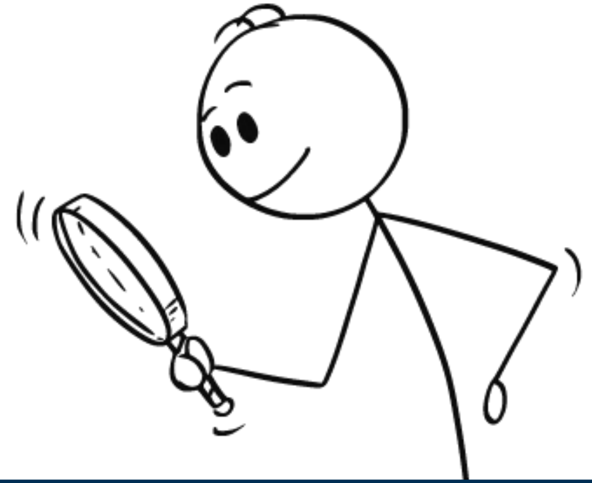


CSE 4/535
Information Retrieval

Sayantan Pal
PhD Student, Department of CSE
338Z Davis Hall

University at Buffalo
Department of CSE

Before we start

1. Project 2 grades will be released by Sunday (November 12th)
2. Project 3 (Final Project will be released this weekend), make sure to form the teams, team registration sheet will be released Monday (November 13th)
3. Midterm 2 - November 13th (Detailed instructions will be shared soon)
4. Today's lecture
   a. Link Analysis - Topic Specific Pagerank, HITS
   b. Latent Semantic Indexing

Recap - Previous Class

1. Anchor Text
2. Page Rank
    a. Power Iterative Method

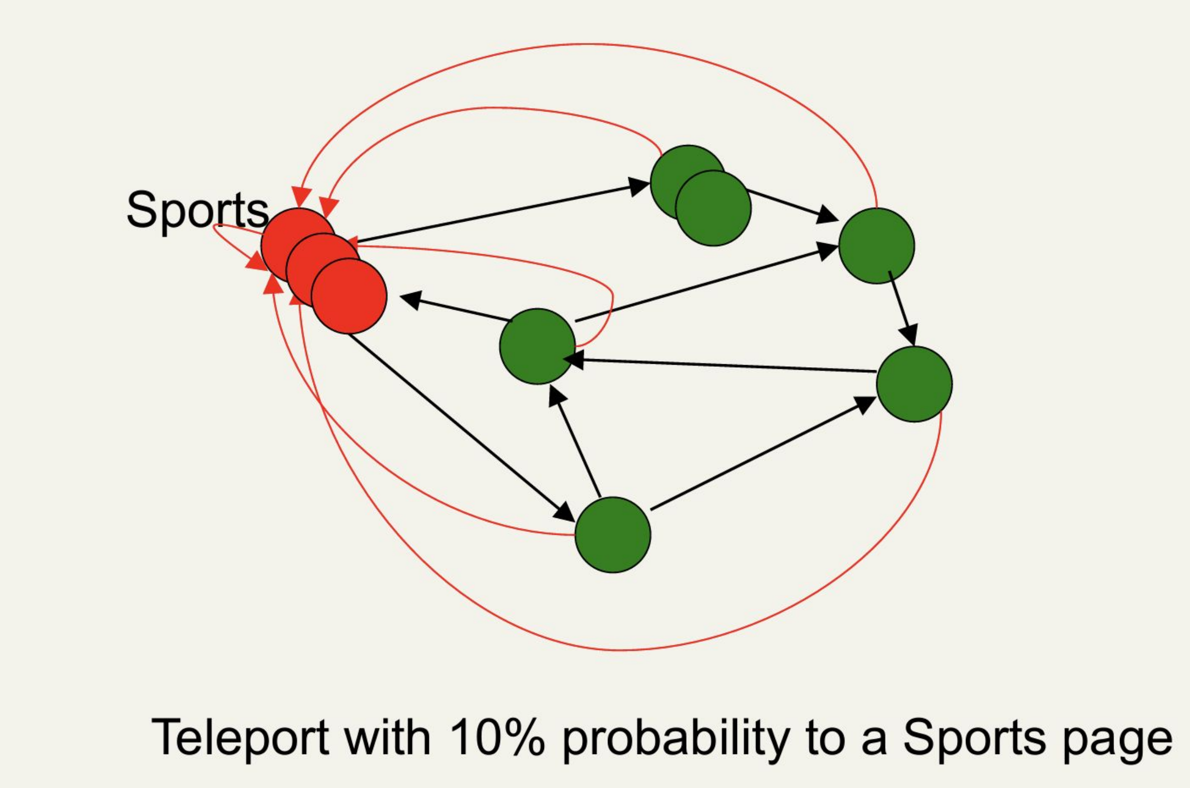
Topic Specific Pagerank

- Conceptually, we use a random surfer who teleports, with say 10% probability, using the following rule:
  - Selects a category (say, one of the 16 top level ODP categories) based on a query & user -specific distribution over the categories
  - Teleport to a page uniformly at random within the chosen category
- Sounds hard to implement: can’t compute PageRank at query time!

Topic Specific Pagerank

- **Offline**:Compute pagerank for *individual* categories
  - Query independent as before
  - Each page has multiple pagerank scores – one for each ODP category, with teleportation only to that category
- **Online**: Distribution of weights over categories computed by query context classification
  - Generate a dynamic pagerank score for each page - weighted sum of category-specific pageranks

Non-uniform Teleportation



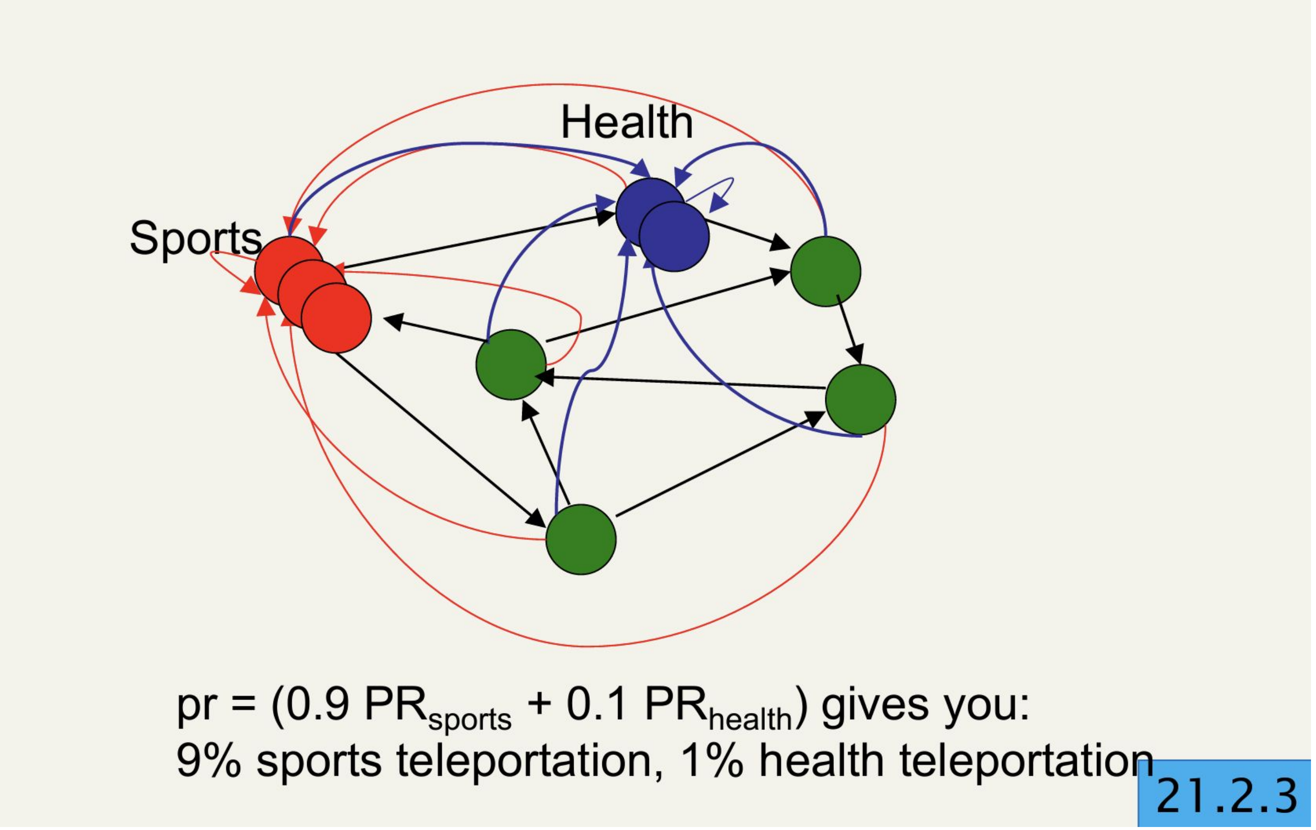
Interpretation of Composite Score

- For a set of personalization vectors $\{\mathbf{v}_j\}$

$$\sum_j [w_j \cdot PR(W, \mathbf{v}_j)] = PR(W, \sum_j [w_j \cdot \mathbf{v}_j])$$

- Weighted sum of rank vectors itself forms a valid rank vector, because PR() is linear wrt $\mathbf{v}_j$

Interpretation



HITS

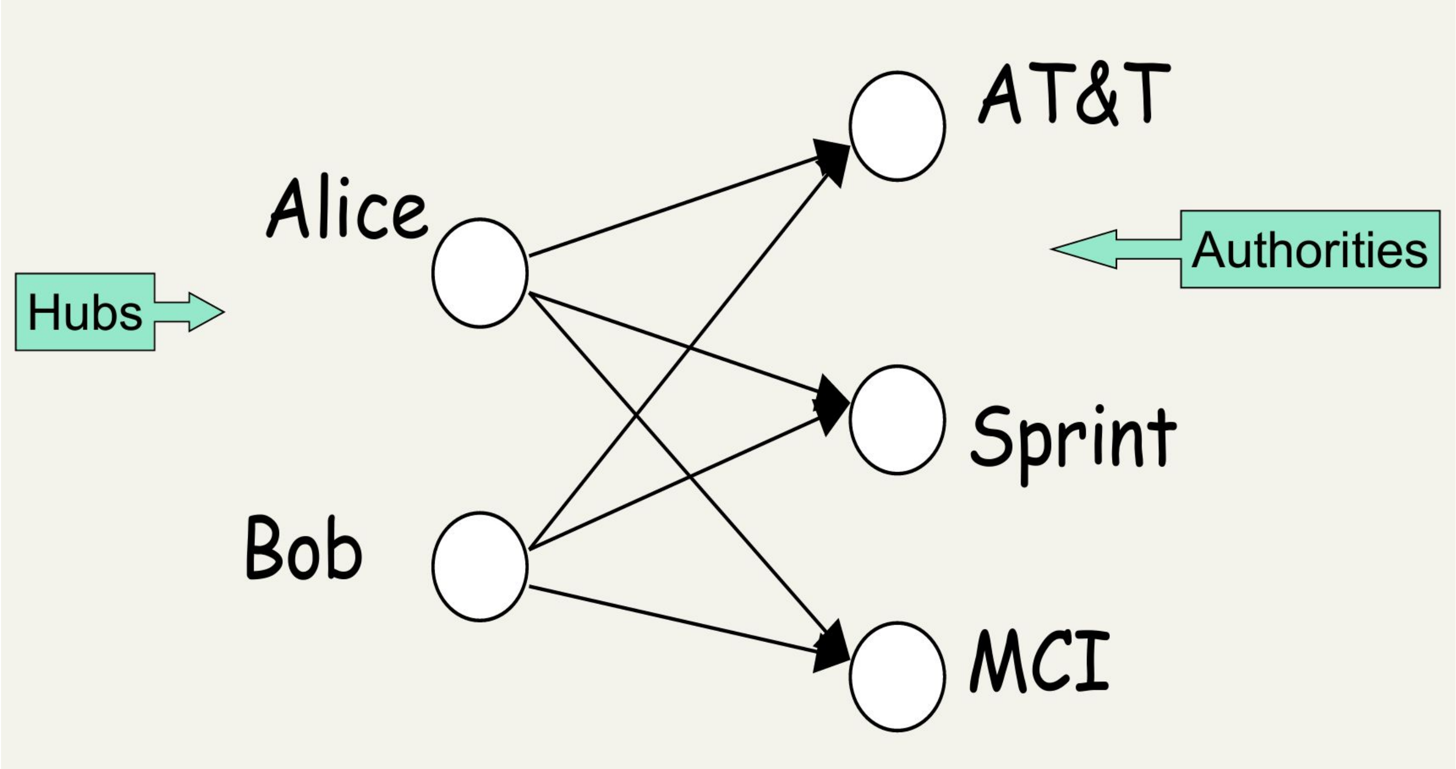
Hyperlink-Induced Topic Search (HITS)

- In response to a query, instead of an ordered list of pages each meeting the query, find <u>two</u> sets of inter-related pages:
  - *Hub pages* are good lists of links on a subject.
    - e.g., "Bob's list of cancer-related links."
  - *Authority pages* occur recurrently on good hubs for the subject.
- Best suited for "broad topic" queries rather than for page-finding queries.
- Gets at a broader slice of common *opinion*.

Hubs and Authorities

- Thus, a good hub page for a topic *points* to many authoritative pages for that topic.
- A good authority page for a topic is *pointed* to by many good hubs for that topic.
- Circular definition - will turn this into an iterative computation.

The hope



High-level scheme

- Extract from the web a base set of pages that *could* be good hubs or authorities.
- From these, identify a small set of top hub and authority pages;

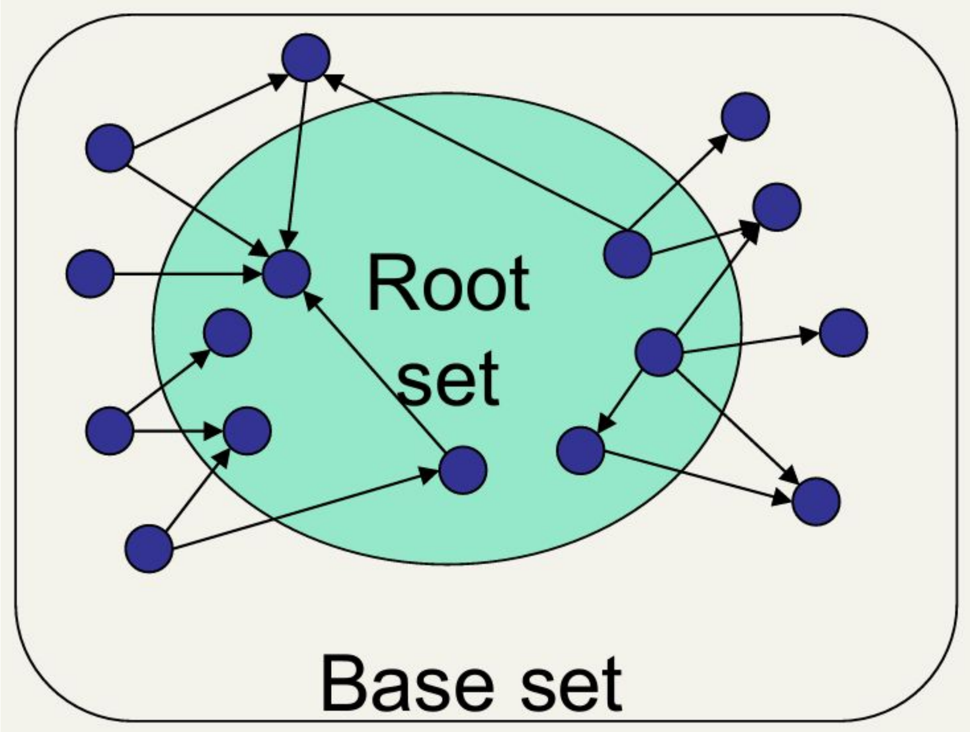
  →iterative algorithm.

Base set

- Given text query (say ***browser***), use a text index to get all pages containing ***browser.***
  - Call this the root set of pages.
- Add in any page that either
  - points to a page in the root set, or
  - is pointed to by a page in the root set.
- Call this the base set.

21.3

Visualization



Assembling the base set

- Root set typically 200-1000 nodes.
- Base set may have up to 5000 nodes.
- How do you find the base set nodes?
  - Follow out-links by parsing root set pages.
  - Get in-links (and out-links) from a *connectivity server.*
  - (Actually, suffices to text-index strings of the form ***href="URL"*** to get in-links to *URL*.)

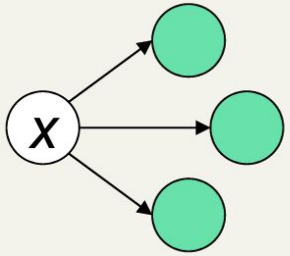
Distilling hubs and authorities

- Compute, for each page x in the base set, a <u>hub score</u> $h(x)$ and an <u>authority score</u> $a(x)$.
- Initialize: for all x, $h(x) \leftarrow 1$; $a(x) \leftarrow 1$;
- Iteratively update all $h(x)$, $a(x)$; Key
- After iterations
  - output pages with highest $h()$ scores as top hubs
  - highest $a()$ scores as top authorities.

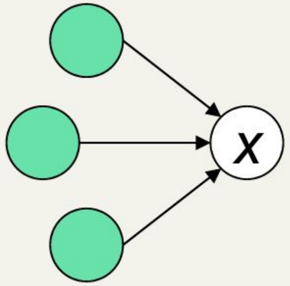
Iterative update

- Repeat the following updates, for all *x*:

$$h(x) \leftarrow \sum_{x \mapsto y} a(y)$$



$$a(x) \leftarrow \sum_{y \mapsto x} h(y)$$



Scaling

- To prevent the *h()* and *a()* values from getting too big, can scale down after each iteration.
- Scaling factor doesn't really matter:
  - we only care about the *relative* values of the scores.

How many iterations?

- Claim: relative values of scores will converge after a few iterations:
  - in fact, suitably scaled, *h()* and *a()* scores settle into a steady state!
  - proof of this comes later.
- We only require the relative orders of the *h()* and *a()* scores - not their absolute values.
- In practice, ~5 iterations get you close to stability.

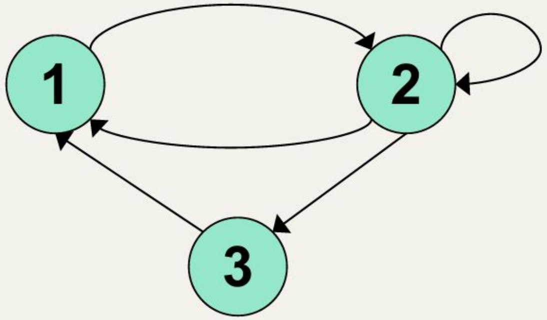
21.3

Things to note

- Pulled together good pages regardless of language of page content.
- Use *only* link analysis after base set assembled
  - iterative scoring is query-independent.
- Iterative computation after text index retrieval - significant overhead.

Proof of convergence

- $n \times n$ <u>adjacency matrix</u> **A**:
  - each of the n pages in the base set has a row and column in the matrix.
  - Entry $A_{ij} = 1$ if page i links to page j, else = 0.



$$\begin{array}{c|ccc|} & 1 & 2 & 3 \\ \hline 1 & 0 & 1 & 0 \\ 2 & 1 & 1 & 1 \\ 3 & 1 & 0 & 0 \\ \hline \end{array}$$

Hub/authority vectors

- View the hub scores *h()* and the authority scores *a()* as vectors with *n* components.
- Recall the iterative updates

$$h(x) \leftarrow \sum_{x \mapsto y} a(y)$$

$$a(x) \leftarrow \sum_{y \mapsto x} h(y)$$

Rewrite in matrix form

- **h=Aa.**
- **a=A^{t}h.**

Recall A^t is the transpose of A.

Substituting, **h=AA^{t}h** and **a=A^{t}Aa.**

Thus, **h** is an eigenvector of **AA^t** and **a** is an eigenvector of **A^{t}A.**

Further, our algorithm is a particular, known algorithm for computing eigenvectors: the *power iteration* method.

Guaranteed to converge.

21.3

Example

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad A^T = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

$$\begin{array}{llllllll}
a(\text{yahoo}) & = & 1 & 1 & 1 & 1 & \cdots & 1 \\
a(\text{amazon}) & = & 1 & 1 & 4/5 & 0.75 & \cdots & 0.732 \\
a(\text{m'soft}) & = & 1 & 1 & 1 & 1 & \cdots & 1
\end{array}$$

$$\begin{array}{llllllll}
h(\text{yahoo}) & = & 1 & 1 & 1 & 1 & \cdots & 1.000 \\
h(\text{amazon}) & = & 1 & 2/3 & 0.71 & 0.73 & \cdots & 0.732 \\
h(\text{m'soft}) & = & 1 & 1/3 & 0.29 & 0.27 & \cdots & 0.268
\end{array}$$

AA^T

	C_1	C_2	C_3
1	3	2	1
2	2	2	0
3	1	0	1

A^TA

	C_1	C_2	C_3
1	2	1	2
2	1	2	1
3	2	1	2

Issues

- Topic Drift
  - Off-topic pages can cause off-topic "authorities" to be returned
    - E.g., the neighborhood graph can be about a "super topic"
- Mutually Reinforcing Affiliates
  - Affiliated pages/sites can boost each others' scores
    - Linkage between affiliated pages is not a useful signal

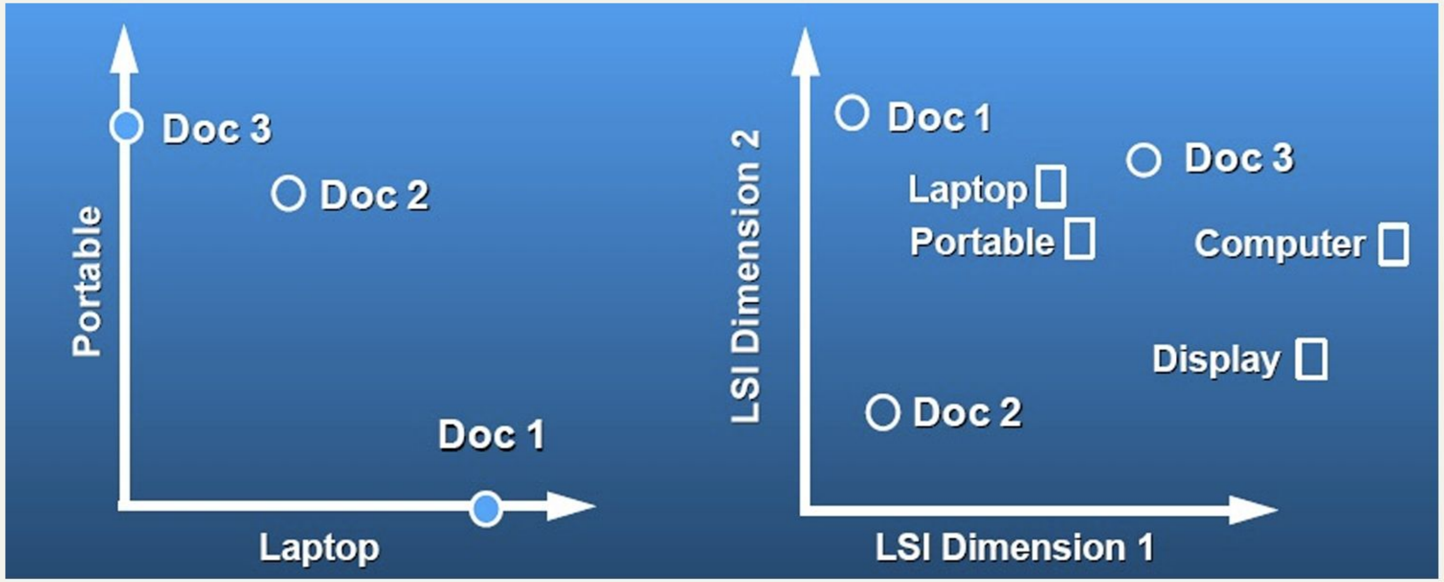
Page Rank and HITS

- Page Rank and HITS are two solutions to the same problem
  - What is the value of an inlink from S to D?
  - In the page rank model, the value of the link depends on the links **into** S
  - In the HITS model, it depends on the value of the other links **out of** S
- The destinies of Page Rank and HITS post-1998 were very different

Latent Semantic Indexing (LSI)

Latent Semantic Analysis

- **Latent semantic space**: illustrating example



courtesy of Susan Dumais

Singular Value Decomposition

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

$$A = \underbrace{U}_{m\times m}\underbrace{\Sigma}_{m\times n}\underbrace{V^T}_{V \text{ is } n\times n}$$

The columns of ***U*** are orthogonal eigenvectors of $\boldsymbol{AA^T}$.

The columns of ***V*** are orthogonal eigenvectors of $\boldsymbol{A^TA}$.

Eigenvalues $\lambda_1 \ldots \lambda_r$ of $\boldsymbol{AA^T}$ are the eigenvalues of $\boldsymbol{A^TA}$.

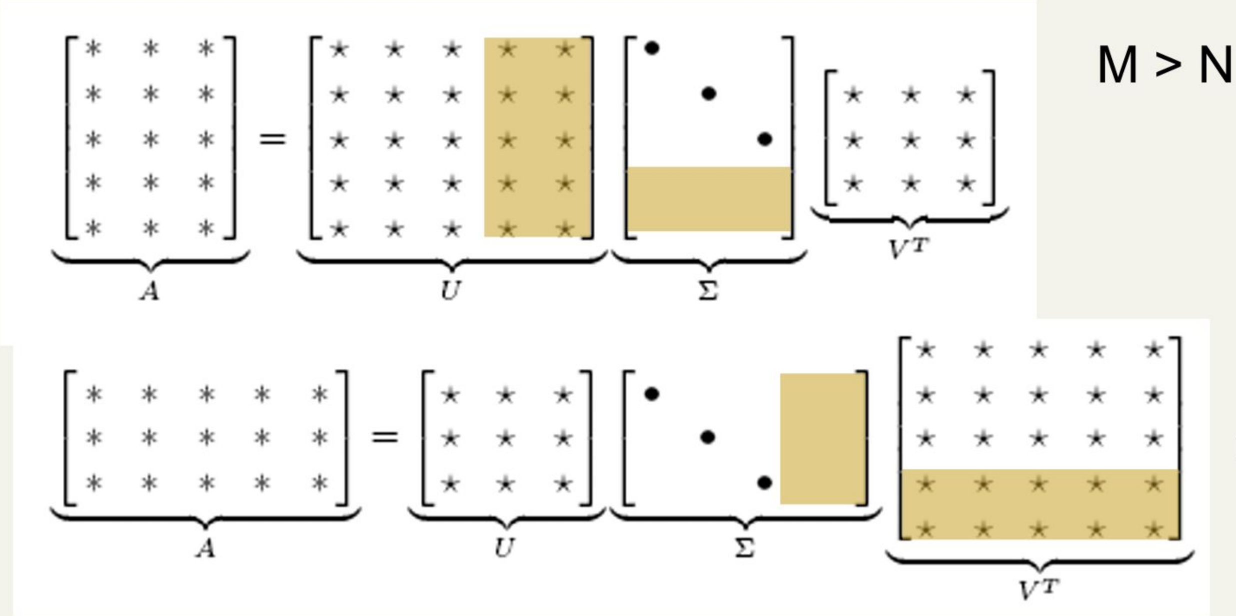
$$\sigma_i = \sqrt{\lambda_i}$$

$$\Sigma = diag(\sigma_1 \ldots \sigma_r)$$

Singular values.

Singular Value Decomposition

- Illustration of SVD dimensions and sparseness



M > N

M < N

SVD example

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

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

$$\underbrace{\begin{bmatrix} 0 & 2/\sqrt{6} & 1/\sqrt{3} \\ 1/\sqrt{2} & -1/\sqrt{6} & 1/\sqrt{3} \\ 1/\sqrt{2} & 1/\sqrt{6} & -1/\sqrt{3} \end{bmatrix}}_{U} \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{3} \\ 0 & 0 \end{bmatrix} \underbrace{\begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}}_{V^T}$$

Typically, the singular values arranged in decreasing order.

Low-rank Approximation

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

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

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

A_k and X are both $m \times n$ matrices.

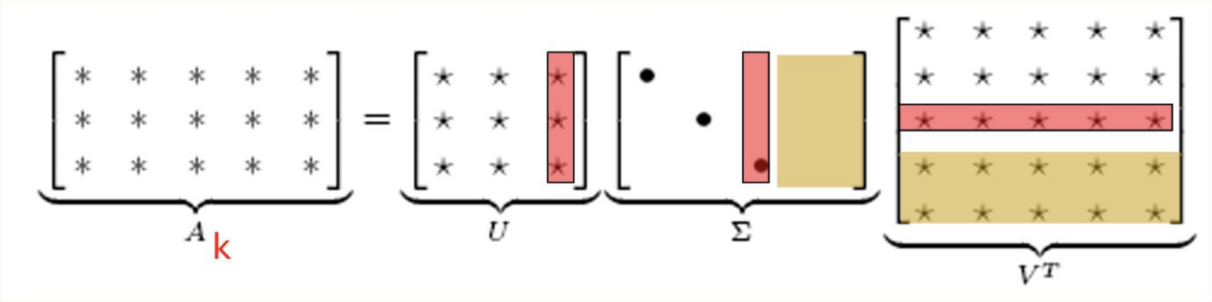
Typically, want $k << r$.

Low-rank Approximation

- Solution via SVD

$$A_k = U \operatorname{diag}(\sigma_1, \ldots, \sigma_k, \underbrace{0, \ldots, 0}) V^T$$

set smallest r-k
singular values to zero



$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

column notation: sum of rank 1 matrices

Approximation error

- How good (bad) is this approximation?
- It's the best possible, measured by the Frobenius norm of the error:

$$\min_{X:rank(X)=k} \|A - X\|_F = \|A - A_k\|_F = \sigma_{k+1}$$

where the σ_i are ordered such that $\sigma_i \geq \sigma_{i+1}$.

Suggests why Frobenius error drops as k increased.

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*

Latent Semantic Analysis via SVD

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

What it is..

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

Method

- Given C, construct its SVD in the form
  $C = U \Sigma V^T$
- Derive from Σ the matrix Σ_k formed by replacing by zeros, the r-k smallest singular values on the diagonal of Σ
- Compute and output $C_k = U \Sigma_k V^T$ as the rank-k approximation to C.

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.
  - Note that q can be any vector, e.g. a new document that needs to be mapped into this space

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)*
- Various extensions
  - Document clustering
  - Relevance feedback (modifying query vector)
- 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.

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

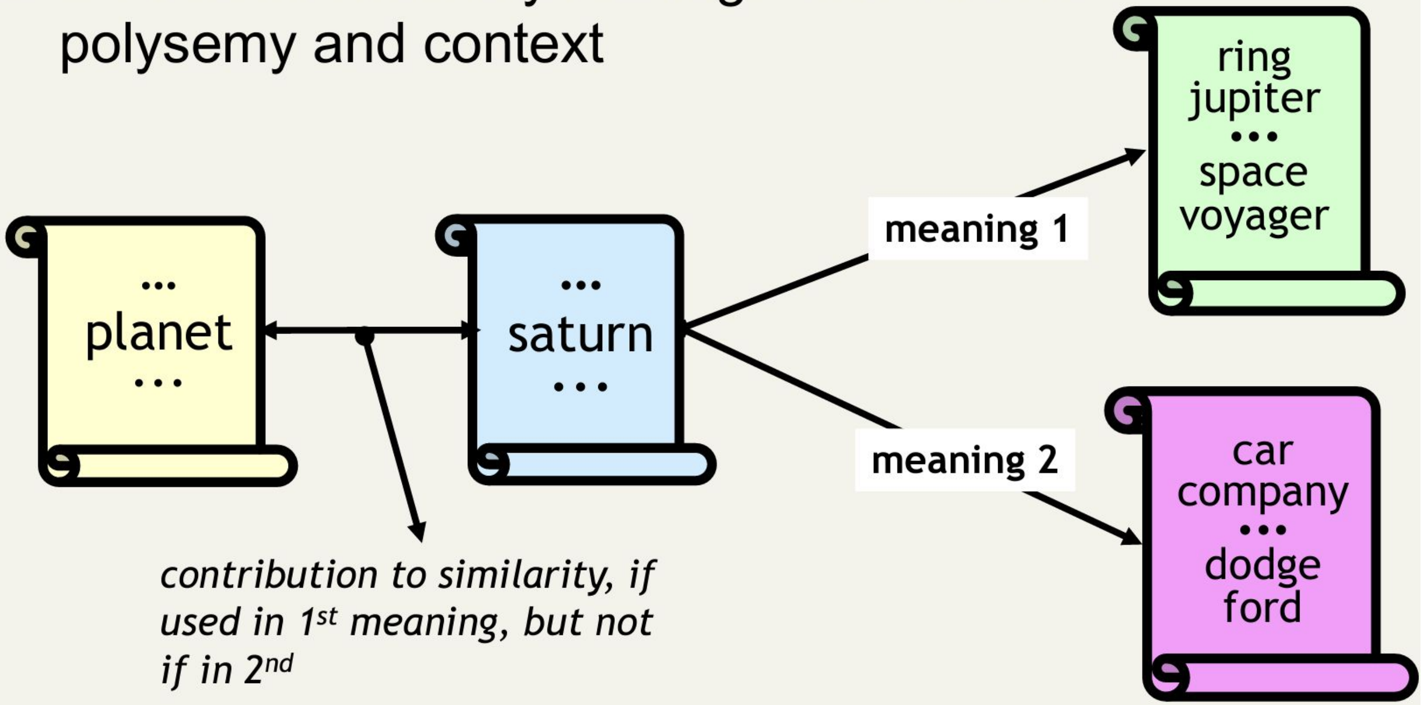
Problems with Lexical Semantics

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

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

Polysemy and Context

- Document similarity on single word level: polysemy and context



LSI: Empirical evidence

- Experiments on TREC 1/2/3 – Dumais
- Lanczos SVD code (available on netlib) due to Berry used in these expts
  - Running times of ~ one day on tens of thousands of docs
- Dimensions – various values 250-350 reported
  - (Under 200 reported unsatisfactory)
- Generally expect recall to improve – what about precision?

LSI: Empirical evidence

- 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

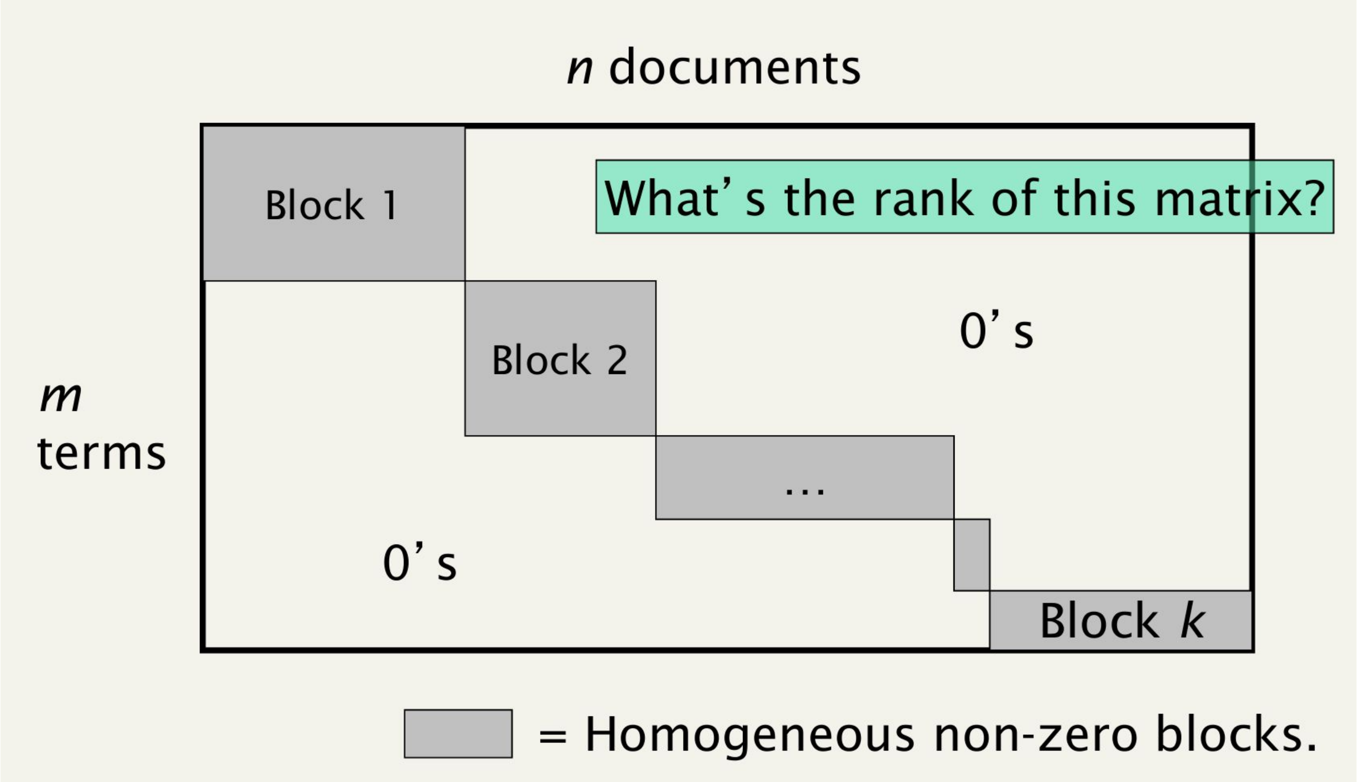
LSI: Failure modes

- Negated phrases
  - TREC topics sometimes negate certain query/terms phrases – automatic conversion of topics to
- Boolean queries
  - As usual, freetext/vector space syntax of LSI queries precludes (say) “Find any doc having to do with the following 5 companies”
- See Dumais for more.

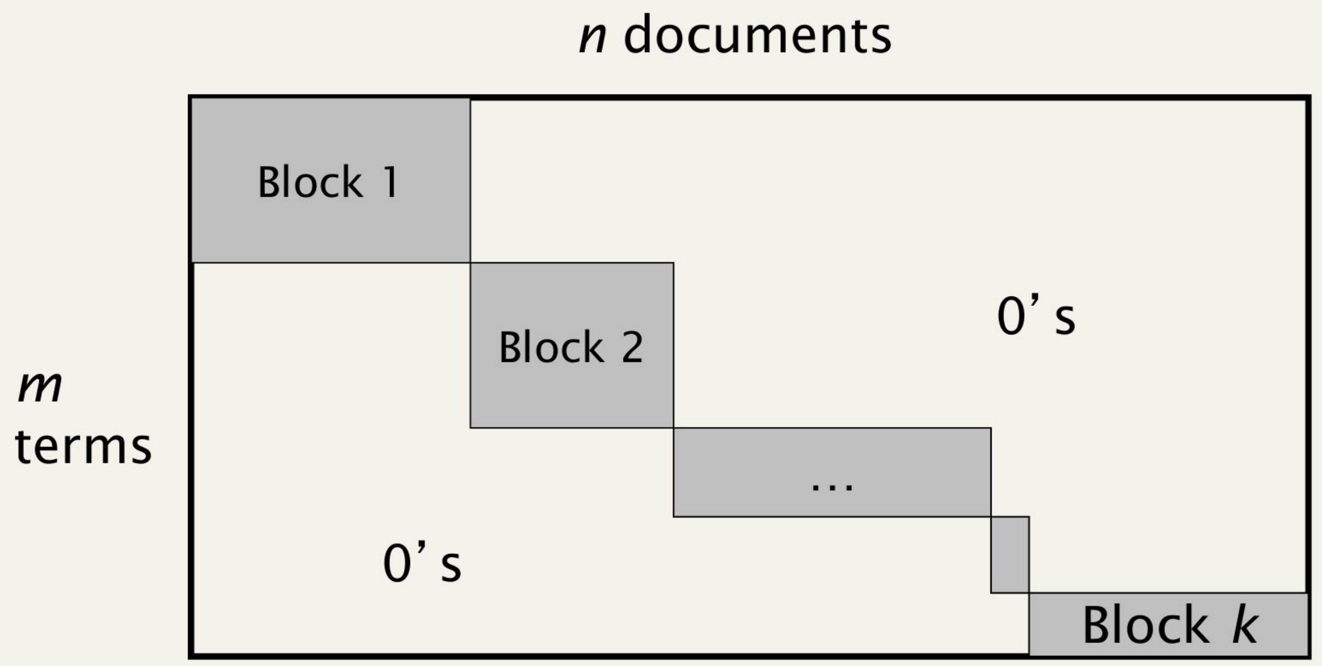
But why is this clustering?

- 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

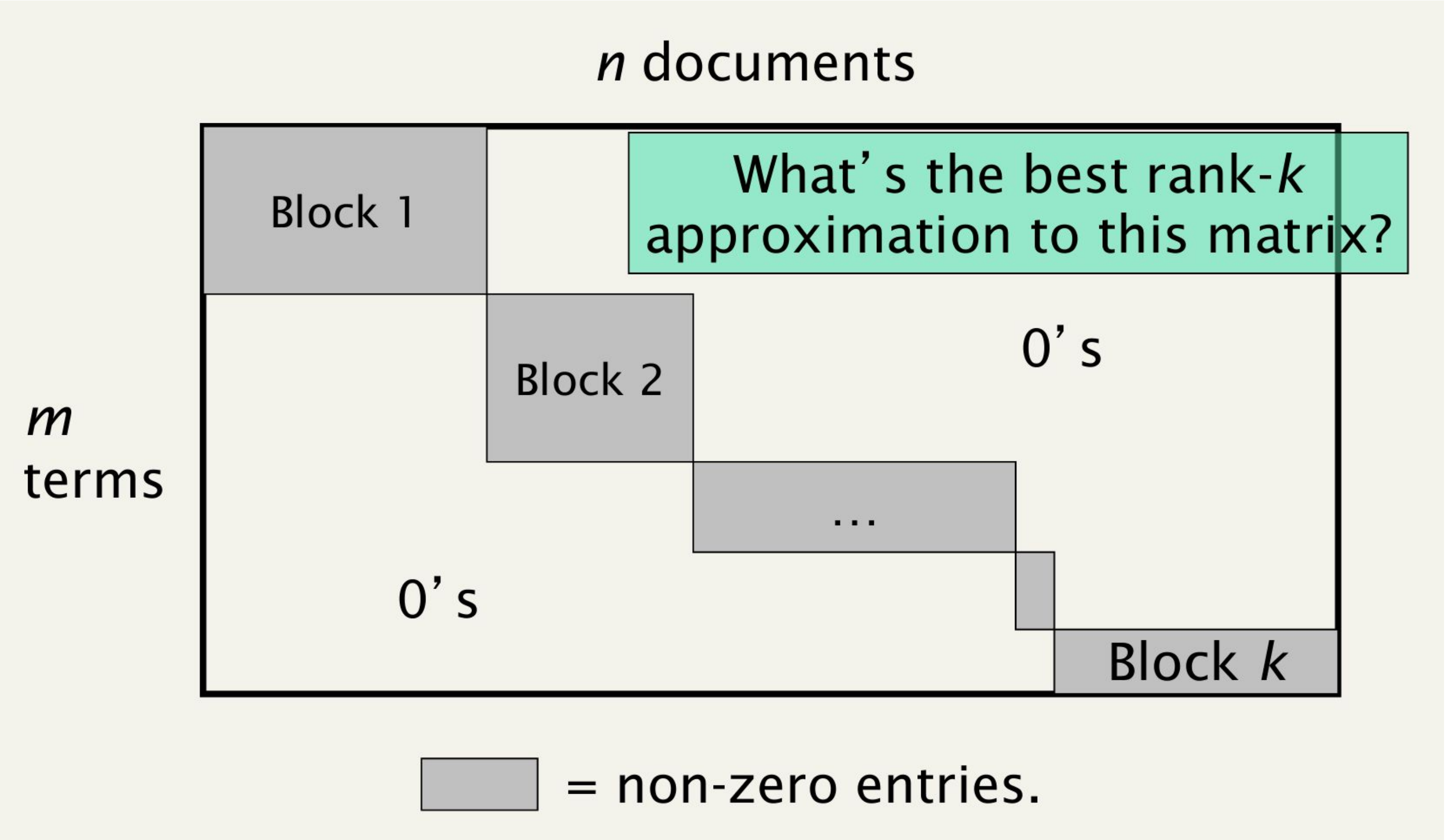


Intuition from block matrices

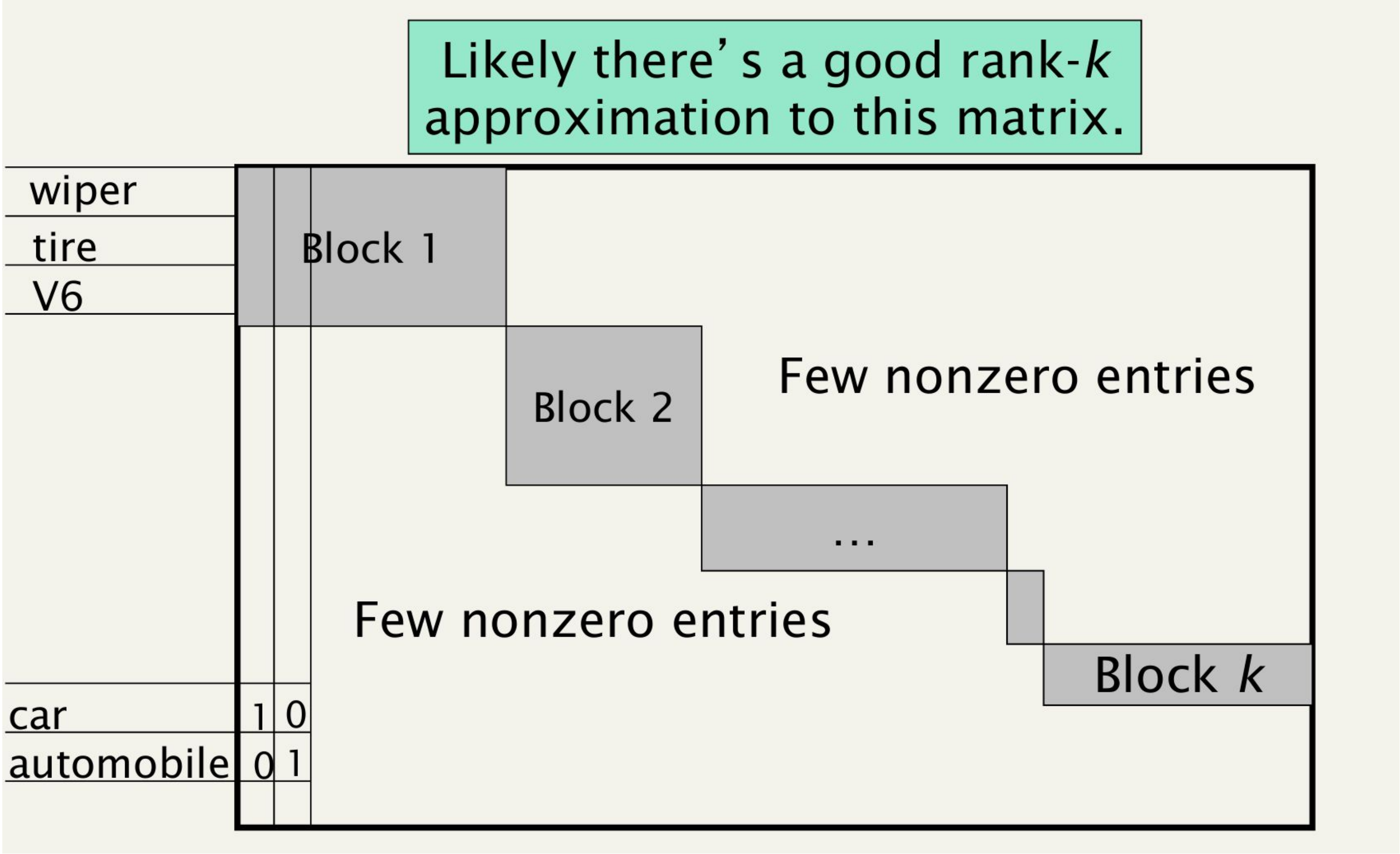


Vocabulary partitioned into k topics (clusters); each doc discusses only one topic.

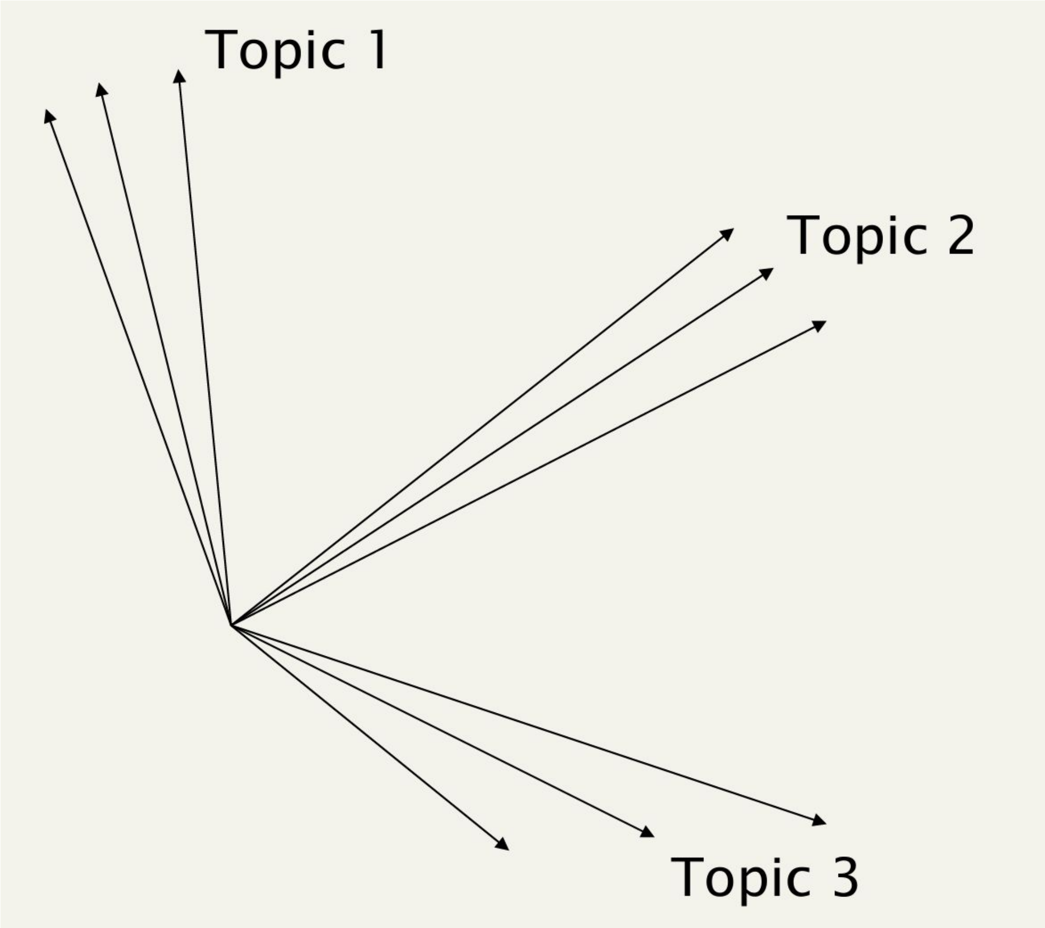
Intuition from block matrices



Intuition from block matrices



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.

LSI has many other applications

- In many settings in pattern recognition and retrieval, we have a feature-object matrix.
  - For text, the terms are features and the docs are objects.
  - Could be opinions and users …
  - This matrix may be redundant in dimensionality.
  - Can work with low-rank approximation.
  - If entries are missing (e.g., users' opinions), can recover if dimensionality is low.
- Powerful general analytical technique
  - Close, principled analog to clustering methods.

Applications of LSI/LSA

- Enterprise search
- Essay scoring
- Data/Text Mining
  - FAA safety incident reports
- Image Processing
  - eigenfaces

References

1. Slides provided by Sougata Saha (Instructor, Fall 2022 - CSE 4/535)
2. Materials provided by Dr. Rohini K Srihari
3. https://nlp.stanford.edu/IR-book/information-retrieval-book.html