

COP509

## Natural Language Processing

---

Latent Semantic Indexing

Prof. Georgina Cosma

Department of Computer Science  
Loughborough University

[g.cosma@lboro.ac.uk](mailto:g.cosma@lboro.ac.uk)

## Learning Outcomes

By the end of this session, you will be able to:

1. Understand what Latent Semantic Indexing (LSI) is and why it was developed
2. Explain how Singular Value Decomposition (SVD) decomposes term-document matrices
3. Perform dimensionality reduction on document representations using the truncated SVD
4. Understand how LSI addresses the problems of synonymy and semantic relatedness
5. Apply LSI for information retrieval including query projection and document ranking
6. Evaluate the strengths and limitations of LSI and appreciate it as a foundation for modern embedding techniques

### Key Insight

LSI provides transparent mathematical intuition for how semantic similarity can be captured in vector spaces — a principle that underlies all modern word embeddings.

# Outline

## 1. Introduction to LSI

- ▶ What is LSI/LSA? The fundamental problem: synonymy and vocabulary mismatch
- ▶ The term-document matrix as starting point

## 2. Singular Value Decomposition (SVD)

- ▶ Decomposing the term-document matrix:  $U$ ,  $\Sigma$ ,  $V^T$
- ▶ Orthonormality and why it matters

## 3. Dimensionality Reduction

- ▶ Zeroing vs. truncation; the reduced matrices  $U_2$ ,  $\Sigma_2$ ,  $V_2^T$
- ▶ Reconstructing  $C_2$ ; the Eckart-Young theorem

## 4. LSI for Information Retrieval

- ▶ Query projection, document ranking, soft clustering
- ▶ Computational challenges and practical solutions

## 5. From LSI to Modern Embeddings

- ▶ LSA vs. LDA; connections to Word2Vec, GloVe, BERT

Part 1

---

## Introduction to LSI

# What is LSI / LSA?

**Latent Semantic Indexing (LSI)** is a technique in natural language processing — in particular distributional semantics — for analysing relationships between a set of documents and the terms they contain by producing a set of **concepts** related to the documents and terms.

## Terminology

- ▶ **LSI** — used when the technique is applied to [information retrieval](#)
- ▶ **LSA** — used for other NLP tasks such as topic modelling, document clustering
- ▶ The two terms are used interchangeably in the literature

## Key Idea

Represent words and documents in a **reduced semantic space** where:

- ▶ Synonyms map to similar positions
- ▶ Semantically related terms are close together
- ▶ Documents about similar topics cluster together

# The Fundamental Problem: Term Mismatch

**Standard vector space models have a critical limitation:**

Consider these two documents:

- ▶ Document 1: “The ship sailed across the ocean”
- ▶ Document 2: “The boat travelled across the sea”

**Problem:** These documents describe the same topic, yet share **no common terms**.

Standard cosine similarity = **0.0** (orthogonal vectors).

## Synonymy

Different words, same meaning  
(ship/boat, ocean/sea)

LSI **primarily** addresses this by mapping related terms to the same latent dimension.

## Polysemy

Same word, different meanings  
(bank = financial institution *or* river bank)

LSI addresses this only partially; contextual models (BERT) handle it fully.

## Recall: The Term-Document Matrix

The starting point for LSI is the standard term-document matrix  $C$ :

	Anthony & 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 similarity between documents and queries.

**Question:** Can we transform this matrix to get a **better** measure of similarity?

**Answer:** Yes — using Singular Value Decomposition (SVD).

Part 2

---

## Singular Value Decomposition

# LSI Overview: The SVD Decomposition

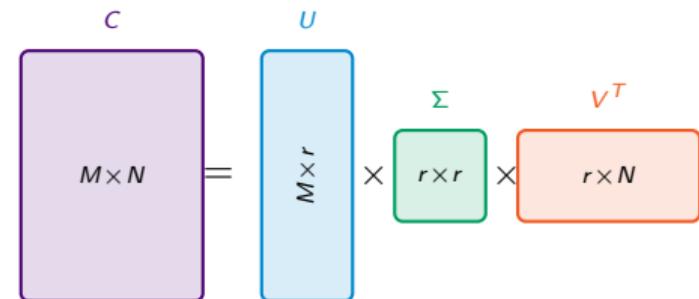
**Core technique:** Decompose the term-document matrix into a product of three matrices.

$$C = U \Sigma V^T$$

Matrix	Size	Meaning
$C$	$M \times N$	Original term-document matrix
$U$	$M \times r$	Terms in semantic space
$\Sigma$	$r \times r$	Importance of each dimension
$V^T$	$r \times N$	Documents in semantic space
$r = \min(M, N)$		for full SVD; $r = k$ after reduction

## Goal

Use the SVD to compute an improved matrix  $C_k$  that captures **semantic** similarity — not just exact word overlap.



Using SVD for this purpose is called Latent Semantic Indexing (LSI).

## The Worked Example: Matrix $C$

For pedagogical clarity we use a small, non-weighted matrix:

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

**Two semantic groups are present:**

**Water-related:** ship, boat, ocean

$d_2$  (boat, ocean) and  $d_3$  (ship) are purely water-related.

**Land-related:** wood, tree

$d_4$  (wood, tree),  $d_5$  (wood),  $d_6$  (tree) are purely land-related.

$d_1$  (ship, ocean, wood) is **mixed** — it contains both water and land terms.

**Goal:** After LSI reduction,  $d_2$  (boat) and  $d_3$  (ship) should be recognised as similar despite sharing no terms.

# SVD Intuition: Hidden Topics

## The Core Idea

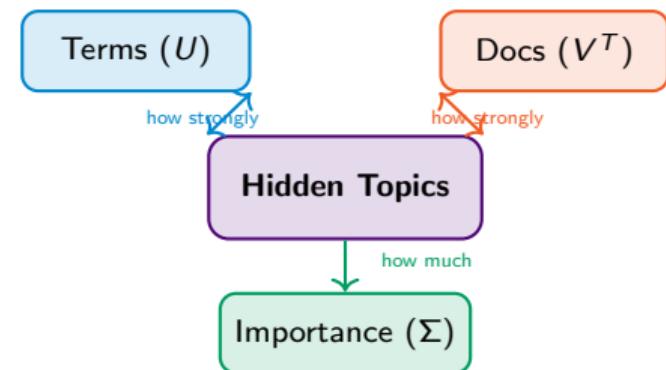
SVD finds the **hidden topics** that best explain the word co-occurrence patterns in your documents.

## What SVD discovers automatically:

- ▶ “ship”, “boat”, and “ocean” co-occur → **water topic**
- ▶ “wood” and “tree” co-occur → **land topic**

## SVD gives us three pieces:

$U$	How strongly each <b>term</b> relates to each hidden topic
$\Sigma$	How <b>important</b> each hidden topic is
$V^T$	How strongly each <b>document</b> relates to each hidden topic



## The Matrix $U$ (Term Matrix)

$U$	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

We can identify topics

Dimension 2 (highlighted):

ship/boat/ocean are negative  
wood/tree are positive

⇒ Dimension 2 captures the water vs land split.

One row per term, one column per  $\min(M, N)$ .

This is an orthonormal matrix

- Row vectors have unit length (values between -1 and +1)
- Each row vector captures a unique, independent dimension of variation
- Think of dimensions as “semantic” dimensions that capture distinct topics

# Interpreting the Dimensions of $U$

## Dimension 1: General importance

Term	$U_{i1}$
ship	-0.44
boat	-0.13
ocean	-0.48
wood	-0.70
tree	-0.26

All values negative — this dimension reflects the overall **frequency/importance** of each term across all documents. Wood appears in most docs, so it has the largest (most negative) value.

## Dimension 2: Water vs Land split

Term	$U_{i2}$	
ocean	-0.51	← Water
boat	-0.33	← Water
ship	-0.30	← Water
wood	+0.35	← Land
tree	+0.65	← Land

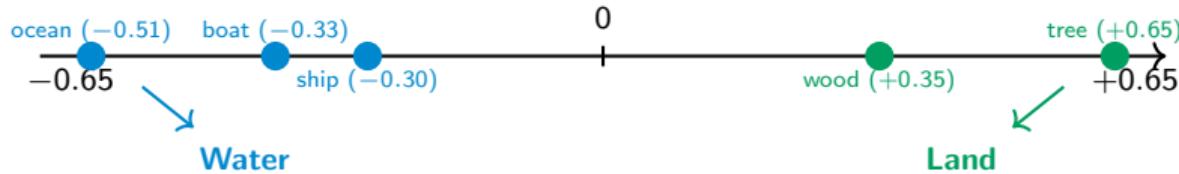
Opposite signs reveal the core semantic split. **SVD** discovered this without being told that ocean and tree are different topics.

## Key Takeaway

Each dimension in  $U$  encodes a different aspect of the term relationships. The most important semantic distinctions appear in the first few dimensions.

## Dimension 2: The Water vs Land Split

Dimension 2 captures the deepest semantic split in our data:



Dimension 2 pattern:  $d_1, d_2, d_3$  are negative (water);  $d_4, d_5, d_6$  are positive (land).

What is orthogonality? (Plain English)

Two vectors are **orthogonal** if they are completely *independent* of each other — knowing one tells you nothing about the other. Think of the x-axis and y-axis on a graph: they are perpendicular and capture entirely different directions.

In the SVD matrices, every dimension (every row/column) is orthogonal to every other. This means **each dimension captures a unique, non-overlapping piece of the semantic structure**. It also guarantees that when we drop the least important dimensions, we lose as little information as possible — and that reconstructing  $C = U\Sigma V^T$  works exactly.

## The Matrix $\Sigma$ (Singular Values)

$\Sigma$	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  $\min(M, N) \times \min(M, N)$ .

### Topic Importance

The diagonal consists of the **singular values** of  $C$ . The magnitude of the singular value measures the **importance of the corresponding semantic dimension**.

Dimensions 1 and 2 (highlighted) are the most important. We will **omit dimensions 3–5**.

### Orthogonality of $V^T$

Allows us to reliably reconstruct documents as linear combinations of the latent concepts, and to measure similarity between documents in this reduced semantic space.

## The Matrix $V^T$ (Document Matrix)

$V^T$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
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

Each column = one document's representation.

Each row = one semantic dimension.

**Row 2 (highlighted):**

$d_1, d_2, d_3$  negative (water topic)

$d_4, d_5, d_6$  positive (land topic)

### Key Insight

Row 2 of  $V^T$  gives the same water/land split seen in column 2 of  $U$  — consistent semantic structure throughout the whole decomposition.

# $C = U\Sigma V^T$ : All Four Matrices Together

Multiplying all components reconstructs the original  $C$  exactly. Each dimension holds information.

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

=

$U$	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

×

$\Sigma$	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

×

$V^T$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
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

## Key Point

The **full** SVD (all 5 dimensions) reconstructs  $C$  perfectly. Each dimension holds some information about the relationships between terms and documents.

## Next Step

We will **reduce** to  $k = 2$  dimensions by discarding the least important ones — keeping the semantic signal and dropping the noise.

The key question: *which dimensions are “noise”?*

## SVD Decomposition: Summary

We have decomposed the term-document matrix  $C$  into a product of three matrices:

$$C = U \Sigma V^T$$

- ▶ 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**  $\Sigma$  is a diagonal matrix where each value reflects the importance of the corresponding semantic dimension

### Next: Why Are We Doing This?

The SVD decomposition on its own simply reconstructs  $C$ . The power of LSI comes from what we do next: applying dimensionality reduction by keeping only the most important dimensions. This is what produces a better similarity measure.

Part 3

---

## Dimensionality Reduction

## How We Use 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 and discard the “details”. These discarded details may:

1. Be **noise** — reduced LSI is less noisy and therefore a better representation
2. Make things **dissimilar that should be similar** — reduced LSI better captures true semantic similarity

**Analogy: Colour vs Black-and-White**

Consider two images of the same flower, one in colour and one in black-and-white. Omitting the colour (a detail) makes the structural similarity easier to see. Similarly, omitting rare or noisy term dimensions helps reveal the underlying semantic similarity between documents.

## Reducing the Dimensionality to 2

We **zero out**  $\sigma_3, \sigma_4, \sigma_5$  in  $\Sigma$  — shaded cells become zero:

$U$  (columns 3–5 now contribute nothing)

	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

$V^T$  (rows 3–5 now contribute nothing)

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

$\Sigma_2$  (dims 3–5 zeroed)

	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

What zeroing  $\sigma_3, \sigma_4, \sigma_5$  means

Setting  $\sigma_j = 0$  multiplies the contribution of column  $j$  of  $U$  and row  $j$  of  $V^T$  by zero — those entire topic directions **vanish** from  $U\Sigma V^T$ . Only the two most important semantic dimensions survive.

Or: **truncate** — physically remove the shaded rows/columns.  
Both methods give identical  $C_2$ .

## Two Methods for Reducing to $k = 2$

### Method 1: Zero out small singular values

Set  $\sigma_3 = \sigma_4 = \sigma_5 = 0$  in  $\Sigma$ . Matrix sizes do not change.

$\Sigma_2$	1	2	3	4	5
1	2.16	0.00	0	0	0
2	0.00	1.59	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

In  $U\Sigma V^T$ , column  $j$  of  $U$  is scaled by  $\sigma_j$ . Setting  $\sigma_j = 0$  means that contribution is multiplied by zero — columns 3–5 of  $U$  and rows 3–5 of  $V^T$  contribute **nothing**.

### Method 2: Truncate matrices

Since those contributions are zero, physically remove them. Matrix sizes do change.

Matrix	Before	After
$U_2$	$5 \times 5$	$5 \times 2$
$\Sigma_2$	$5 \times 5$	$2 \times 2$
$V_2^T$	$5 \times 6$	$2 \times 6$

$$C_2 = U_2 \Sigma_2 V_2^T$$

For large collections (millions of documents), carrying zero-contribution columns wastes memory and computation. Truncation avoids this entirely.

Both methods produce **identical**  $C_2$ . Method 1 explains *why* it works; Method 2 is how it is implemented in practice.

## Original $C$ vs Reduced $C_2 = U\Sigma_2 V^T$

Both reconstructions side by side — spot the difference:

Original  $C$  (exact, integer, sparse)

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

$$\text{sim}(d_2, d_3) = 0 \text{ (no shared terms)}$$

boat and ship appear unrelated.

Reduced  $C_2$  (approx., real-valued, dense)

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

$$\text{sim}(d_2, d_3) \approx 0.52 \text{ (semantic overlap)}$$

boat and ship are now similar!

### What changed?

$C_2$  is a **two-dimensional representation** of the matrix. Every entry is now non-zero — the latent semantic structure has been made explicit. We have performed a **dimensionality reduction** to two dimensions.

## The Truncated Matrices: $U_2$ , $\Sigma_2$ , $V_2^T$

After truncating to  $k = 2$ , the three matrices are:

$U_2$  ( $5 \times 2$ ): First 2 cols of  $U$

$\Sigma_2$  ( $2 \times 2$ ): Top-left submatrix

Size change summary:

	1	2
ship	-0.44	-0.30
boat	-0.13	-0.33
ocean	-0.48	-0.51
wood	-0.70	+0.35
tree	-0.26	+0.65

	1	2
1	2.16	0.00
2	0.00	1.59

	Full	Truncated
$U$	$5 \times 5$	$5 \times 2$
$\Sigma$	$5 \times 5$	$2 \times 2$
$V^T$	$5 \times 6$	$2 \times 6$

$V_2^T$  ( $2 \times 6$ ): First 2 rows of  $V^T$

	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
row 1	-0.75	-0.28	-0.20	-0.45	-0.33	-0.12
row 2	-0.29	-0.53	-0.19	+0.63	+0.22	+0.41

Row 1: all negative (general term importance). Row 2: water docs ( $d_1-d_3$ ) negative; land docs ( $d_4-d_6$ ) positive.

$$\text{Reconstruction: } C_2 = U_2 \Sigma_2 V_2^T$$

**Step 1: Compute  $U_2 \Sigma_2$  — scale each column of  $U_2$  by its singular value**

	$U_2$		$U_2 \Sigma_2$	
	col 1	col 2	$\times 2.16$	$\times 1.59$
ship	-0.44	-0.30	-0.95	-0.47
boat	-0.13	-0.33	-0.28	-0.53
ocean	-0.48	-0.51	-1.03	-0.81
wood	-0.70	+0.35	-1.52	+0.56
tree	-0.26	+0.65	-0.57	+1.03

**Step 2: Multiply  $(U_2 \Sigma_2)$  by  $V_2^T$  to give  $C_2$**

Entry  $c_{ij} = \text{dot product of row } i \text{ (term } i\text{) from } (U_2 \Sigma_2) \text{ with column } j \text{ (document } j\text{) from } V_2^T.$

**Example: how much does “ship” relate to document  $d_2$ ?**

$d_2$  is the document containing only “boat” and “ocean”. We compute entry  $c_{\text{ship}, d_2}$  by taking the **ship row** of  $(U_2 \Sigma_2)$  and the  $d_2$  **column** of  $V_2^T$  and computing the dot product:

$$(-0.95) \times (-0.28) + (-0.47) \times (-0.53) = 0.266 + 0.249 = \mathbf{0.52}$$

This positive value reflects that “ship” and  $d_2$  (boat, ocean) are in the same semantic neighbourhood — even though “ship” does not appear in  $d_2$  at all.

$C_2$	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$	$d_6$
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

$C_2$  is **dense** and **real-valued** — every term now has a non-zero entry for every document.

## Why the Reduced Matrix is “Better”

Similarity of  $d_2$  (boat, ocean) and  $d_3$  (ship):

Original space ( $C$ ):

	$d_2$	$d_3$
ship	0	1
boat	1	0
ocean	1	0
wood	0	0
tree	0	0

$$d_2 \cdot d_3 = 0 + 0 + 0 + 0 + 0 = \textcolor{red}{0.0}$$

No shared terms  $\Rightarrow$  zero similarity.

Reduced space ( $C_2$ ):

	$d_2$	$d_3$
ship	0.52	0.28
boat	0.36	0.16
ocean	0.72	0.36
wood	0.12	0.20
tree	-0.39	-0.08

$$0.52 \times 0.28 + 0.36 \times 0.16 + 0.72 \times 0.36 + \\ 0.12 \times 0.20 + (-0.39) \times (-0.08) \approx \textcolor{teal}{0.52}$$

### What Changed?

“boat” and “ship” are **semantically similar** (both water-related). Dimensionality reduction forces synonyms to share dimensions, making this relationship explicit. The “unnecessary detail” — the noise — has been removed.

# How LSI Addresses Synonymy

The dimensionality reduction forces different words to share the same semantic dimension.

## Low cost: map synonyms together

“happy”, “joyful”, “delighted” → same dimension

They co-occur with similar words, so collapsing them loses little information.

Small penalty

## High cost: map unrelated words together

“happy” + “automobile” → same dimension

They co-occur with very different words, so collapsing them destroys semantic distinctions.

Large penalty

SVD selects the “least costly” mapping:

- ▶ Automatically maps synonyms to similar positions in the reduced space
- ▶ Avoids collapsing unrelated concepts
- ▶ **No manual synonym dictionary needed!**

The mathematical framework naturally minimises information loss, preserving important semantic distinctions while merging redundant information.

# SVD is Mathematically Optimal: Eckart-Young Theorem

## Eckart-Young Theorem

Keeping the  $k$  largest singular values gives the **optimal rank- $k$  approximation** of  $C$  in terms of Frobenius norm:

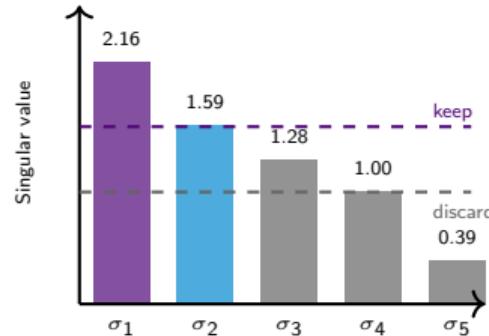
$$\|C - C_k\|_F = \sqrt{\sum_i \sum_j (c_{ij} - c'_{ij})^2}$$

No other rank- $k$  matrix is closer to  $C$ .

## What this means for LSI:

- ▶ Among all possible  $k$ -dimensional representations, SVD gives the best one
- ▶ **LSI uses the “best possible” matrix**

## Singular values in our example:



$\sigma_1, \sigma_2$  (purple/blue) retained;  $\sigma_3, \sigma_4, \sigma_5$  (grey) discarded as noise.

## Important Caveat

SVD is optimal for *reconstruction error*, not necessarily for downstream tasks such as retrieval. Neural methods

# Questions?

Prof. Georgina Cosma

[g.cosma@lboro.ac.uk](mailto:g.cosma@lboro.ac.uk)

Department of Computer Science, Loughborough University