# Distributional Semantics

CS 585, Fall 2017

Introduction to Natural Language Processing

http://people.cs.umass.edu/~brenocon/inlp2017

Brendan O'Connor
College of Information and Computer Sciences
University of Massachusetts Amherst

*[Slides from SLP3/Dan Jurafsky and David Belanger]*

# Why vector models of meaning? computing the similarity between words

"**fast**" is similar to "**rapid**"

"**tall**" is similar to "**height**"

Question answering:

*Q: "How **tall** is Mt. Everest?"*

*Candidate A: "The official **height** of Mount Everest is 29029 feet"*

# 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**

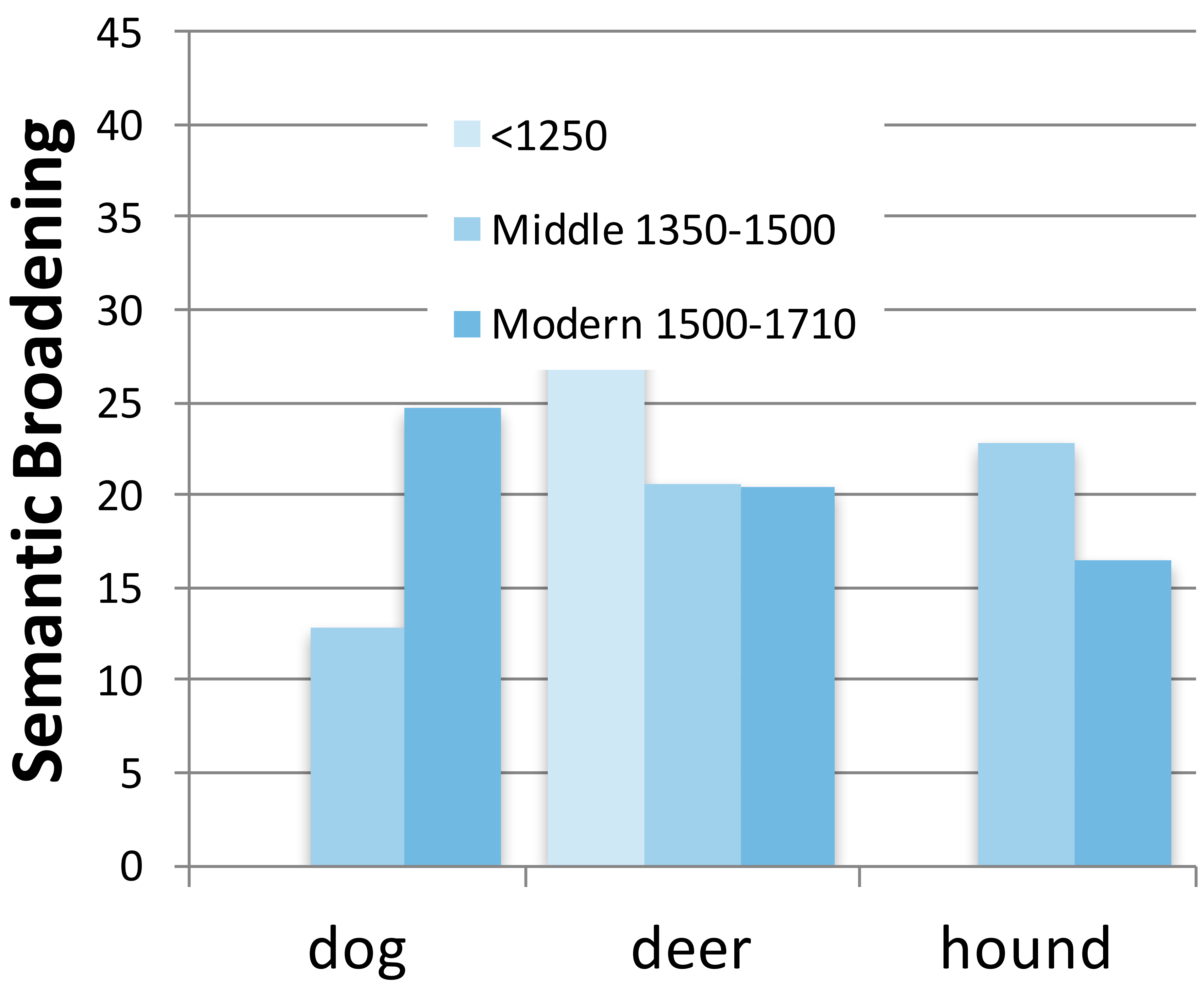
Mainframes usually are referred those computers with fast, advanced processing capabilities that could perform by itself tasks that may require a lot of Personal Computers (PC) Machines. Usually mainframes would have lots of RAMs, very large secondary storage devices, and very fast processors to cater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

# Word similarity for historical linguistics: semantic change over time

Kulkarni, Al-Rfou, Perozzi, Skiena 2015

Sagi, Kaufmann Clark 2013




# 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.”

# 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

- Nida example:

```
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

- Nida example:

```
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.

Question:

What do ‘art’ and ‘pharmaceuticals’ have in common?

What are contexts that they would both have?
What are contexts that they wouldn’t share?

https://brenocon.com/blog/2009/09/seeing-how-art-and-pharmaceuticals-are-linguistically-similar-in-web-text/

# Comparing Context Vectors

| common contexts for "art" but not "pharmaceuticals" [7394 total] | common contexts for both "art" and "pharmaceuticals" [165 total] | common contexts for "pharmaceuticals" but not "art" [206 total] |
|---|---|---|
| 'm into _ | areas such as _ | a greater amount of _ |
| 's interested in _ | prices of _ | standards for _ |
| A collection of _ | storage of _ | marketer of _ |
| _ has been described by | producers of _ | market for _ |
| structure of _ | _ designed for | prescriptions for _ |
| study in _ | the provision of _ | the supply of _ |
| _ have been shown in | _ sold in | the availability of _ |
| The knowledge of _ | the same way as _ | advertising for _ |
| _ is a commodity | _ are among | the appropriate use of _ |
| _ is a creation | The production of _ | shipment of _ |
| _ is a world | the analysis of _ | a cocktail of _ |
| an exhibition of _ | advances in _ | classes of _ |
| the commercialization of _ | specialising in _ | a complete inventory of _ |
| the confinement of _ | a career in _ | _ related downloads |
| _ is cast in | _ stolen from | new generations of _ |

https://brenocon.com/blog/2009/09/seeing-how-art-and-pharmaceuticals-are-linguistically-similar-in-web-text/

# Four kinds of vector models

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

# Shared intuition

# Shared intuition

- Model the meaning of a word by “embedding” in a vector space.

# Shared intuition

- Model the meaning of a word by “embedding” in a vector space.
- The meaning of a word is a vector of numbers
  - Vector models are also called “**embeddings**”.

# Shared intuition

- Model the meaning of a word by "embedding" in a vector space.
- The meaning of a word is a vector of numbers
  - Vector models are also called "**embeddings**".
- Contrast: word meaning is represented in many computational linguistic applications by a vocabulary index ("word number 545")

# Shared intuition

- Model the meaning of a word by “embedding” in a vector space.
- The meaning of a word is a vector of numbers
  - Vector models are also called “**embeddings**”.
- Contrast: word meaning is represented in many computational linguistic applications by a vocabulary index (“word number 545”)
- Old philosophy joke:

  Q: What’s the meaning of life?

  A: LIFE’

# Term-document matrix

- Each cell: count of term *t* in a document *d*: $\mathrm{tf}_{t,d}$:
  - Each document is a count vector in $\mathbb{N}^{v}$: a column below

| | As You Like It | Twelfth Night | Julius Caesar | Henry V |
|---|---|---|---|---|
| battle | 1 | 1 | 8 | 15 |
| soldier | 2 | 2 | 12 | 36 |
| fool | 37 | 58 | 1 | 5 |
| clown | 6 | 117 | 0 | 0 |

# Term-document matrix

- Two documents are similar if their vectors are similar

| | As You Like It | Twelfth Night | Julius Caesar | Henry V |
|---|---|---|---|---|
| battle | 1 | 1 | 8 | 15 |
| soldier | 2 | 2 | 12 | 36 |
| fool | 37 | 58 | 1 | 5 |
| clown | 6 | 117 | 0 | 0 |

# Term-document matrix

- Two documents are similar if their vectors are similar

| | As You Like It | Twelfth Night | Julius Caesar | Henry V |
|---|---|---|---|---|
| battle | 1 | 1 | 8 | 15 |
| soldier | 2 | 2 | 12 | 36 |
| fool | 37 | 58 | 1 | 5 |
| clown | 6 | 117 | 0 | 0 |

# Term-document matrix

- Two documents are similar if their vectors are similar

| | As You Like It | Twelfth Night | Julius Caesar | Henry V |
|---|---|---|---|---|
| battle | 1 | 1 | 8 | 15 |
| soldier | 2 | 2 | 12 | 36 |
| fool | 37 | 58 | 1 | 5 |
| clown | 6 | 117 | 0 | 0 |

- Embedding documents in a word space
  - not our main goal here, but quite common: e.g. in IR

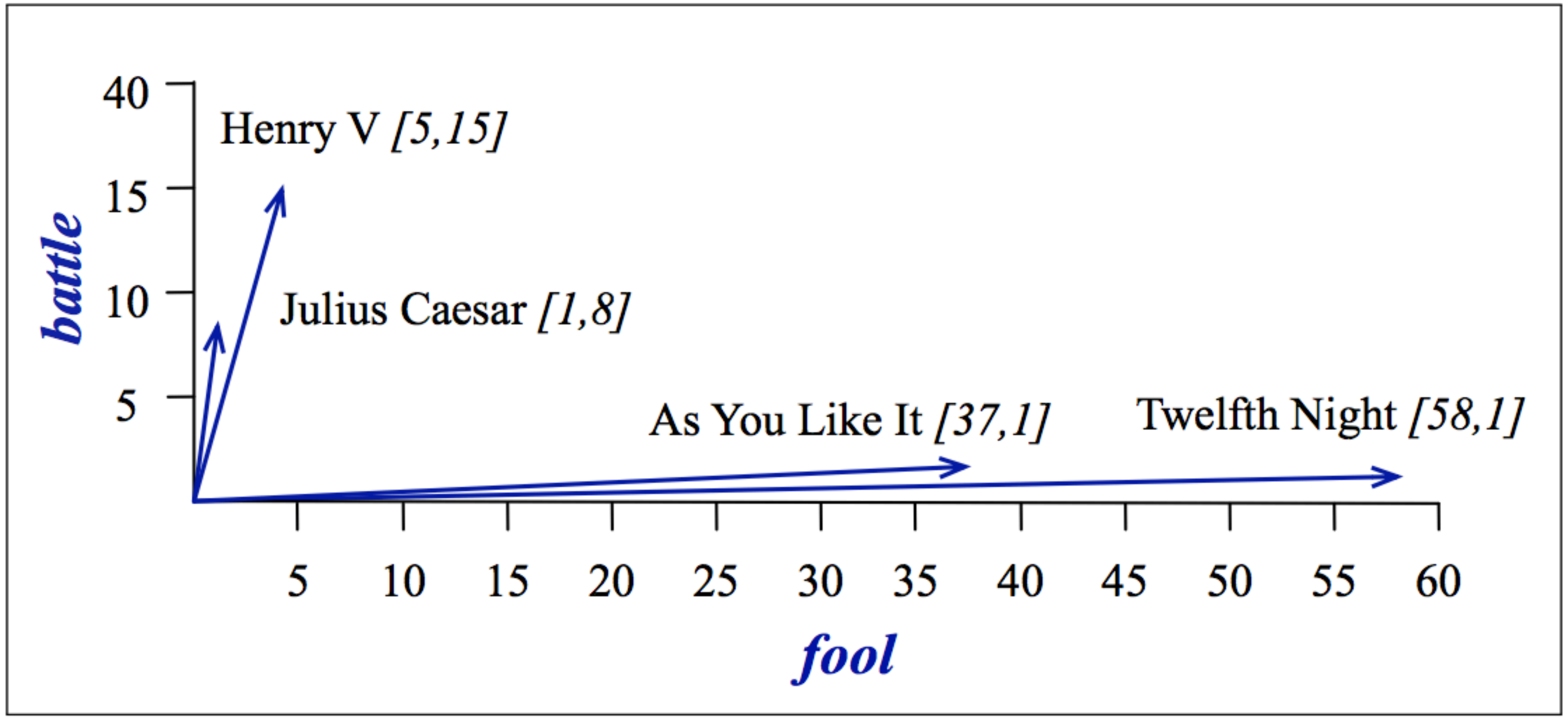



**Figure 15.3** A spatial visualization of the document vectors for the four Shakespeare play documents, showing just two of the dimensions, corresponding to the words *battle* and *fool*. The comedies have high values for the *fool* dimension and low values for the *battle* dimension.

# The words in a term-document matrix

- Each word is a count vector in $\mathbb{N}^D$: a row below

| | As You Like It | Twelfth Night | Julius Caesar | Henry V |
|---|---|---|---|---|
| battle | 1 | 1 | 8 | 15 |
| soldier | 2 | 2 | 12 | 36 |
| fool | 37 | 58 | 1 | 5 |
| clown | 6 | 117 | 0 | 0 |

# The words in a term-document matrix

- Each word is a $\boxed{\text{count vector}}$ in $\mathbb{N}^D$: a row below

| | As You Like It | Twelfth Night | Julius Caesar | Henry V |
|---|---|---|---|---|
| battle | 1 | 1 | 8 | 15 |
| soldier | 2 | 2 | 12 | 36 |
| fool | 37 | 58 | 1 | 5 |
| clown | 6 | 117 | 0 | 0 |

# The words in a term-document matrix

- Two **words** are similar if their vectors are similar

| | As You Like It | Twelfth Night | Julius Caesar | Henry V |
|---|---|---|---|---|
| battle | 1 | 1 | 8 | 15 |
| soldier | 2 | 2 | 12 | 36 |
| fool | 37 | 58 | 1 | 5 |
| clown | 6 | 117 | 0 | 0 |

# The words in a term-document matrix

- Two **words** are similar if their vectors are similar

| | As You Like It | Twelfth Night | Julius Caesar | Henry V |
|---|---|---|---|---|
| battle | 1 | 1 | 8 | 15 |
| soldier | 2 | 2 | 12 | 36 |
| fool | 37 | 58 | 1 | 5 |
| clown | 6 | 117 | 0 | 0 |

# The words in a term-document matrix

- Two **words** are similar if their vectors are similar

| | As You Like It | Twelfth Night | Julius Caesar | Henry V |
|---|---|---|---|---|
| battle | 1 | 1 | 8 | 15 |
| soldier | 2 | 2 | 12 | 36 |
| fool | 37 | 58 | 1 | 5 |
| clown | 6 | 117 | 0 | 0 |

# Term-context matrix for word similarity

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

| | 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 | |

# Term-context matrix for word similarity

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

| | 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 | |

# Term-context matrix for word similarity

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

| | 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 | |

# Term-context matrix for word similarity

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

| | 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 | |

# Term-context matrix for word similarity

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

| | 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 | |

# The word-word or word-context matrix

- Instead of entire documents, use smaller contexts
  - Paragraph
  - Window of $\pm$ 4 words
- A word is now defined by a vector over counts of context words
- Instead of each vector being of length D
- Each vector is now of length |V|
- The word-word matrix is |V|x|V|

# Word-Word matrix
# Sample contexts ± 7 words

| | | |
|---|---|---|
| sugar, a sliced lemon, a tablespoonful of | **apricot** | preserve or 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 | |
| … | … | | | | | | |

# Word-Word matrix
# Sample contexts $\pm$ 7 words

| | | |
|---|---|---|
| sugar, a sliced lemon, a tablespoonful of | **apricot** | preserve or 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 | |
| … | … | | | | | | |

# Word-Word matrix
# Sample contexts $\pm$ 7 words

| | | |
|---|---|---|
| sugar, a sliced lemon, a tablespoonful of | **apricot** | preserve or 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 | |
| … | … | | | | | | |

# Word-Word matrix
# Sample contexts $\pm$ 7 words

| | | |
|---|---|---|
| sugar, a sliced lemon, a tablespoonful of | **apricot** | preserve or 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 | |
| … | … | | | | | | |

# Word-Word matrix
# Sample contexts ± 7 words

| | | | |
|---|---|---|---|
| sugar, a sliced lemon, a tablespoonful of | **apricot** | preserve or 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 | |
| … | … | | | | | | |

# Word-word matrix

- We showed only 4x6, but the real matrix is 50,000 x 50,000
  - So it's very **sparse**
    - Most values are 0.
  - That's OK, since there are lots of efficient algorithms for sparse matrices.
- The size of windows depends on your goals
  - The shorter the windows , the more **syntactic** the representation

    $\pm$ 1-3 very syntacticy
  - The longer the windows, the more **semantic** the representation

    $\pm$ 4-10 more semanticy

# 2 kinds of co-occurrence between 2 words

(Schütze and Pedersen, 1993)

- First-order co-occurrence (**syntagmatic association**):
  - They are typically nearby each other.
  - *wrote* is a first-order associate of *book* or *poem*.
- Second-order co-occurrence (**paradigmatic association**):
  - They have similar neighbors.
  - *wrote* is a second- order associate of words like *said* or *remarked*.

*which gets **syntactic** sim?  which gets **topical** sim?*

# Distributional similarity!!

- stopped here 10/31

- Midterm pickup -- my office after class, or any office hours next week
- Small HW3 - co-occurrence and distributional similarity

# Distributional similarity
# “Paradigmatic”

- 1. Represent a word as a context vector
  - of frequencies, or better, positive-only PMI
- 2. Calculate word-to-word similarity as a function of two vectors
- (3. Reduce dimensionality of context vectors)

# Problem with raw counts

- Raw word frequency is not a great measure of association between words
  - It’s very skewed
    - “the” and “of” are very frequent, but maybe not the most discriminative
- We’d rather have a measure that asks whether a context word is **particularly informative** about the target word.
  - Positive Pointwise Mutual Information (PPMI)

# Pointwise mutual information

- Do words **w** and **c** occur more often than if they were independent?
- Does **c** occur more often around w, relative to its baseline frequency?

$$\mathrm{PMI}(w,c) = \log \frac{P(w,c)}{P(w)P(c)} = \log \frac{P(c \mid w)}{P(c)}$$

# Issues with PMI

- Words with small counts
  - Easy to get very high PMI scores
  - Solution: Frequency thresholding. Only use words/contexts with e.g. count(w) >= 20
    - You can’t learn about rare words very well anyway...

# Issues with PMI

- Words with small counts
  - Easy to get very high PMI scores
  - Solution: Frequency thresholding. Only use words/contexts with e.g. count(w) >= 20
    - You can't learn about rare words very well anyway...

- Positive PMI
  - Negative PMI scores are weird
    - Hard to assess without large corpora
    - Is "unrelatedness" meaningful?
  - Solution: just clip negative values. Works well for dist. sim., at least.
    - max(0,z) = "positive part" or "rectified linear unit" function

$$\mathrm{PPMI} = \max\left(0,\ \log \frac{P(x,y)}{P(x)P(y)}\right)$$

# Measuring similarity

- Given 2 target words *v* and *w*
- We’ll need a way to measure their similarity.
- Most measure of vectors similarity are based on the:
- **Dot product** or **inner product** from linear algebra

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

  - High when two vectors have large values in same dimensions.
  - Low (in fact 0) for **orthogonal vectors** with zeros in complementary
31 distribution

# Solution: cosine

- Just divide the dot product by the length of the two vectors!

$$\frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|}$$

- This turns out to be the cosine of the angle between them!

$$\begin{aligned} \vec{a} \cdot \vec{b} &= |\vec{a}||\vec{b}| \cos \theta \\ \frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|} &= \cos \theta \end{aligned}$$

# Cosine

Dot product

Unit vectors

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

$v_i$ is the PPMI value for word $v$ in context $i$
$w_i$ is the PPMI value for word $w$ in context $i$.

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

Dot product

Unit vectors

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

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

Which pair of words is more similar?

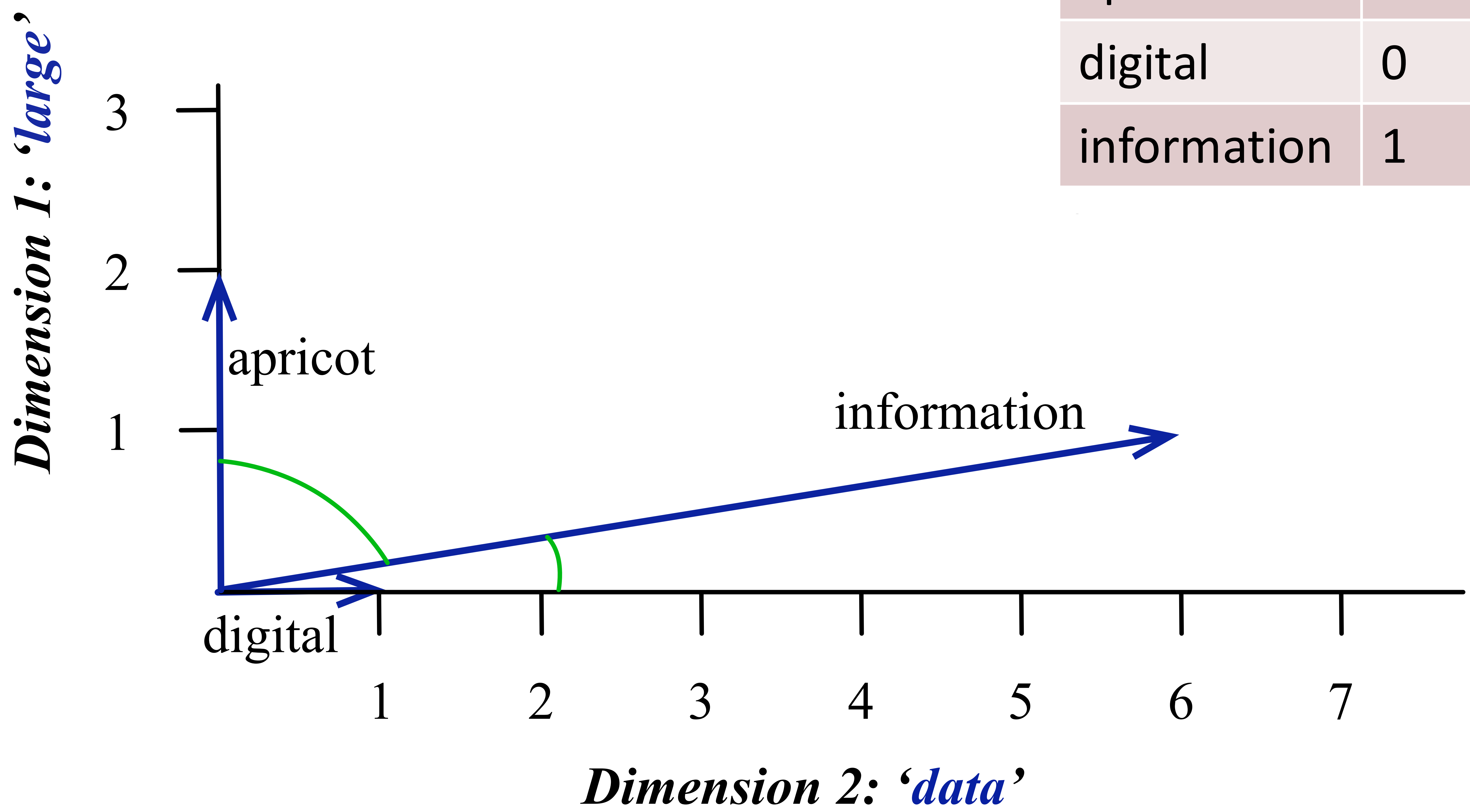
cosine(apricot,information) = $\frac{2+0+0}{\sqrt{2+0+0}\sqrt{1+36+1}} = \frac{2}{\sqrt{2}\sqrt{38}} = .23$

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

# Visualization

| | large | data |
|---|---|---|
| apricot | 2 | 0 |
| digital | 0 | 1 |
| information | 1 | 6 |




7

# WORD EMBEDDINGS

# <u>Latent</u> (reduced-dim) word embeddings

Sparse Context Vector (10 million+ dimensional):

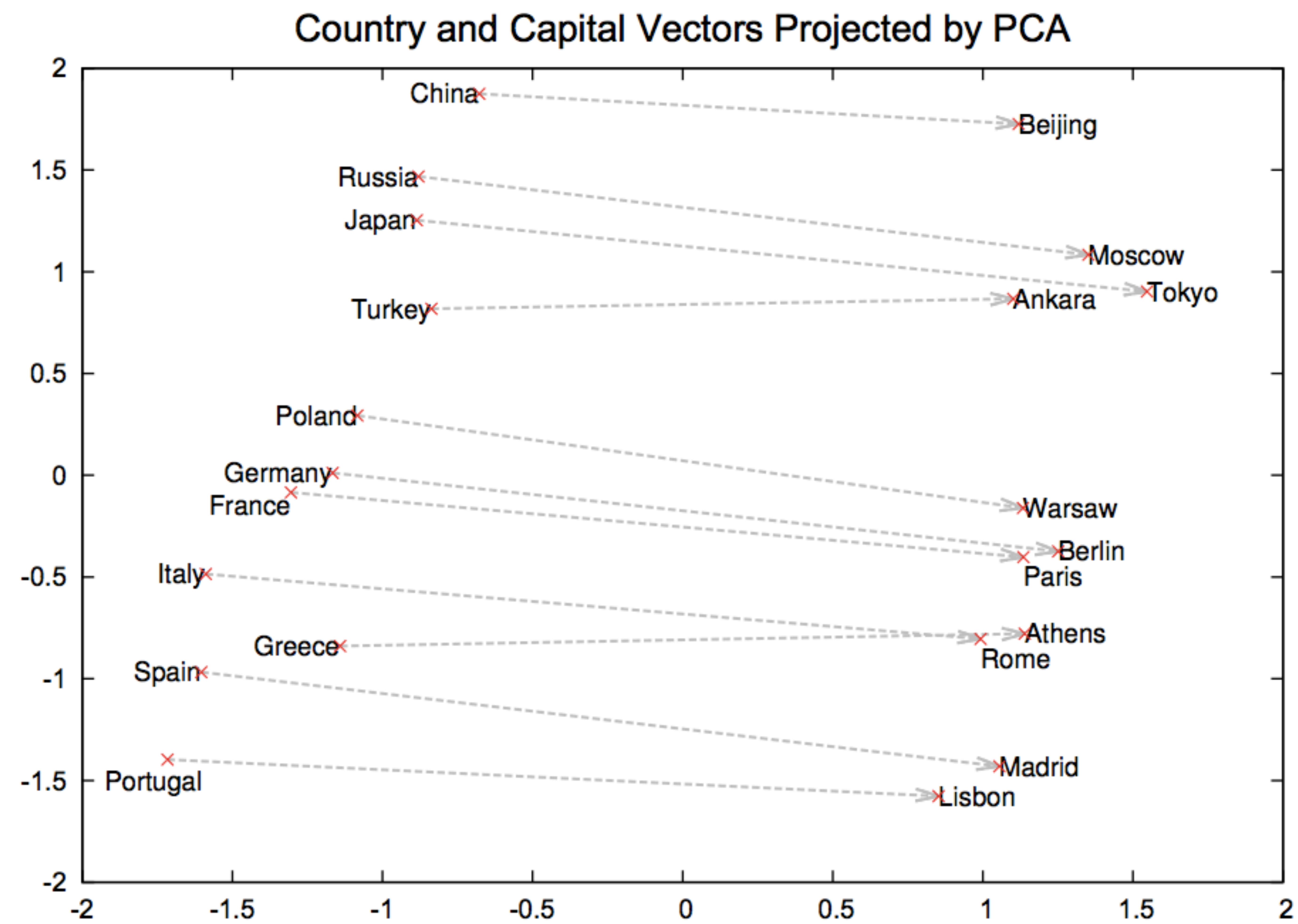
$$V_i = \boxed{[0, 1, 0, 0, 0, 4, 0, 0, 0, 2, 0, 0, 1, \ldots]}$$

[This can be directly used, but maybe too slow, sparse]

Instead represent every word type as a low-dimensional dense vector (about 100 dimensional ).

$$E_i = \boxed{[.253, 458, 4.56, 78.5, 120, \ldots]}$$

These don't come directly from the data. They need to be learned.


Country and Capital Vectors Projected by PCA
China
Beijing
Russia
Japan
Moscow
Tokyo
Turkey
Ankara
Poland
Germany
France
Warsaw
Berlin
Paris
Italy
Greece
Athens
Rome
Spain
Madrid
Portugal
Lisbon
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

# Nearest Neighbors

- deals --> checks approvals vents stickers cuts
- warned --> suggested speculated predicted stressed argued
- ability --> willingness inability eagerness disinclination desire
- dark --> comfy wild austere cold tinny
- possibility --> possiblity possibilty dangers notion likelihood

# Nearest Neighbors

- deals --> checks approvals vents stickers cuts
- warned --> suggested speculated predicted stressed argued
- ability --> willingness inability eagerness disinclination desire
- dark --> comfy wild austere cold tinny
- possibility --> possiblity possibilty dangers notion likelihood

Question:

What are the pros and cons of representing word types with such small vectors?

Pro:

It requires less annotated data to train an ML model on low dimensional features.

Con:

You can't capture all of the subtlety of language in 100 dimensions. (...can you?)

# Learning Embeddings by Preserving Similarity

- Given long, sparse context cooccurrence vectors $V_i$ and $V_j$

- Goal: Choose Embeddings $E_i$ and $E_j$ such that similarity is approximately preserved

$$V_i^\top V_j \approx E_i^\top E_j$$

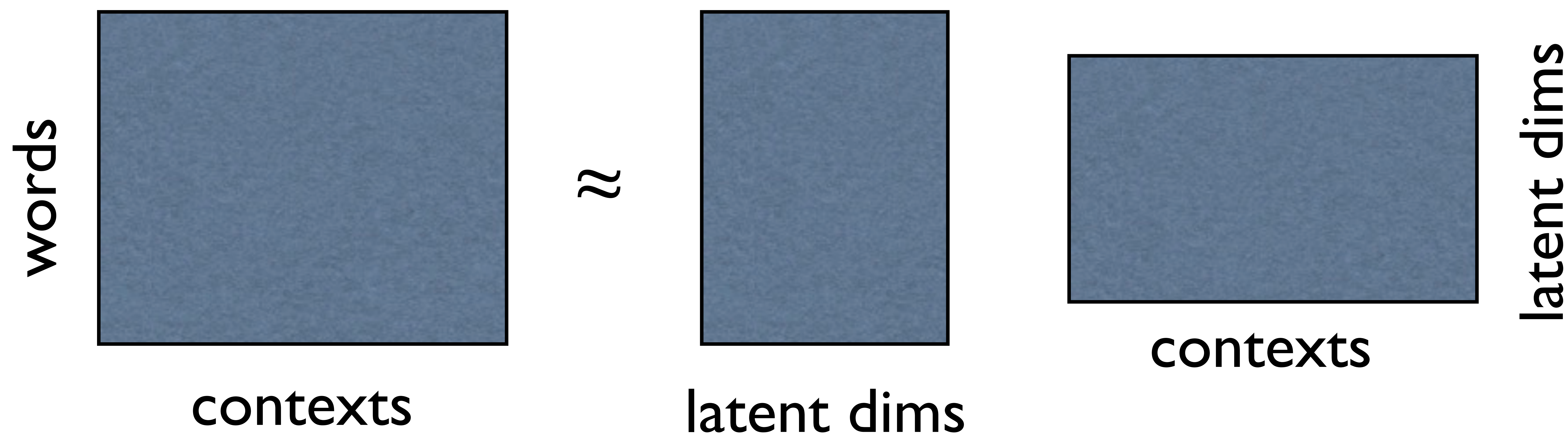
For all words jointly?

Use eigendecomposition / singular value decomposition / matrix factorization

# Matrix factorization

V ≈ E B

(counts) (embeddings)




Reconstruct the co-occurrence matrix

$$V_{i,c} \approx \sum_k E_{i,k} B_{k,c}$$

Singular Value Decomposition learns E,B
(or other matrix factorization techniques)

Preserve pairwise distances between words i, j

$$V_i^\intercal V_j \approx E_i^\intercal E_j$$

Eigen Decomposition learns E

- “Distributional / Word Embedding” models
  - Typically, they learn embeddings to be good at word-context factorization, which seems to often give useful embeddings
  - e.g.: *word2vec* (Mikolov): fast software! Views problem as context prediction
- Pre-trained embeddings resources
  - *GLOVE*, *word2vec*, etc.
  - Make sure it’s trained on a corpus sufficiently similar to what you care about!
- How to use?
  - Similarity lookups
  - Latent dimensions as features for model
    (though works better in neural, not linear, models...)

# Extensions

- Alternative: Task-specific embeddings (always better...)

- Multilingual embeddings
- Better contexts: direction, syntax, morphology / characters...
- Phrases and meaning composition
  - vector(hardly awesome) = g(vector(hardly), vector(awesome))