw away C, or merge them



Summary: How to learn 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**.

(Dense) Word embeddings you can download!

- **Word2vec**
  https://code.google.com/archive/p/word2vec/
- **Fasttext**

  http://www.fasttext.cc/
- **Glove**
  http://nlp.stanford.edu/projects/glove/

Evaluating embeddings

- Compare to human scores on word similarity-type tasks:
  - WordSim-353 (Finkelstein et al., 2002)
  - Stanford Contextual Word Similarity (SCWS) dataset (Huang et al., 2012)
- TOEFL dataset:
  - *Levied is closest in meaning to:*
    - ***imposed****, believed, requested, correlated*

Outline

- Neural language models with skip-grams (Word2vec)
  - Task, training algorithm, training data construction, training objective
- Properties of embeddings
- Embeddings and bias

Properties of embeddings

- Nearest words to some embeddings (Mikolov et al. 2013)

target:	Redmond	Havel	ninjutsu	graffiti	capitulate
	Redmond Wash.	Vaclav Havel	ninja	spray paint	capitulation
	Redmond Washington	president Vaclav Havel	martial arts	grafitti	capitulated
	Microsoft	Velvet Revolution	swordsmanship	taggers	capitulating

Properties of embeddings

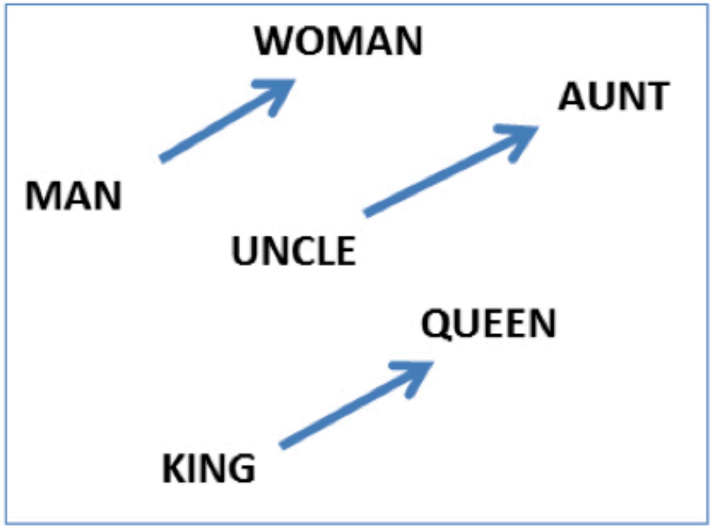
Similarity depends on window size C

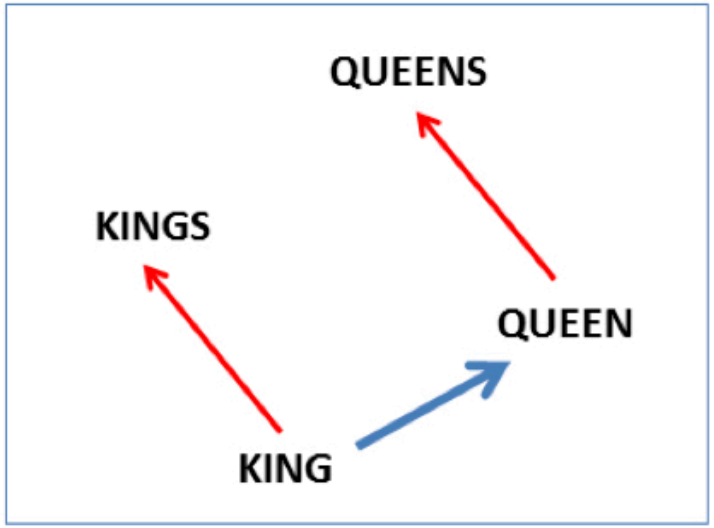
- C = ±2 The nearest words to *Hogwarts:*
  - *Sunnydale*
  - *Evernight*
- C = ±5 The nearest words to *Hogwarts:*
  - *Dumbledore*
  - *Malfoy*
  - *halfblood*

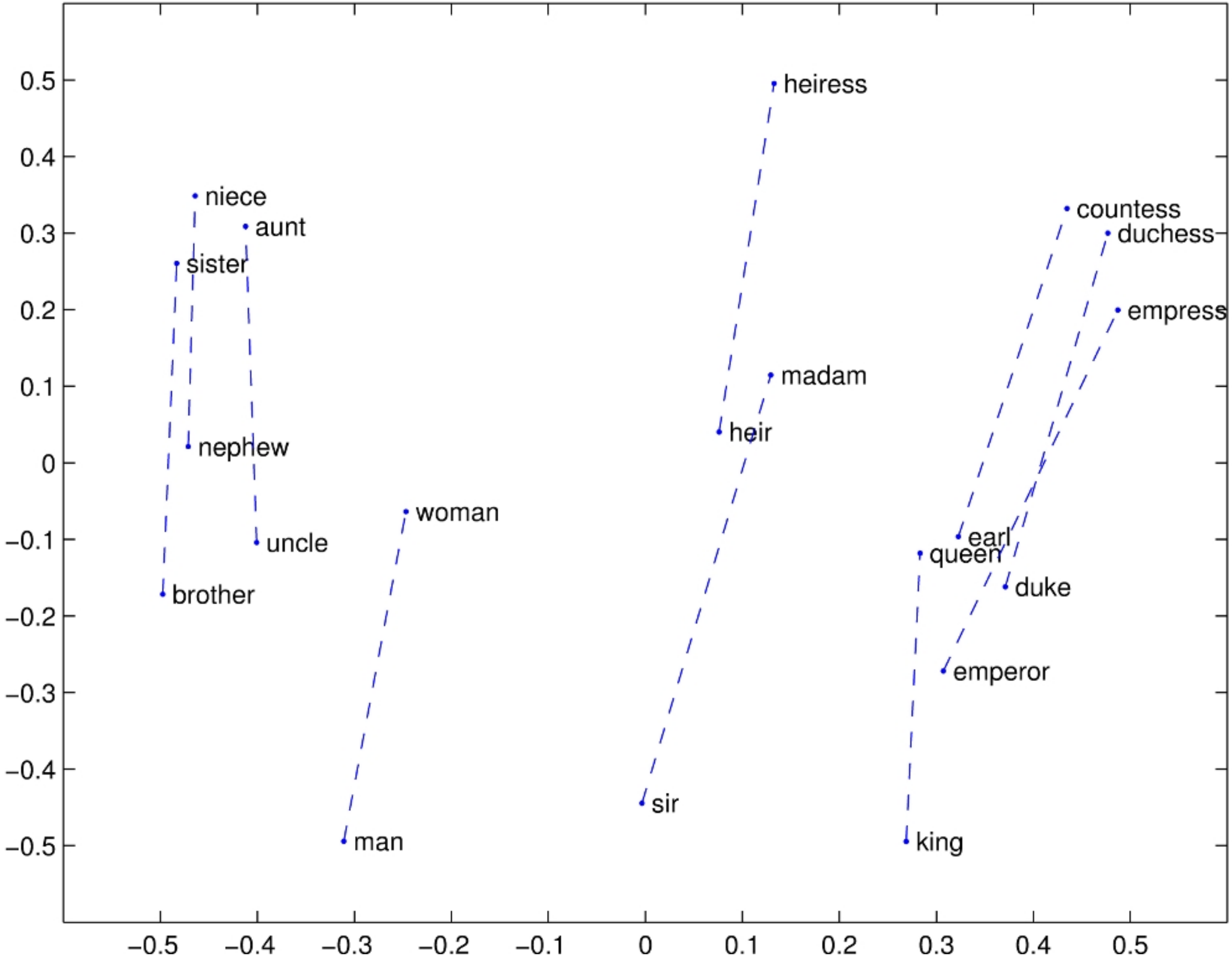
Analogy: Embeddings capture relational meaning!

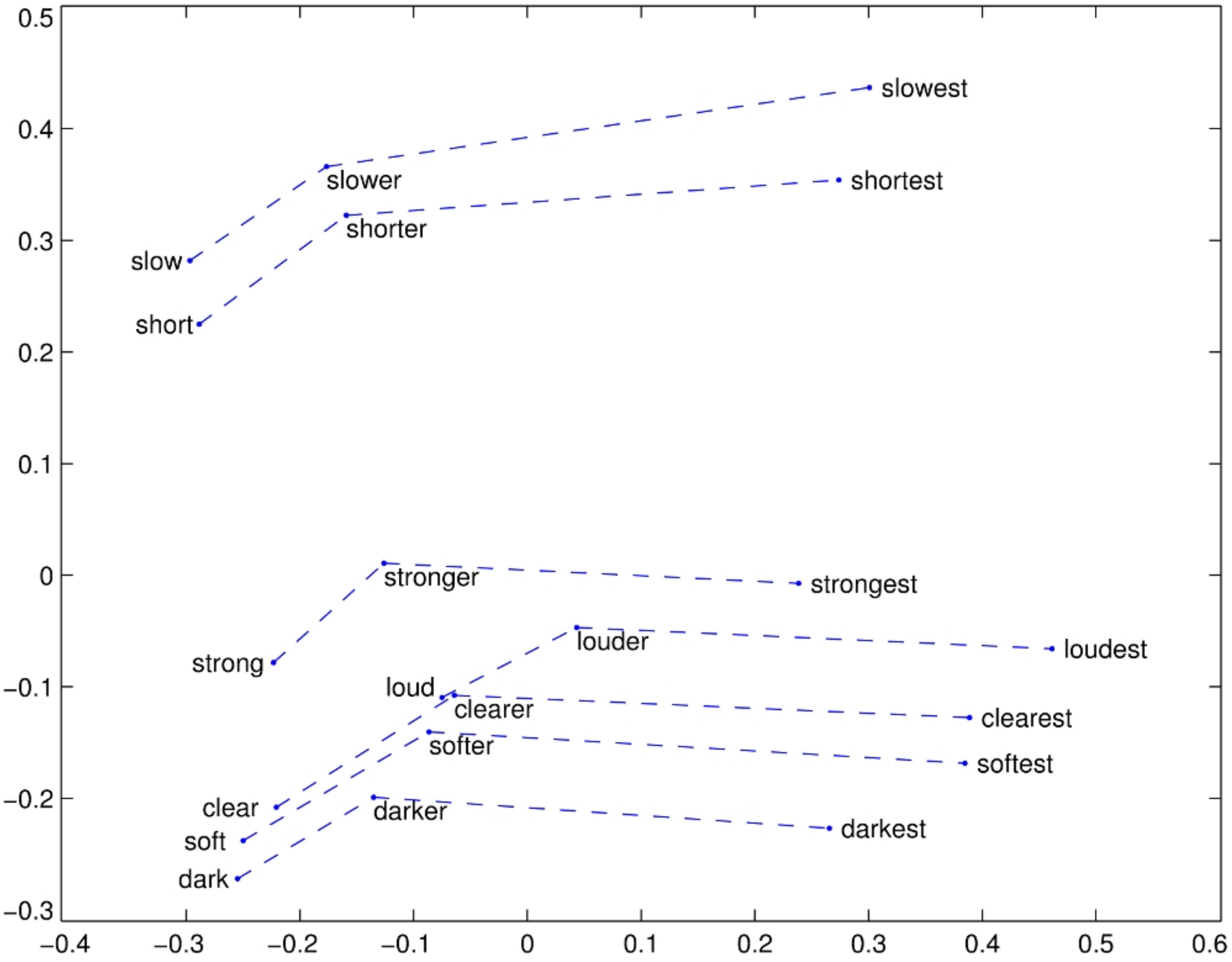
vector(*'king'*) - vector(*'man'*) + vector(*'woman'*) $\approx$ vector('queen')

vector(*'Paris'*) - vector(*'France'*) + vector(*'Italy'*) $\approx$ vector('Rome')






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

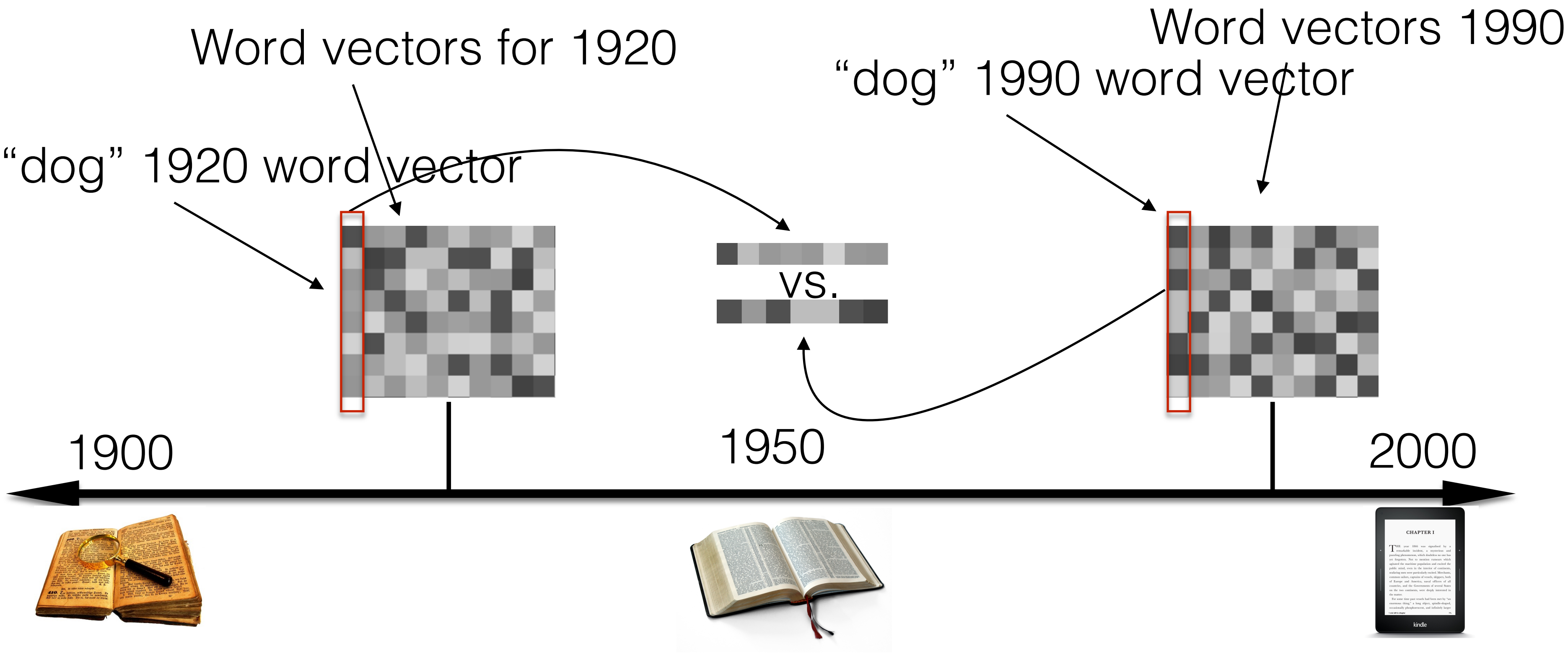


slowest
slower
shortest
shorter
slow
short
stronger
strongest
louder
loudest
strong
loud
clearer
clearest
softer
softest
clear
darker
darkest
soft
dark
0.5
0.4
0.3
0.2
0.1
0
−0.1
−0.2
−0.3
−0.4
−0.3
−0.2
−0.1
0
0.1
0.2
0.3
0.4
0.5
0.6

Embeddings can help study word history!

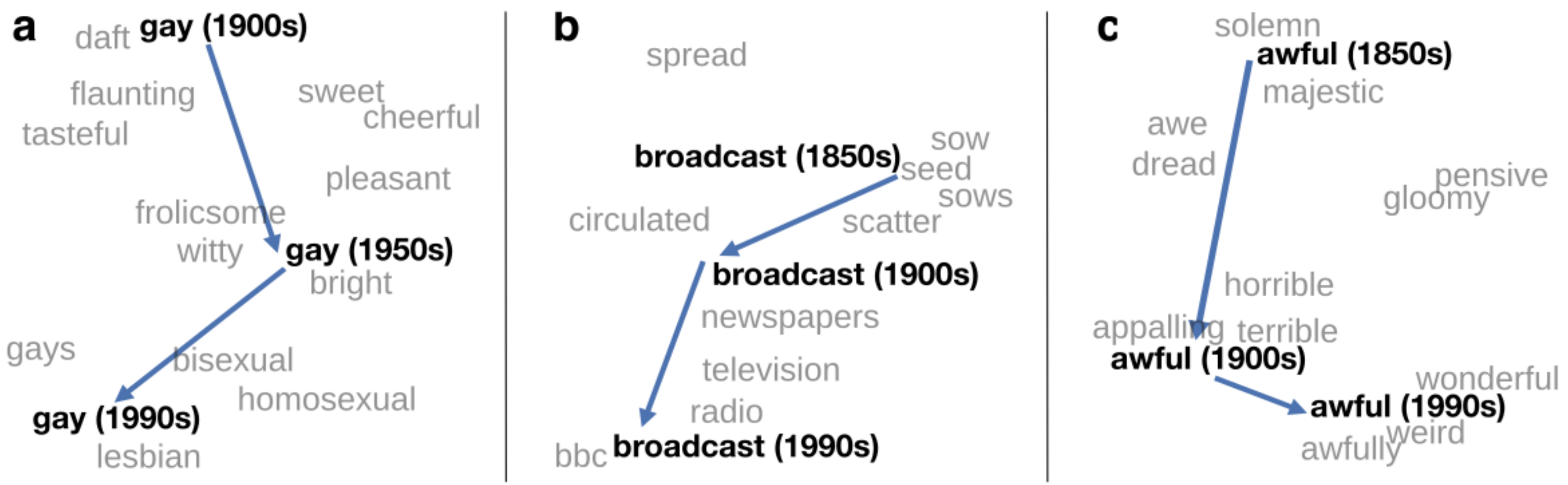
- Train embeddings on old books to study changes in word meaning!!

Diachronic word embeddings for studying language change!



Visualizing changes

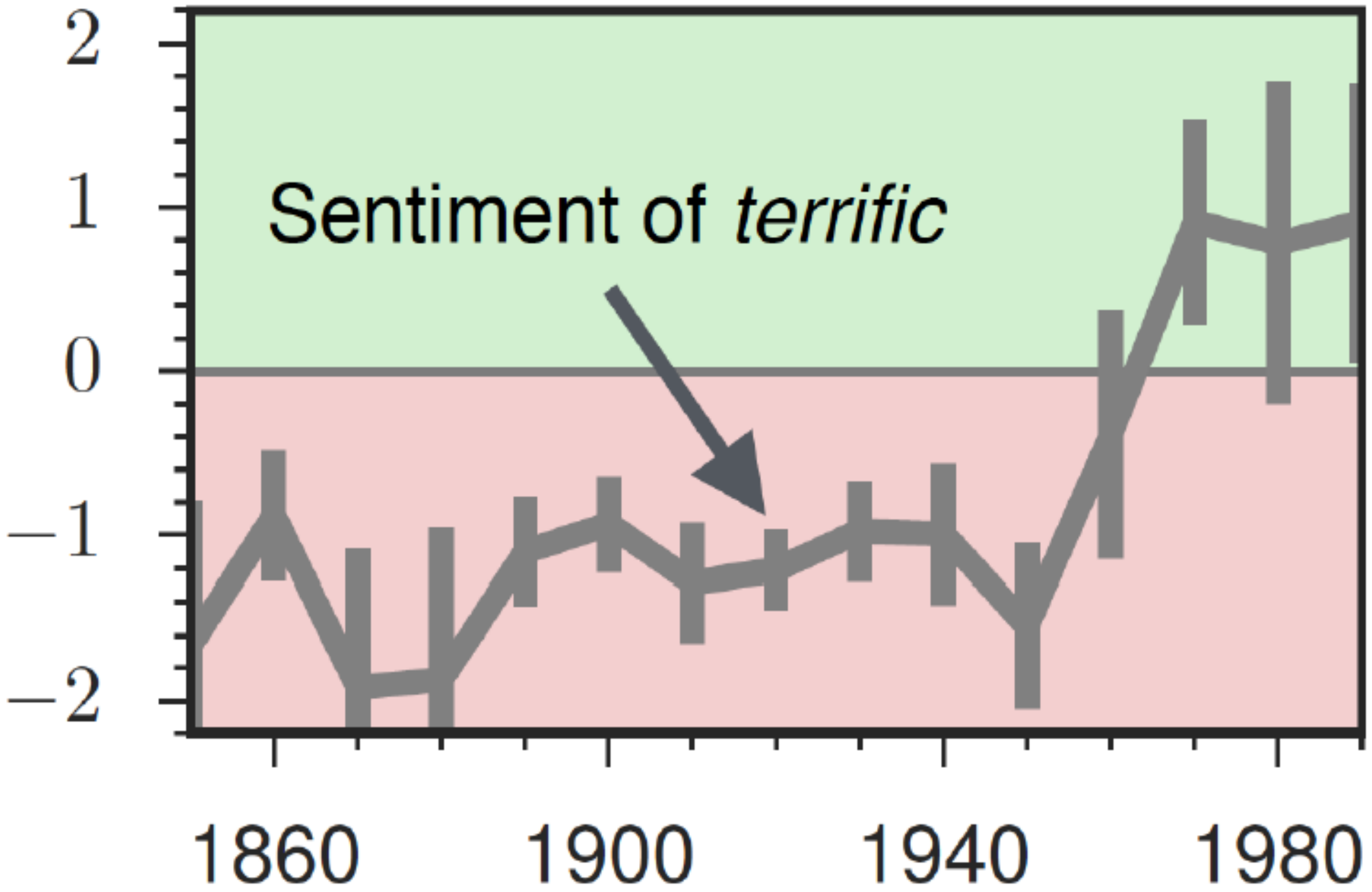
Project 300 dimensions down into 2



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

The evolution of sentiment words

Negative words change faster than positive words



Outline

- Neural language models with skip-grams (Word2vec)
  - Task, training algorithm, training data construction, training objective
- Properties of embeddings
- Embeddings and bias

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

Embeddings reflect cultural bias

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

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

- 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.
- Caliskan et al. replication with embeddings:
  - African-American names (*Leroy, Shaniqua*) had a higher GloVe (another word embeddings learning method) cosine similarity 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.

Embeddings as a window onto history

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

- The cosine similarity of embeddings for decade X for occupations or adjectives (e.g. teacher or smart) to male vs female names
  - Find its correlation with the actual percentage of women teachers in decade X

History of biased framings of women

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

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

Embeddings reflect ethnic stereotypes over time

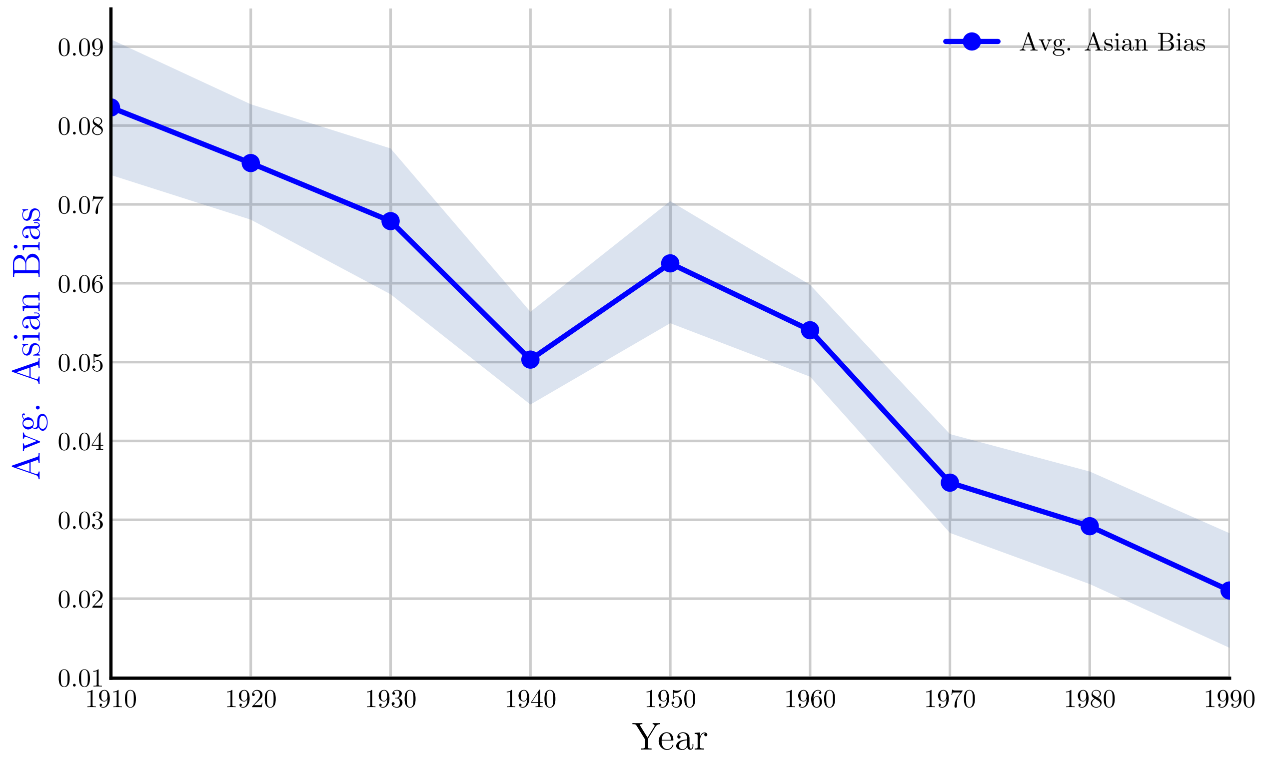
Garg, Nikhil, Schiebinger, Londa, Jurafsky, Dan, and Zou, James (2018). Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, *115*(16), E3635–E3644

- 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.

Change in linguistic framing 1910-1990

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

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



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

Directions

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