

# Introduction to Information Retrieval

<http://informationretrieval.org>

## IIR 18: Latent Semantic Indexing

Hinrich Schütze

Institute for Natural Language Processing, Universität Stuttgart

2011-08-29

# Models and Methods

- 1 Boolean model and its limitations (30)
- 2 Vector space model (30)
- 3 Probabilistic models (30)
- 4 Language model-based retrieval (30)
- 5 Latent semantic indexing (30)
- 6 Learning to rank (30)

# Take-away

# Take-away

- Singular Value Decomposition (SVD): The math behind LSI

# Take-away

- Singular Value Decomposition (SVD): The math behind LSI
- SVD used for dimensionality reduction

# Take-away

- **Singular Value Decomposition (SVD)**: The math behind LSI
- SVD used for **dimensionality reduction**
- **Latent Semantic Indexing (LSI)**: SVD used in information retrieval

# Outline

- 1 Singular Value Decomposition
- 2 Dimensionality reduction
- 3 Latent Semantic Indexing

# Recall: Term-document matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
anthony	5.25	3.18	0.0	0.0	0.0	0.35
brutus	1.21	6.10	0.0	1.0	0.0	0.0
caesar	8.59	2.54	0.0	1.51	0.25	0.0
calpurnia	0.0	1.54	0.0	0.0	0.0	0.0
cleopatra	2.85	0.0	0.0	0.0	0.0	0.0
mercy	1.51	0.0	1.90	0.12	5.25	0.88
worser	1.37	0.0	0.11	4.15	0.25	1.95
...						

# Recall: Term-document matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
anthony	5.25	3.18	0.0	0.0	0.0	0.35
brutus	1.21	6.10	0.0	1.0	0.0	0.0
caesar	8.59	2.54	0.0	1.51	0.25	0.0
calpurnia	0.0	1.54	0.0	0.0	0.0	0.0
cleopatra	2.85	0.0	0.0	0.0	0.0	0.0
mercy	1.51	0.0	1.90	0.12	5.25	0.88
worser	1.37	0.0	0.11	4.15	0.25	1.95
...						

This matrix is the basis for computing [the similarity between documents and queries](#).

# Recall: Term-document matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
anthony	5.25	3.18	0.0	0.0	0.0	0.35
brutus	1.21	6.10	0.0	1.0	0.0	0.0
caesar	8.59	2.54	0.0	1.51	0.25	0.0
calpurnia	0.0	1.54	0.0	0.0	0.0	0.0
cleopatra	2.85	0.0	0.0	0.0	0.0	0.0
mercy	1.51	0.0	1.90	0.12	5.25	0.88
worser	1.37	0.0	0.11	4.15	0.25	1.95
...						

This matrix is the basis for computing **the similarity between documents and queries**.

This lecture: Can we transform this matrix, so that we get a **better measure of similarity** between documents and queries?

# Latent semantic indexing: Overview

# Latent semantic indexing: Overview

- We will **decompose** the term-document matrix into a product of matrices.

# Latent semantic indexing: Overview

- We will **decompose** the term-document matrix into a product of matrices.
- The particular decomposition we'll use: **singular value decomposition** (SVD).

# Latent semantic indexing: Overview

- We will **decompose** the term-document matrix into a product of matrices.
- The particular decomposition we'll use: **singular value decomposition** (SVD).
- SVD:  $C = U\Sigma V^T$  (where  $C$  = term-document matrix)

# Latent semantic indexing: Overview

- We will **decompose** the term-document matrix into a product of matrices.
- The particular decomposition we'll use: **singular value decomposition** (SVD).
- SVD:  $C = U\Sigma V^T$  (where  $C$  = term-document matrix)
- We will then use the SVD to compute a **new, improved term-document matrix**  $C'$ .

# Latent semantic indexing: Overview

- We will **decompose** the term-document matrix into a product of matrices.
- The particular decomposition we'll use: **singular value decomposition** (SVD).
- SVD:  $C = U\Sigma V^T$  (where  $C$  = term-document matrix)
- We will then use the SVD to compute a **new, improved term-document matrix**  $C'$ .
- We'll get **better similarity** values out of  $C'$  (compared to  $C$ ).

# Latent semantic indexing: Overview

- We will **decompose** the term-document matrix into a product of matrices.
- The particular decomposition we'll use: **singular value decomposition** (SVD).
- SVD:  $C = U\Sigma V^T$  (where  $C$  = term-document matrix)
- We will then use the SVD to compute a **new, improved term-document matrix**  $C'$ .
- We'll get **better similarity** values out of  $C'$  (compared to  $C$ ).
- Using SVD for this purpose is called **latent semantic indexing** or LSI. □

# Example of $C = U\Sigma V^T$ : The matrix $C$

$C$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
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

This is a standard term-document matrix.

# Example of $C = U\Sigma V^T$ : The matrix $C$

$C$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
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

This is a standard term-document matrix.

Actually, we use a non-weighted matrix here to simplify the example.



Example of  $C = U\Sigma V^T$ : The matrix  $U$ 

$U$	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

# Example of $C = U\Sigma V^T$ : The matrix $U$

$U$	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

Square matrix,  $M \times M$

# Example of $C = U\Sigma V^T$ : The matrix $U$

$U$	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

Square matrix,  $M \times M$

This is an **orthonormal matrix**: (i) Row vectors have unit length.  
(ii) Any two distinct row vectors are orthogonal to each other.

# Example of $C = U\Sigma V^T$ : The matrix $U$

$U$	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

Square matrix,  $M \times M$

This is an **orthonormal matrix**: (i) Row vectors have unit length.  
 (ii) Any two distinct row vectors are orthogonal to each other.

Think of the dimensions as “semantic” dimensions that capture distinct topics like politics, sports, economics. 2 = water/land

# Example of $C = U\Sigma V^T$ : The matrix $U$

$U$	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

Square matrix,  $M \times M$

This is an **orthonormal matrix**: (i) Row vectors have unit length.  
 (ii) Any two distinct row vectors are orthogonal to each other.

Think of the dimensions as “semantic” dimensions that capture distinct topics like politics, sports, economics. 2 = water/land

Each number  $u_{ij}$  in the matrix indicates how strongly related term  $i$  is to the topic represented by semantic dimension  $j$ . □

Example of  $C = U\Sigma V^T$ : The matrix  $\Sigma$ 

$\Sigma$	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

# Example of $C = U\Sigma V^T$ : The matrix $\Sigma$

$\Sigma$	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

This is a **square, diagonal matrix** of dimensionality  $\min(M, N) \times \min(M, N)$ .

# Example of $C = U\Sigma V^T$ : The matrix $\Sigma$

$\Sigma$	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

This is a **square, diagonal matrix** of dimensionality  $\min(M, N) \times \min(M, N)$ .

The diagonal consists of the **singular values** of  $C$ .

# Example of $C = U\Sigma V^T$ : The matrix $\Sigma$

$\Sigma$	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

This is a **square, diagonal matrix** of dimensionality  $\min(M, N) \times \min(M, N)$ .

The diagonal consists of the **singular values** of  $C$ .

The magnitude of the singular value measures the **importance of the corresponding semantic dimension**.

# Example of $C = U\Sigma V^T$ : The matrix $\Sigma$

$\Sigma$	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

This is a **square, diagonal matrix** of dimensionality  $\min(M, N) \times \min(M, N)$ .

The diagonal consists of the **singular values** of  $C$ .

The magnitude of the singular value measures the **importance of the corresponding semantic dimension**.

We'll make use of this by **omitting unimportant dimensions**. □

Example of  $C = U\Sigma V^T$ : The matrix  $V^T$ 

$V^T$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
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
6	0.00	0.00	0.00	-0.58	0.58	0.58

Example of  $C = U\Sigma V^T$ : The matrix  $V^T$ 

$V^T$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
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
6	0.00	0.00	0.00	-0.58	0.58	0.58

$N \times N$  square matrix.

# Example of $C = U\Sigma V^T$ : The matrix $V^T$

$V^T$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
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
6	0.00	0.00	0.00	-0.58	0.58	0.58

$N \times N$  square matrix. Drop row 6 – only want  $\min(M, N)$  LSI dims.

# Example of $C = U\Sigma V^T$ : The matrix $V^T$

$V^T$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
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
6	0.00	0.00	0.00	-0.58	0.58	0.58

$N \times N$  square matrix. Drop row 6 – only want  $\min(M, N)$  LSI dims.

Again: This is an **orthonormal matrix**: (i) Column vectors have unit length. (ii) Any two distinct column vectors are orthogonal to each other.

# Example of $C = U\Sigma V^T$ : The matrix $V^T$

$V^T$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
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
6	0.00	0.00	0.00	-0.58	0.58	0.58

$N \times N$  square matrix. Drop row 6 – only want  $\min(M, N)$  LSI dims.

Again: This is an **orthonormal matrix**: (i) Column vectors have unit length. (ii) Any two distinct column vectors are orthogonal to each other.

These are again the semantic dimensions from matrices  $U$  and  $\Sigma$  that capture distinct topics like politics, sports, economics.

## Example of $C = U\Sigma V^T$ : The matrix $V^T$

$V^T$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
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
6	0.00	0.00	0.00	-0.58	0.58	0.58

$N \times N$  square matrix. Drop row 6 – only want  $\min(M, N)$  LSI dims.

Again: This is an **orthonormal matrix**: (i) Column vectors have unit length. (ii) Any two distinct column vectors are orthogonal to each other.

These are again the semantic dimensions from matrices  $U$  and  $\Sigma$  that capture distinct topics like politics, sports, economics.

Each  $v_{ij}$  in the matrix indicates how strongly related document  $i$  is to the topic represented by semantic dimension  $j$ . □

# Example of $C = U\Sigma V^T$ : All four matrices

$C$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$							
ship	1.00	0.00	1.00	0.00	0.00	0.00							
boat	0.00	1.00	0.00	0.00	0.00	0.00							
ocean	1.00	1.00	0.00	0.00	0.00	0.00							
wood	1.00	0.00	0.00	1.00	1.00	0.00							
tree	0.00	0.00	0.00	1.00	0.00	1.00							
$U$	1	2	3	4	5	$\Sigma$	1	2	3	4	5		
ship	-0.44	-0.30	0.57	0.58	0.25	1	2.16	0.00	0.00	0.00	0.00		
boat	-0.13	-0.33	-0.59	0.00	0.73	2	0.00	1.59	0.00	0.00	0.00		
ocean	-0.48	-0.51	-0.37	0.00	-0.61	3	0.00	0.00	1.28	0.00	0.00		
wood	-0.70	0.35	0.15	-0.58	0.16	4	0.00	0.00	0.00	1.00	0.00		
tree	-0.26	0.65	-0.41	0.58	-0.09	5	0.00	0.00	0.00	0.00	0.39		
$V^T$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$							
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							

LSI is decomposition of  $C$  into a representation of the terms, a representation of the documents and a representation of the importance of the “semantic” dimensions.



# LSI: Summary

# LSI: Summary

- We've decomposed the term-document matrix  $C$  into a product of three matrices:  $U\Sigma V^T$ .

# LSI: Summary

- We've decomposed the term-document matrix  $C$  into a product of three matrices:  $U\Sigma V^T$ .
- The term matrix  $U$  – consists of one (row) vector for each term

# LSI: Summary

- We've decomposed the term-document matrix  $C$  into a product of three matrices:  $U\Sigma V^T$ .
- The term matrix  $U$  – consists of one (row) vector for each term
- The document matrix  $V^T$  – consists of one (column) vector for each document

# LSI: Summary

- We've decomposed the term-document matrix  $C$  into a product of three matrices:  $U\Sigma V^T$ .
- The term matrix  $U$  – consists of one (row) vector for each term
- The document matrix  $V^T$  – consists of one (column) vector for each document
- The singular value matrix  $\Sigma$  – diagonal matrix with singular values, reflecting importance of each dimension

# LSI: Summary

- We've decomposed the term-document matrix  $C$  into a product of three matrices:  $U\Sigma V^T$ .
- The term matrix  $U$  – consists of one (row) vector for each term
- The document matrix  $V^T$  – consists of one (column) vector for each document
- The singular value matrix  $\Sigma$  – diagonal matrix with singular values, reflecting importance of each dimension
- Next: Why are we doing this? □

# Outline

- 1 Singular Value Decomposition
- 2 Dimensionality reduction
- 3 Latent Semantic Indexing

# How we use the SVD in LSI

# How we use the SVD in LSI

- Key property: Each singular value tells us how important its dimension is.

# How we use the SVD in LSI

- Key property: Each singular value tells us how important its dimension is.
- By setting less important dimensions to zero, we keep the important information, but get rid of the “details”.

# How we use the SVD in LSI

- Key property: Each singular value tells us how important its dimension is.
- By setting less important dimensions to zero, we keep the important information, but get rid of the “details”.
- These details may

# How we use the SVD in LSI

- Key property: Each singular value tells us how important its dimension is.
- By setting less important dimensions to zero, we keep the important information, but get rid of the “details”.
- These details may
  - be **noise** – in that case, reduced LSI is a better representation because it is less noisy.

# How we use the SVD in LSI

- Key property: Each singular value tells us how important its dimension is.
- By setting less important dimensions to zero, we keep the important information, but get rid of the “details”.
- These details may
  - be **noise** – in that case, reduced LSI is a better representation because it is less noisy.
  - **make things dissimilar that should be similar** – again, the reduced LSI representation is a better representation because it represents similarity better.

# How we use the SVD in LSI

- Key property: Each singular value tells us how important its dimension is.
- By setting less important dimensions to zero, we keep the important information, but get rid of the “details”.
- These details may
  - be **noise** – in that case, reduced LSI is a better representation because it is less noisy.
  - **make things dissimilar that should be similar** – again, the reduced LSI representation is a better representation because it represents similarity better.
- Analogy for “fewer details is better”

# How we use the SVD in LSI

- Key property: Each singular value tells us how important its dimension is.
- By setting less important dimensions to zero, we keep the important information, but get rid of the “details”.
- These details may
  - be **noise** – in that case, reduced LSI is a better representation because it is less noisy.
  - **make things dissimilar that should be similar** – again, the reduced LSI representation is a better representation because it represents similarity better.
- Analogy for “fewer details is better”
  - Image of a blue flower

# How we use the SVD in LSI

- Key property: Each singular value tells us how important its dimension is.
- By setting less important dimensions to zero, we keep the important information, but get rid of the “details”.
- These details may
  - be **noise** – in that case, reduced LSI is a better representation because it is less noisy.
  - **make things dissimilar that should be similar** – again, the reduced LSI representation is a better representation because it represents similarity better.
- Analogy for “fewer details is better”
  - Image of a blue flower
  - Image of a yellow flower

# How we use the SVD in LSI

- Key property: Each singular value tells us how important its dimension is.
- By setting less important dimensions to zero, we keep the important information, but get rid of the “details”.
- These details may
  - be **noise** – in that case, reduced LSI is a better representation because it is less noisy.
  - **make things dissimilar that should be similar** – again, the reduced LSI representation is a better representation because it represents similarity better.
- Analogy for “fewer details is better”
  - Image of a blue flower
  - Image of a yellow flower
  - Omitting color makes it easier to see the similarity



# Reducing the dimensionality to 2

$U$	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	
$\Sigma_2$	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	0.00	0.00	0.00	
4	0.00	0.00	0.00	0.00	0.00	
5	0.00	0.00	0.00	0.00	0.00	
$V^T$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
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	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00

# Reducing the dimensionality to 2

$U$	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	
$\Sigma_2$	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	0.00	0.00	0.00	
4	0.00	0.00	0.00	0.00	0.00	
5	0.00	0.00	0.00	0.00	0.00	
$V^T$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
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	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00

Actually, we only zero out singular values in  $\Sigma$ . This has the effect of setting the corresponding dimensions in  $U$  and  $V^T$  to zero when computing the product  $C = U\Sigma V^T$ .  $\square$

# Reducing the dimensionality to 2

$C_2$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$					
ship	0.85	0.52	0.28	0.13	0.21	-0.08					
boat	0.36	0.36	0.16	-0.20	-0.02	-0.18					
ocean	1.01	0.72	0.36	-0.04	0.16	-0.21					
wood	0.97	0.12	0.20	1.03	0.62	0.41					
tree	0.12	-0.39	-0.08	0.90	0.41	0.49					
$U$	1	2	3	4	5	$\Sigma_2$	1	2	3	4	5
ship	-0.44	-0.30	0.57	0.58	0.25	1	2.16	0.00	0.00	0.00	0.00
boat	-0.13	-0.33	-0.59	0.00	0.73	2	0.00	1.59	0.00	0.00	0.00
ocean	-0.48	-0.51	-0.37	0.00	-0.61	3	0.00	0.00	0.00	0.00	0.00
wood	-0.70	0.35	0.15	-0.58	0.16	4	0.00	0.00	0.00	0.00	0.00
tree	-0.26	0.65	-0.41	0.58	-0.09	5	0.00	0.00	0.00	0.00	0.00
$V^T$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$					
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					



Recall unreduced decomposition  $C = U\Sigma V^T$ 

$C$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$							
ship	1.00	0.00	1.00	0.00	0.00	0.00							
boat	0.00	1.00	0.00	0.00	0.00	0.00							
ocean	1.00	1.00	0.00	0.00	0.00	0.00	=						
wood	1.00	0.00	0.00	1.00	1.00	0.00							
tree	0.00	0.00	0.00	1.00	0.00	1.00							
$U$	1	2	3	4	5	$\Sigma$	1	2	3	4	5		
ship	-0.44	-0.30	0.57	0.58	0.25	1	2.16	0.00	0.00	0.00	0.00		
boat	-0.13	-0.33	-0.59	0.00	0.73	2	0.00	1.59	0.00	0.00	0.00		
ocean	-0.48	-0.51	-0.37	0.00	-0.61	3	0.00	0.00	1.28	0.00	0.00	×	
wood	-0.70	0.35	0.15	-0.58	0.16	4	0.00	0.00	0.00	1.00	0.00		
tree	-0.26	0.65	-0.41	0.58	-0.09	5	0.00	0.00	0.00	0.00	0.39		
$V^T$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$							
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							



Original matrix  $C$  vs. reduced  $C_2 = U\Sigma_2V^T$ 

$C$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
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

$C_2$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
ship	0.85	0.52	0.28	0.13	0.21	-0.08
boat	0.36	0.36	0.16	-0.20	-0.02	-0.18
ocean	1.01	0.72	0.36	-0.04	0.16	-0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	-0.39	-0.08	0.90	0.41	0.49

Original matrix  $C$  vs. reduced  $C_2 = U\Sigma_2V^T$ 

$C$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
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

$C_2$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
ship	0.85	0.52	0.28	0.13	0.21	-0.08
boat	0.36	0.36	0.16	-0.20	-0.02	-0.18
ocean	1.01	0.72	0.36	-0.04	0.16	-0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	-0.39	-0.08	0.90	0.41	0.49

We can view  $C_2$  as a **two-dimensional** representation of the matrix  $C$ . We have performed a **dimensionality reduction** to two dimensions.



# Why the reduced matrix $C_2$ is better than $C$

$C$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
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

$C_2$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
ship	0.85	0.52	0.28	0.13	0.21	-0.08
boat	0.36	0.36	0.16	-0.20	-0.02	-0.18
ocean	1.01	0.72	0.36	-0.04	0.16	-0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	-0.39	-0.08	0.90	0.41	0.49

# Why the reduced matrix $C_2$ is better than $C$

$C$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
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

Similarity of  $d_2$  and  $d_3$  in the original space: 0.

$C_2$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
ship	0.85	0.52	0.28	0.13	0.21	-0.08
boat	0.36	0.36	0.16	-0.20	-0.02	-0.18
ocean	1.01	0.72	0.36	-0.04	0.16	-0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	-0.39	-0.08	0.90	0.41	0.49

# Why the reduced matrix $C_2$ is better than $C$

$C$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
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

$C_2$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
ship	0.85	0.52	0.28	0.13	0.21	-0.08
boat	0.36	0.36	0.16	-0.20	-0.02	-0.18
ocean	1.01	0.72	0.36	-0.04	0.16	-0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	-0.39	-0.08	0.90	0.41	0.49

Similarity of  $d_2$  and  $d_3$  in the original space: 0.

Similarity of  $d_2$  and  $d_3$  in the reduced space:  $0.52 * 0.28 + 0.36 * 0.16 + 0.72 * 0.36 + 0.12 * 0.20 + -0.39 * -0.08 \approx 0.52$

# Why the reduced matrix $C_2$ is better than $C$

$C$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
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

$C_2$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
ship	0.85	0.52	0.28	0.13	0.21	-0.08
boat	0.36	0.36	0.16	-0.20	-0.02	-0.18
ocean	1.01	0.72	0.36	-0.04	0.16	-0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	-0.39	-0.08	0.90	0.41	0.49

“boat” and “ship” are semantically similar. The “reduced” similarity measure reflects this.  $\square$

# Outline

- 1 Singular Value Decomposition
- 2 Dimensionality reduction
- 3 Latent Semantic Indexing

# Why we use LSI in information retrieval

# Why we use LSI in information retrieval

- LSI takes documents that are semantically similar (= talk about the same topics), ...

# Why we use LSI in information retrieval

- LSI takes documents that are semantically similar (= talk about the same topics), ...
- ... but are not similar in the vector space (because they use different words) ...

# Why we use LSI in information retrieval

- LSI takes documents that are semantically similar (= talk about the same topics), ...
- ...but are not similar in the vector space (because they use different words) ...
- ...and re-represents them in a reduced vector space ...

# Why we use LSI in information retrieval

- LSI takes documents that are semantically similar (= talk about the same topics), ...
- ...but are not similar in the vector space (because they use different words) ...
- ...and re-represents them in a reduced vector space ...
- ...in which they have higher similarity.

# Why we use LSI in information retrieval

- LSI takes documents that are semantically similar (= talk about the same topics), ...
- ...but are not similar in the vector space (because they use different words) ...
- ...and re-represents them in a reduced vector space ...
- ...in which they have higher similarity.
- Thus, LSI addresses the problems of **synonymy** and **semantic relatedness**.

# Why we use LSI in information retrieval

- LSI takes documents that are semantically similar (= talk about the same topics), ...
- ...but are not similar in the vector space (because they use different words) ...
- ...and re-represents them in a reduced vector space ...
- ...in which they have higher similarity.
- Thus, LSI addresses the problems of **synonymy** and **semantic relatedness**.
- Standard vector space: Synonyms contribute nothing to document similarity.

# Why we use LSI in information retrieval

- LSI takes documents that are semantically similar (= talk about the same topics), ...
- ... but are not similar in the vector space (because they use different words) ...
- ... and re-represents them in a reduced vector space ...
- ... in which they have higher similarity.
- Thus, LSI addresses the problems of **synonymy** and **semantic relatedness**.
- Standard vector space: Synonyms contribute nothing to document similarity.
- Desired effect of LSI: Synonyms contribute strongly to document similarity.



# How LSI addresses synonymy and semantic relatedness

- The dimensionality reduction forces us to omit a lot of “detail”.

# How LSI addresses synonymy and semantic relatedness

- The dimensionality reduction forces us to omit a lot of “detail”.
- We have to map different words (= different dimensions of the full space) to the same dimension in the reduced space.

# How LSI addresses synonymy and semantic relatedness

- The dimensionality reduction forces us to omit a lot of “detail”.
- We have to map different words (= different dimensions of the full space) to the same dimension in the reduced space.
- The “cost” of mapping synonyms to the same dimension is much less than the cost of collapsing unrelated words.

# How LSI addresses synonymy and semantic relatedness

- The dimensionality reduction forces us to omit a lot of “detail”.
- We have to map different words (= different dimensions of the full space) to the same dimension in the reduced space.
- The “cost” of mapping synonyms to the same dimension is much less than the cost of collapsing unrelated words.
- SVD selects the “least costly” mapping (see below).

# How LSI addresses synonymy and semantic relatedness

- The dimensionality reduction forces us to omit a lot of “detail”.
- We have to map different words (= different dimensions of the full space) to the same dimension in the reduced space.
- The “cost” of mapping synonyms to the same dimension is much less than the cost of collapsing unrelated words.
- SVD selects the “least costly” mapping (see below).
- Thus, it will map synonyms to the same dimension.

# How LSI addresses synonymy and semantic relatedness

- The dimensionality reduction forces us to omit a lot of “detail”.
- We have to map different words (= different dimensions of the full space) to the same dimension in the reduced space.
- The “cost” of mapping synonyms to the same dimension is much less than the cost of collapsing unrelated words.
- SVD selects the “least costly” mapping (see below).
- Thus, it will map synonyms to the same dimension.
- But it will avoid doing that for unrelated words. □

# LSI: Comparison to other approaches

# LSI: Comparison to other approaches

- **Relevance feedback** and **query expansion** are used to **increase recall** in information retrieval – if query and documents have no terms in common.

# LSI: Comparison to other approaches

- Relevance feedback and query expansion are used to increase recall in information retrieval – if query and documents have no terms in common.
- LSI increases recall and hurts precision.

# LSI: Comparison to other approaches

- Relevance feedback and query expansion are used to increase recall in information retrieval – if query and documents have no terms in common.
- LSI increases recall and hurts precision.
- Thus, it addresses the same problems as (pseudo) relevance feedback and query expansion . . .

# LSI: Comparison to other approaches

- Relevance feedback and query expansion are used to increase recall in information retrieval – if query and documents have no terms in common.
- LSI increases recall and hurts precision.
- Thus, it addresses the same problems as (pseudo) relevance feedback and query expansion . . .
- . . . and it has the same problems. □

# Implementation

# Implementation

- Compute SVD of term-document matrix

# Implementation

- Compute SVD of term-document matrix
- Reduce the space and compute reduced document representations

# Implementation

- Compute SVD of term-document matrix
- Reduce the space and compute reduced document representations
- Map the query into the reduced space  $\vec{q}_k = \Sigma_k^{-1} U_k^T \vec{q}$ .

# Implementation

- Compute SVD of term-document matrix
- Reduce the space and compute reduced document representations
- Map the query into the reduced space  $\vec{q}_k = \Sigma_k^{-1} U_k^T \vec{q}$ .
- This follows from:  $C_k = U \Sigma_k V^T \Rightarrow \Sigma_k^{-1} U^T C = V_k^T$

# Implementation

- Compute SVD of term-document matrix
- Reduce the space and compute reduced document representations
- Map the query into the reduced space  $\vec{q}_k = \Sigma_k^{-1} U_k^T \vec{q}$ .
- This follows from:  $C_k = U \Sigma_k V^T \Rightarrow \Sigma_k^{-1} U^T C = V_k^T$
- Compute similarity of  $q_k$  with all reduced documents in  $V_k$ .

# Implementation

- Compute SVD of term-document matrix
- Reduce the space and compute reduced document representations
- Map the query into the reduced space  $\vec{q}_k = \Sigma_k^{-1} U_k^T \vec{q}$ .
- This follows from:  $C_k = U \Sigma_k V^T \Rightarrow \Sigma_k^{-1} U^T C = V_k^T$
- Compute similarity of  $q_k$  with all reduced documents in  $V_k$ .
- Output ranked list of documents as usual

# Implementation

- Compute SVD of term-document matrix
- Reduce the space and compute reduced document representations
- Map the query into the reduced space  $\vec{q}_k = \Sigma_k^{-1} U_k^T \vec{q}$ .
- This follows from:  $C_k = U \Sigma_k V^T \Rightarrow \Sigma_k^{-1} U^T C = V_k^T$
- Compute similarity of  $q_k$  with all reduced documents in  $V_k$ .
- Output ranked list of documents as usual
- Exercise: What is the fundamental problem with this approach?



# Optimality

# Optimality

- SVD is **optimal** in the following sense.

# Optimality

- SVD is **optimal** in the following sense.
- Keeping the  $k$  largest singular values and setting all others to zero gives you the optimal approximation of the original matrix  $C$ . **Eckart-Young theorem**

# Optimality

- SVD is **optimal** in the following sense.
- Keeping the  $k$  largest singular values and setting all others to zero gives you the optimal approximation of the original matrix  $C$ . **Eckart-Young theorem**
- Optimal: no other matrix of the same rank (= with the same underlying dimensionality) approximates  $C$  better.

# Optimality

- SVD is **optimal** in the following sense.
- Keeping the  $k$  largest singular values and setting all others to zero gives you the optimal approximation of the original matrix  $C$ . **Eckart-Young theorem**
- Optimal: no other matrix of the same rank (= with the same underlying dimensionality) approximates  $C$  better.
- Measure of approximation is Frobenius norm:

$$\|C\|_F = \sqrt{\sum_i \sum_j c_{ij}^2}$$

# Optimality

- SVD is **optimal** in the following sense.
- Keeping the  $k$  largest singular values and setting all others to zero gives you the optimal approximation of the original matrix  $C$ . **Eckart-Young theorem**
- Optimal: no other matrix of the same rank (= with the same underlying dimensionality) approximates  $C$  better.
- Measure of approximation is Frobenius norm:  
$$\|C\|_F = \sqrt{\sum_i \sum_j c_{ij}^2}$$
- So LSI uses the “best possible” matrix.

# Optimality

- SVD is **optimal** in the following sense.
- Keeping the  $k$  largest singular values and setting all others to zero gives you the optimal approximation of the original matrix  $C$ . **Eckart-Young theorem**
- Optimal: no other matrix of the same rank (= with the same underlying dimensionality) approximates  $C$  better.
- Measure of approximation is Frobenius norm:  
$$\|C\|_F = \sqrt{\sum_i \sum_j c_{ij}^2}$$
- So LSI uses the “best possible” matrix.
- There is only one best possible matrix – unique solution (modulo signs).

# Optimality

- SVD is **optimal** in the following sense.
- Keeping the  $k$  largest singular values and setting all others to zero gives you the optimal approximation of the original matrix  $C$ . **Eckart-Young theorem**
- Optimal: no other matrix of the same rank (= with the same underlying dimensionality) approximates  $C$  better.
- Measure of approximation is Frobenius norm:
$$\|C\|_F = \sqrt{\sum_i \sum_j c_{ij}^2}$$
- So LSI uses the “best possible” matrix.
- There is only one best possible matrix – unique solution (modulo signs).
- Caveat: There is only a tenuous relationship between the Frobenius norm and cosine similarity between documents. □

# Data for graphical illustration of LSI

# Data for graphical illustration of LSI

- $c_1$  Human machine interface for lab abc computer applications
- $c_2$  A survey of user opinion of computer system response time
- $c_3$  The EPS user interface management system
- $c_4$  System and human system engineering testing of EPS
- $c_5$  Relation of user perceived response time to error measurement
- $m_1$  The generation of random binary unordered trees
- $m_2$  The intersection graph of paths in trees
- $m_3$  Graph minors IV Widths of trees and well quasi ordering
- $m_4$  Graph minors A survey

# Data for graphical illustration of LSI

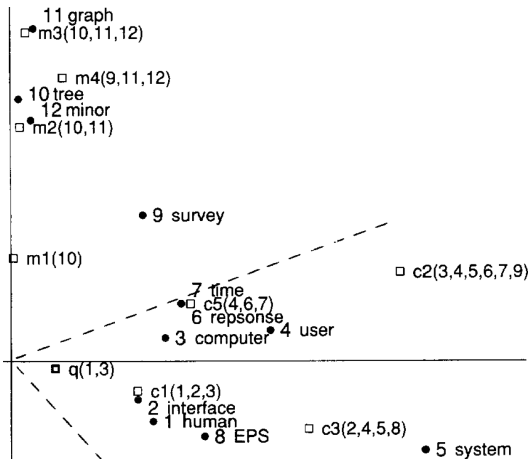
- $c_1$  Human machine interface for lab abc computer applications
- $c_2$  A survey of user opinion of computer system response time
- $c_3$  The EPS user interface management system
- $c_4$  System and human system engineering testing of EPS
- $c_5$  Relation of user perceived response time to error measurement
- $m_1$  The generation of random binary unordered trees
- $m_2$  The intersection graph of paths in trees
- $m_3$  Graph minors IV Widths of trees and well quasi ordering
- $m_4$  Graph minors A survey

**The matrix  $C$**

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1



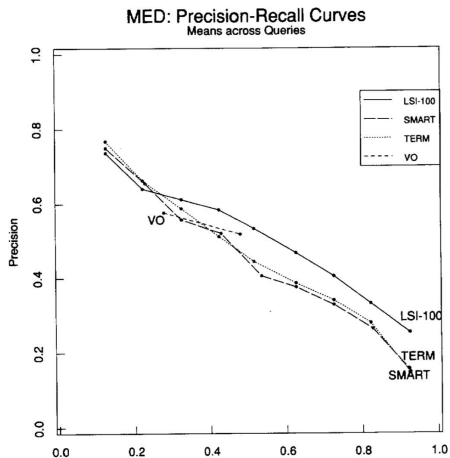
# Graphical illustration of LSI: Plot of $C_2$



2-dimensional plot of  $C_2$  (scaled dimensions). Circles = terms. Open squares = documents (component terms in parentheses).  $q$  = query “human computer interaction”.

The dotted cone represents the region whose points are within a cosine of .9 from  $q$ . All documents about human-computer documents (c1-c5) are near  $q$ , even c3/c5 although they share no terms. None of the graph theory documents (m1-m4) are near  $q$ . □

# LSI performs better than vector space on MED collection



LSI-100 = LSI reduced to 100 dimensions; SMART = SMART implementation of vector space model



# Take-away

- **Singular Value Decomposition (SVD)**: The math behind LSI
- SVD used for **dimensionality reduction**
- **Latent Semantic Indexing (LSI)**: SVD used in information retrieval

# Resources

- Chapter 18 of Introduction to Information Retrieval
- Resources at <http://informationretrieval.org/essir2011>
  - Latent semantic indexing by Deerwester et al. (original paper)
  - Probabilistic LSI by Hofmann
  - Word space: LSI for words