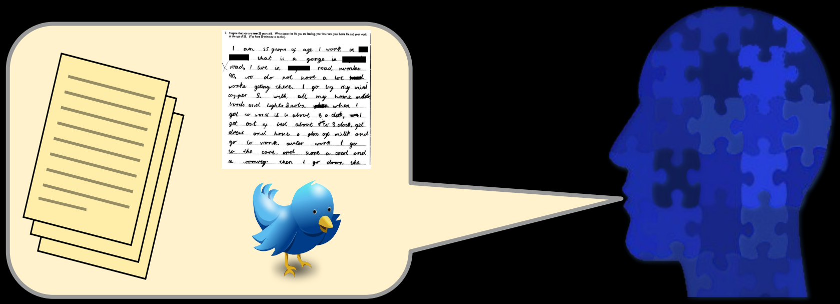
Vector Semantics and Embeddings

CSE392 - Spring 2019
Special Topic in CS

Tasks



- Vectors which represent words or sequences

how?

- Dimensionality Reduction
- Recurrent Neural Network and Sequence Models

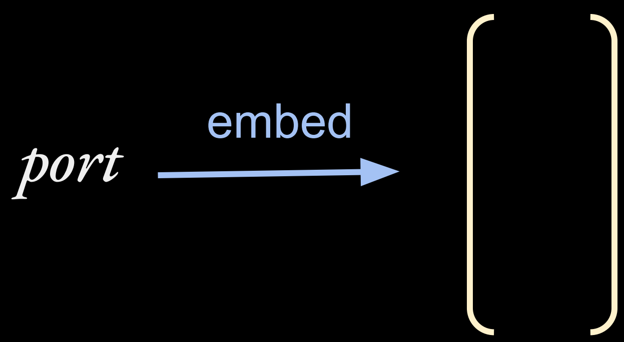
Objective

To embed: convert a token (or sequence) to a vector that represents meaning.

Objective

To embed: convert a token (or sequence) to a vector that is useful to perform NLP tasks.

Objective

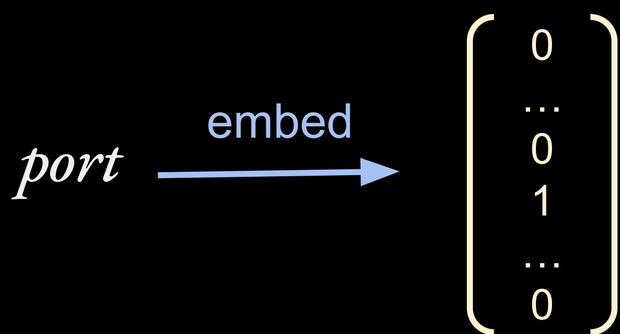


Objective

$$\textit{port} \xrightarrow{\text{embed}} \begin{bmatrix} 0 \\ \cdots \\ 0 \\ 1 \\ \cdots \\ 0 \end{bmatrix}$$

Objective

one-hot is sparse vector



Prefer dense vectors

- Less parameters (weights) for machine learning model.
- May generalize better implicitly.
- May capture synonyms

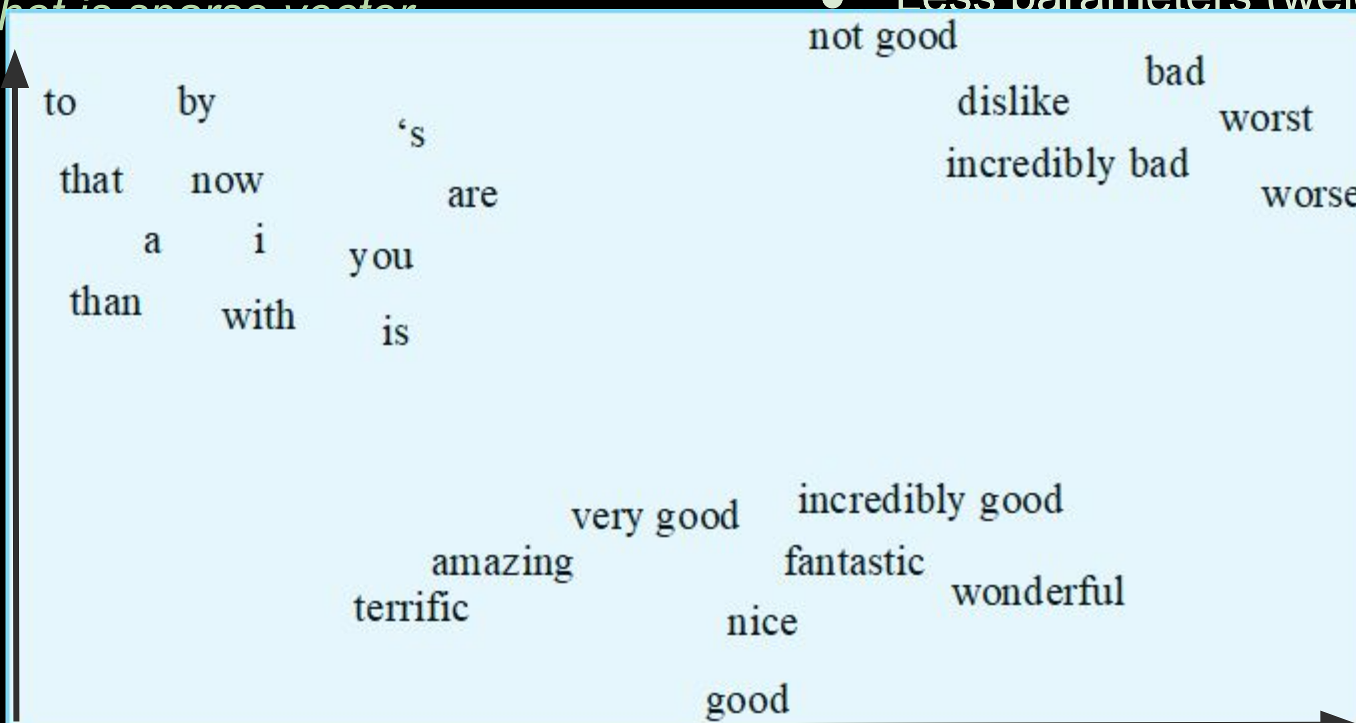
For deep learning, in practice, they work better. Why? Roughly, less parameters becomes increasingly important when you are learning multiple layers of weights rather than just a single layer.

Objective

Prefer dense vectors

- Less parameters (weights) for
  el.
  implicitly.
  s

  ce, they work
  rameters
  nt when you are
  ghts rather than



(Jurafsky, 2012)

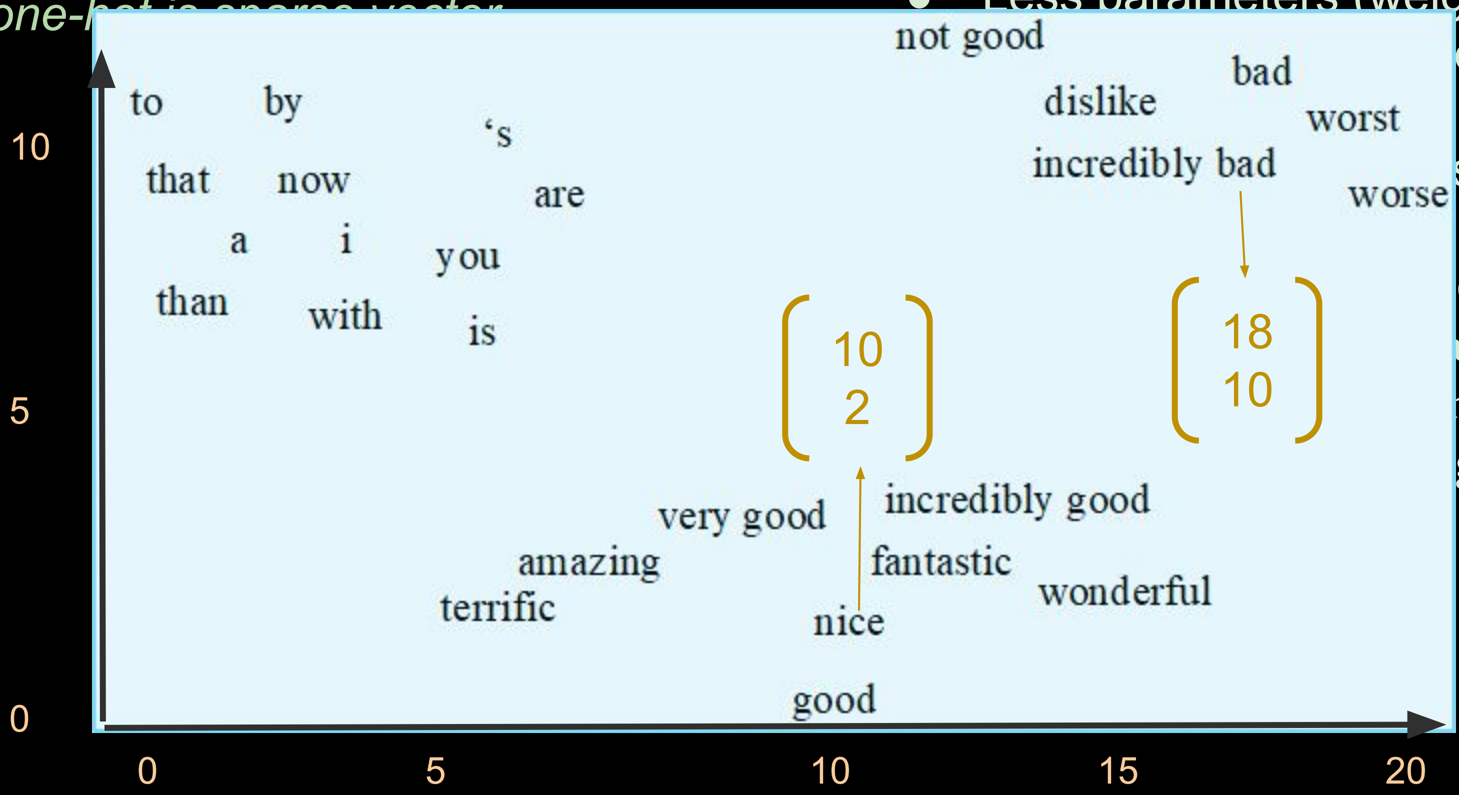
Objective

one-h...

Prefer dense vectors

- Less parameters (weights) for ...el.
- ...implicitly.
- ...s

...ce, they work
...rameters
...nt when you are
...ghts rather than



(Jurafsky, 2012)

Objective

To embed: convert a token (or sequence) to a vector that represents **meaning.**

Objective

To embed: convert a token (or sequence) to a vector that represents **meaning.**

Wittgenstein, 1945: "*The meaning of a word is its use in the language*"

Distributional hypothesis -- A word's meaning is defined by all the different contexts it appears in (i.e. how it is "distributed" in natural language).

Firth, 1957: "*You shall know a word by the company it keeps*"

Objective

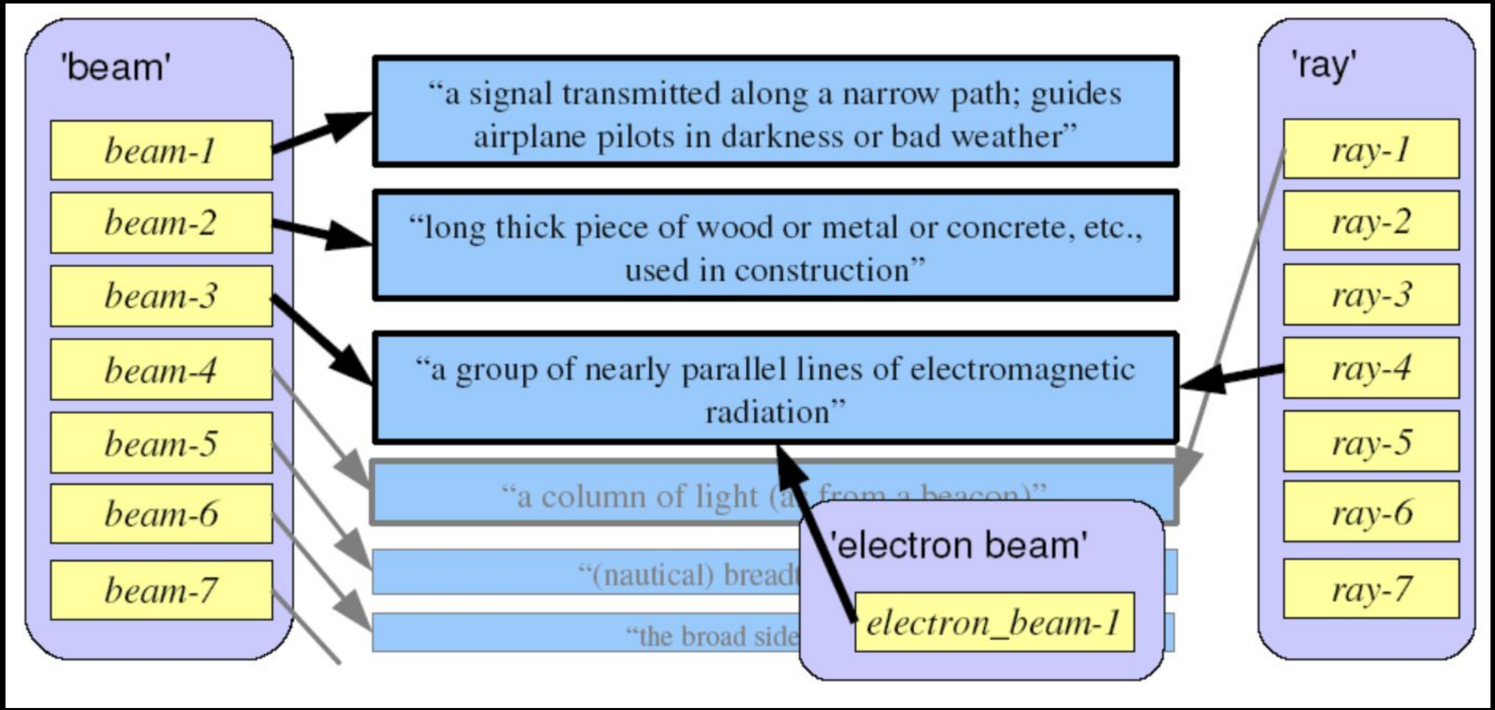
To embed: convert a token (or sequence) to a vector that represents **meaning.**

Wittgenstein, 1945: "*The meaning of a word is its use in the language*"

Distributional hypothesis -- A word's meaning is defined by all the different contexts it appears in (i.e. how it is "distributed" in natural language).

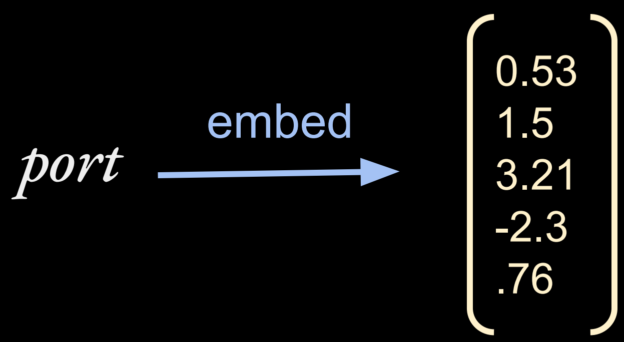
Firth, 1957: "*You shall know a word by the company it keeps*"

Distributional Hypothesis

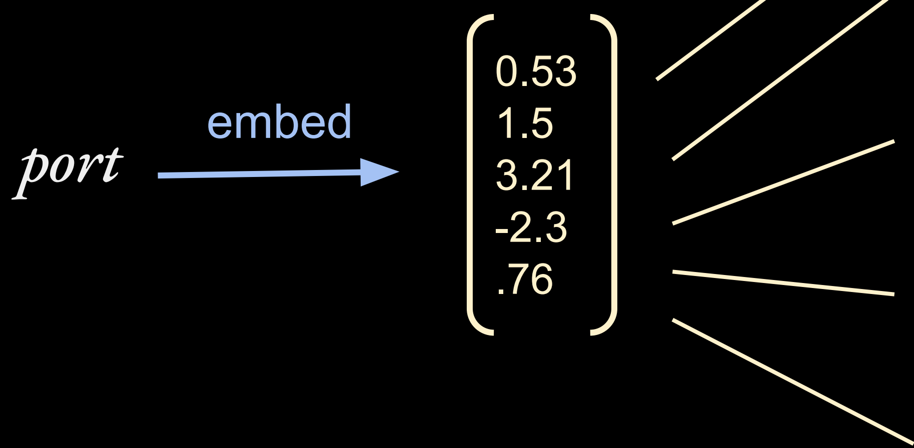


The nail hit the beam behind the wall.

Objective



Objective



port.n.1 (a place (seaport or airport) where people and merchandise can enter or leave a country)

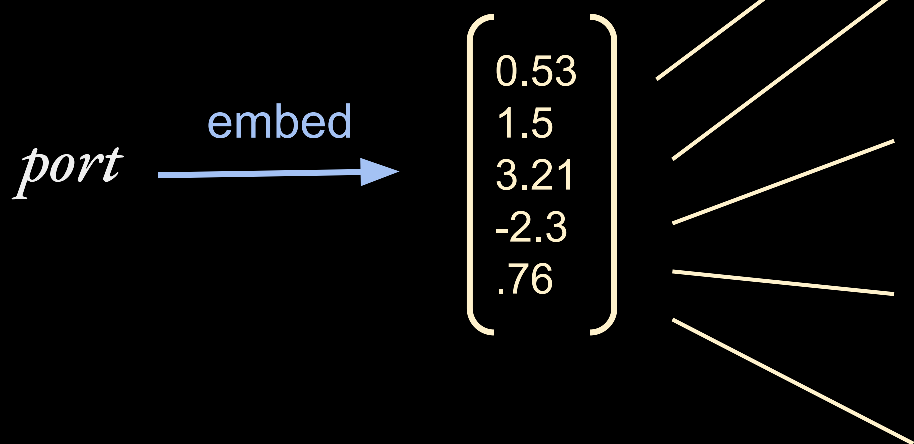
port.n.2 port wine (sweet dark-red dessert wine originally from Portugal)

port.n.3, embrasure, porthole (an opening (in a wall or ship or armored vehicle) for firing through)

larboard, **port**.n.4 (the left side of a ship or aircraft to someone who is aboard and facing the bow or nose)

interface, **port**.n.5 ((computer science) computer circuit consisting of the hardware and associated circuitry that links one device with another (especially a computer and a hard disk drive or other peripherals))

Objective



port.n.1 (a place (seaport or airport) where people and merchandise can enter or leave a country)

port.n.2 port wine (sweet dark-red dessert wine originally from Portugal)

port.n.3, embrasure, porthole (an opening (in a wall or ship or armored vehicle) for firing through)

larboard, **port**.n.4 (the left side of a ship or aircraft to someone who is aboard and facing the bow or nose)

interface, **port**.n.5 ((computer science) computer circuit consisting of the hardware and associated circuitry that links one device with another (especially a computer and a hard disk drive or other peripherals))

How?

1. One-hot representation
2. Selectors (represent context by “multi-hot” representation)
3. From PCA/Singular Value Decomposition
   (Know as “Latent Semantic Analysis” in some circumstances)

Tf-IDF: Term Frequency, Inverse Document Frequency,

PMI: Point-wise mutual information, ...etc…

How?

1. One-hot representation
2. Selectors (represent context by “multi-hot” representation)
3. From PCA/Singular Value Decomposition
   (Know as “Latent Semantic Analysis” in some circumstances)

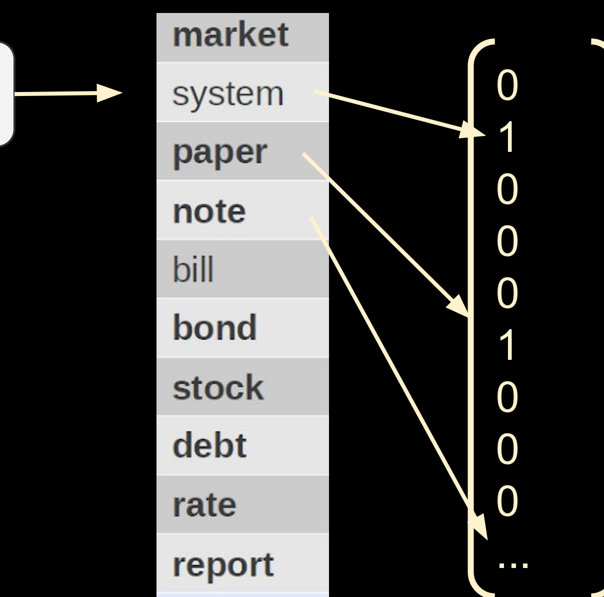
“Neural Embeddings”:

4. Word2vec
5. Fasttext
6. Glove
7. Bert

How?

1. One-hot represent
2. **Selectors (represent context by “multi-hot” representation)**
3. From PCA/Singular Value Decomposition
   (Know as “Latent Semantic Analysis” in some circumstances)

…, word1, word2, **bill**, word3, word4, ...



4. Word2vec
5. Fasttext
6. Glove
7. Bert

How?

1. One-hot represent
2. Selectors (represent context by “multi-hot” representation)
3. From PCA/Singular Value Decomposition
   (Know as “Latent Semantic Analysis” in some circumstances)

“Neural Embeddings”:

4. Word2vec
5. Fasttext
6. Glove
7. Bert

How?

1. One-hot represent
2. Selectors (represent context by “multi-hot” representation)
3. From PCA/Singular Value Decomposition
   (Know as “Latent Semantic Analysis” in some circumstances)

“Neural Embeddings”:

4. Word2vec
5. Fasttext
6. Glove
7. Bert

How?

1. One-hot represent
2. Selectors (represent context by “multi-hot” representation)
3. From PCA/Singular Value Decomposition
   (Know as “Latent Semantic Analysis” in some circumstances)

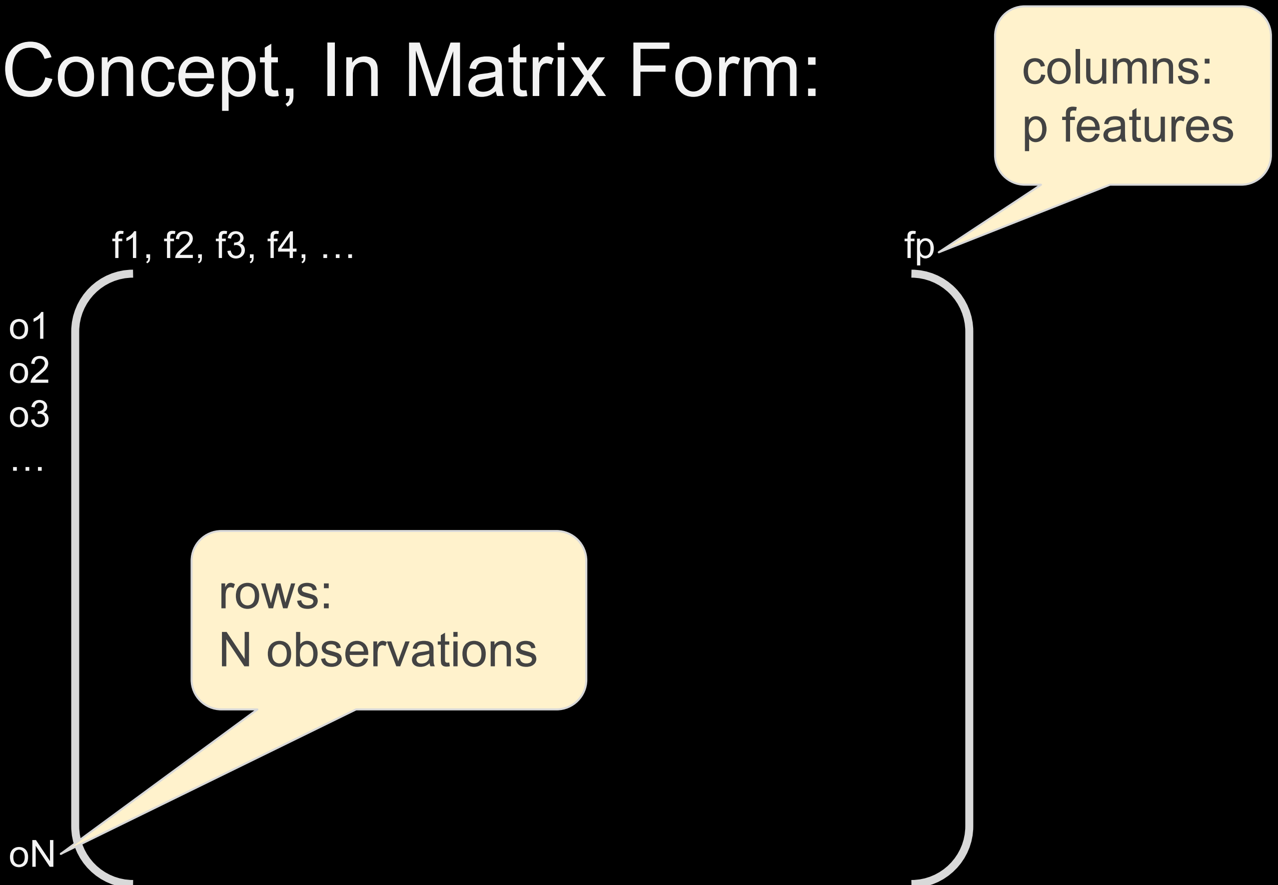
“Neural Embeddings”:

4. Word2vec
5. Fasttext
6. Glove
7. Bert

SVD-Based Embeddings

Singular Value Decomposition...

Concept, In Matrix Form:

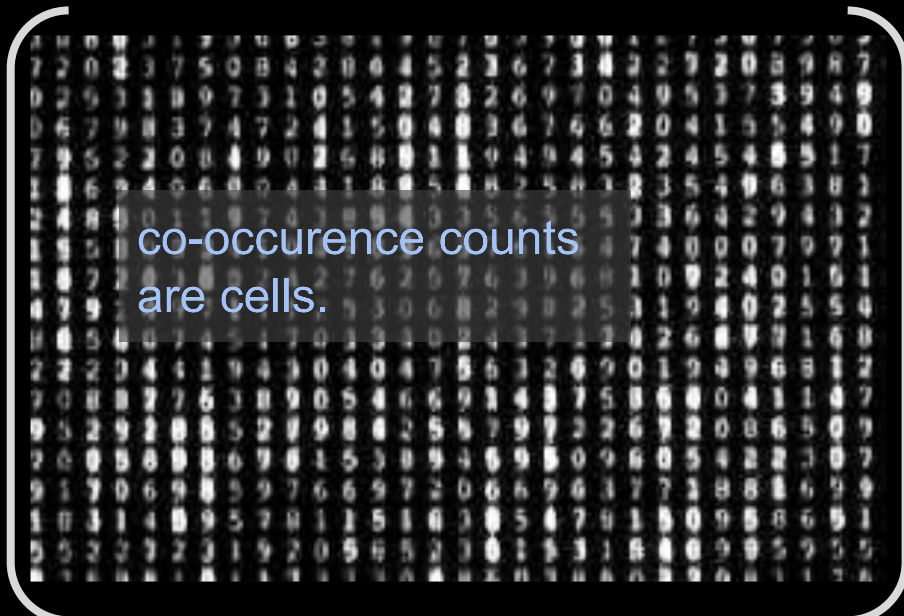


SVD-Based Embeddings

context words are features

f1, f2, f3, f4, … fp

o1
o2
o3
…
oN



target words are observations

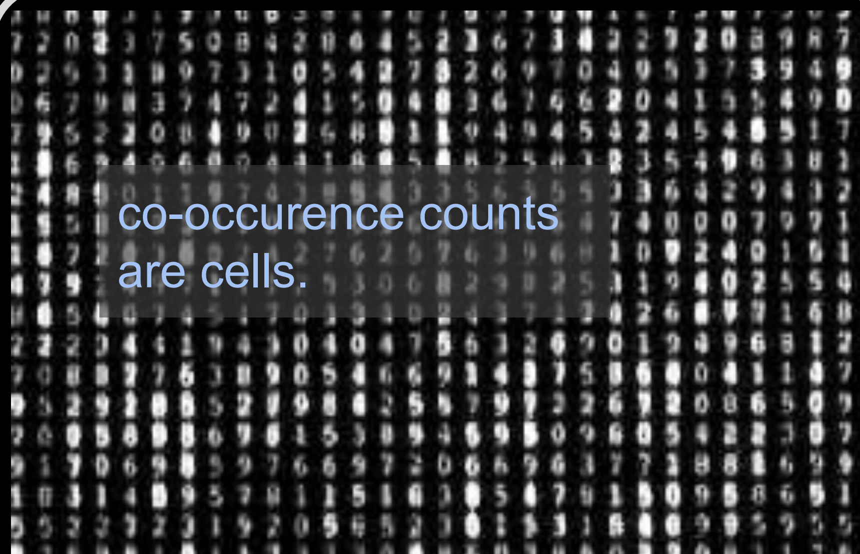
SVD-Based Embeddings

Dimensionality reduction
-- try to represent with only p’ dimensions

context words are features

f1, f2, f3, f4, … fp

o1
o2
o3
…
oN

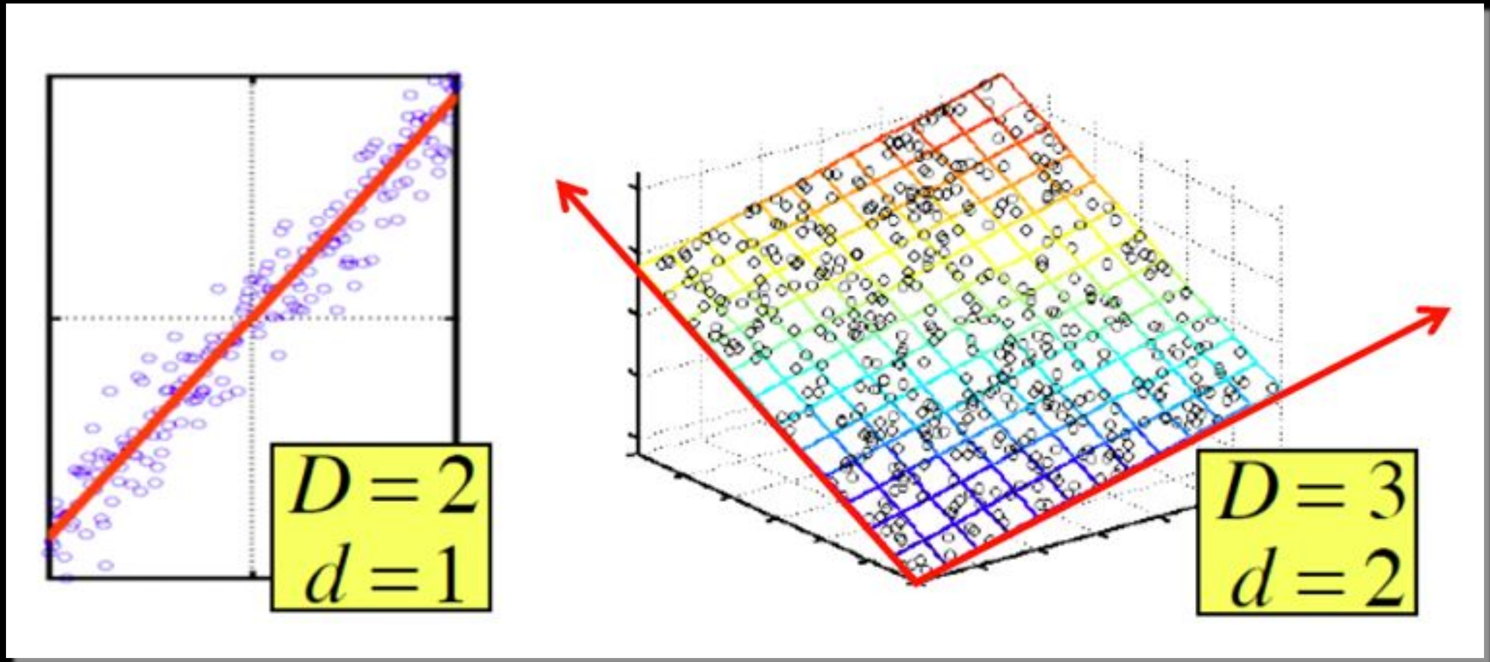


target words are
observations

c1, c2, c3, c4, … cp’

o1
o2
o3
…
oN

Concept: Dimensionality Reduction in 3-D, 2-D, and 1-D



Data (or, at least, what we want from the data) may be accurately represented with less dimensions.

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Dimensionality Reduction

Rank: Number of linearly independent columns of A.
(i.e. columns that can't be derived from the other columns through addition).

Q: What is the rank of this matrix?

$$\begin{pmatrix} 1 & -2 & 3 \\ 2 & -3 & 5 \\ 1 & 1 & 0 \end{pmatrix}$$

Dimensionality Reduction

Rank: Number of linearly independent columns of A.
(i.e. columns that can't be derived from the other columns).

Q: What is the rank of this matrix?

$$\begin{pmatrix} 1 & -2 & 3 \\ 2 & -3 & 5 \\ 1 & 1 & 0 \end{pmatrix}$$

A: 2. The 1st is just the sum of the second two columns

… we can represent as linear combination of 2 vectors:

$$\begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} \begin{pmatrix} -2 \\ -3 \\ 1 \end{pmatrix}$$

Dimensionality Reduction - PCA

Linear approximates of data in r dimensions.

Found via *Singular Value Decomposition:*

$$X_{[nxp]} = U_{[nxr]} D_{[rxr]} V_{[pxr]}^T$$

X: original matrix, U: “left singular vectors”,

D: “singular values” (diagonal), V: “right singular vectors”

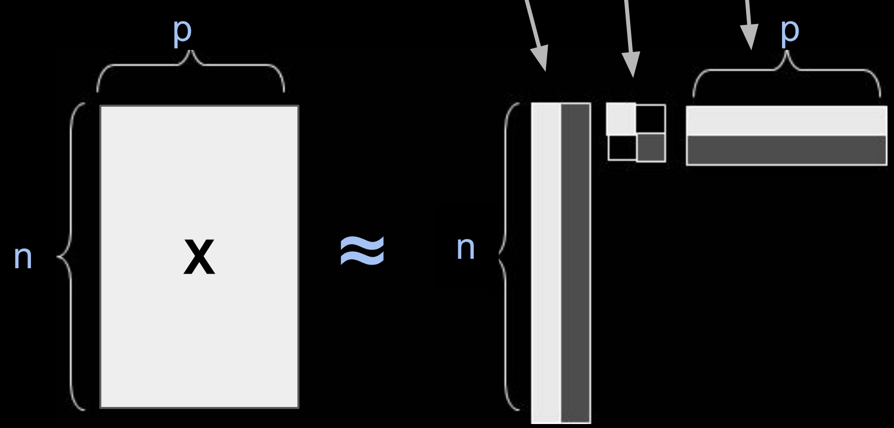
Dimensionality Reduction - PCA

Linear approximates of data in r dimensions.

Found via *Singular Value Decomposition:*

$$X_{[nxp]} = U_{[nxr]} D_{[rxr]} V_{[pxr]}^T$$

X: original matrix, U: "left singular vectors",
D: "singular values" (diagonal), V: "right singular vectors"



Dimensionality Reduction - PCA - Example

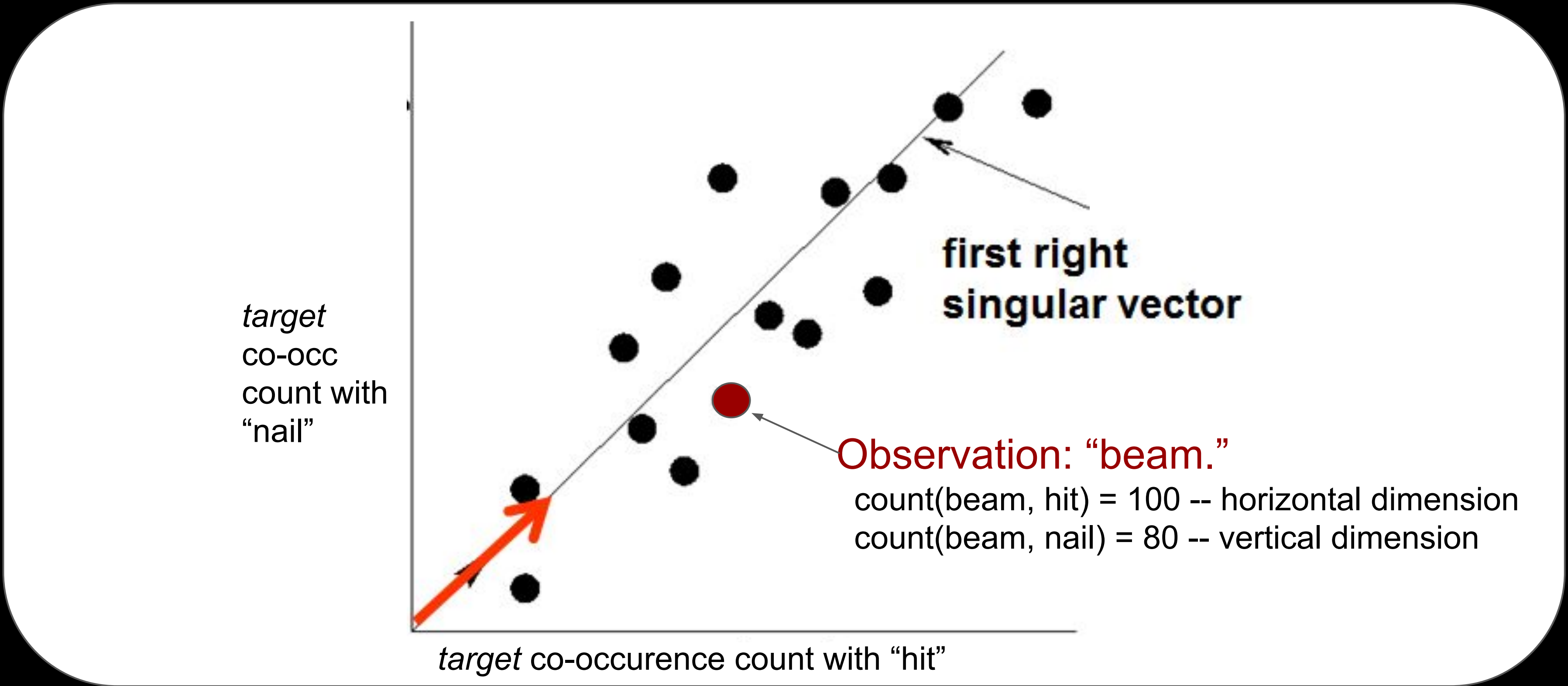
$$X_{[n \times p]} = U_{[n \times r]} D_{[r \times r]} V_{[p \times r]}^T$$

Word co-occurrence counts:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

Dimensionality Reduction - PCA - Example

$$X_{[nxp]} \cong U_{[nxr]} D_{[rxr]} V_{[pxr]}^T$$



Dimensionality Reduction - PCA

Linear approximates of data in r dimensions.

Found via *Singular Value Decomposition:*

$$X_{[nxp]} \cong U_{[nxr]} D_{[rxr]} V_{[pxr]}^T$$

X: original matrix,
D: "singular values" (diagonal),
U: "left singular vectors",
V: "right singular vectors"

Projection (dimensionality reduced space) in 3 dimensions:

$$(U_{[nx3]} D_{[3x3]} V_{[px3]}^T)$$

To reduce features in new dataset, A:

$$A_{[\,m \times p\,]} VD = A_{small[\,m \times 3\,]}$$

Dimensionality Reduction - PCA

Linear approximates of data in r dimensions.

Found via *Singular Value Decomposition:*

$$X_{[nxp]} \cong U_{[nxr]} D_{[rxr]} V_{[pxr]}^T$$

X: original matrix,
D: "singular values" (diagonal),
U: "left singular vectors",
V: "right singular vectors"

To check how well the original matrix can be reproduced:

$Z_{[nxp]} = U D V^T$, How does Z compare to X?

To reduce features in new dataset, A:

$$A_{[\, m \times p \,]} VD = A_{small[\, m \times 3 \,]}$$

Dimensionality Reduction - PCA

Linear approximates of data in r dimensions.

Found via *Singular Value Decomposition:*

$$X_{[nxp]} \cong U_{[nxr]} D_{[rxr]} V_{[pxr]}^T$$

X: original matrix, U: “left singular vectors”,
D: “singular values” (diagonal), V: “right singular vectors”

To check how well the original matrix can be reproduced:

$Z_{[nxp]} = U D V^T$, How does Z compare to original X?

To reduce features in new dataset:

$$A_{[m \times p]} VD = A_{small[m \times 3]}$$

Dimensionality Reduction - PCA

- **Goal: Minimize the sum of reconstruction errors:**

$$\sum_{i=1}^{N}\sum_{j=1}^{D}\left\|x_{ij}-z_{ij}\right\|^{2}$$

  - where $\boldsymbol{x}_{ij}$ are the “old” and $\boldsymbol{z}_{ij}$ are the “new” coordinates

X: original mat ... ar vectors”,
D: “singular va ... ular vectors”

To check how well the original matrix can be reproduced:

$Z_{[nxp]} = U\ D\ V^T$, How does Z compare to original X?

To reduce features in new dataset:

$A_{[\ m\ x\ p\ ]}VD = A_{small[\ m\ x\ 3\ ]}$

Dimensionality Reduction - PCA

This is the objective
that SVD Solves

- **Goal: Minimize the sum of reconstruction errors:**

$$\sum_{i=1}^{N}\sum_{j=1}^{D}\left\| x_{ij} - z_{ij} \right\|^2$$

  - where $\boldsymbol{x}_{ij}$ are the "old" and $\boldsymbol{z}_{ij}$ are the "new" coordinates

X: original mat … ar vectors",
D: "singular va … ular vectors"

To check how well the original matrix can be reproduced:

$Z_{[nxp]} = U\ D\ V^T$, How does Z compare to original X?

To reduce features in new dataset:

$A_{[\ m\ x\ p\ ]}\ VD = A_{small[\ m\ x\ 3\ ]}$

Dimensionality Reduction - PCA

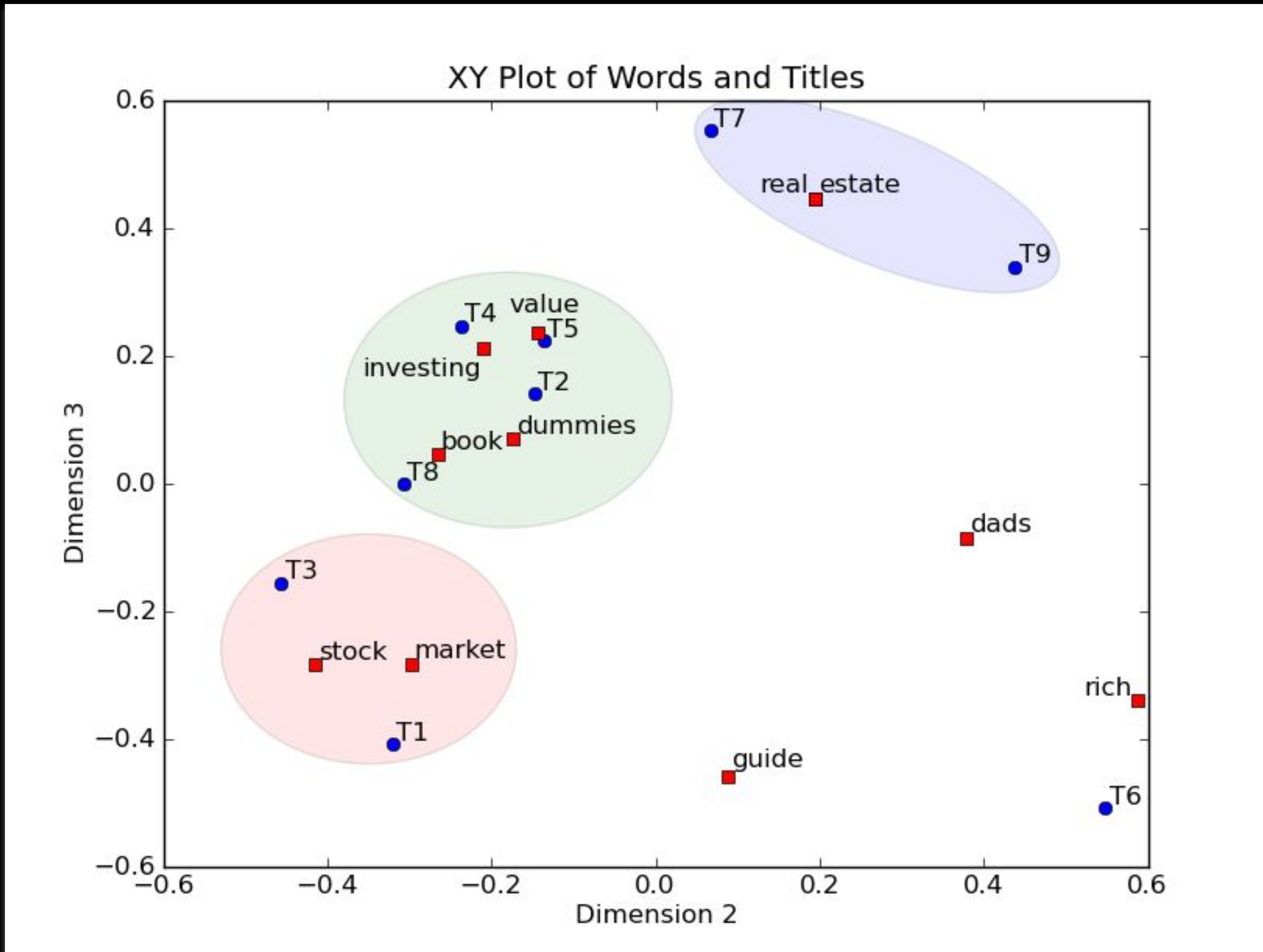
Linear approximates of data in r dimensions.

Found via *Singular Value Decomposition:*

$$X_{[nxp]} \cong U_{[nxr]} D_{[rxr]} V_{[pxr]}^{T}$$

U, D, and V are unique

D: always positive


XY Plot of Words and Titles
Dimension 3
Dimension 2
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
T7
real estate
T9
T4
value
T5
investing
T2
book
dummies
T8
dads
T3
stock
market
rich
T1
guide
T6

(TechnoWiki)

How?

1. One-hot represent
2. Selectors (represent context by “multi-hot” representation)
3. From PCA/Singular Value Decomposition
   (Know as “Latent Semantic Analysis” in some circumstances)

“Neural Embeddings”:

4. Word2vec
5. Fasttext
6. Glove
7. Bert

How?

1. One-hot represent
2. Selectors (represent context by “multi-hot” representation)
3. From PCA/Singular Value Decomposition
   (Know as “Latent Semantic Analysis” in some circumstances)

“Neural Embeddings”:

4. **Word2vec**
5. Fasttext
6. Glove
7. Bert

Word2Vec

Principal: Predict missing word.

Similar to language modeling but predicting context, rather than next word.

p(context | word)

Word2Vec

Principal: Predict missing word.

Similar to language modeling but predicting context, rather than next word.

p(context | word)

To learn, maximize

Word2Vec

Principal: Predict missing word.

Similar to language modeling but predicting context, rather than next word.

p(context | word)

To learn, maximize.
In practice, minimize

$$J = 1 - p(context \mid word)$$

Word2Vec: Context

p(context | word)

2 Versions of Context:

- Continuous bag of words (CBOW): Predict word from context
- Skip-Grams (SG): predict context words from target

Word2Vec: Context

p(context | word)

2 Versions of Context:

- Continuous bag of words (CBOW): Predict word from context
- **Skip-Grams (SG): predict context words from target**

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

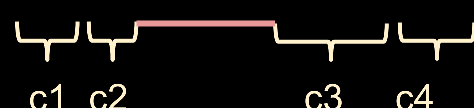
(Jurafsky, 2017)

Word2Vec: Context

p(context | word)

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

The nail hit the beam behind the wall.



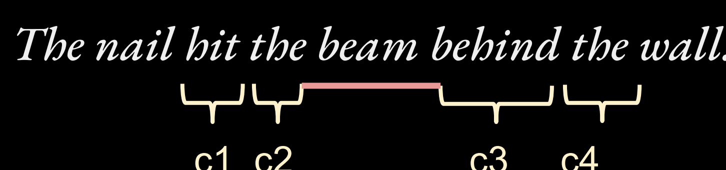
(Jurafsky, 2017)

Word2Vec: Context

p(context | word)

x = (hit, beam), y = 1
x = (the, beam), y = 1
x = (behind, beam), y = 1
...

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



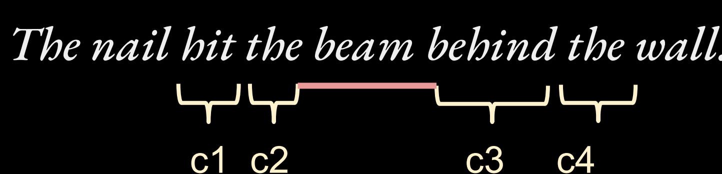
(Jurafsky, 2017)

Word2Vec: Context

p(context | word)

x = (hit, beam), y = 1
x = (the, beam), y = 1
x = (behind, beam), y = 1
…
x = (happy, beam), y = 0
x = (think, beam), y = 0
...

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



(Jurafsky, 2017)

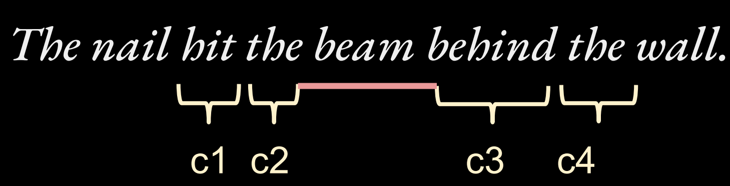
Word2Vec: Context

p(context | word)

x = (hit, beam), y = 1
x = (the, beam), y = 1
x = (behind, beam), y = 1
…
x = (happy, beam), y = 0
x = (think, beam), y = 0
…

k negative example (y=0) for every positive.
How?

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



(Jurafsky, 2017)

Word2Vec: Context

p(context | word)

x = (hit, beam), y = 1
x = (the, beam), y = 1
x = (behind, beam), y = 1
…
x = (happy, beam), y = 0
x = (think, beam), y = 0
...

k negative example (y=0) for every positive.
How? Randomly draw from unigram distribution

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

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

The nail hit the beam behind the wall.

c1 c2 c3 c4

(Jurafsky, 2017)

Word2Vec: Context

p(context | word)

x = (hit, beam), y = 1
x = (the, beam), y = 1
x = (behind, beam), y = 1
…
x = (happy, beam), y = 0
x = (think, beam), y = 0
…

k negative example (y=0) for every positive.
How? Randomly draw from unigram distribution adjusted:

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

$\alpha = 0.75$

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

The nail hit the beam behind the wall.

c1 c2 c3 c4

(Jurafsky, 2017)

Word2Vec: Context

p(context | word)

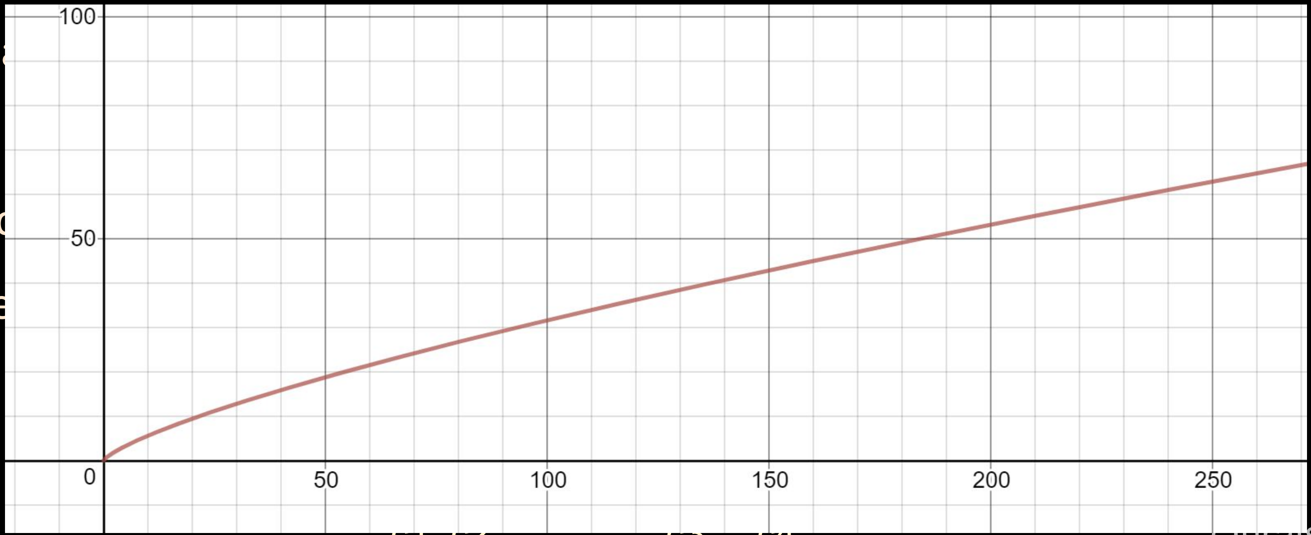
x = (hit, beam), y = 1
x = (the, beam), y = 1
x = (behind, beam), y = 1
…
x = (happy, beam), y = 0
x = (think, beam), y = 0
...

k negative example (y=0) for every positive.
How? Randomly draw from unigram distribution adjusted:

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

$\alpha = 0.75$

1. Treat the t
2. **Randomly**
3. Use logisti
4. Use the we



c1 c2 c3 c4

(Jurafsky, 2017)

Word2Vec: Context

x = (hit, beam), y = 1
x = (the, beam), y = 1
x = (behind, beam), y = 1
…
x = (happy, beam), y = 0
x = (think, beam), y = 0
…

single context:

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

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

The nail hit the beam behind the wall.

c1 c2 c3 c4

(Jurafsky, 2017)

Word2Vec: Context

Logistic: $\sigma(z) = 1 / (1 + e^{-z})$

x = (hit, beam), y = 1
x = (the, beam), y = 1
x = (behind, beam), y = 1
…
x = (happy, beam), y = 0
x = (think, beam), y = 0
…

single context:

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

All Contexts

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

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

The nail hit the beam behind the wall.

c1 c2 c3 c4

(Jurafsky, 2017)

Word2Vec: Context

x = (hit, beam), y = 1
x = (the, beam), y = 1
x = (behind, beam), y = 1
…
x = (happy, beam), y = 0
x = (think, beam), y = 0
…

single context:

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

Intuition: t□c is a measure of similarity:

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos \theta$$

But, it is not a probability! To make it one, apply logistic activation:

$$\sigma(z) = 1 / (1 + e^{-z})$$

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

The nail hit the beam behind the wall.

c1 c2 c3 c4

(Jurafsky, 2017)

Word2Vec: Context

Intuition: t□c is a measure of similarity:

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos\theta$$

But, it is not a probability! To make it one, apply logistic activation:

$$\sigma(z) = 1 / (1 + e^{-z})$$

x = (hit, beam), y = 1
x = (the, beam), y = 1
x = (behind, beam), y = 1
…
x = (happy, beam), y = 0
x = (think, beam), y = 0
…

single context:

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

all contexts

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

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

The nail hit the beam behind the wall.

c1 c2 c3 c4

(Jurafsky, 2017)

Word2Vec: How to Learn?

$P(y=1 | c, t)$

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**

The nail hit the beam behind the wall.

c1 c2 c3 c4

(Jurafsky, 2017)

Word2Vec: How to Learn?

$P(y=1 | c, t)$

Assume 300 * |vocab| weights (parameters) for each of c and t

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**

The nail hit the beam behind the wall.

c1 c2 c3 c4

(Jurafsky, 2017)

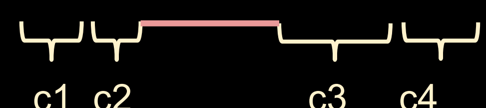
Word2Vec: How to Learn?

$P(y=1 | c, t)$

Assume 300 * |vocab| weights (parameters) for each of c and t
Start with random vectors (or all 0s)

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**

The nail hit the beam behind the wall.



(Jurafsky, 2017)

Word2Vec: How to Learn?

$P(y=1 | c, t)$

Assume 300 * |vocab| weights (parameters) for each of c and t
Start with random vectors (or all 0s)

Goal:

Maximize similarity of (c, t) in positive data (y = 1)
Minimize similarity of (c, t) in negative data (y = 0)

The nail hit the beam behind the wall.

c1 c2 c3 c4

(Jurafsky, 2017)

Word2Vec: How to Learn?

$P(y=1|\ c,\ t)$

Assume 300 * |vocab| weights (parameters) for each of c and t
Start with random vectors (or all 0s)

Goal:

Maximize similarity of (c, t) in positive data (y = 1)
Minimize similarity of (c, t) in negative data (y = 0)

$$\sum_{(c,t)} (y) log P(y = 1|c, t) + (y - 1) log P(y = 0|c, t)$$

Word2Vec: How to Learn?

P(y=1| c, t)

Assume 300 * |vocab| weights (parameters) for each of c and t
Start with random vectors (or all 0s)

Goal:

Maximize similarity of (c, t) in positive data (y = 1)
Minimize similarity of (c, t) in negative data (y = 0)

$$\sum_{(c,t)} (y) log P(y = 1|c, t) + (y - 1) log P(y = 0|c, t)$$

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

Word2Vec: How to Learn?

P(y=1| c,

Assum for each of c and t
Start w

Optimized using gradient descent type methods.

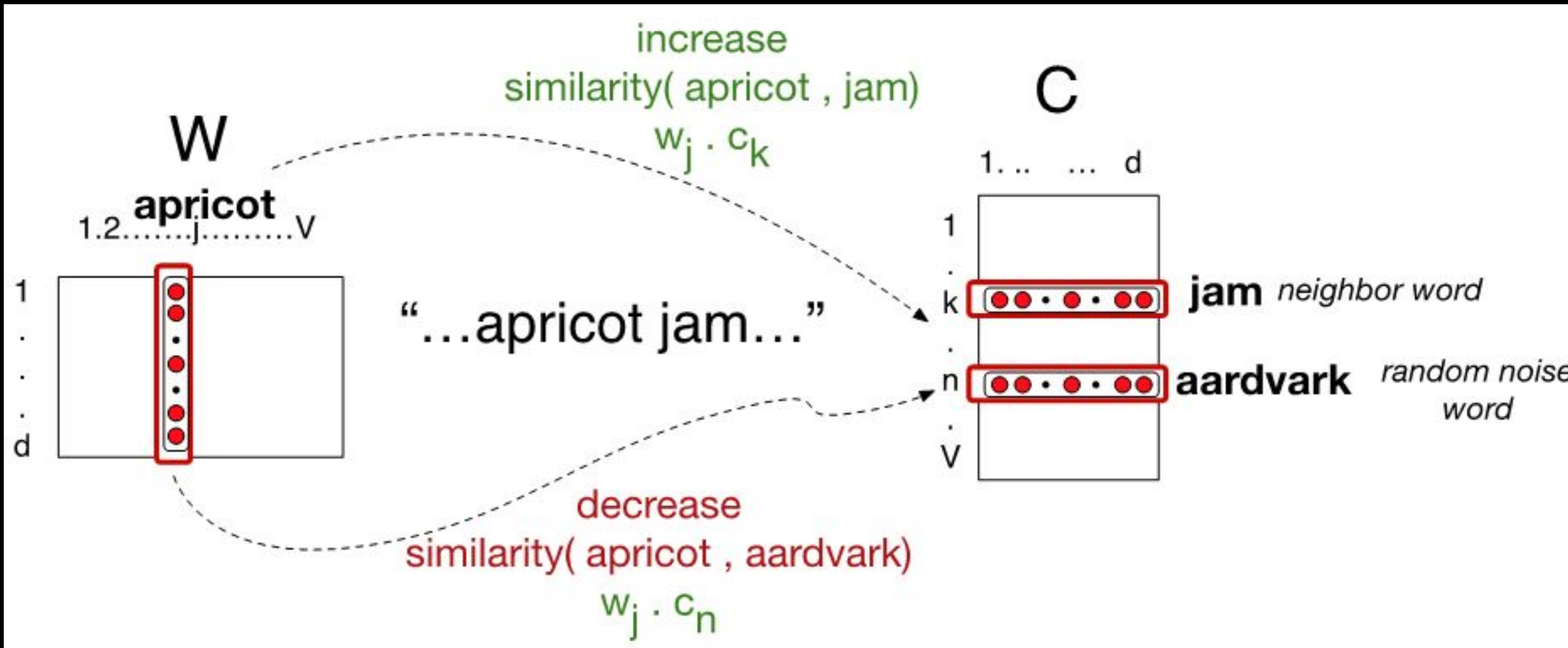
Goal:

Maximize ilarity of (c, t) in positive data (y = 1)
Minimize milarity of (c, t) in negative data (y = 0)

$$\sum_{(c,t)} (y) log P(y = 1|c,t) + (y-1) log P(y = 0|c,t)$$

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

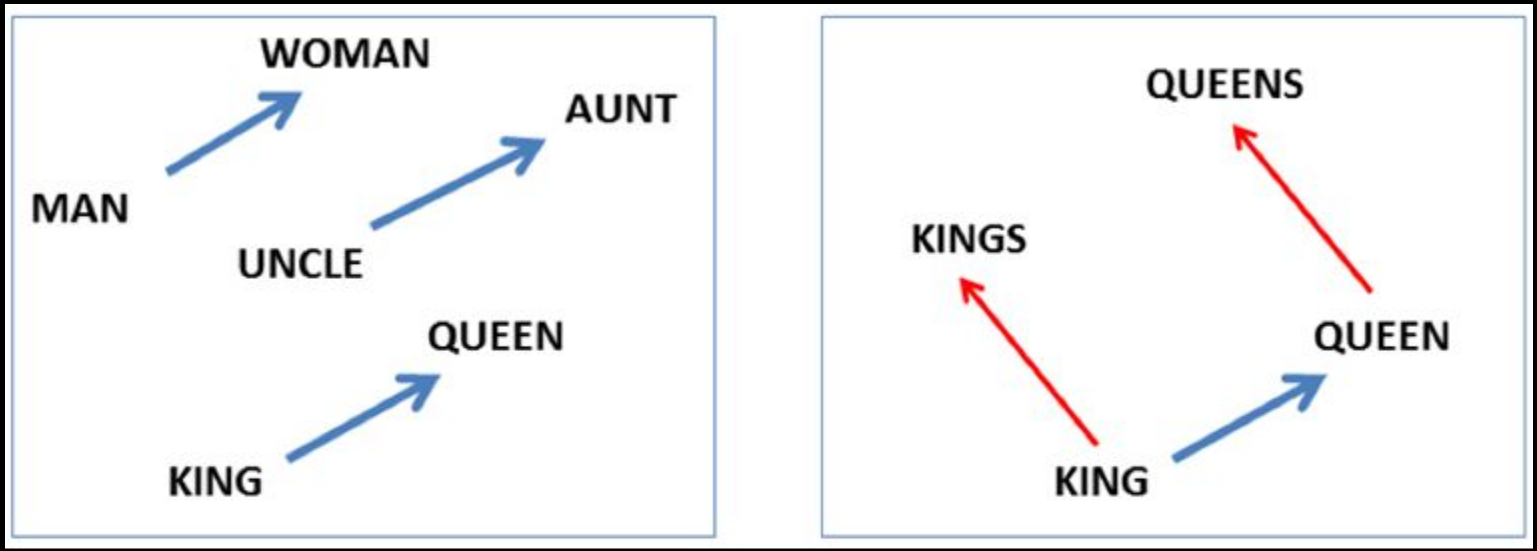
Word 2 Vec



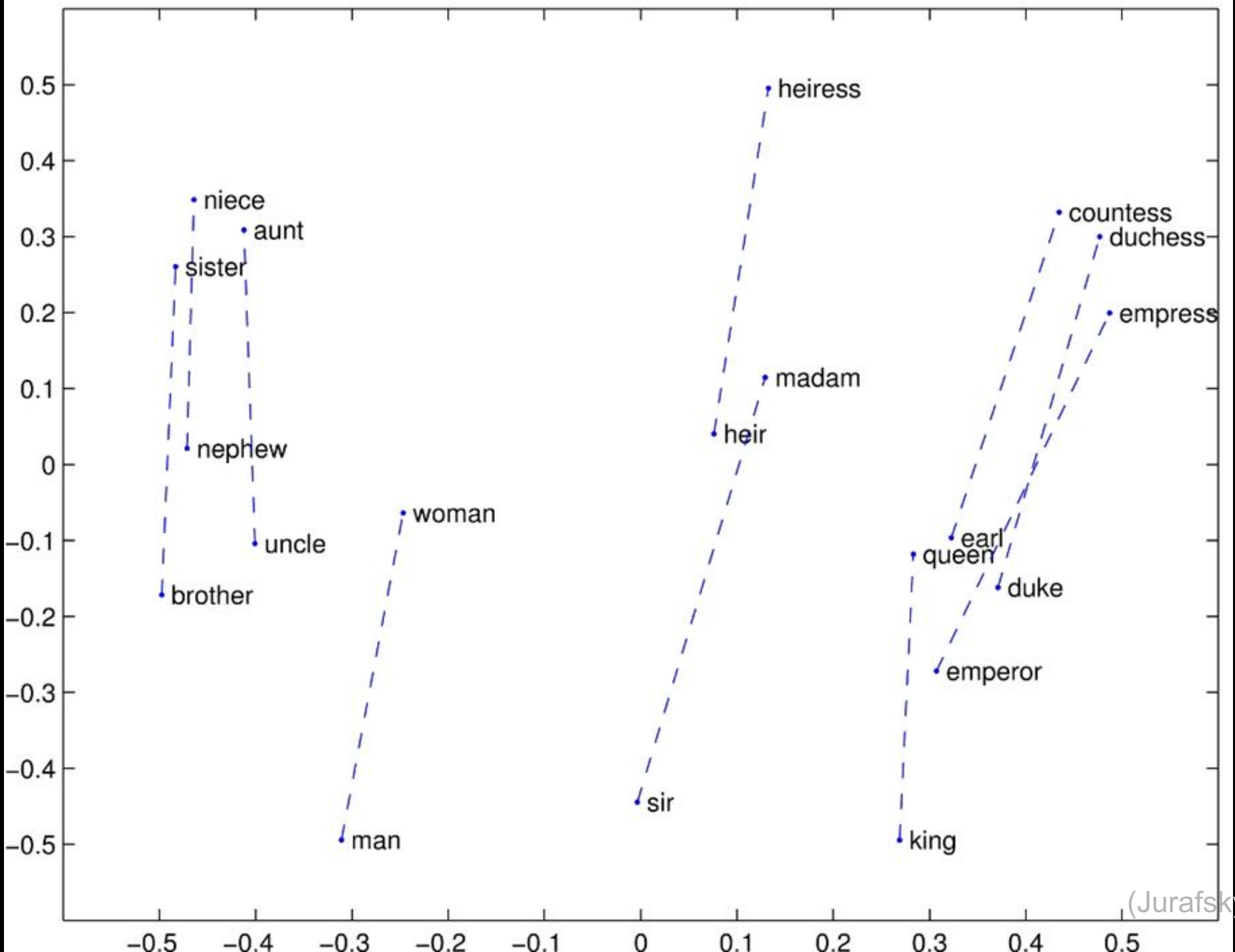
$$\sum_{(c,t)} (y) log P(y = 1|c, t) + (y - 1) log P(y = 0|c, t)$$

(Jurafsky, 2017)

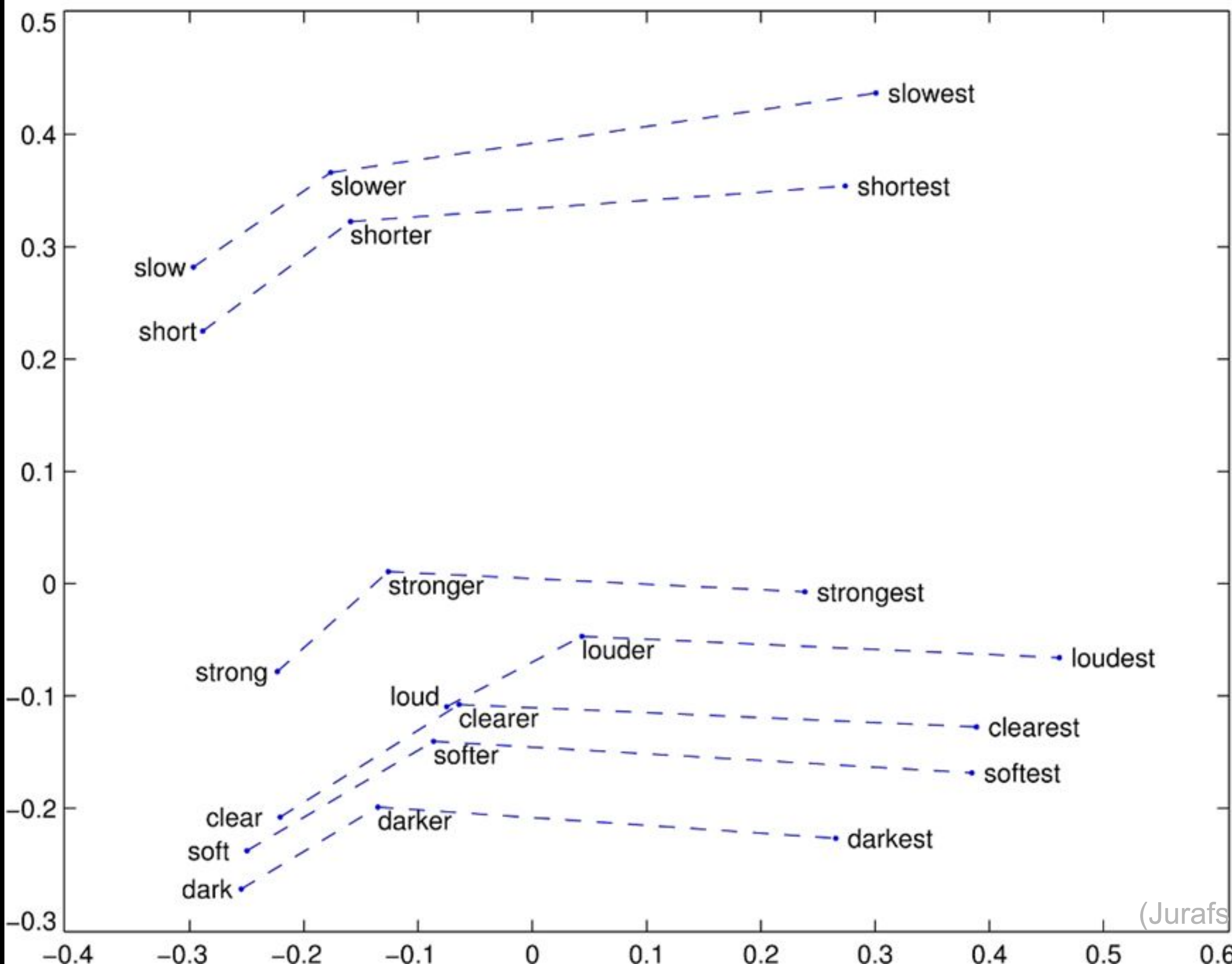
Word2Vec captures analogies (kind of)



(Jurafsky, 2017)



(Jurafsky, 2017)



(Jurafsky, 2017)

Word2Vec: Quantitative Evaluations

Compare to manually annotated pairs of words: WordSim-353 (Finkelstein et al., 2002)

Compare to words in context (Huang et al., 2012)

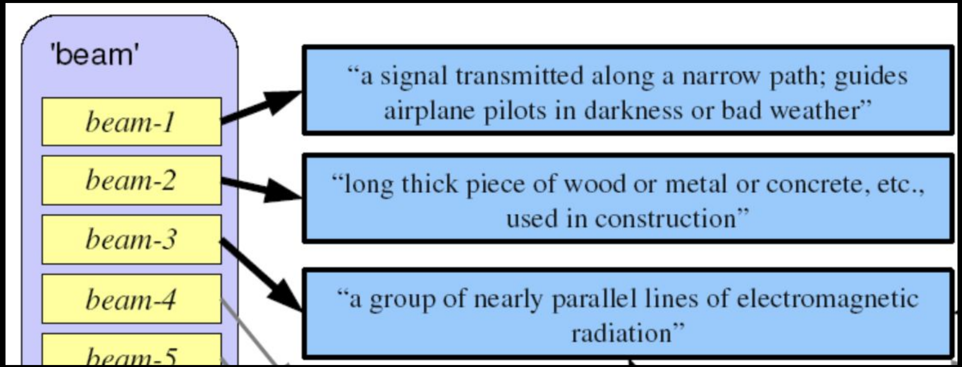
Answer TOEFL synonym questions.

Current Trends in Embeddings

1. Contextual word embeddings (a different embedding depending on context):

   The nail hit the beam behind the wall.

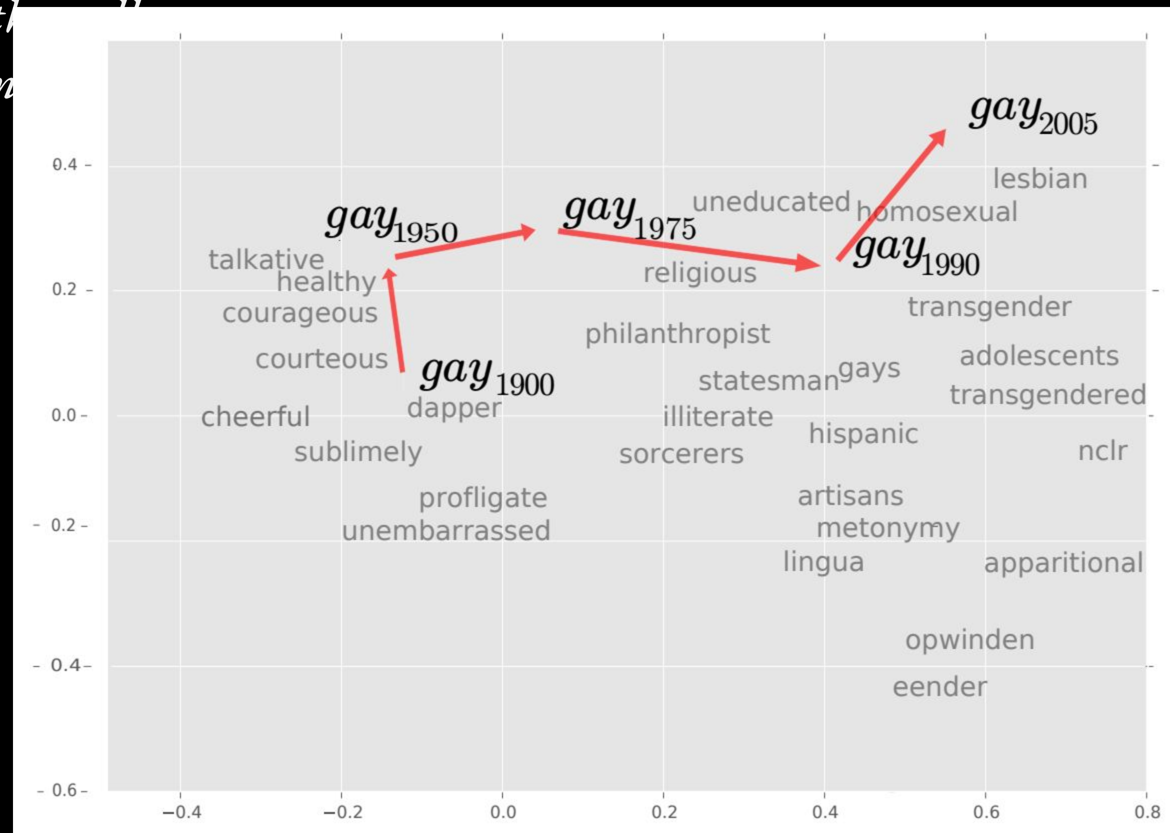
   They reflected a beam off the moon.



Current Trends in Embeddings

1. Contextual word embeddings (a different embedding depending on context):
   The nail hit the beam behind th
   They reflected a beam off the m

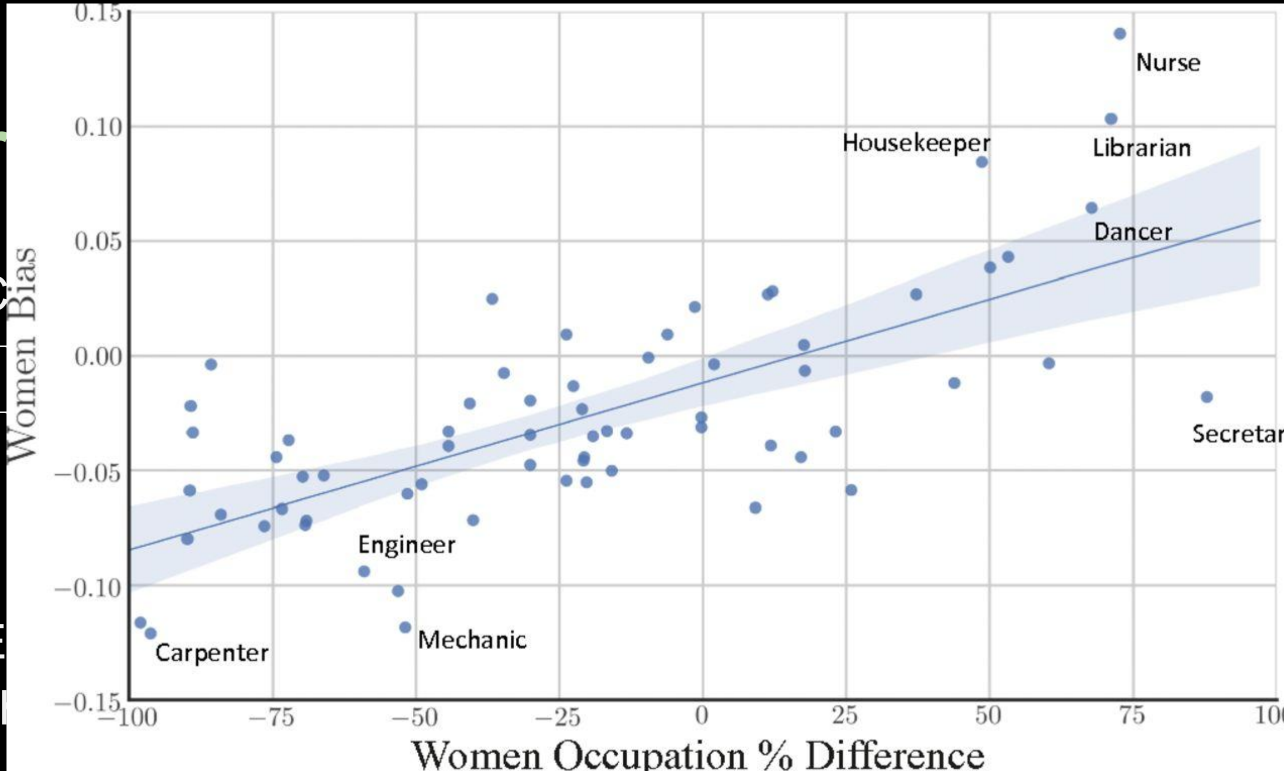
2. Embeddings can capture changes in word meaning.



(Kulkarni et al.,2015)

Cur

1. C g on context):
   T
   T



2. E
   cl

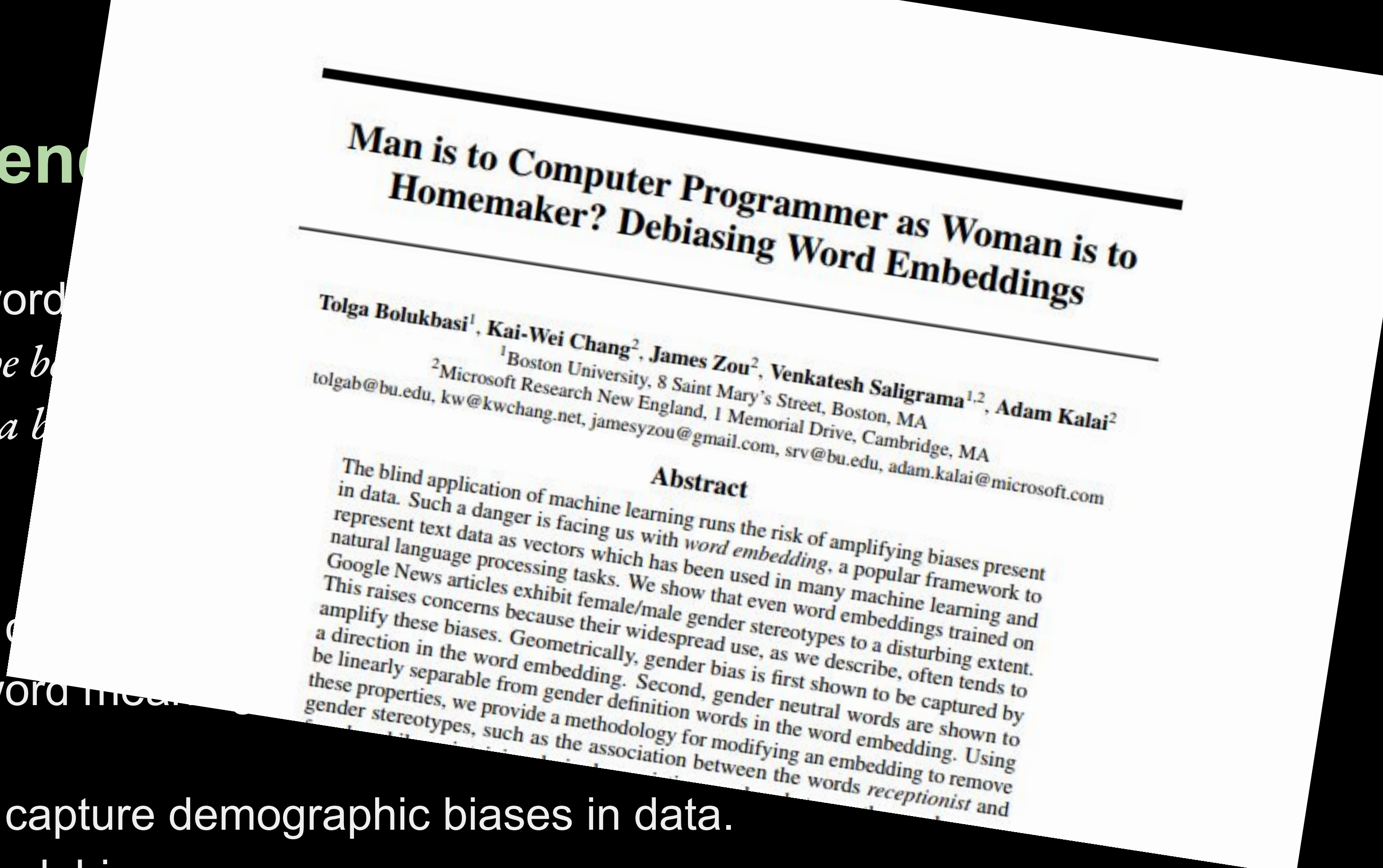
(Garg et al., 2018)

3. Embeddings capture demographic biases in data.

Current Tren

1. Contextual word
The nail hit the b
They reflected a b

2. Embeddings c
changes in word mea



Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings

Tolga Bolukbasi[1], Kai-Wei Chang[2], James Zou[2], Venkatesh Saligrama[1,2], Adam Kalai[2]
[1]Boston University, 8 Saint Mary's Street, Boston, MA
[2]Microsoft Research New England, 1 Memorial Drive, Cambridge, MA
tolgab@bu.edu, kw@kwchang.net, jamesyzou@gmail.com, srv@bu.edu, adam.kalai@microsoft.com

Abstract

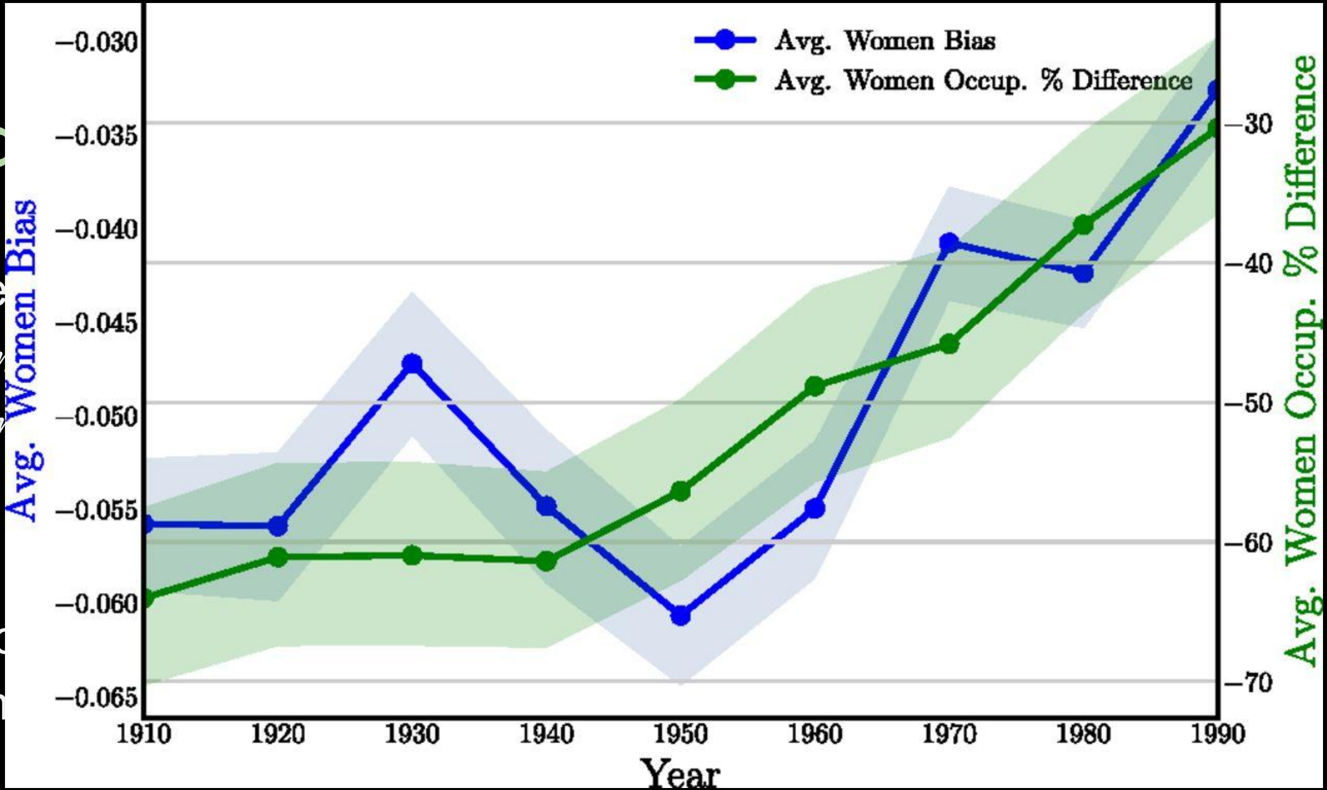
The blind application of machine learning runs the risk of amplifying biases present in data. Such a danger is facing us with *word embedding*, a popular framework to represent text data as vectors which has been used in many machine learning and natural language processing tasks. We show that even word embeddings trained on Google News articles exhibit female/male gender stereotypes to a disturbing extent. This raises concerns because their widespread use, as we describe, often tends to amplify these biases. Geometrically, gender bias is first shown to be captured by a direction in the word embedding. Second, gender neutral words are shown to be linearly separable from gender definition words in the word embedding. Using these properties, we provide a methodology for modifying an embedding to remove gender stereotypes, such as the association between the words *receptionist* and

3. Embeddings capture demographic biases in data.
    a. Efforts to debias
    b. Useful for tracking bias over time. (Garg et al., 2018)

Current Trenc

1. Contextual word e
   The nail hit the bear
   They reflected a bear

2. Embeddings can c
   changes in word n



3. Embeddings capture demographic biases in data.
   a. Efforts to debias
   b. Useful for tracking bias over time.

(Garg et al., 2018)

Vector Semantics and Embeddings

Take-Aways

- Dense representation of meaning is desirable.
- Approach 1: Dimensionality reduction techniques
- Approach 2: Learning representations by trying to predict held-out words.
- Word2Vec skipgram model attempts to solve by predicting target word from context word:
  maximize similarity between true pairs; minimize similarity between random pairs.
- Embeddings do in fact seem to capture meaning in applications
- Dimensionality reduction techniques just as good by some evaluations.
- Current Trends: Integrating context, Tracking changes in meaning.