Statistical Learning and Data Mining
CS 363D/ SSC 358

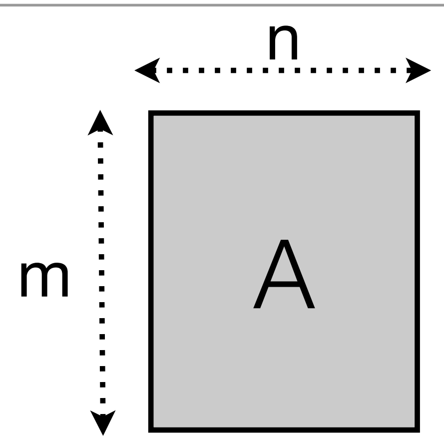
Lecture: SVD, Vector Space Document Model

Prof. Pradeep Ravikumar
pradeepr@cs.utexas.edu

Outline

- Matrices (Norms, Singular Value Decomposition (SVD), Eigenvalues)
- Running Example: Analysis of Text Documents

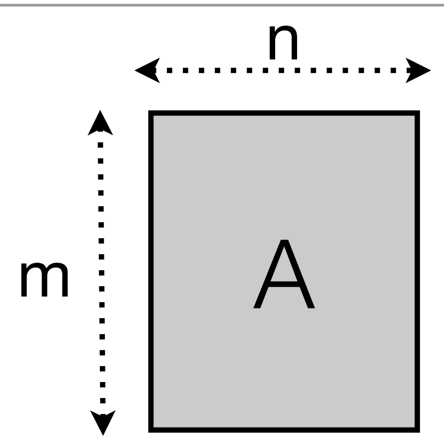
Matrices



- A matrix $A \in \mathbb{R}^{m \times n}$ can also be viewed as a **linear transformation**:

$$\begin{aligned} A : \mathbb{R}^n &\mapsto \mathbb{R}^m \\ x &\mapsto Ax \end{aligned}$$

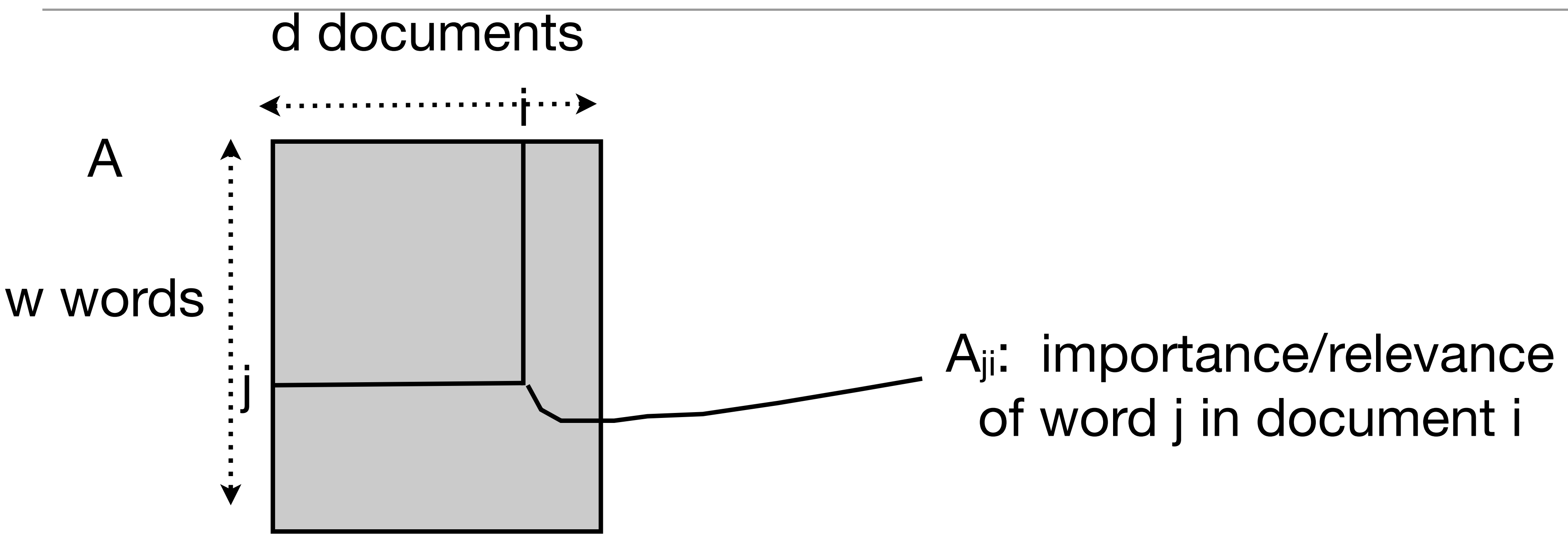
Matrices



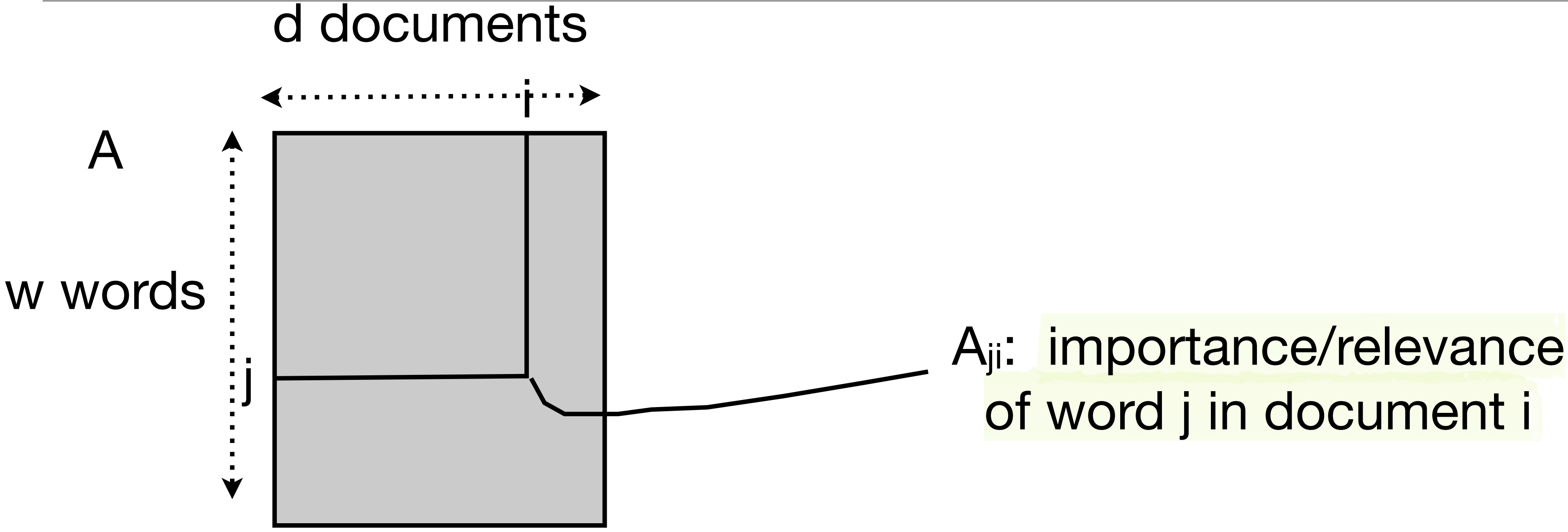
- A matrix $A \in \mathbb{R}^{m \times n}$ can also be viewed as a **linear transformation**:

$$\begin{aligned} A : \mathbb{R}^n &\mapsto \mathbb{R}^m \\ x &\mapsto Ax \\ \alpha x + \beta y &\mapsto A(\alpha x + \beta y) = \alpha Ax + \beta Ay \quad \textbf{: Linear Transformation} \end{aligned}$$

Vector Space Model for Documents



Vector Space Model for Documents

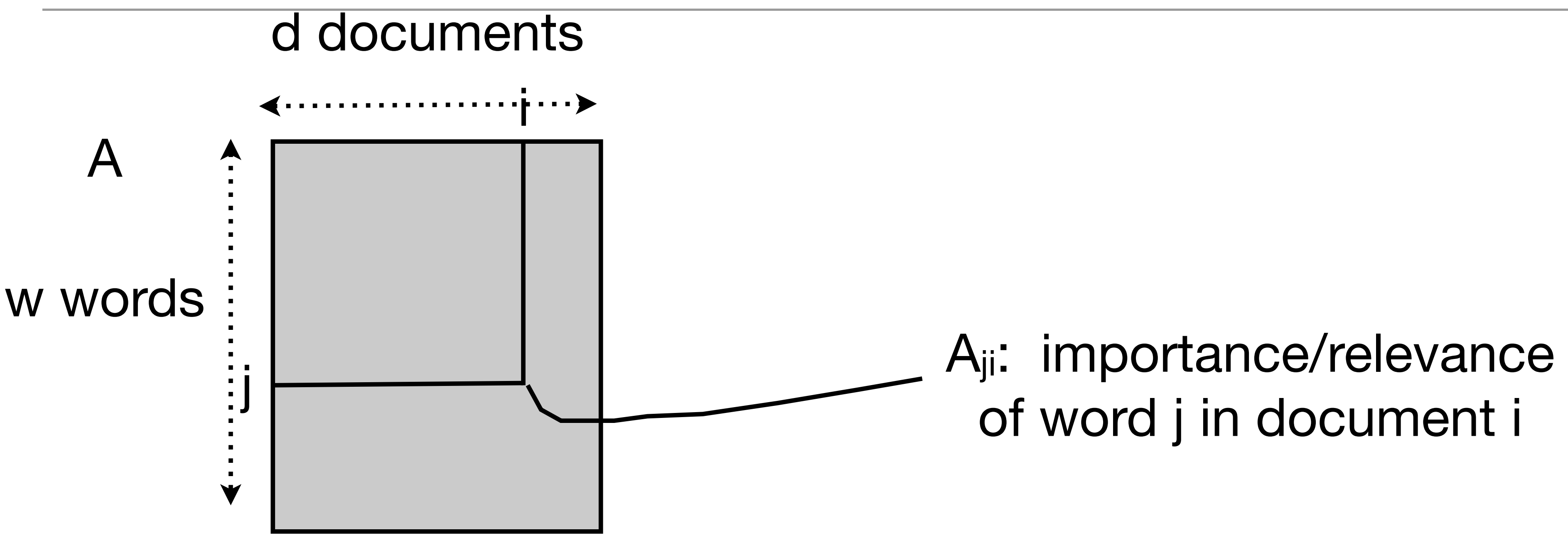


- Extract all unique words, ignoring case
- Eliminate **stop-words** : “a” , “and” , “the” , ...
- Eliminate non-content-bearing high-frequency and low-frequency words (using heuristic criteria)
- ….
- For each document, count no. of occurrences of each word

Vector Space Model for Documents

- Extract all unique words, ignoring case
- Eliminate **stop-words** : “a” , “and” , “the” , ...
- Eliminate non-content-bearing high-frequency and low-frequency words (using heuristic criteria)
- Extract word phrases (“New York”)
- Reduce words to their root/stem (eliminating plurals, tenses, pre/suffixes)
- Assign a unique integer between 1 and w to remaining w words
- For each document, count no. of occurrences of each word

Vector Space Model for Documents



- $A_{ji} = t_{ji} \times g_j \times s_i$

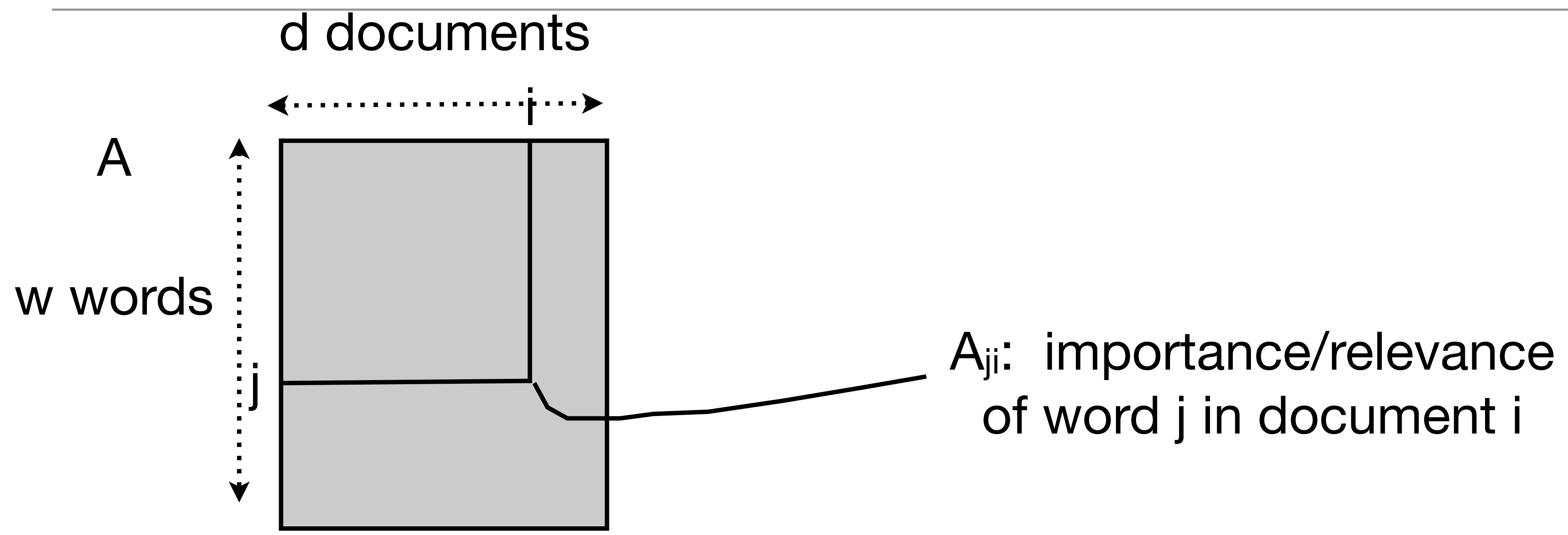
  t_{ji}: doc-term frequency; no. of times word j in document i

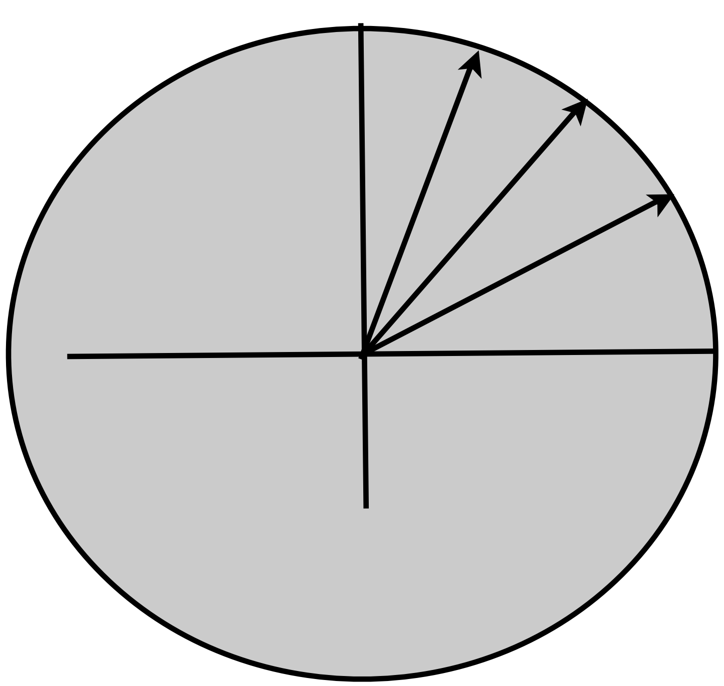
  g_j: importance of word i in entire document collection;
  e.g. $\log \frac{d}{d_j}$, where d_j is no. of documents that contains word j

  $s_i = 1/\sqrt{\sum_{j=1}^{w} (t_{ji} g_j)^2}$: normalization for document i.

- Note that columns of A are normalized: $\|a_i\|_2 = 1$.

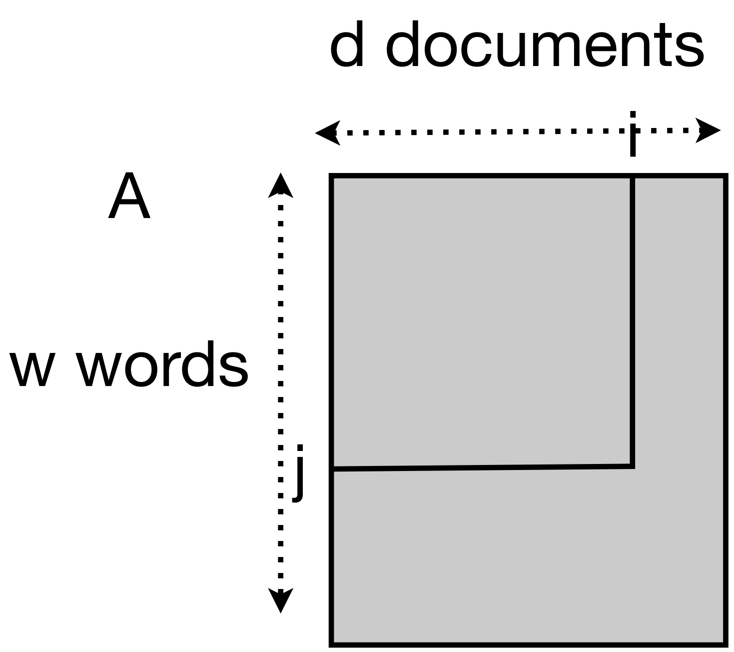
Vector Space Model for Documents





- Note that columns of A are normalized: $\|a_i\|_2 = 1$.

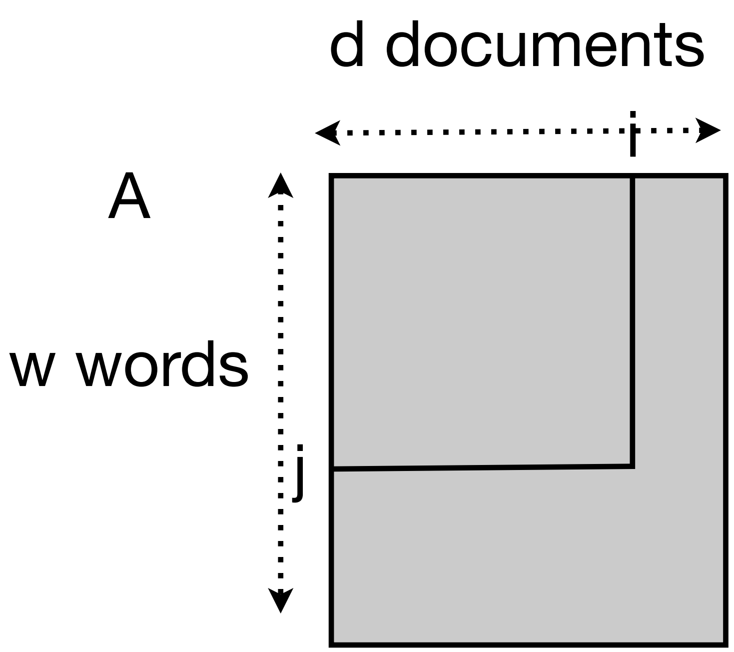
Query



Suppose we query
"data mining"

$$q = \begin{array}{c|l} \left[\begin{array}{c} 0 \\ 0 \\ \vdots \\ 1/\sqrt{2} \\ \vdots \\ 1/\sqrt{2} \\ \vdots \\ 0 \end{array}\right] & \begin{array}{l} \\ \\ \\ \leftarrow \text{data} \\ \\ \leftarrow \text{mining} \\ \\ \\ \end{array} \end{array} \quad w$$

Query

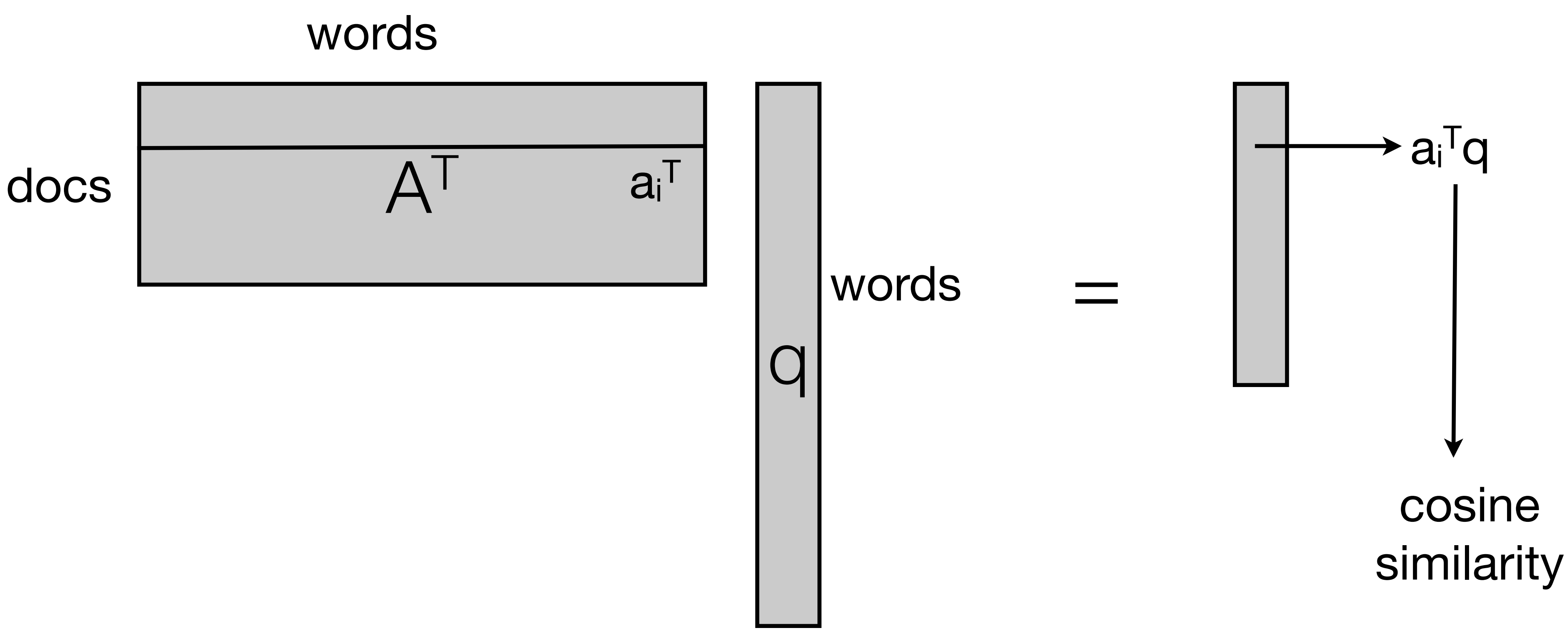


Suppose we query
“data mining”

$$q = \left[\begin{array}{c} 0 \\ 0 \\ \vdots \\ 1/\sqrt{2} \\ \vdots \\ 1/\sqrt{2} \\ \vdots \\ 0 \end{array}\right] \begin{array}{l} \\ \\ w \\ \longleftarrow \text{data} \\ \\ \longleftarrow \text{mining} \\ \\ \\ \end{array}$$

$A^T q$: Scores of documents with respect to query

Query Retrieval



$$a_i^T q = \|a_i\|_2 \|q\|_2 \cos\theta$$

$A^T q$: Scores of documents with respect to query

Caveats with using word-document matrix

- **Size:** Even after pruning and following pre-processing steps outlined earlier, the number of words would be in the tens of thousands

- **Word Senses:**

  - Synonymy: different words have similar meaning
    e.g. searching for MRI, or "Magnetic Resonance Imaging"

  - Polysemy: One words may have different meanings depending on context
    e.g. "mining" could refer to "data mining" or "coal mining"

Caveats with using word-document matrix

- **Size:** Even after pruning and following pre-processing steps outlined earlier, the number of words would be in the tens of thousands

- **Word Senses:**

  - Synonymy: di
    e.g. searching ging”

  - Polysemy: On depending on context
    e.g. “mining” ining”

Caveats with using word-document matrix

- Imagine that we could convert word-document matrix, into an ideal "semantic term" - document matrix

- Imagine that given a query (which like a document is a set of words), we can convert it into a set of "semantic terms"

  - Then we could compute query-document similarities as before

  - We humans do this all the time

  - Think of this ideal "semantic term" - document matrix as "approximating" our word-document matrix

How Good is my Matrix Approximation?

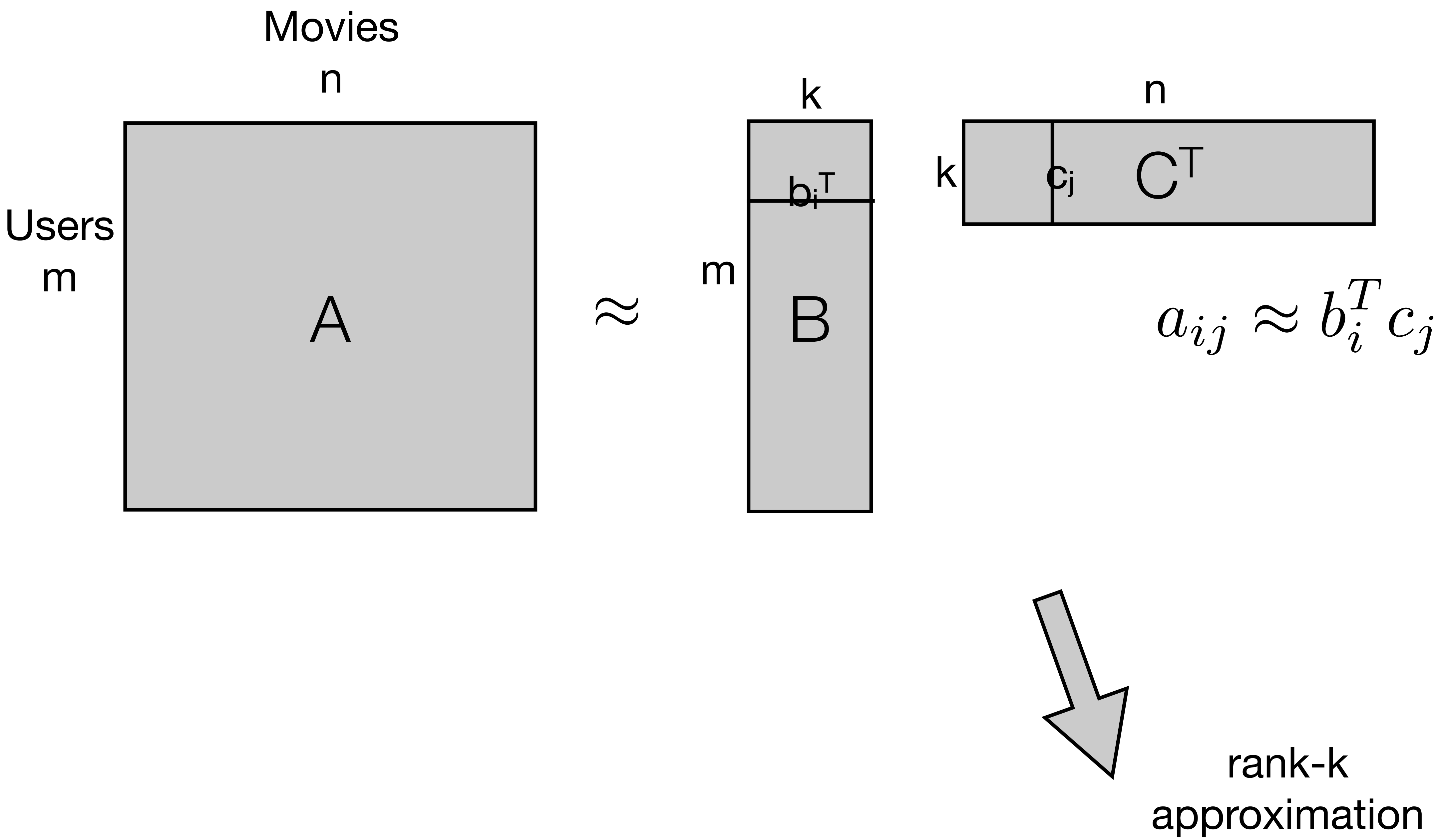
- Bill Gates, Lord Kelvin: You can't really make progress unless you can **measure**!

- Suppose I want to **approximate** a matrix A by another matrix B.
  - How good is B as an approximation?
  - Matrices also have norms ||| A |||
  - Use a matrix norm to measure approximation error: ||| A - B |||
  - A popular matrix norm is the Frobenius norm:

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

How do I Approximate my Matrix?

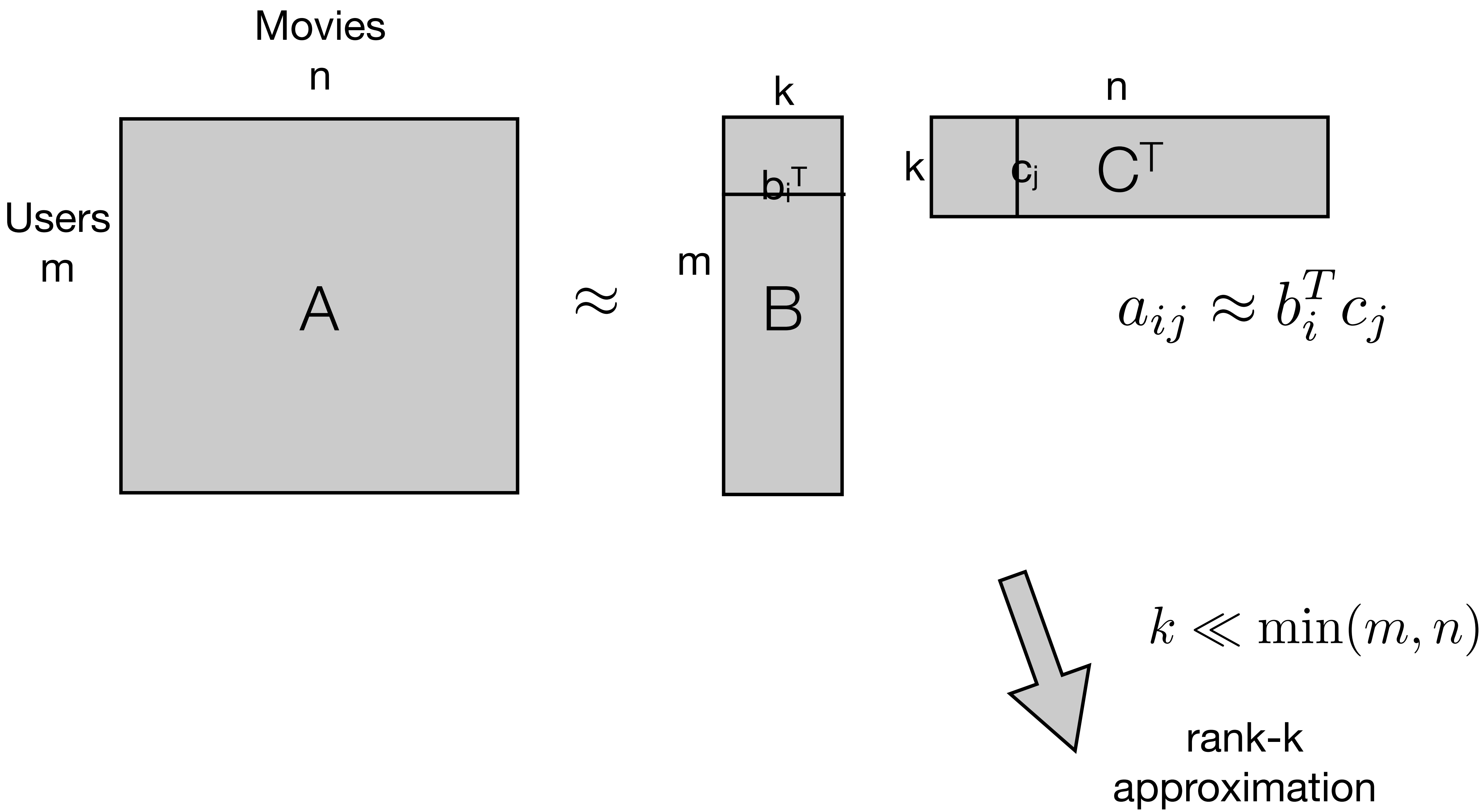
- A popular way to approximate a big matrix: Low-rank Approximation



$$a_{ij} \approx b_i^T c_j$$

Matrix Approximation

- A popular way to approximate a big matrix: Low-rank Approximation



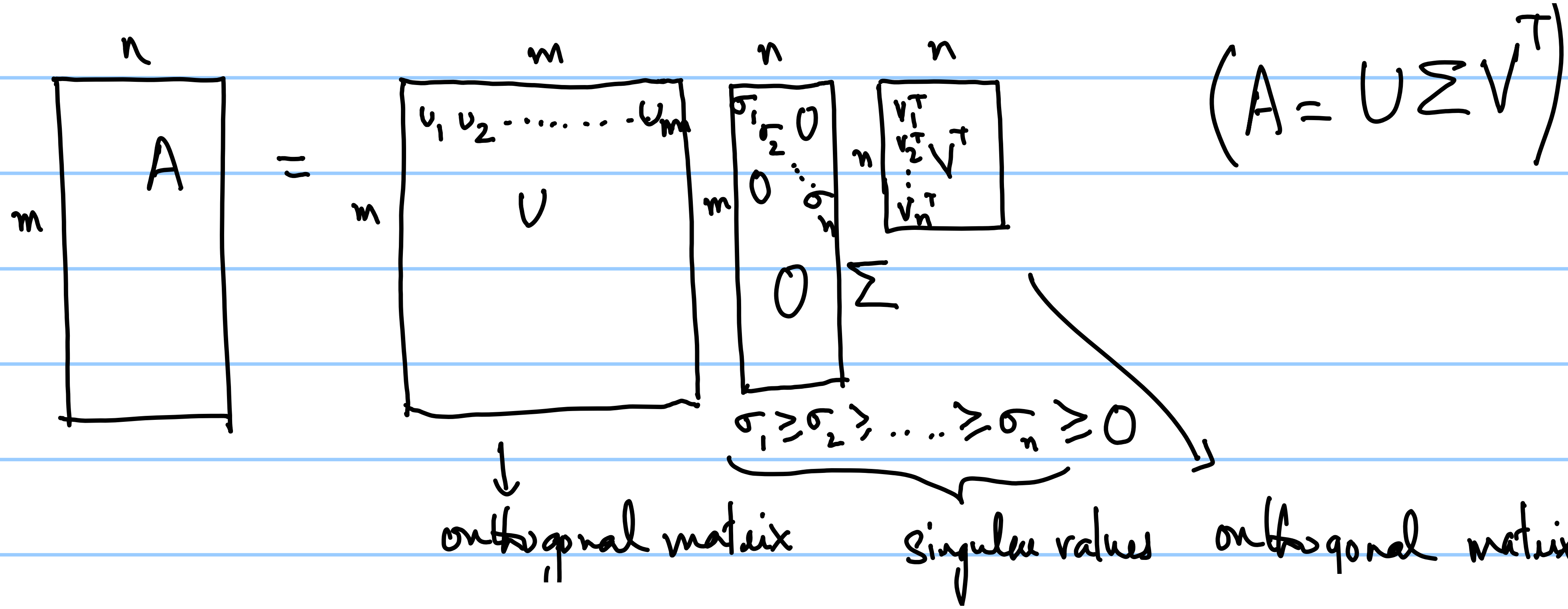
Best Low-rank Approximation

- Given A, and k, what is the best rank-k approximation?
- Find matrices B, C **of rank-**k which solve:

$$\min_{B,C} \|A - B\,C^T\|_F.$$

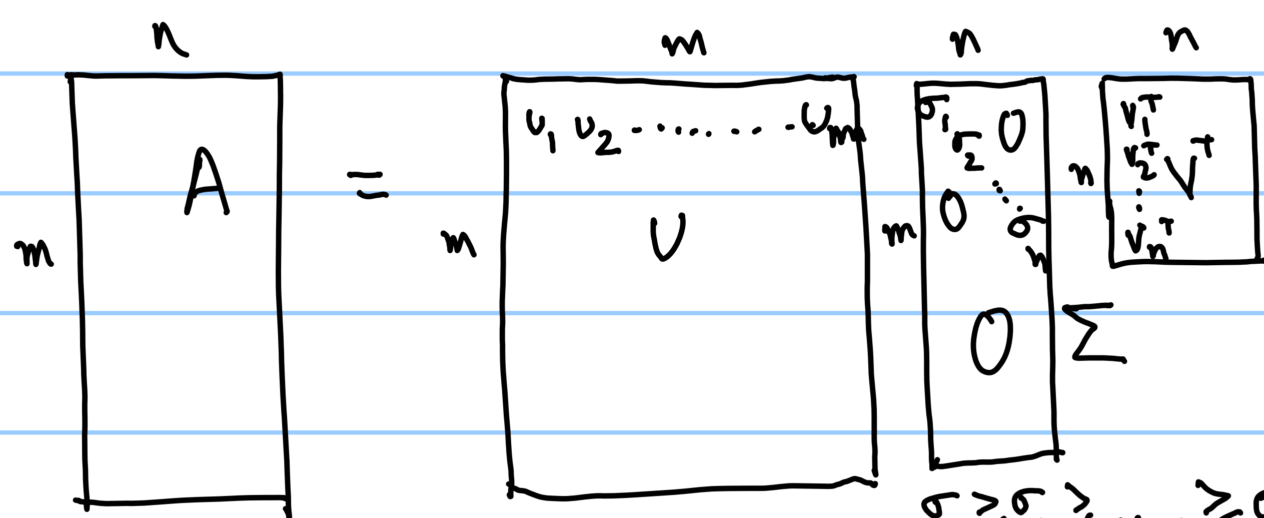
- Solution is given by SVD: Singular Value Decomposition of A

SVD; Singular Value Decomposition



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

SVD; Singular Value Decomposition



$$\left(A = U\Sigma V^T\right)$$

$$\underbrace{\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0}_{\text{singular values}}$$

orthogonal matrix

$$U^T U = I$$

$$\text{|||}$$

$$u_i^T u_j = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

left singular vectors

orthogonal matrix

$$V^T V = I$$

$$v_i^T v_j = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

right singular vectors

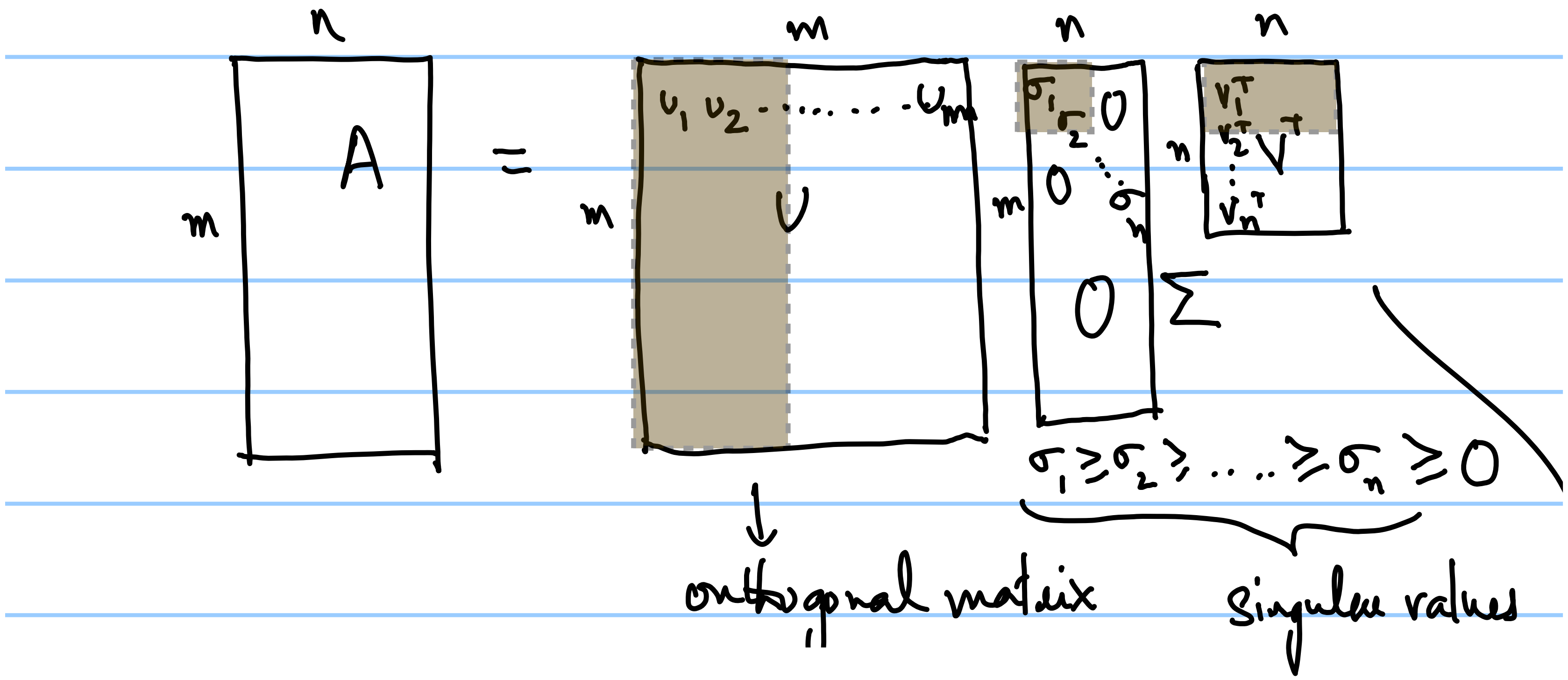
Low-Rank Approximation Using SVD

$$\begin{aligned}\mathbf{A}_k &= \mathbf{U}_k \boldsymbol{\Sigma}_k \mathbf{V}_k^T \\ &= [\mathbf{u}_1, \mathbf{u}_2, \cdots \mathbf{u}_k] \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \multicolumn{4}{c}{\dotfill} \\ 0 & 0 & \cdots & \sigma_k \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_k^T \end{bmatrix}\end{aligned}$$

where $U_k^T U_k = I$, $V_k^T V_k = I$, and

$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_k$.

Low-Rank Approximation Using SVD



Low-Rank Approximation Using SVD

$$
\begin{aligned}
\mathbf{A}_k &= \mathbf{U}_k \boldsymbol{\Sigma}_k \mathbf{V}_k^T \\
&= [\mathbf{u}_1, \mathbf{u}_2, \cdots \mathbf{u}_k] \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \multicolumn{4}{c}{\dotfill} \\ 0 & 0 & \cdots & \sigma_k \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_k^T \end{bmatrix}
\end{aligned}
$$

Important Result: Among all rank-k approximations of A, the best is A_k:

$\min_{B:\, \mathrm{rank}(B) \leq k} \|A - B\|_F \quad \leftarrow$ minimizing $B = A_k$.

Latent Semantic Indexing (LSI)

$$
\begin{aligned}
\mathbf{A}_k &= \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^T \\
&= [\mathbf{u}_1, \mathbf{u}_2, \cdots \mathbf{u}_k] \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \sigma_k \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_k^T \end{bmatrix}
\end{aligned}
$$

- Use A_k instead of A for computing query-document similarities.
- Use $A_k^T\, q$ instead of $A^T\, q$.

LSI Contd.

Note that $\mathbf{A}_k^T\mathbf{q} = (\mathbf{V}_k\mathbf{\Sigma}_k)\left(\mathbf{U}_k^T\mathbf{q}\right)$.

$$\mathbf{U}_k^T\mathbf{q} \quad = \quad \begin{bmatrix} \mathbf{u}_1^T\mathbf{q} \\ \mathbf{u}_2^T\mathbf{q} \\ \vdots \\ \mathbf{u}_k^T\mathbf{q} \end{bmatrix},$$

- Each component $u_i^T q$ of the vector $U_k^T q$ is the **projection** of query vector q onto the singular vector u_i.
- The w-dimensional query vector q is reduced to k dimensions
- The singular vectors (u_i's) do not span all possible documents, but span "important" part of the space

LSI Contd.

Note that $\mathbf{A}_k^T \mathbf{q} = (\mathbf{V}_k \mathbf{\Sigma}_k) \left(\mathbf{U}_k^T \mathbf{q} \right)$.

$$A = U_k \Sigma_k V_k^T + U_{w-k} \Sigma_{w-k} V_{w-k}^T$$

$$U_k^T A = \Sigma_k V_k^T \qquad \ldots\ U \text{ is orthonormal}$$

LSI Contd.

Note that $\mathbf{A}_k^T\mathbf{q} = (\mathbf{V}_k\mathbf{\Sigma}_k)\left(\mathbf{U}_k^T\mathbf{q}\right)$.

$$A = U_k\Sigma_k V_k^T + U_{w-k}\Sigma_{w-k}V_{w-k}^T$$

$$U_k^T A = \Sigma_k V_k^T \qquad \ldots\ U \text{ is orthonormal}$$

- Thus, $V_k\Sigma_k$ is the projection of the documents (columns of A) onto U_k.
- So that $A_k^T q = (V_k\Sigma_k)(U_k^T q)$ can be interpreted as dot-product between projected documents and projected query!

LSI: Alternatively

1. For the entire document collection, form $\mathbf{V}_k\mathbf{\Sigma}_k$.

2. For a new query $\mathbf{q}$, form $\mathbf{U}_k^T\mathbf{q}$.

3. Compute $\mathbf{z} = (\mathbf{V}_k\mathbf{\Sigma}_k)(\mathbf{U}_k^T\mathbf{q})$ and return the document i with large $\mathbf{z}_i$ values as being the most relevant.

Example

Suppose we are give the following $d = 9$ documents:

c1: Human machine interface for Lab ABC computer applications

c2: A survey of user opinion of computer system response time

c3: The EPS user interface management system

c4: System and human system engineering testing of EPS

c5: Relation of user-perceived response time to error measurement.

m1: The generation of random, binary, unordered trees

m2: The intersection graph of paths in trees

m3: Graph minors IV: Widths of trees and well-quasi-ordering

m4: Graph minors: A survey

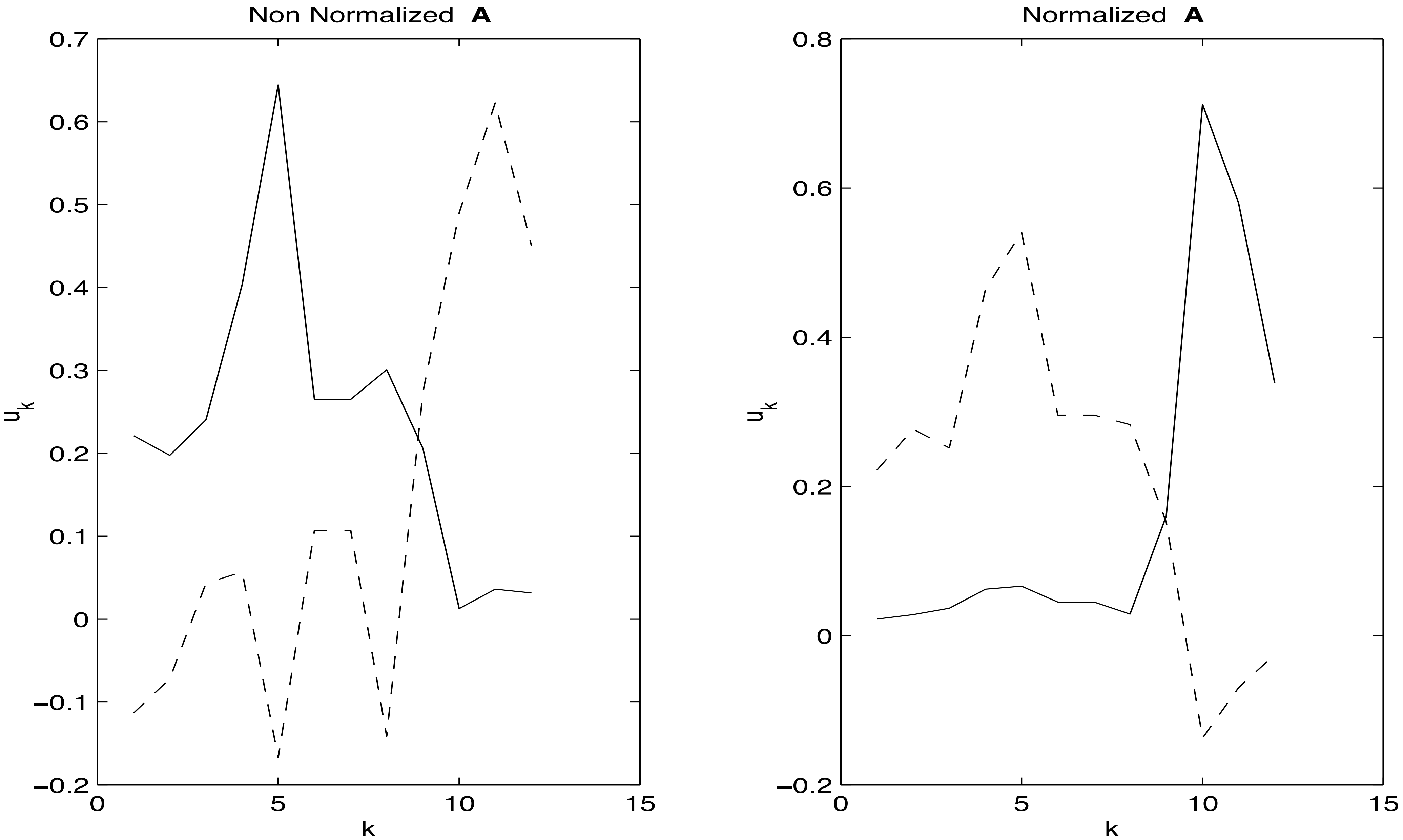
Term-Doc. Matrix (Vector Space Model)

Documents

Terms	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

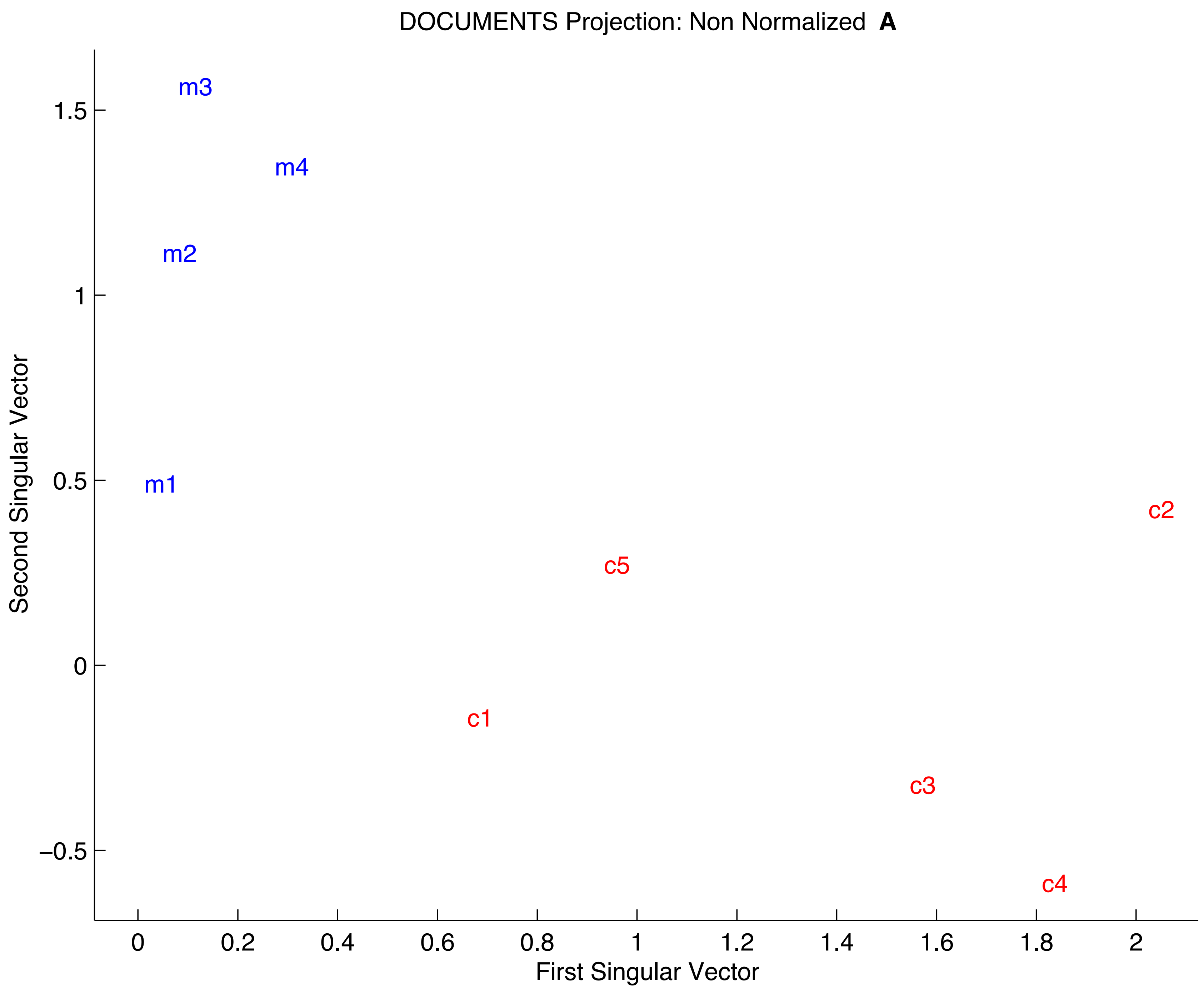
A: Nine document vectors, each in a 12 dimensional word space

Normalized vs Unnormalized **A**

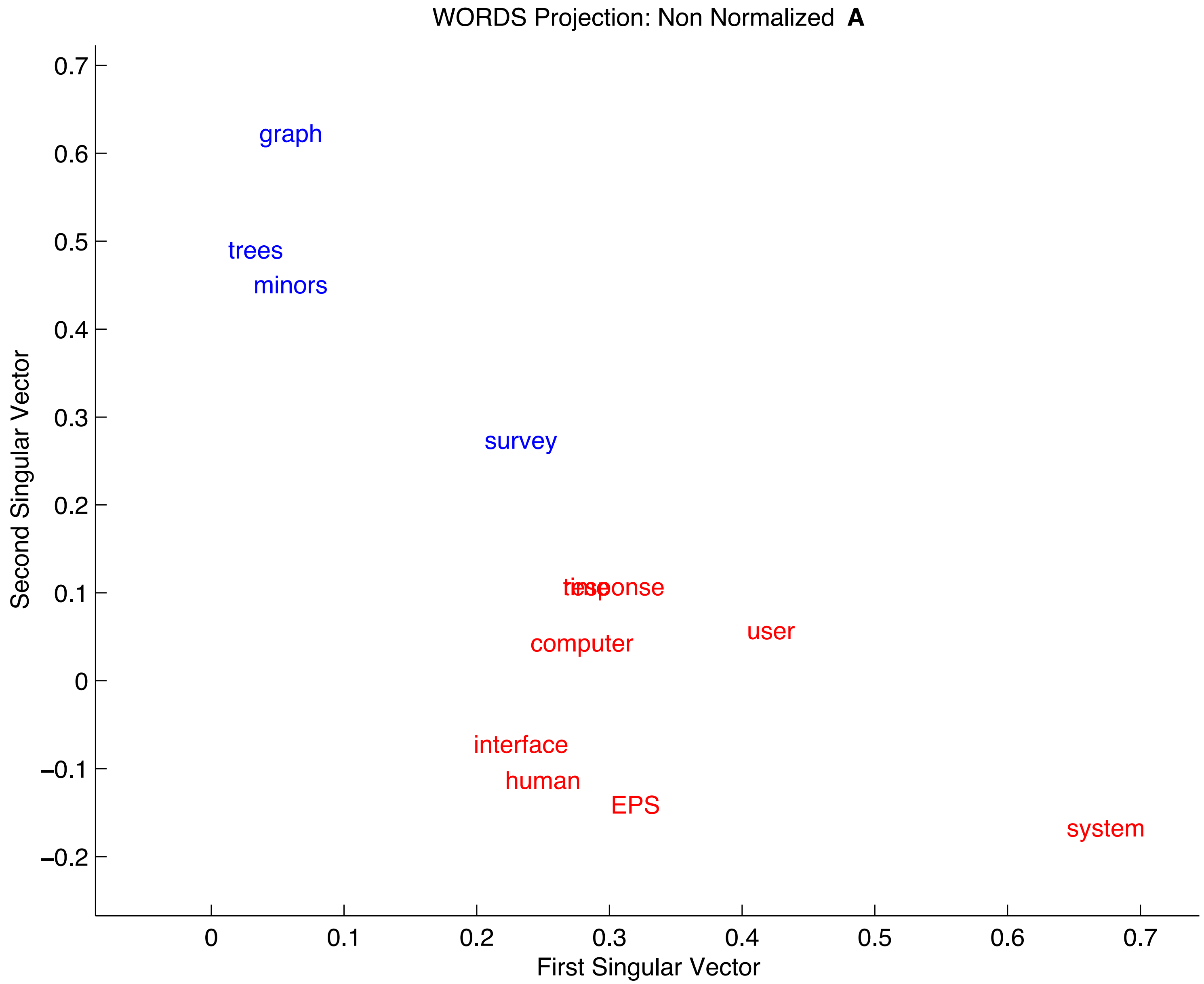


First two left-singular vectors (solid line: first; dashed-line: second)

Documents projected onto 1st two sing. vectors



Words projected onto same two right sing. vectors



: Words projected on 2-D space from **A**

LSI: Drawbacks

- Computationally expensive: (a) typically A is large, and (b) many singular vectors are required (k = 100 to 500) and the SVD software seems to take time that is quadratic in k

- A common question: how do singular vectors capture the concepts of a document collection?

- LSI needs long queries to work well; but typically we use short queries when searching. Thus, LSI is not used in any commercial engine.

- Uses the vector space model for documents, which has its caveats (sequencing information of words in the document is lost, we wouldn't want a document just containing the few words of a query etc.)