CS 6120/CS4120: Natural Language Processing

Instructor: Prof. Lu Wang
College of Computer and Information Science
Northeastern University
Webpage: www.ccs.neu.edu/home/luwang

Outline

- Vector Semantics
- Sparse representation
  - Pointwise Mutual Information (PMI)
- Dense representation
  - Singular Value Decomposition (SVD)
  - Neural Language Model (Word2Vec)
  - Brown cluster

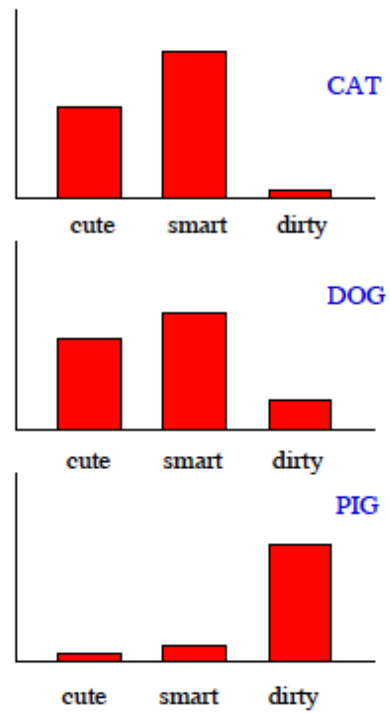
Why vector models of meaning? computing the similarity between words

“**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”*


CAT
cute
smart
dirty
DOG
cute
smart
dirty
PIG
cute
smart
dirty

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

Sample Word-Word 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	
…	…						

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

Pointwise mutual information:

Do events x and y co-occur more than if they were independent?

$$\text{PMI}(X,Y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

PPMI(w,context)

	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pineapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.57	-	0.47	-

PPMI versus add-2 smoothed PPMI

$$\text{PMI}(X,Y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

PPMI(w,context)

	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pineapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.57	-	0.47	-

PPMI(w,context) [add-2]

	computer	data	pinch	result	sugar
apricot	0.00	0.00	0.56	0.00	0.56
pineapple	0.00	0.00	0.56	0.00	0.56
digital	0.62	0.00	0.00	0.00	0.00
information	0.00	0.58	0.00	0.37	0.00

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 (raw counts)

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

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

Sec. 6.3

Cosine for computing similarity

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.

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

Using syntax to define a word's context

- Zellig Harris (1968)

  "The meaning of entities, and the meaning of grammatical relations among them, is related to the restriction of combinations of these entities relative to other entities"
- **Two words are similar if they have similar syntactic contexts**

Duty and **responsibility** have similar syntactic distribution:

Modified by adjectives	additional, administrative, assumed, collective, congressional, constitutional …
Objects of verbs	assert, assign, assume, attend to, avoid, become, breach..

Co-occurrence vectors based on syntactic dependencies

Dekang Lin, 1998 "Automatic Retrieval and Clustering of Similar Words"

- Each dimension: a context word in one of R grammatical relations
  - Consider word "cell", and phrase "cell absorbs nutrients"
  - **Subject-of**- "***absorb***"
- Instead of a vector of $|V|$ features, a vector of $R|V|$

- Each dimension: a context word in one of R grammatical relations
  - Consider word "cell"
  - **Subject-of**- "***absorb***"
- Instead of a vector of *|V|* features, a vector of *R|V|*
- Example: counts for the word *cell* :

cell	
1	subj-of, absorb
1	subj-of, adapt
1	subj-of, behave
	⋮
16	pobj-of, inside
30	pobj-of, into
	⋮
3	nmod-of, abnormality
8	nmod-of, anemia
1	nmod-of, architecture
	⋮
6	obj-of, attack
11	obj-of, call

Syntactic dependencies for dimensions

- Alternative (Padó and Lapata 2007):
  - Instead of having a |V| x R|V| matrix
  - Have a |V| x |V| matrix
  - Counts of words that occur in one of R dependencies (subject, object, etc).
  - So M(“cell”,”absorb”) =

  count(subj(cell,absorb))

  + count(obj(cell,absorb))

  + count(pobj(cell,absorb))+…

PMI applied to dependency relations

Hindle, Don. 1990. Noun Classification from Predicate-Argument Structure. ACL

Object of "drink"	Count	PMI
tea	2	11.8
liquid	2	10.5
wine	2	9.3
anything	3	5.2
it	3	1.3

- "`Drink it`" more common than "`drink wine`"
- But "`wine`" is a better "drinkable" thing than "`it`"

Alternative to PPMI for measuring association

- Recall that we studied tf-idf…
- The combination of two factors
  - **Term frequency** (Luhn 1957): frequency of the word (can be logged)
  - **Inverse document frequency** (IDF) (Spark Jones 1972)
    - N is the total number of documents
    - df_i = "document frequency of word *i*"
    - = number of documents with word *I*

$$\mathrm{idf}_i = \log\left(\frac{N}{df_i}\right)$$

  - w_{ij} *: for word i in document j*

$$w_{ij} = tf_{ij} \cdot idf_i$$

tf-idf not generally used for word-word similarity

- But is by far the most common weighting when we are considering the relationship of words to documents

Evaluating similarity (Revisit)

- Extrinsic (task-based, end-to-end) Evaluation:
  - Question Answering
  - Spell Checking
  - Essay grading
- Intrinsic Evaluation:
  - Correlation between algorithm and human word similarity ratings
    - Wordsim353: 353 noun pairs rated 0-10. *sim(plane,car)=5.77*
  - Taking TOEFL multiple-choice vocabulary tests
    - `Levied is closest in meaning to:`
      `imposed, believed, requested, correlated`

Summary

- Distributional (vector) models of meaning
  - **Sparse** (PPMI-weighted word-word co-occurrence matrices)
  - **Dense**:
    - Word-word SVD (50-2000 dimensions)
    - Skip-grams and CBOW (100-1000 dimensions)

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

Three methods for getting short dense vectors

- Singular Value Decomposition (SVD)
- “Neural Language Model” – inspired by predictive models
- Brown clustering

Singular Value Decomposition (SVD)

Rank of a Matrix

- What is the rank of a matrix A?

Rank of a Matrix

- What is the rank of a matrix A?
- Number of linearly independent columns of A

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix}$$

Rank of a Matrix

- What is the rank of a matrix A?
- Number of linearly independent columns of A

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix}$$

- Rank is 2
- We can rewrite A as two “basis” vectors: [1 2 1] [-2 -3 1]

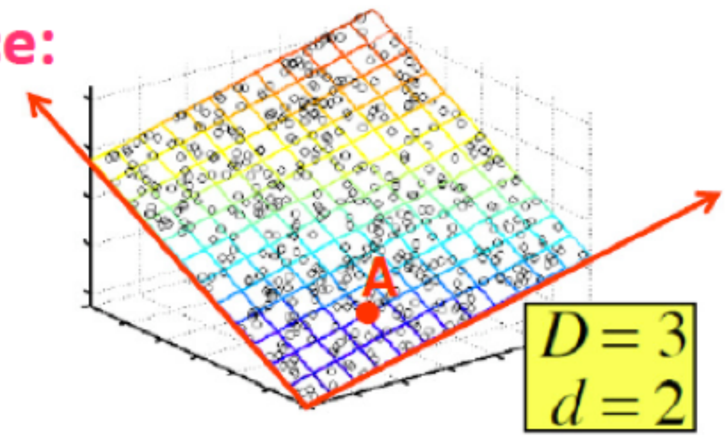
Rank as "Dimensionality"

Cloud of points 3D space:

- Think of point positions as a matrix:

$$\begin{array}{l c}
 & \begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix} \begin{array}{c} \text{A} \\ \text{B} \\ \text{C} \end{array}
\end{array}$$

1 row per point:

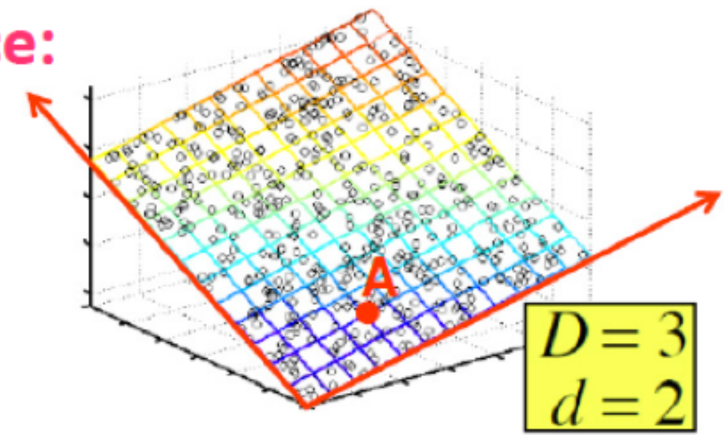


Rank as “Dimensionality”

Cloud of points 3D space:

- Think of point positions as a matrix:

1 row per point:
$$\begin{array}{c} \begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix} \end{array} \begin{array}{l} A \\ B \\ C \end{array}$$

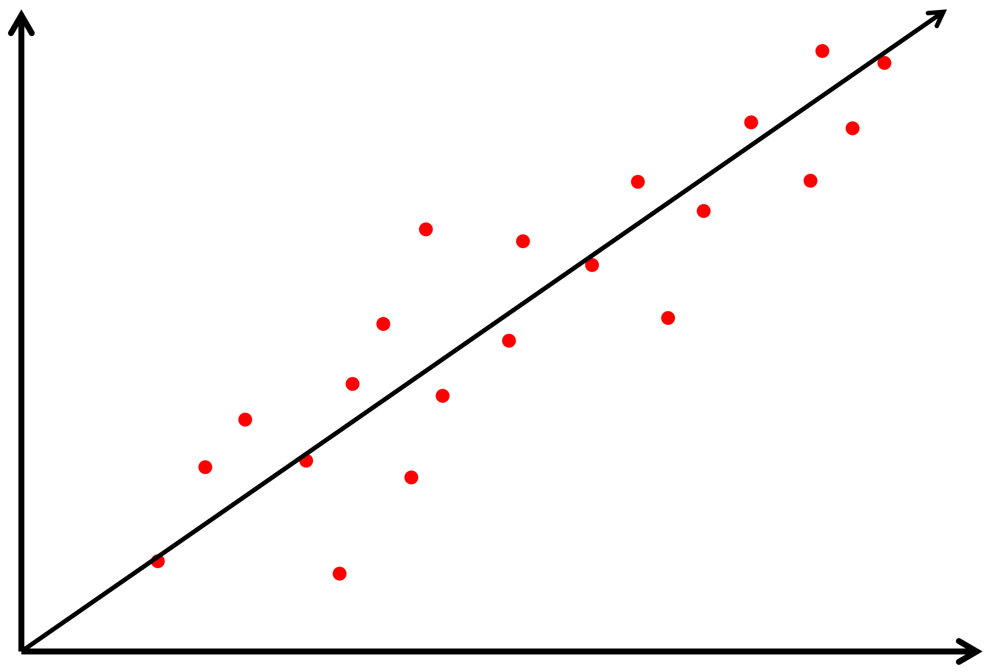


- Rewrite the coordinates in a more efficient way!
  - Old basis vectors: [1 0 0], [0 1 0], [0 0 1]
  - New basis vectors: [1 2 1], [-2 -3 1]

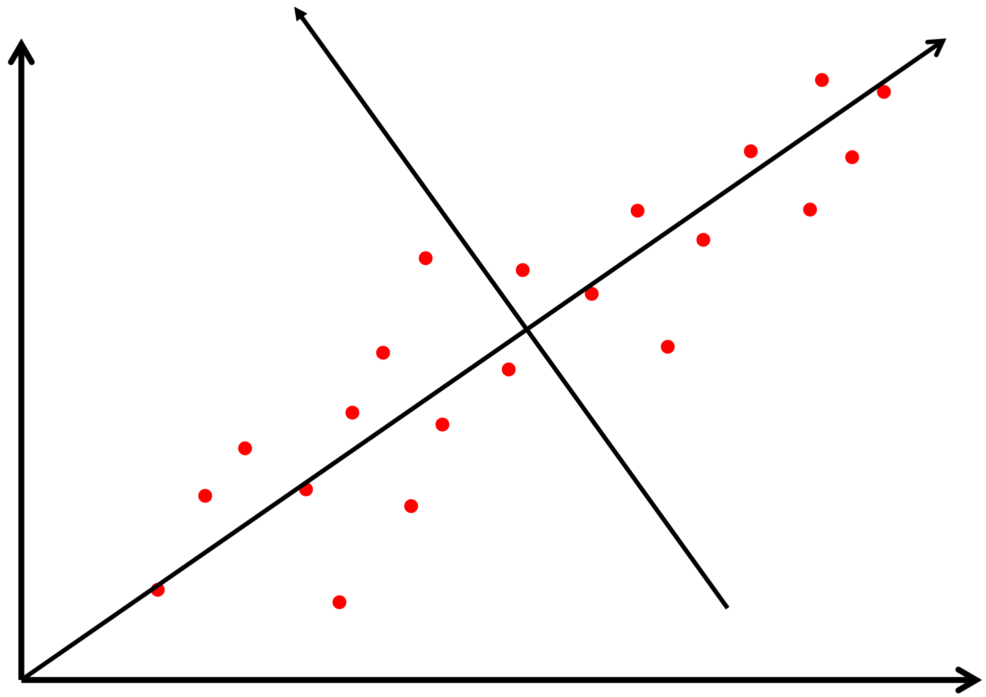
Intuition of Dimensionality Reduction

- Approximate an N-dimensional dataset using fewer dimensions
- By first rotating the axes into a new space
- In which the highest order dimension captures the most variance in the original dataset
- And the next dimension captures the next most variance, etc.

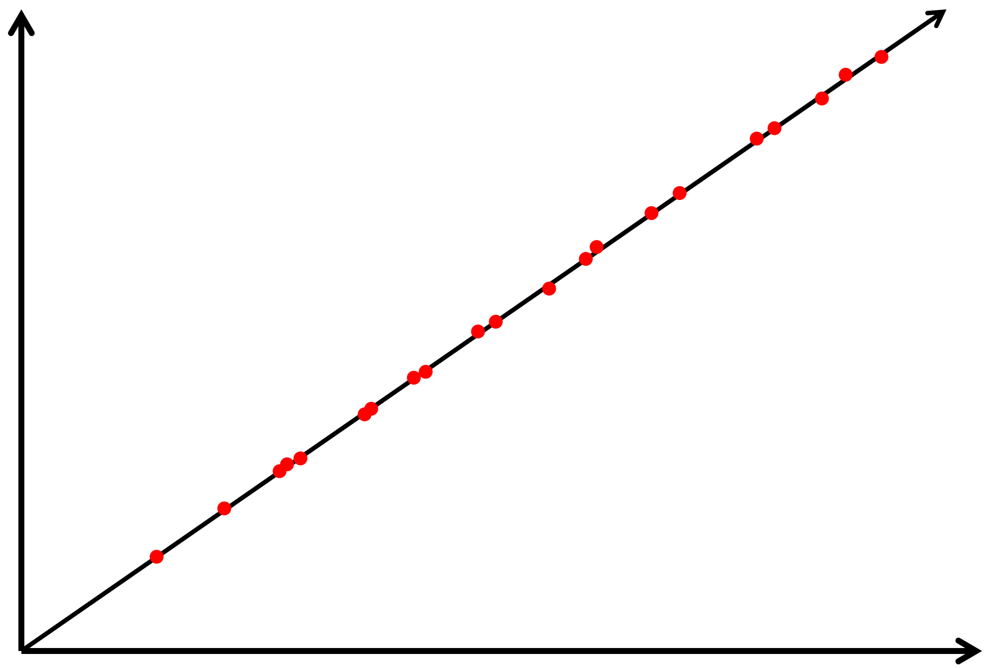
Sample Dimensionality Reduction



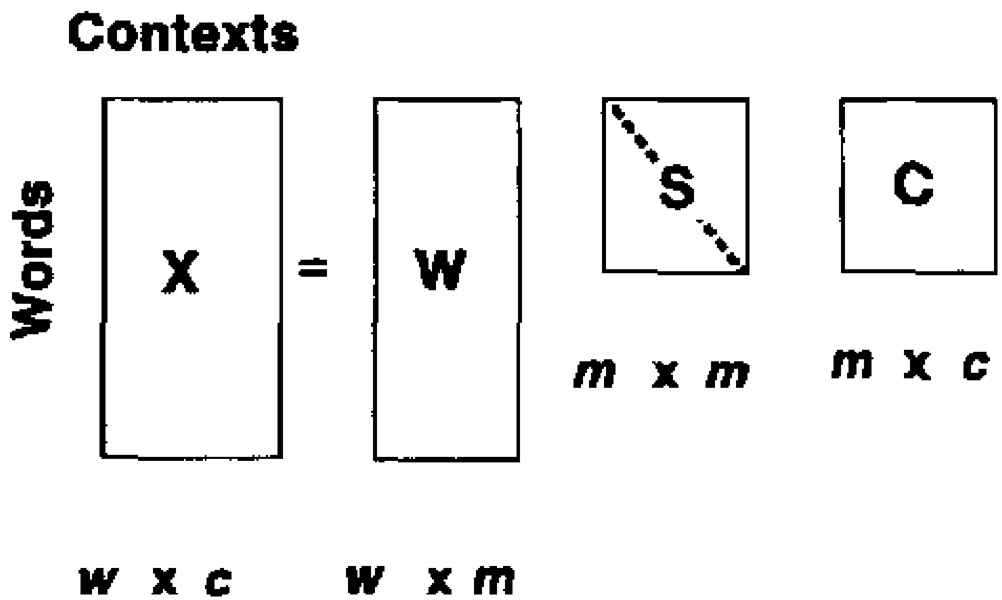
Sample Dimensionality Reduction



Sample Dimensionality Reduction



Singular Value Decomposition



(assuming the matrix has rank m)

Landuaer and Dumais 1997

Singular Value Decomposition

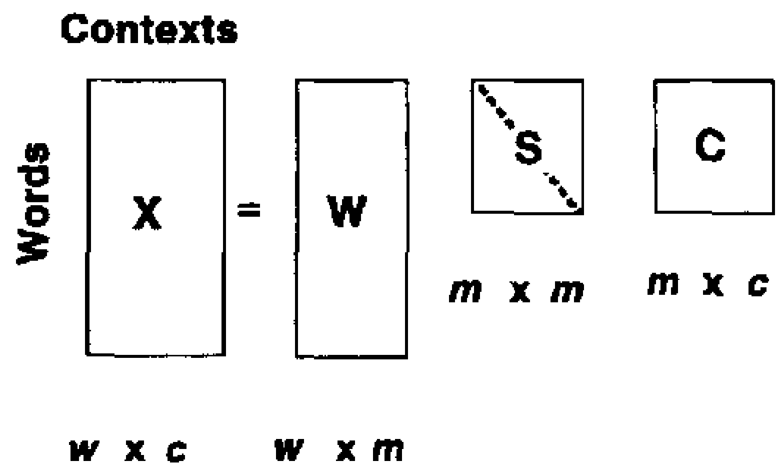
*Any rectangular w x c matrix **X** equals the product of 3 matrices:*

W: rows corresponding to original but m columns represents a dimension in a new latent space, such that

- m column vectors are orthogonal to each other
- Columns are ordered by the amount of variance in the dataset each new dimension accounts for

S: diagonal *m* x *m* matrix of **singular values** expressing the importance of each dimension.

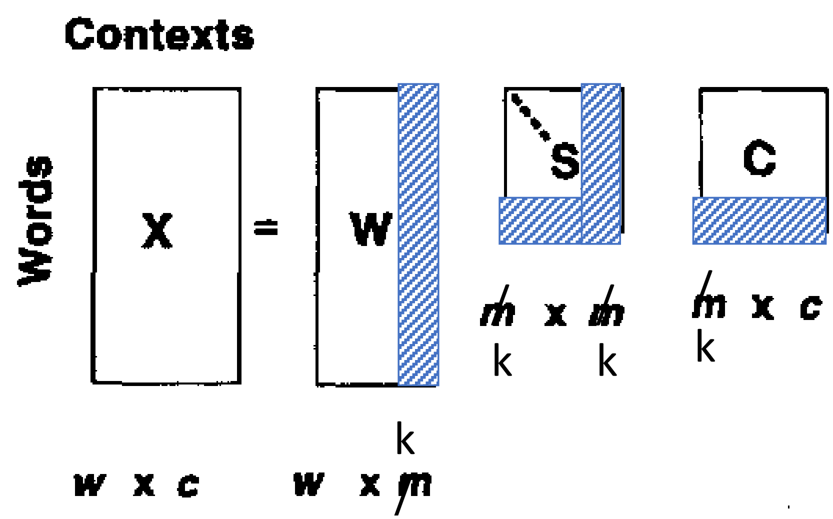
C: columns corresponding to original but m rows corresponding to singular values



SVD applied to term-document matrix: Latent Semantic Analysis

Deerwester et al (1988)

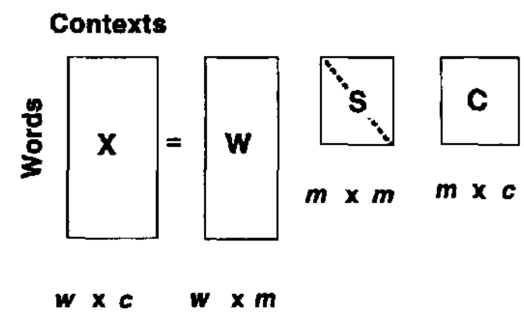
- If instead of keeping all m dimensions, we just keep the top k singular values. Let's say 300.
- The result is a least-squares approximation to the original X
- But instead of multiplying, we'll just make use of W.
- Each row of W:
  - A k-dimensional vector
  - Representing word W



SVD on Term-Document Matrix: Example

- The matrix X

	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1



Matrix **W**

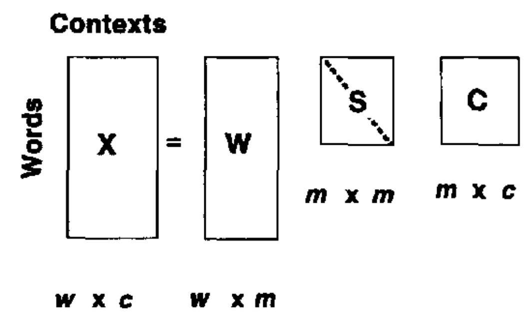
	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
wood	−0.70	0.35	0.15	−0.58	0.16
tree	−0.26	0.65	−0.41	0.58	−0.09

Matrix **S**

	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	1.28	0.00	0.00
4	0.00	0.00	0.00	1.00	0.00
5	0.00	0.00	0.00	0.00	0.39

Matrix **C**

	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.28	−0.75	0.45	−0.20	0.12	−0.33
4	0.00	0.00	0.58	0.00	−0.58	0.58
5	−0.53	0.29	0.63	0.19	0.41	−0.22



Matrix **W**

	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
wood	−0.70	0.35	0.15	−0.58	0.16
tree	−0.26	0.65	−0.41	0.58	−0.09

Matrix **S**

	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	1.28	0.00	0.00
4	0.00	0.00	0.00	1.00	0.00
5	0.00	0.00	0.00	0.00	0.39

Matrix **C**

	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.28	−0.75	0.45	−0.20	0.12	−0.33
4	0.00	0.00	0.58	0.00	−0.58	0.58
5	−0.53	0.29	0.63	0.19	0.41	−0.22

Reduce dimension: The Matrix W

	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
wood	−0.70	0.35	0.15	−0.58	0.16
tree	−0.26	0.65	−0.41	0.58	−0.09

	1	2	3	4	5
ship	−0.44	−0.30	0.00	0.00	0.00
boat	−0.13	−0.33	0.00	0.00	0.00
ocean	−0.48	−0.51	0.00	0.00	0.00
wood	−0.70	0.35	0.00	0.00	0.00
tree	−0.26	0.65	0.00	0.00	0.00

Reduce dimension: The Matrix S

1	2	3	4	5
2.16	0.00	0.00	0.00	0.00
0.00	1.59	0.00	0.00	0.00
0.00	0.00	1.28	0.00	0.00
0.00	0.00	0.00	1.00	0.00
0.00	0.00	0.00	0.00	0.39

$\Rightarrow$

1	2	3	4	5
2.16	0.00	0.00	0.00	0.00
0.00	1.59	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00

Reduce dimension: The Matrix C

d_1	d_2	d_3	d_4	d_5	d_6
−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
−0.29	−0.53	−0.19	0.63	0.22	0.41
0.28	−0.75	0.45	−0.20	0.12	−0.33
0.00	0.00	0.58	0.00	−0.58	0.58
−0.53	0.29	0.63	0.19	0.41	−0.22

d_1	d_2	d_3	d_4	d_5	d_6
−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
−0.29	−0.53	−0.19	0.63	0.22	0.41
0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00

Reduce dimension: The Matrix W

	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

	1	2	3	4	5
ship	−0.44	−0.30	0.00	0.00	0.00
boat	−0.13	−0.33	0.00	0.00	0.00
ocean	−0.48	−0.51	0.00	0.00	0.00
wood	−0.70	0.35	0.00	0.00	0.00
tree	−0.26	0.65	0.00	0.00	0.00

Reduce dimension: The Matrix W

	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

	1	2	3	4	5
ship	−0.44	−0.30	0.00	0.00	0.00
boat	−0.13	−0.33	0.00	0.00	0.00
ocean	−0.48	−0.51	0.00	0.00	0.00
wood	−0.70	0.35	0.00	0.00	0.00
tree	−0.26	0.65	0.00	0.00	0.00

Similarity between ***ship*** and ***boat*** ***vs*** ***ship*** and ***wood*** ?

Reduce dimension: The Matrix W

	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

	1	2	3	4	5
ship	−0.44	−0.30	0.00	0.00	0.00
boat	−0.13	−0.33	0.00	0.00	0.00
ocean	−0.48	−0.51	0.00	0.00	0.00
wood	−0.70	0.35	0.00	0.00	0.00
tree	−0.26	0.65	0.00	0.00	0.00

More details

- 300 dimensions are commonly used
- The cells are commonly weighted by a product of two weights (TF-IDF)
    - Local weight: Log term frequency
    - Global weight: either idf or an entropy measure

Let’s return to PPMI word-word matrices

- Can we apply to SVD to them?

SVD applied to term-term matrix

$$\underbrace{\begin{bmatrix} \\ \\ X \\ \\ \\ \end{bmatrix}}_{|V| \times |V|} = \underbrace{\begin{bmatrix} \\ \\ W \\ \\ \\ \end{bmatrix}}_{|V| \times |V|} \underbrace{\begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_V \end{bmatrix}}_{|V| \times |V|} \underbrace{\begin{bmatrix} \\ \\ C \\ \\ \\ \end{bmatrix}}_{|V| \times |V|}$$

(assuming the matrix has rank |V|, may not be true)

SVD applied to term-term matrix

$$\underbrace{\begin{bmatrix} \\ \\ X \\ \\ \\ \end{bmatrix}}_{|V| \times |V|} = \underbrace{\begin{bmatrix} \\ \\ W \\ \\ \\ \end{bmatrix}}_{|V| \times |V|} \underbrace{\begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_V \end{bmatrix}}_{|V| \times |V|} \underbrace{\begin{bmatrix} \\ \\ C \\ \\ \\ \end{bmatrix}}_{|V| \times |V|}$$

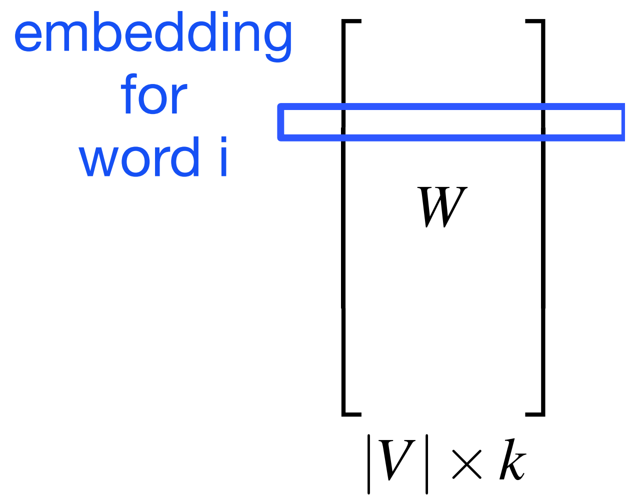
(assuming the matrix has rank |V|, may not be true)

Truncated SVD on term-term matrix

$$\underset{|V| \times |V|}{\begin{bmatrix} \\ \\ X \\ \\ \\ \end{bmatrix}} = \underset{|V| \times k}{\begin{bmatrix} \\ \\ W \\ \\ \\ \end{bmatrix}} \underset{k \times k}{\begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_k \end{bmatrix}} \underset{k \times |V|}{\begin{bmatrix} & C & \end{bmatrix}}$$

Truncated SVD produces embeddings

- Each row of W matrix is a k-dimensional representation of each word *w*
- K might range from 50 to 1000
- Generally we keep the top k dimensions, but some experiments suggest that getting rid of the top 1 dimension or even the top 50 dimensions is helpful (Lapesa and Evert 2014).



Embeddings versus sparse vectors

- Dense SVD embeddings sometimes work better than sparse PPMI matrices at tasks like word similarity
  - Denoising: low-order dimensions may represent unimportant information
  - Truncation may help the models generalize better to unseen data.
  - Having a smaller number of dimensions may make it easier for classifiers to properly weight the dimensions for the task.
  - Dense models may do better at capturing higher order co-occurrence.