CSEP 517
Natural Language Processing

Word Embeddings

Luke Zettlemoyer

(Slides adapted from Danqi Chen, Greg Durrett, Chris Manning, Dan Jurafsky)

How to represent words?

N-gram language models

It is 76 F and ____.

$$P(w \mid \text{it is 76 F and})$$

[0.0001, 0.1, 0, 0, 0.002, ..., 0.3, ..., 0]

red sunny

Text classification

$$P(y = 1 \mid x) = \sigma(\theta^\intercal w + b)$$

I like this movie. 👍

I don't like this movie. 👎

$w^{(1)}$ [0, 1, 0, 0, 0, ..., 1, ..., 1]

$w^{(2)}$ [0, 1, 0, 1, 0, ..., 1, ..., 1]

don't

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 1 0 0 0 0]

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

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

Challenge: How to compute similarity of two 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!

…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

"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

"words that occur in similar contexts tend to have similar meanings"

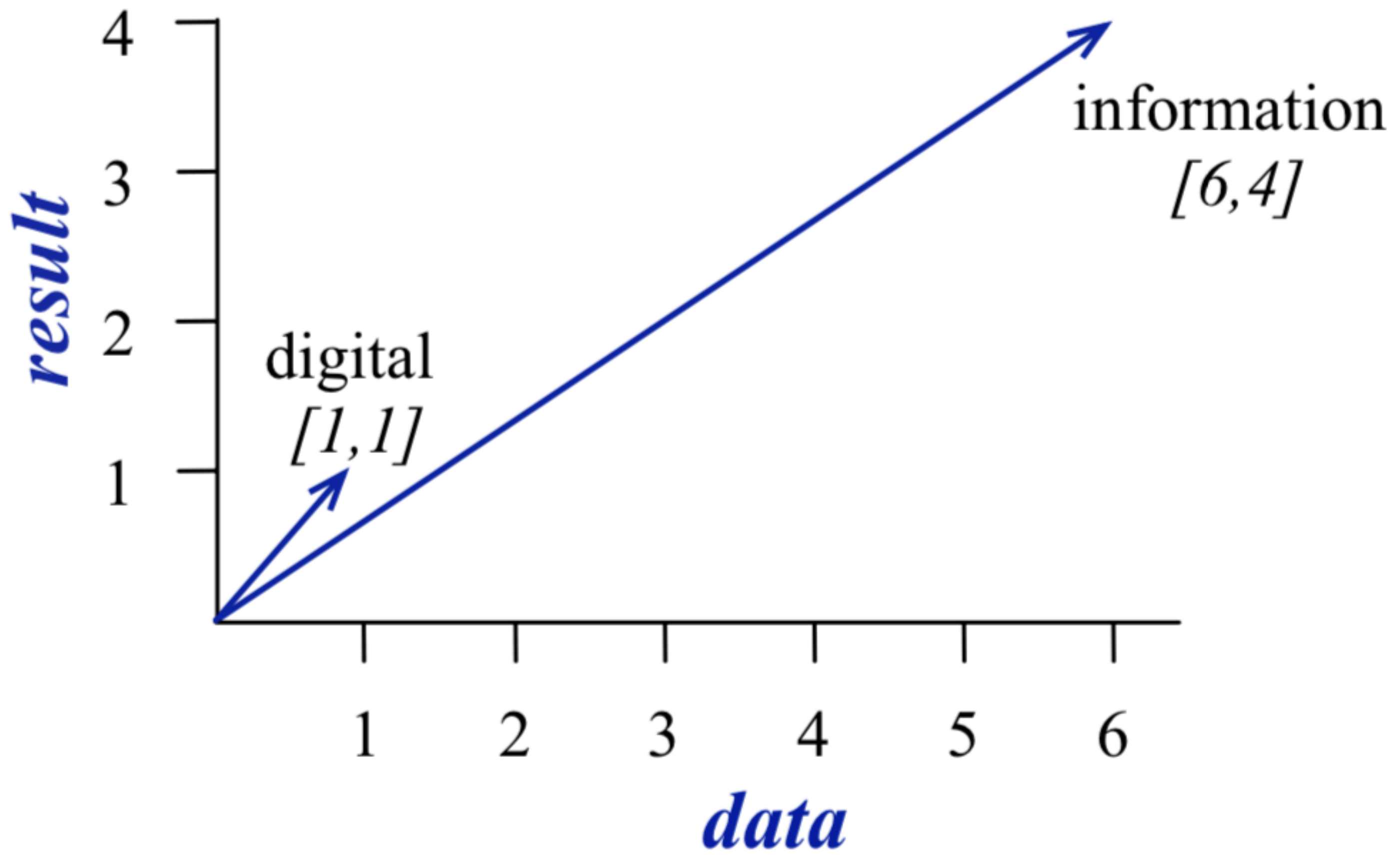
Words as vectors

- 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 context vectors to represent the meaning of words!
  - word-word co-occurrence matrix:

	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	

Words as vectors



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

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

What is the range of cos($\cdot$)?

Words as vectors

Problem: not all counts are equal, words can randomly co-occur

- Solution: re-weight by how likely it is for the two words to co-occur by simple chance
- PPMI = Positive Pointwise Mutual Information

$$\text{PPMI}(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P(c)}, 0)$$

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

	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

Sparse vs 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 focus of this lecture
  - The basis of all the modern NLP systems

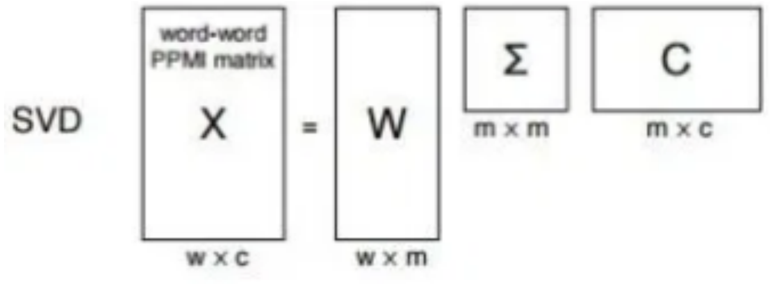
Dense vectors

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

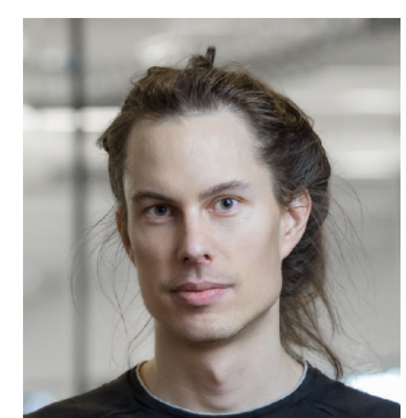
Why dense vectors?

- Short vectors are easier to use as features in ML systems
- Dense vectors may generalize better than storing explicit counts
- They do better at capturing synonymy
  - w_1 co-occurs with “car”, w_2 co-occurs with “automobile”

- Different methods for getting dense vectors:
  - Singular value decomposition (SVD)
  - word2vec and friends: “learn” the vectors!



Word2vec and friends



(Mikolov et al, 2013): Distributed Representations of Words and Phrases and their Compositionality

Word2vec

- 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
    - Twitter: 27B
    - Common Crawl: 840B
- Output: $f : V \rightarrow \mathbb{R}^d$

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

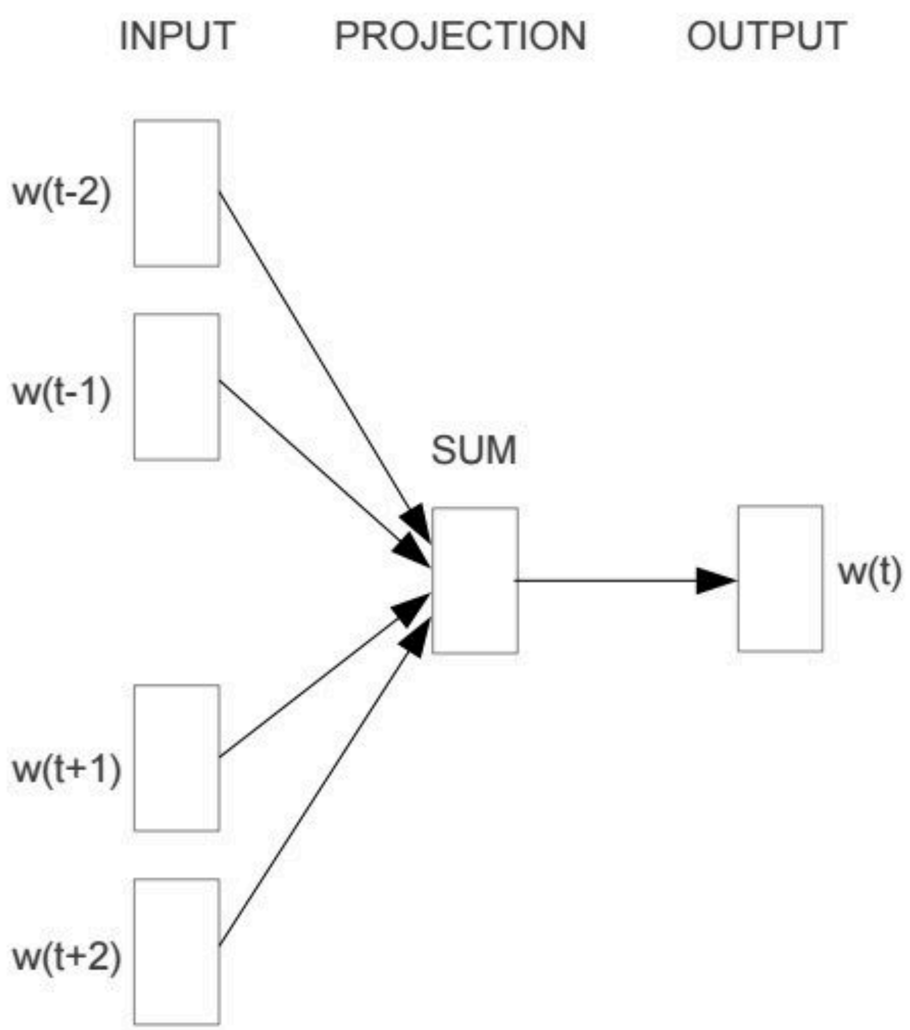
$$v_{\text{the}} = \begin{pmatrix} 0.234 \\ 0.266 \\ 0.239 \\ -0.199 \end{pmatrix} \qquad v_{\text{language}} = \begin{pmatrix} 0.290 \\ -0.441 \\ 0.762 \\ 0.982 \end{pmatrix}$$

Word2vec

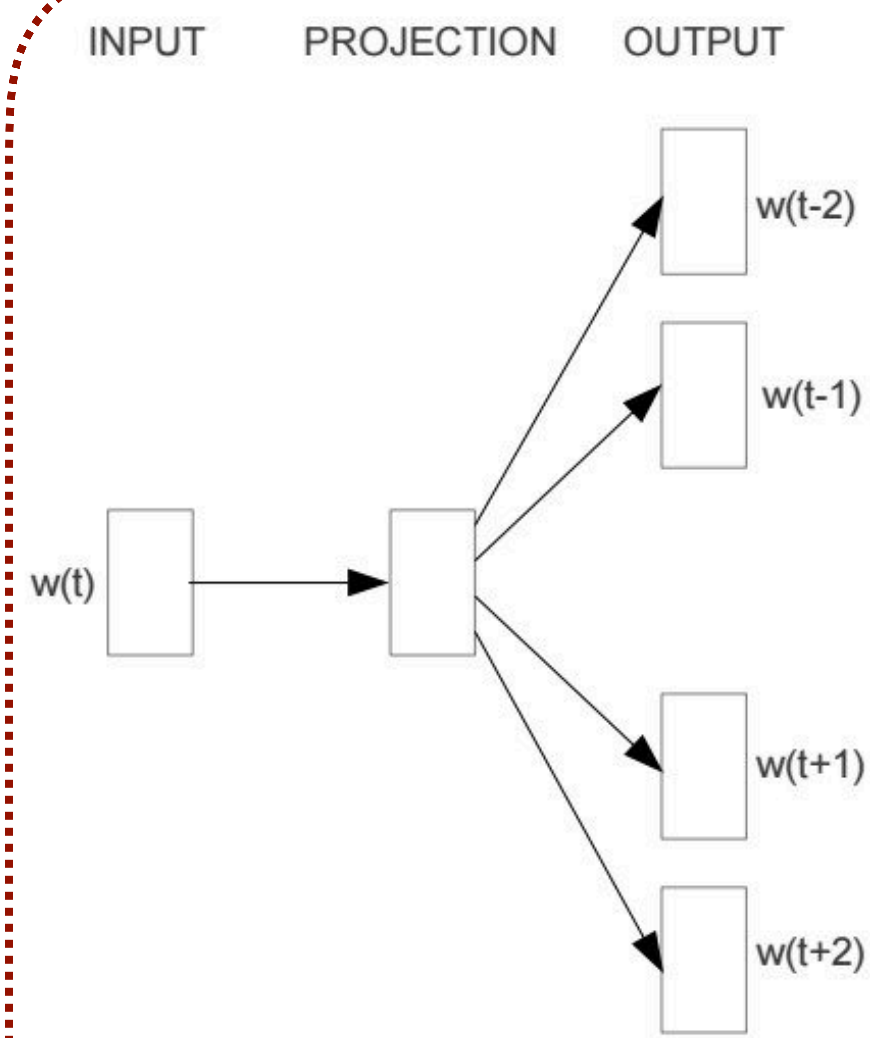
word = “sweden”

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

Word2vec



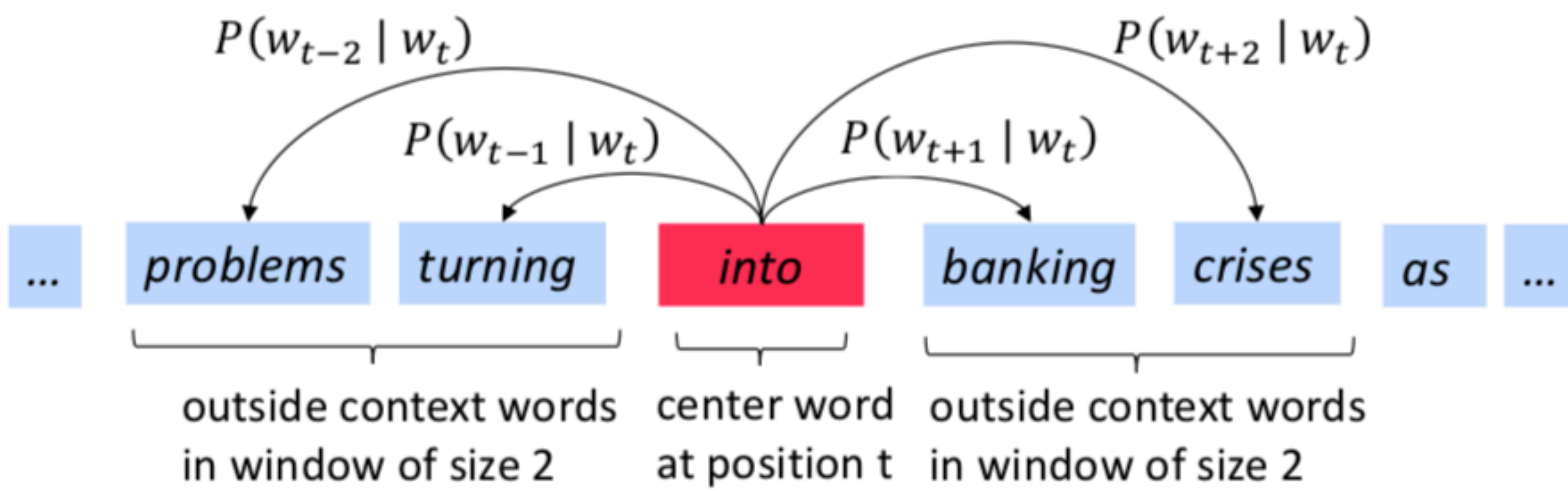
Continuous Bag of Words (CBOW)



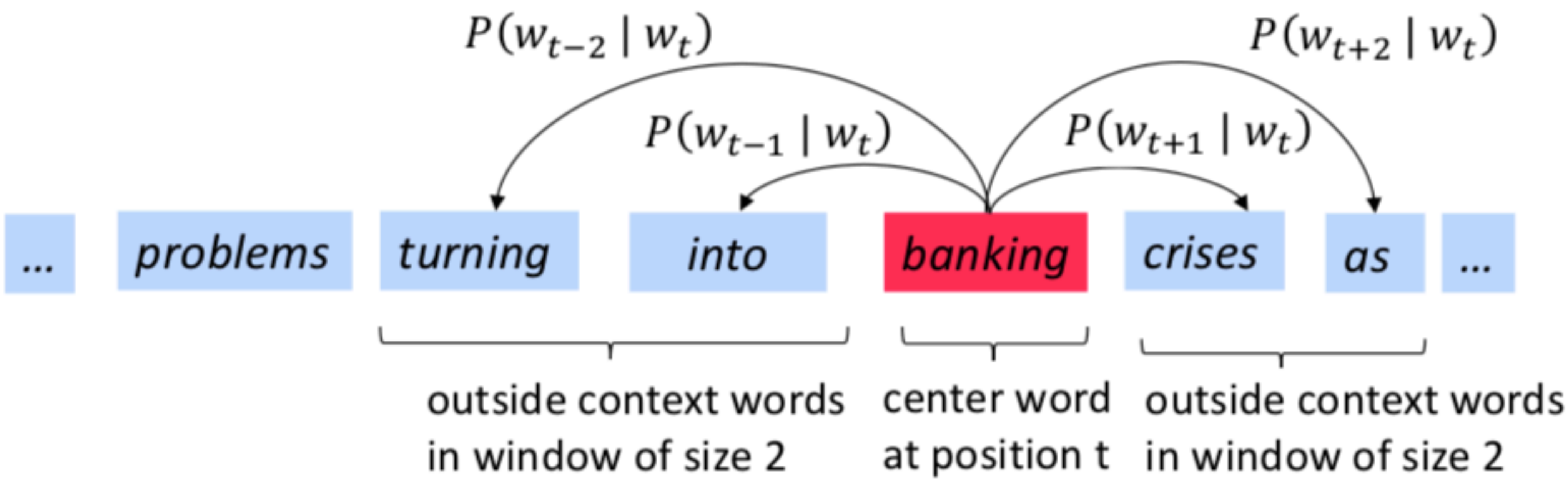
Skip-grams

Skip-gram

- The idea: we want to use words to **predict** their context words
- Context: a fixed window of size $2m$



Skip-gram


$P(w_{t-2} \mid w_t)$
$P(w_{t-1} \mid w_t)$
$P(w_{t+1} \mid w_t)$
$P(w_{t+2} \mid w_t)$
...
problems
turning
into
banking
crises
as
...
outside context words
in window of size 2
center word
at position t
outside context words
in window of size 2

Skip-gram: objective function

- For each position $t = 1,2,\ldots T$, predict context words within context size m, given center word w_j:

all the parameters to be optimized

$$\mathcal{L}(\theta) = \prod_{t=1}^{T} \prod_{-m \leq j \leq m, j \neq 0} P(w_{t+j} \mid w_t; \theta)$$

- The objective function $J(\theta)$ is the (average) negative log likelihood:

$$J(\theta) = -\frac{1}{T} \log \mathcal{L}(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{-m \leq j \leq m, j \neq 0} \log P(w_{t+j} \mid w_t; \theta)$$

How to define $P(w_{t+j} \mid w_t; \theta)$?

- We have two sets of vectors for each word in the vocabulary

$\mathbf{u}_i \in \mathbb{R}^d$: embedding for target word *i*

$\mathbf{v}_{i'} \in \mathbb{R}^d$: embedding for context word *i'*

- Use inner product $\mathbf{u}_i \cdot \mathbf{v}_{i'}$ to measure how likely word *i* appears with context word *i'*, the larger the better

"softmax" we learned last time!

$$P(w_{t+j} \mid w_t) = \frac{\exp(\mathbf{u}_{w_t} \cdot \mathbf{v}_{w_{t+j}})}{\sum_{k \in V} \exp(\mathbf{u}_{w_t} \cdot \mathbf{v}_k)}$$

$\theta = \{\{\mathbf{u}_k\}, \{\mathbf{v}_k\}\}$ are all the parameters in this model!

Q: Why two sets of vectors?

Any issues?

How to train the model

Calculating all the gradients together!

$$\theta = \{\{\mathbf{u}_k\}, \{\mathbf{v}_k\}\}$$

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0}\log P(w_{t+j} \mid w_t; \theta) \qquad \nabla_\theta J(\theta) = ?$$

Q: How many parameters are in total?

We can apply stochastic gradient descent (SGD)!

$$\theta^{(t+1)} = \theta^{(t)} - \eta\nabla_\theta J(\theta)$$

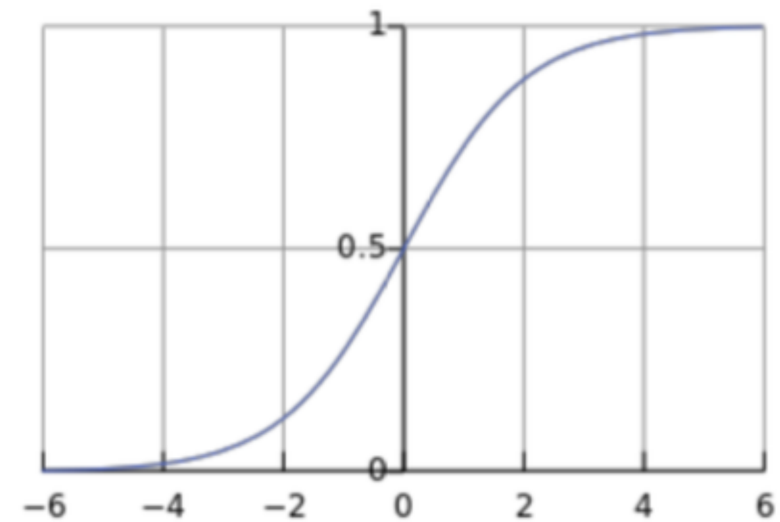
Skip-gram with negative sampling (SGNS)

Idea: recast problem as binary classification!

- Target word is positive example
- All words not in context are negative

$$P(D = 1 \mid t, c) = \sigma(\mathbf{u}_t \cdot \mathbf{v}_c)$$

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



positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

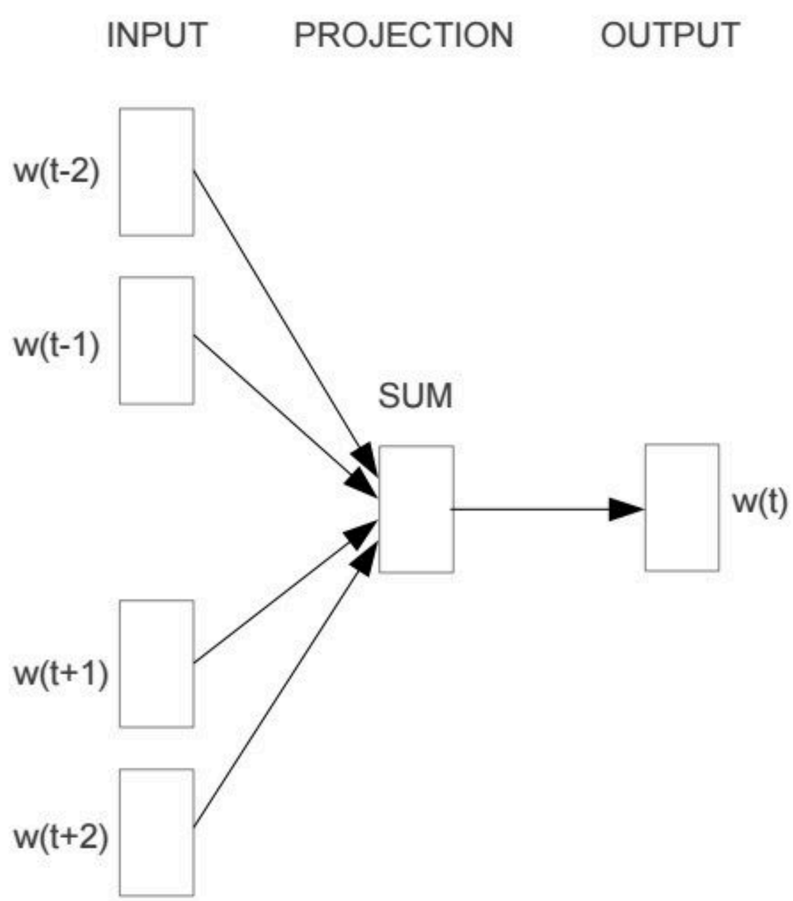
negative examples -

t	c	t	c
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

To compute loss, pick K random words as negative examples:

$$J(\theta) = -P(D = 1 \mid t, c) - \frac{1}{K} \sum_{i=1}^{K} P(D = 0 \mid t_i, c)$$

Continuous Bag of Words (CBOW)



$$L(\theta) = \prod_{t=1}^{T} P\left(w_t \mid \{w_{t+j}\}, -m \leq j \leq m, j \neq 0\right)$$

$$\bar{\mathbf{v}}_t = \frac{1}{2m} \sum_{-m \leq j \leq m, j \neq 0} \mathbf{v}_{t+j}$$

$$P(w_t \mid \{w_{t+j}\}) = \frac{\exp(\mathbf{u}_{w_t} \cdot \bar{\mathbf{v}}_t)}{\sum_{k \in V} \exp(\mathbf{u}_k \cdot \bar{\mathbf{v}}_t)}$$

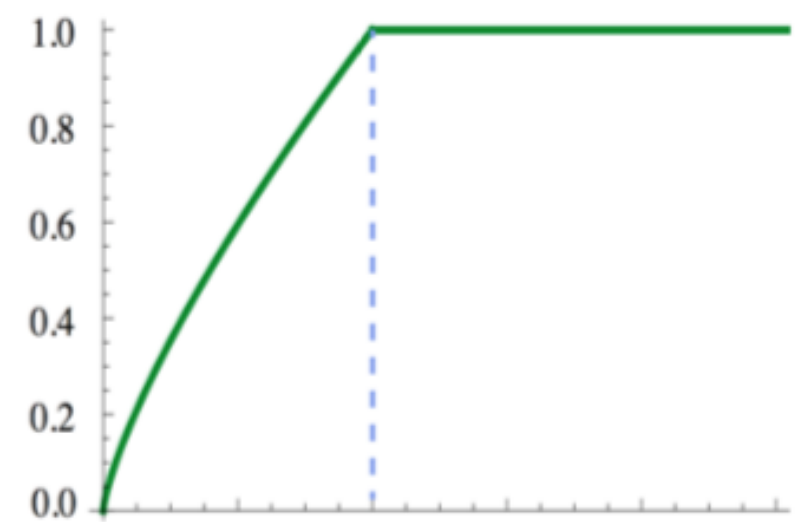
GloVe: **Glo**bal **Ve**ctors

- Let's take the global co-occurrence statistics: $X_{i,j}$

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

- Training faster
- Scalable to very large corpora

$f \sim$



(Pennington et al, 2014): GloVe: Global Vectors for Word Representation

GloVe: **Glo**bal **Ve**ctors

Nearest words to frog:

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

litoria

leptodactylidae

rana

eleutherodactylus

(Pennington et al, 2014): GloVe: Global Vectors for Word Representation

FastText: Sub-Word Embeddings

- Similar as Skip-gram, but break words into n-grams with n = 3 to 6

where:

3-grams: <wh, whe, her, ere, re>

4-grams: <whe, wher, here, ere>

5-grams: <wher, where, here>

6-grams: <where, where>

- Replace $\mathbf{u}_i \cdot \mathbf{v}_j$ by $\sum_{g \in n\text{-grams}(w_i)} \mathbf{u}_g \cdot \mathbf{v}_j$

- More to come! Contextualized word embeddings

(Bojanowski et al, 2017): Enriching Word Vectors with Subword Information

Trained word embeddings available

- word2vec: https://code.google.com/archive/p/word2vec/
- GloVe: https://nlp.stanford.edu/projects/glove/
- FastText: https://fasttext.cc/

Download pre-trained word vectors

- Pre-trained word vectors. This data is made available under the Public Domain Dedication and License v1.0 whose full text can be found at: http://www.opendatacommons.org/licenses/pddl/1.0/.
  - Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 50d, 100d, 200d, & 300d vectors, 822 MB download): glove.6B.zip
  - Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors, 1.75 GB download): glove.42B.300d.zip
  - Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors, 2.03 GB download): glove.840B.300d.zip
  - Twitter (2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 100d, & 200d vectors, 1.42 GB download): glove.twitter.27B.zip
- Ruby script for preprocessing Twitter data

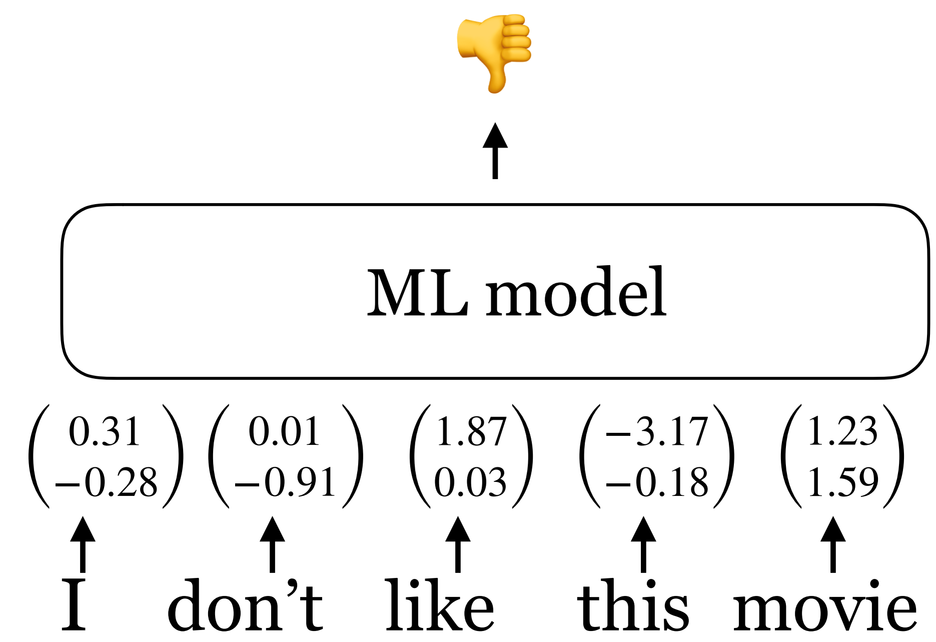
Differ in algorithms, text corpora, dimensions, cased/uncased...

Evaluating Word Embeddings

Extrinsic vs intrinsic evaluation

Extrinsic evaluation

- Let’s plug these word embeddings into a real NLP system and see whether this improves performance
- Could take a long time but still the most important evaluation metric



Intrinsic evaluation

- Evaluate on a specific/intermediate subtask
- Fast to compute
- Not clear if it really helps the downstream task

Intrinsic evaluation

Word similarity

Example dataset: wordsim-353
353 pairs of words with human judgement
http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/

Word 1	Word 2	Human (mean)
tiger	cat	7.35
tiger	tiger	10
book	paper	7.46
computer	internet	7.58
plane	car	5.77
professor	doctor	6.62
stock	phone	1.62
stock	CD	1.31
stock	jaguar	0.92

Cosine similarity:

$$\cos(\boldsymbol{u}_i, \boldsymbol{u}_j) = \frac{\boldsymbol{u}_i \cdot \boldsymbol{u}_j}{||\boldsymbol{u}_i||_2 \times ||\boldsymbol{u}_j||_2}.$$

Metric: Spearman rank correlation

Intrinsic evaluation

Word Similarity

Model	Size	WS353	MC	RG	SCWS	RW
SVD	6B	35.3	35.1	42.5	38.3	25.6
SVD-S	6B	56.5	71.5	71.0	53.6	34.7
SVD-L	6B	65.7	<u>72.7</u>	75.1	56.5	37.0
CBOW†	6B	57.2	65.6	68.2	57.0	32.5
SG†	6B	62.8	65.2	69.7	<u>58.1</u>	37.2
GloVe	6B	<u>65.8</u>	<u>72.7</u>	<u>77.8</u>	53.9	<u>38.1</u>
SVD-L	42B	74.0	76.4	74.1	58.3	39.9
GloVe	42B	<u>**75.9**</u>	<u>**83.6**</u>	<u>**82.9**</u>	<u>**59.6**</u>	<u>**47.8**</u>
CBOW*	100B	68.4	79.6	75.4	59.4	45.5

Intrinsic evaluation

Word analogy

man: woman ≈ king: ?

$$\arg\max_i \left(\cos(\mathbf{u}_i, \mathbf{u}_b - \mathbf{u}_a + \mathbf{u}_c)\right)$$

semantic

Chicago:Illinois≈Philadelphia: ?

syntactic

bad:worst ≈ cool: ?

More examples at

http://download.tensorflow.org/data/questions-words.txt

What can go wrong with word embeddings?

- What’s wrong with learning a word’s “meaning” from its usage?
- What data are we learning from?
- What are we going to learn from this data?

What do we mean by bias?

- Identify *she - he* axis in word vector space, project words onto this axis

Extreme *she* occupations

1. homemaker	2. nurse	3. receptionist
4. librarian	5. socialite	6. hairdresser
7. nanny	8. bookkeeper	9. stylist
10. housekeeper	11. interior designer	12. guidance counselor

Extreme *he* occupations

1. maestro	2. skipper	3. protege
4. philosopher	5. captain	6. architect
7. financier	8. warrior	9. broadcaster
10. magician	11. figher pilot	12. boss

Bolukbasi et al. (2016)

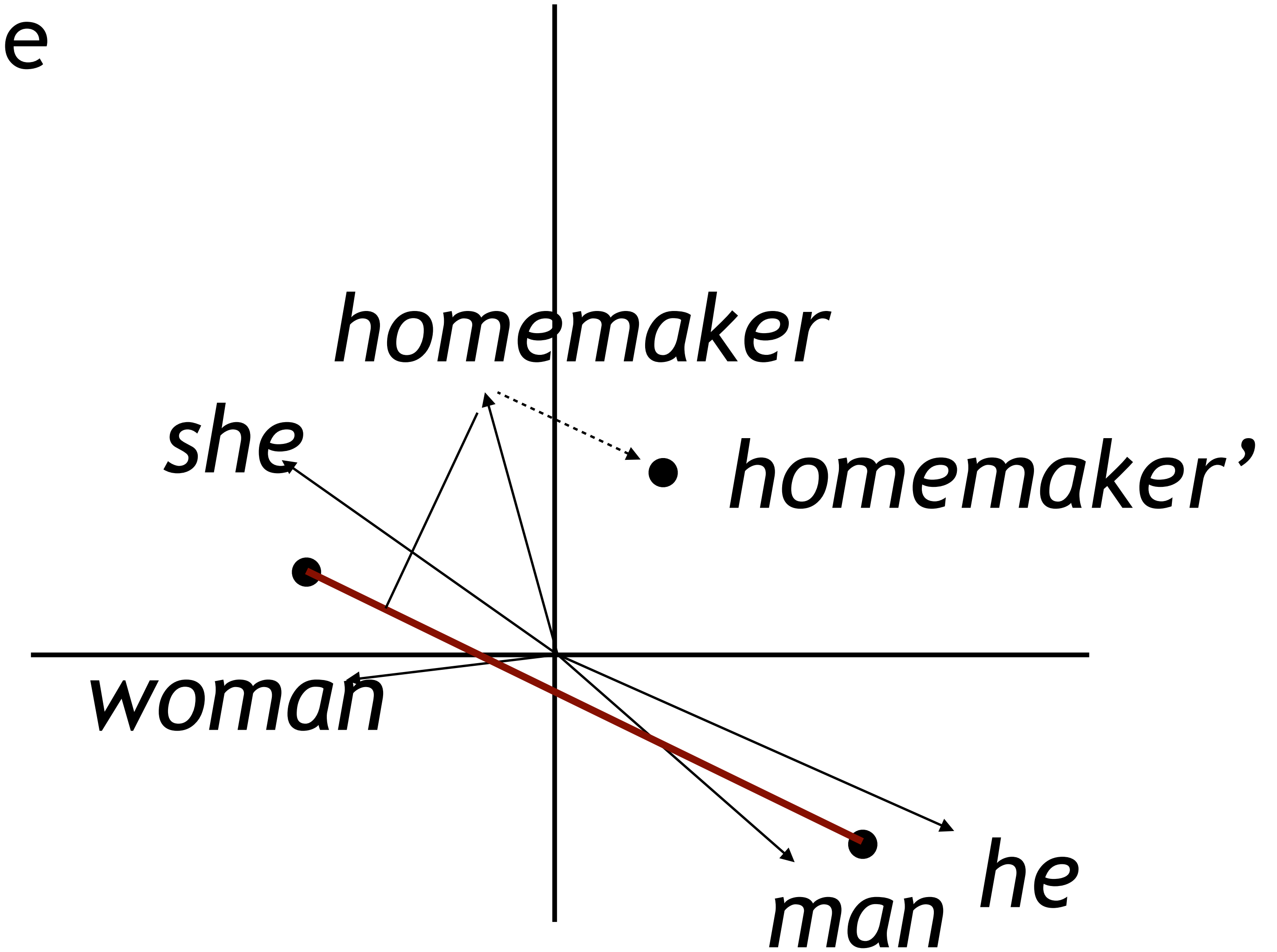
- Nearest neighbor of (b - a + c)

Racial Analogies	
black → homeless	caucasian → servicemen
caucasian → hillbilly	asian → suburban
asian → laborer	black → landowner
Religious Analogies	
jew → greedy	muslim → powerless
christian → familial	muslim → warzone
muslim → uneducated	christian → intellectually

Manzini et al. (2019)

Debiasing

- Identify gender subspace with gendered words
- Project words onto this subspace
- Subtract those projections from the original word

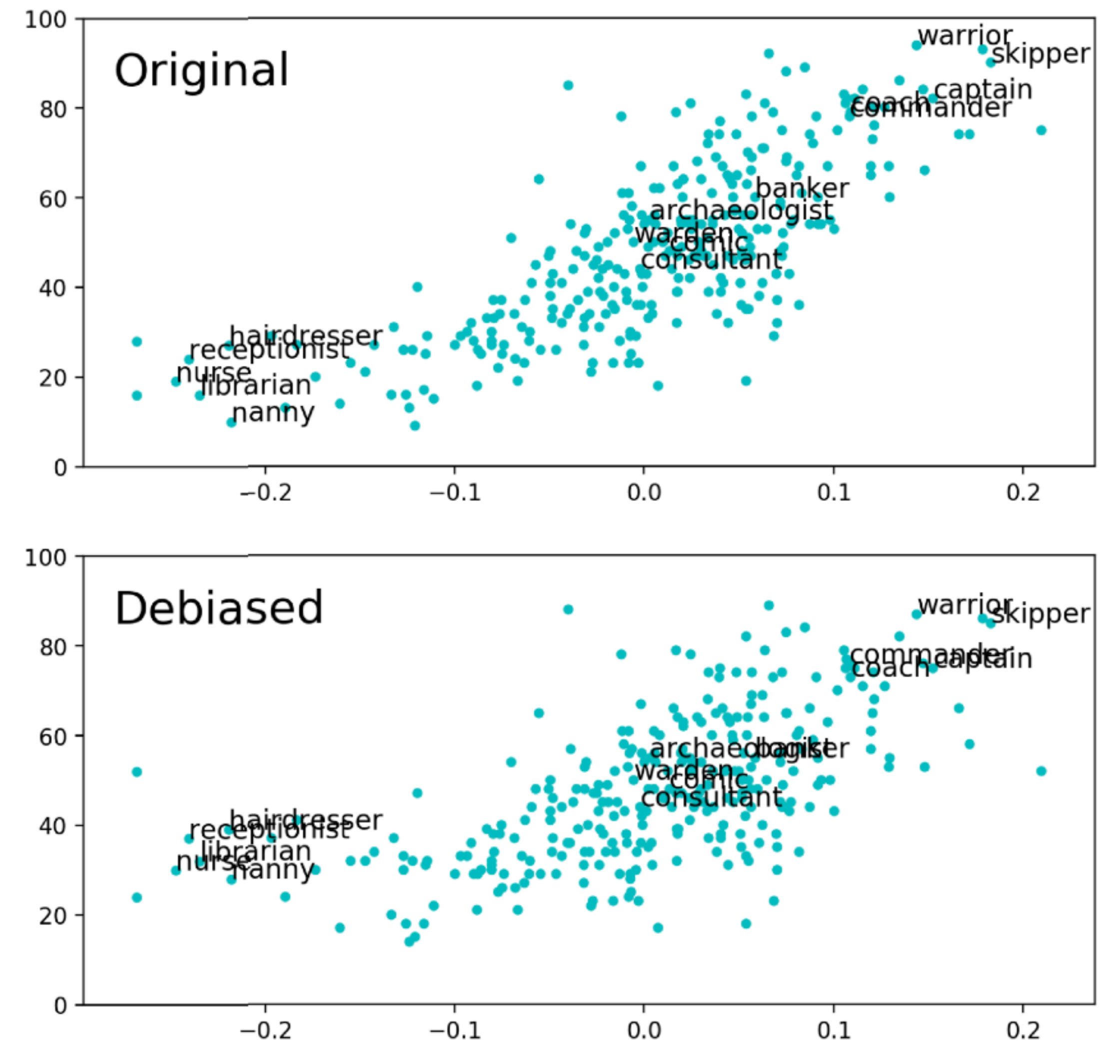


Bolukbasi et al. (2016)

Hardness of Debiasing

- Not that effective...and the male and female words are still clustered together

- Bias pervades the word embedding space and isn't just a local property of a few words



(a) The plots for HARD-DEBIASED embedding, before (top) and after (bottom) debiasing.

Gonen and Goldberg (2019)