

# **Multimedia Content Management: Latent Semantic Analysis**

---

Ralf Moeller  
Hamburg Univ. of Technology

# Acknowledgement

- Slides taken from presentation material for the following book:

Introduction  
to  
Information  
Retrieval

Christopher D. Manning  
*Stanford University*

Prabhakar Raghavan  
*Yahoo! Research*

Hinrich Schütze  
*University of Stuttgart*

 CAMBRIDGE  
UNIVERSITY PRESS

# Eigenvalues & Eigenvectors

- **Eigenvectors** (for a square  $m \times m$  matrix  $S$ )

$$S\mathbf{v} = \lambda\mathbf{v}$$

(right) eigenvector  $\mathbf{v} \in \mathbb{R}^m \neq \mathbf{0}$       eigenvalue  $\lambda \in \mathbb{R}$

*Example*

$$\begin{pmatrix} 6 & -2 \\ 4 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \end{pmatrix} = 2 \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

- How many eigenvalues are there at most?

$$S\mathbf{v} = \lambda\mathbf{v} \iff (S - \lambda I)\mathbf{v} = \mathbf{0}$$

only has a non-zero solution if  $|S - \lambda I| = 0$

this is a  $m$ -th order equation in  $\lambda$  which can have **at most  $m$  distinct solutions** (roots of the characteristic polynomial) - can be complex even though  $S$  is real.

# Singular Value Decomposition

For an  $m \times n$  matrix  $\mathbf{A}$  of rank  $r$  there exists a factorization (Singular Value Decomposition = **SVD**) as follows:

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

$m \times m$     $m \times n$     $V$  is  $n \times n$

The columns of  $\mathbf{U}$  are orthogonal eigenvectors of  $\mathbf{A}\mathbf{A}^T$ .

The columns of  $\mathbf{V}$  are orthogonal eigenvectors of  $\mathbf{A}^T\mathbf{A}$ .

Eigenvalues  $\lambda_1 \dots \lambda_r$  of  $\mathbf{A}\mathbf{A}^T$  are the eigenvalues of  $\mathbf{A}^T\mathbf{A}$ .

$$\sigma_i = \sqrt{\lambda_i}$$
$$\Sigma = \text{diag}(\sigma_1 \dots \sigma_r) \leftarrow \text{Singular values.}$$

# SVD example

$$\text{Let } A = \begin{bmatrix} 1 & -1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Thus  $m=3$ ,  $n=2$ . Its SVD is

$$\begin{bmatrix} 0 & 2/\sqrt{6} & 1/\sqrt{3} \\ 1/\sqrt{2} & -1/\sqrt{6} & 1/\sqrt{3} \\ 1/\sqrt{2} & 1/\sqrt{6} & -1/\sqrt{3} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{3} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$$

*As opposed to the presentation in the example, typically, the singular values are arranged in decreasing order.*

# Low-rank Approximation

- SVD can be used to compute optimal **low-rank approximations**.
- Approximation problem: Find  $A_k$  of rank  $k$  such that

$$A_k = \arg \min_{X: \text{rank}(X)=k} \|A - X\|_F \longleftarrow \text{Frobenius norm}$$

$$\|A\|_F \equiv \sqrt{\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2}.$$

$A_k$  and  $X$  are both  $m \times n$  matrices.  
Typically, want  $k \ll r$ .

# Low-rank Approximation

- Solution via SVD

$$A_k = U \operatorname{diag}(\sigma_1, \dots, \sigma_k, \underbrace{0, \dots, 0}_{\text{set smallest } r-k \text{ singular values to zero}}) V^T$$

*set smallest  $r-k$   
singular values to zero*

$$\underbrace{\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}}_{A_k} = \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_U \underbrace{\begin{bmatrix} \bullet & & & & \\ & \bullet & & & \\ & & & & \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}}_{V^T}$$

# SVD Low-rank approximation

- Whereas the term-doc matrix  $A$  may have  $m=50000$ ,  $n=10$  million (and rank close to 50000)
- We can construct an approximation  $A_{100}$  with rank 100.
  - ♦ Of all rank 100 matrices, it would have the lowest Frobenius error.
- Great ... but why would we??
- Answer: *Latent Semantic Indexing*

C. Eckart, G. Young, *The approximation of a matrix by another of lower rank.* *Psychometrika*, 1, 211-218, 1936.

# What it is

- From term-doc matrix  $A$ , we compute the approximation  $A_k$ .
- There is a row for each term and a column for each doc in  $A_k$
- Thus docs live in a space of  $k \ll r$  dimensions
  - ◆ These dimensions are not the original axes
- But why?

# Vector Space Model: Pros

- **Automatic** selection of index terms
- **Partial matching** of queries and documents (*dealing with the case where no document contains all search terms*)
- **Ranking** according to **similarity score** (*dealing with large result sets*)
- **Term weighting** schemes (*improves retrieval performance*)
- Geometric foundation

# Problems with Lexical Semantics

- Ambiguity and association in natural language
  - ◆ **Polysemy**: Words often have a **multitude of meanings** and different types of usage (*more severe in very heterogeneous collections*).
  - ◆ The vector space model is unable to discriminate between different meanings of the same word.

$$\text{sim}_{\text{true}}(d, q) < \cos(\angle(\vec{d}, \vec{q}))$$

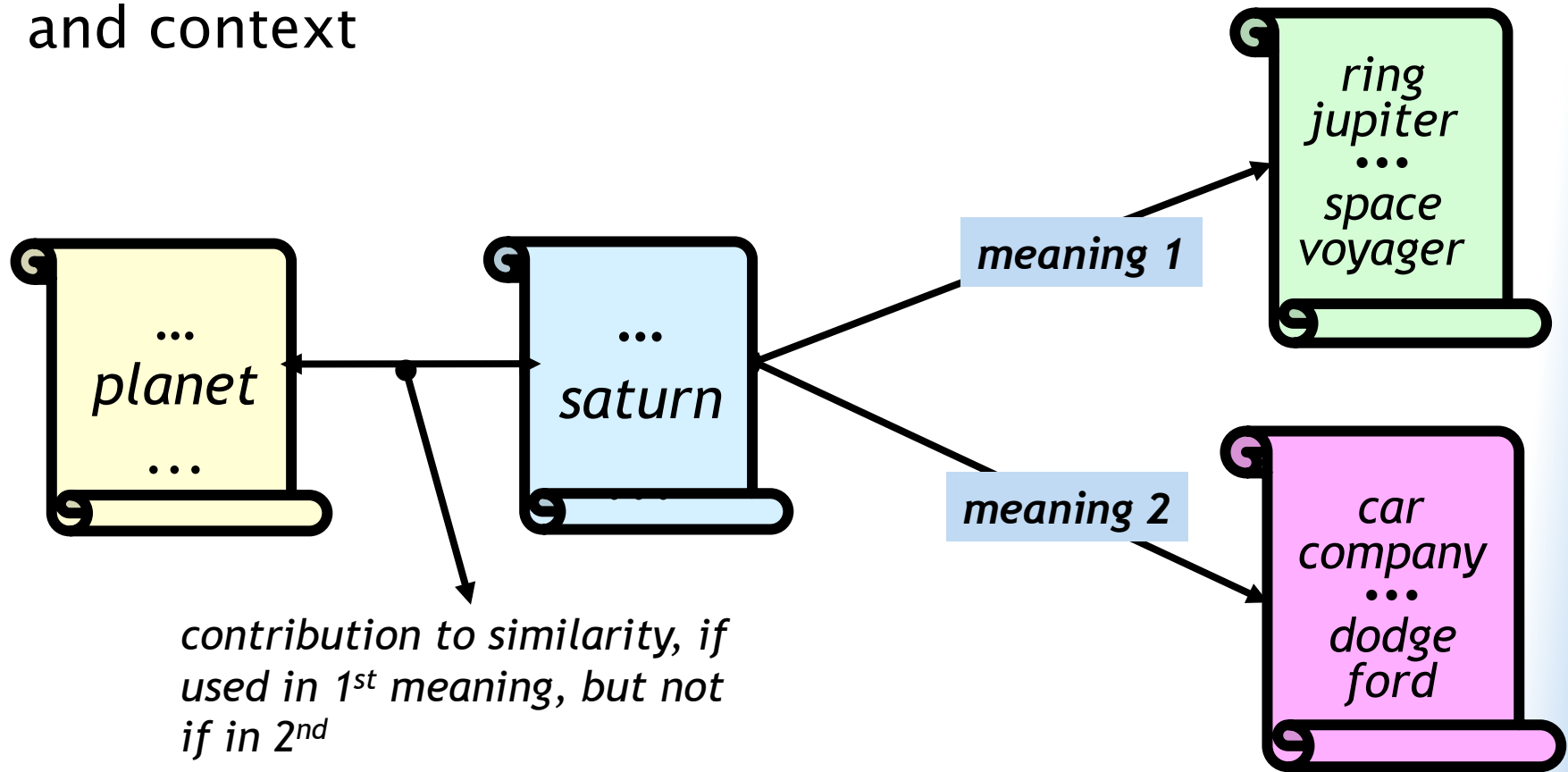
# Problems with Lexical Semantics

- ◆ **Synonymy**: Different terms may have an **dential or a similar meaning** (weaker: words indicating the same topic).
- ◆ No associations between words are made in the vector space representation.

$$\text{sim}_{\text{true}}(d, q) > \cos(\angle(\vec{d}, \vec{q}))$$

# Polysemy and Context

- Document similarity on single word level: polysemy and context



# Latent Semantic Indexing (LSI)

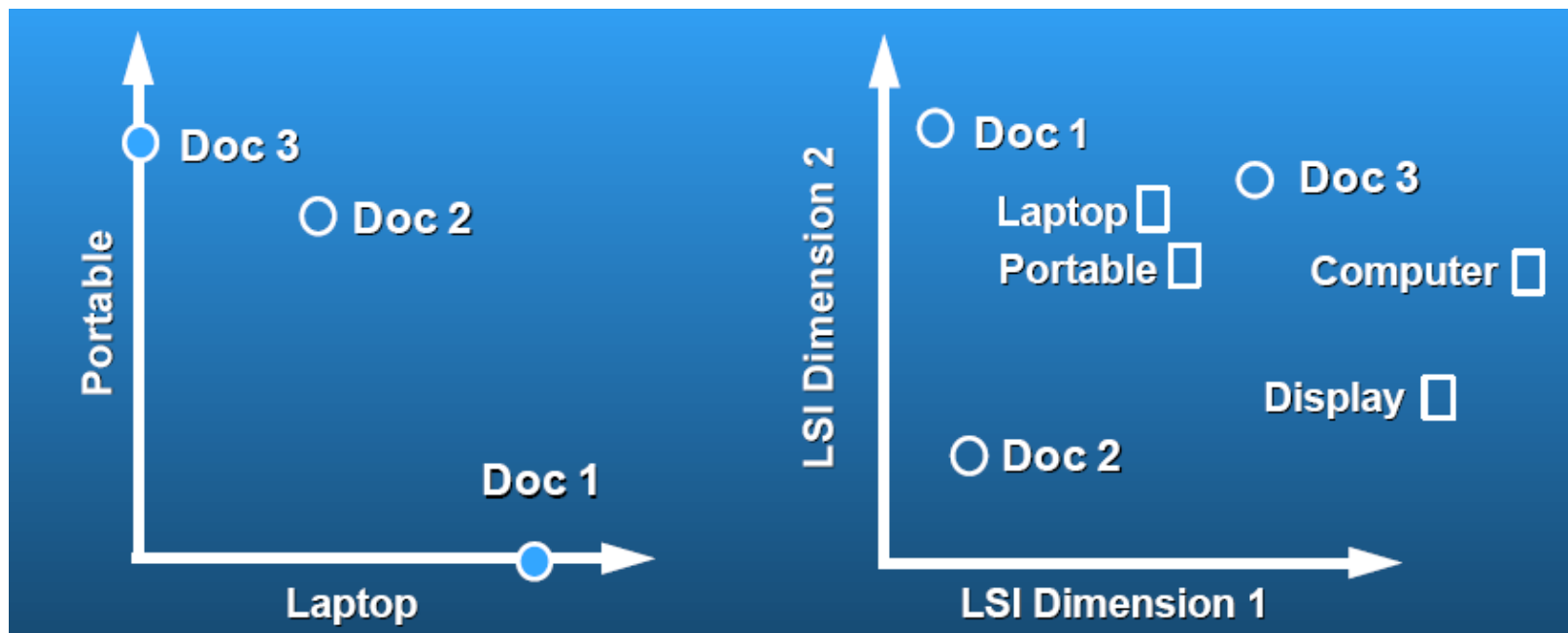
- Perform a **low-rank approximation** of **document-term matrix** (typical rank **100–300**)
- General idea
  - ◆ Map documents (*and* terms) to a **low-dimensional** representation.
  - ◆ Design a mapping such that the low-dimensional space reflects **semantic associations** (latent semantic space).
  - ◆ Compute document similarity based on the **inner product** in this **latent semantic space**

# Goals of LSI

- Similar terms map to similar location in low dimensional space
- Noise reduction by dimension reduction

# Latent Semantic Analysis

- **Latent semantic space:** illustrating example



*courtesy of Susan Dumais*

# Performing the maps

- Each row and column of  $A$  gets mapped into the  $k$ -dimensional LSI space, by the SVD.
- Claim - this is not only the mapping with the best (Frobenius error) approximation to  $A$ , but in fact *improves* retrieval.
- A query  $q$  is also mapped into this space, by

$$q_k = q^T U_k \Sigma_k^{-1}$$

- ♦ Query NOT a sparse vector.

# Empirical evidence: TREC

- Generally expect recall to improve – what about precision?
- Precision at or above median TREC precision
  - ◆ Top scorer on almost 20% of TREC topics
- Slightly better on average than straight vector spaces
- Effect of dimensionality:

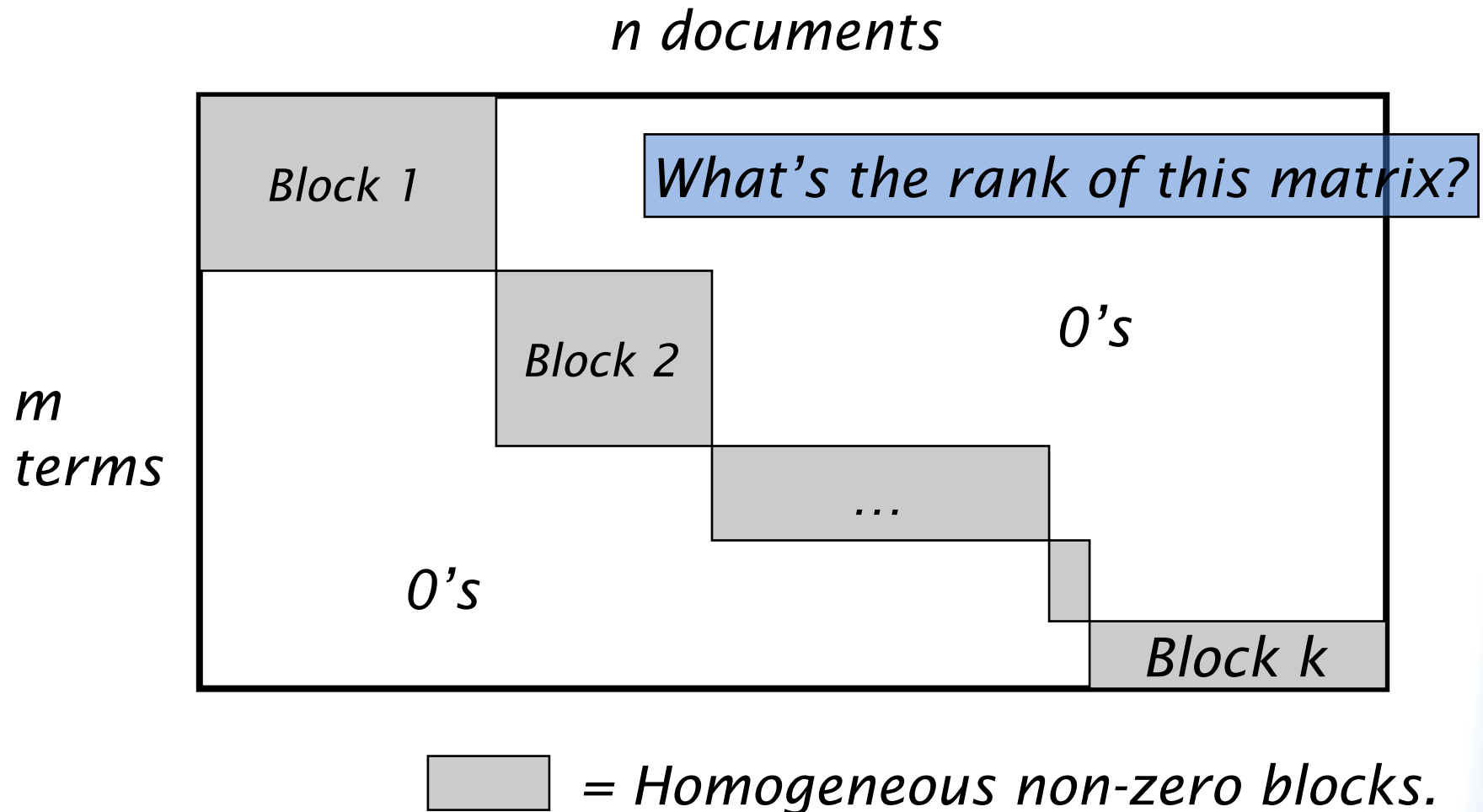
Dimensions	Precision
250	0.367
300	0.371
346	0.374

*TREC = Text REtrieval Conference benchmarks*

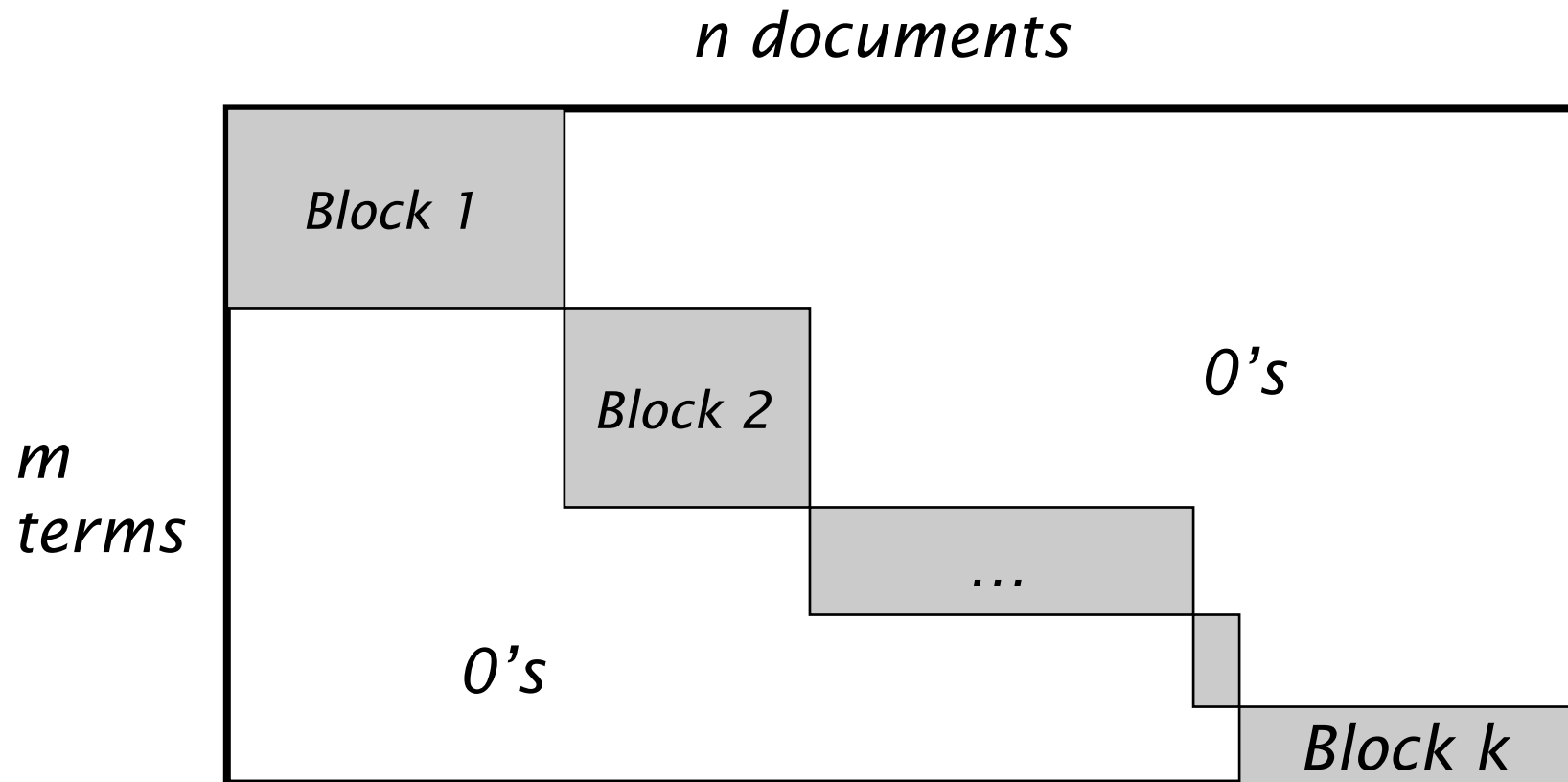
# But why is this clustering?

- We've talked about docs, queries, retrieval and precision here.
- What does this have to do with clustering?
- Intuition: Dimension reduction through LSI brings together “related” axes in the vector space.

# Intuition from block matrices



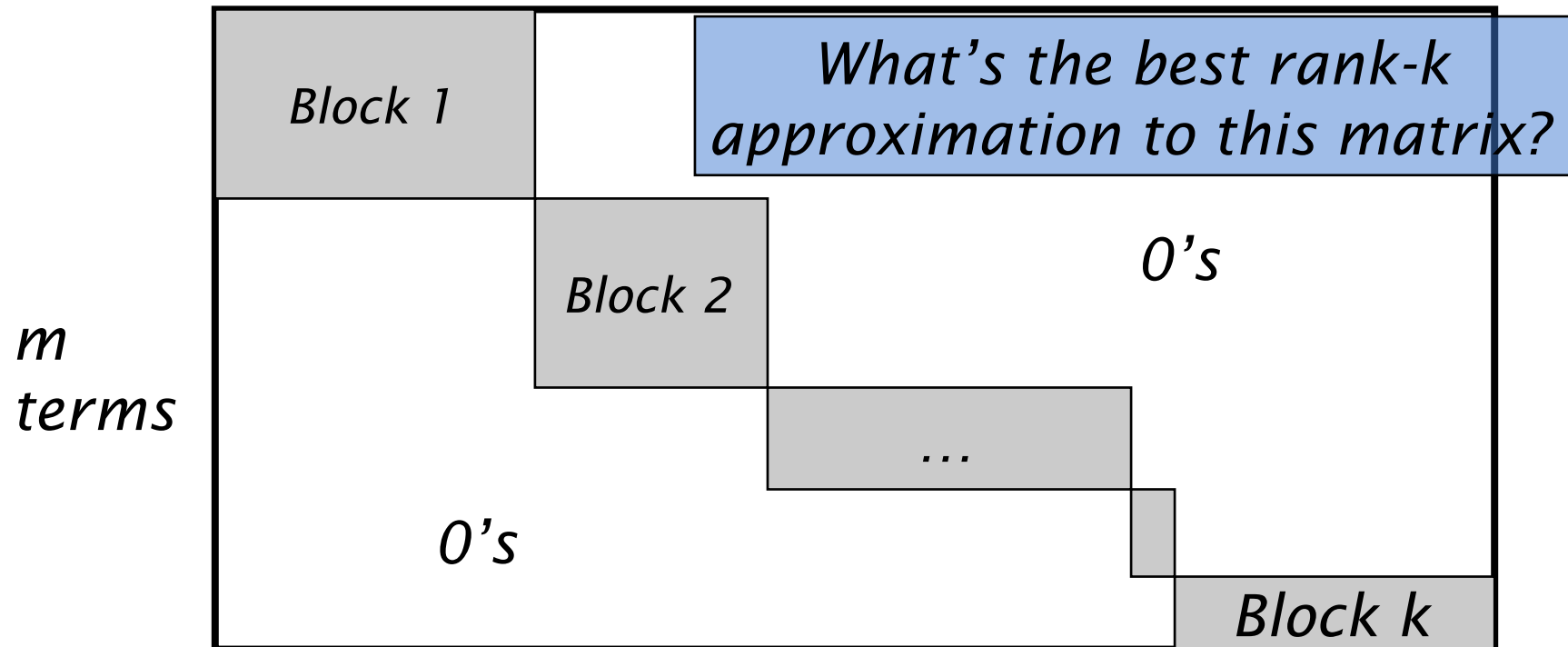
# Intuition from block matrices



*Vocabulary partitioned into  $k$  topics (clusters);  
each doc discusses only one topic.*

# Intuition from block matrices

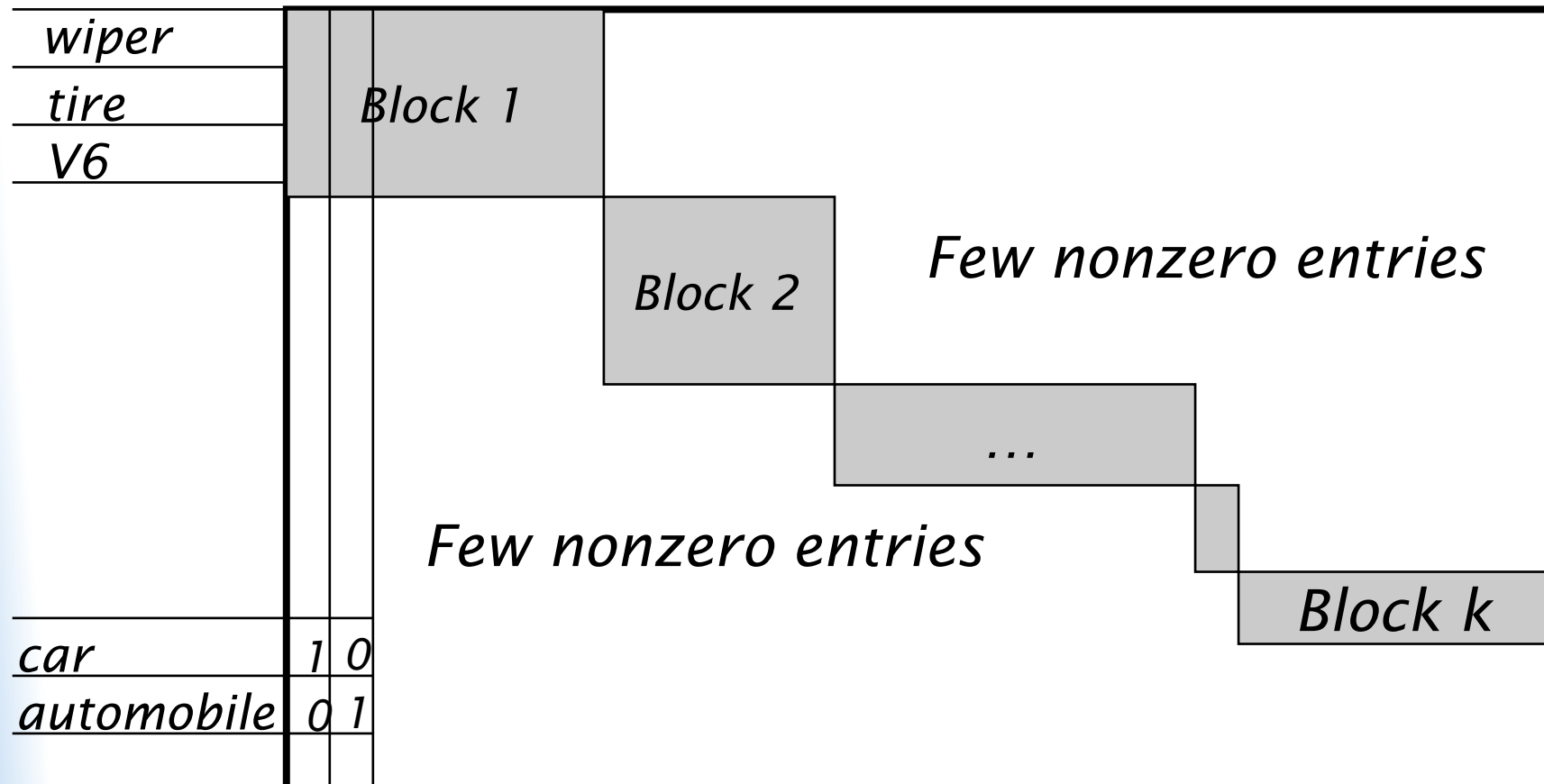
$n$  documents



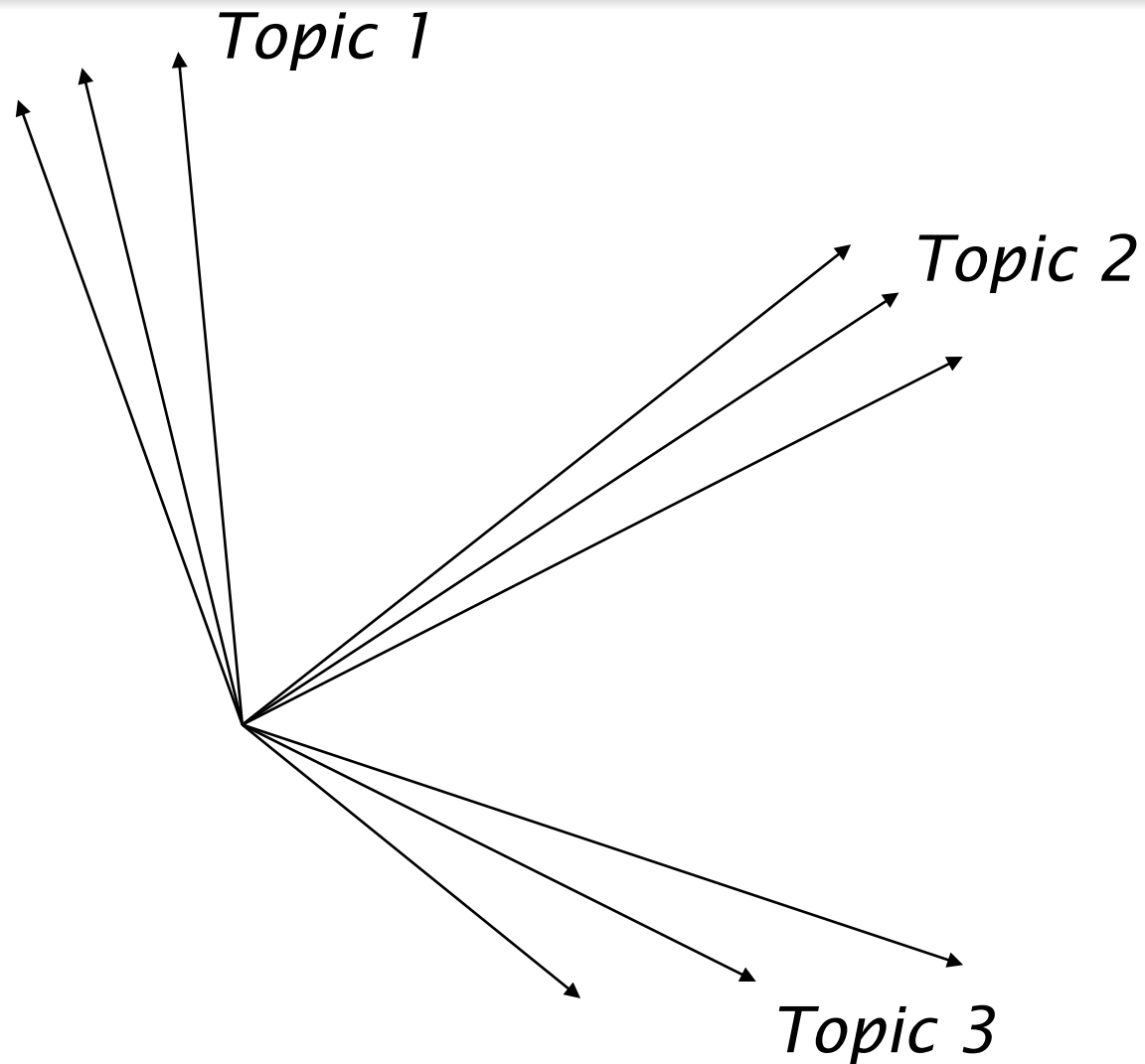
 = non-zero entries.

# Intuition from block matrices

*Likely there's a good rank-k approximation to this matrix.*



# Simplistic picture



# Some wild extrapolation

- The “dimensionality” of a corpus is the number of distinct topics represented in it.
- More mathematical wild extrapolation:
  - ◆ if  $A$  has a rank  $k$  approximation of low Frobenius error, then there are no more than  $k$  distinct topics in the corpus.